مجانية
لجميع طلاب
البكالوريا
المصرية
★★★★★

وفق أحدث
تطوير لمنهج
البكالوريا
المصرية
2026

مستر
إبراهيم نجم

برمجة البكالوريا المصرية

شرح المنهج

(عربي وإنجليزي)

★ تعلم بذكاء... برمج مستقبلك ★

ملزمة التأسيس المجانية

ابدأ رحلتك في البرمجة من الصفر

ARTIFICIAL INTELLIGENCE
الذكاء الاصطناعي
تعرف على عالم الذكاء الاصطناعي وتطبيقاته في الحياة اليومية

ICT
تكنولوجيا المعلومات والاتصالات
اكتسب مهارات استخدام التكنولوجيا بذكاء واحتراف

COMPUTER SCIENCE
علوم الحاسب
افهم أساسيات علوم الحاسب والخوارزميات والتفكير المنطقي

PYTHON
البرمجة
تعلم البرمجة خطوة بخطوة بأسلوب بسيط ومبسط

باركود الطالب

امسح الباركود
هنا

الاسم : .....
الصف : .....
المدرسة : .....

مميزات الملزمة

- مناسب للمبتدئين من الصفر
- تطوير مهارات التفكير وحل المشكلات
- أسئلة واختيارات تأسيسية
- أمثلة وتدريبات متنوعة
- شرح مبسط للمفاهيم الأساسية

تابعنا على
MrIbrahimNegm

امسح الكود
لزيارة منصة
مستر إبراهيم نجم
ibrahim.in-masr.com

للتواصل والاستفسار
01224194220

مرحبًا بك

عزيزي الطالب،

- أهلاً بك في ملزمة التأسيس لبرمجة البكالوريا المصرية 2026.
- صُممت هذه الملزمة لتساعدك على بناء أساس قوي في البرمجة بأسلوب بسيط ومنظم يناسب جميع الطلاب.
- هدفنا ليس حفظ المعلومات فقط، بل تنمية مهارات التفكير والتحليل وحل المشكلات.

تعلم بذكاء... برمج مستقبلك.

Ibrahim Negm

أ / إبراهيم نجم

برمجة البكالوريا المصرية

محتويات الملزمة

دليلك الشامل لبناء أساس قوي في البرمجة خطوة بخطوة حتى الاحتراف.

"النجاح لا يأتي بالصدفة، بل بالعلم والعمل والمثابرة."

ما هي البرمجة؟

البرمجة هي كتابة الأوامر والتعليمات بلغة يفهمها الحاسوب لكي يقوم بتنفيذها وإخراج النتائج المطلوبة.

كيف تعمل البرمجة؟

1. نكتب الأوامر (البرنامج)
2. يفهمها الحاسوب
3. يقوم بتنفيذها ويعطي النتائج

أمثلة على البرمجة في حياتنا

- تطبيقات الهاتف التي نستخدمها
- الألعاب الإلكترونية
- المواقع الإلكترونية
- السيارات الذكية
- الأجهزة الذكية

لماذا نتعلم البرمجة؟

- تنمي التفكير المنطقي
- تساعد على حل المشكلات
- تفتح مجالات عمل ممتازة
- تجعل أفكارك حقيقة

تعلم بذكاء... برمج مستقبلك.

أين نستخدم البرمجة؟

البرمجة موجودة في **كل مكان حولنا**، وتستخدم في مجالات كثيرة **تسهل حياتنا** اليومية.

أمثلة من الحياة اليومية

الألعاب الإلكترونية

جميع الألعاب التي نلعبها تعتمد على البرمجة.

المواقع الإلكترونية

كل موقع على الإنترنت تم إنشاؤه بواسطة البرمجة.

الهواتف الذكية

التطبيقات التي نستخدمها يومياً يتم برمجتها.

السيارات الحديثة

البرمجة تتحكم في المحرك، الأمان، والتكييف وغيرها.

التسوق الإلكتروني

المتاجر الإلكترونية وأنظمة الدفع تعمل بالبرمجة.

المستشفيات

تستخدم البرمجة في حفظ البيانات وإدارة الأجهزة.

تذكر: البرمجة ليست مجرد كتابة أكواد، بل هي **حل للمشكلات** وصناعة **للمستقبل**.

لماذا نتعلم البرمجة؟

تعلم البرمجة مهارة أساسية في عصر التكنولوجيا،
وتفتح أمامك **فرصاً كثيرة** في المستقبل.

فوائد تعلم البرمجة

تبني مستقبلك
تعلم البرمجة اليوم يضمن لك مستقبلاً أفضل غدا.

تفتح فرص عمل مميزة
مجال البرمجة من أكثر المجالات طلباً في سوق العمل.

تنمي التفكير المنطقي
تساعدك على تحليل المشكلات وحلها خطوة بخطوة.

تعزز الثقة بالنفس
إكمال أي برنامج بنجاح يعزز شعورك بالإنجاز.

مفيدة في كل المجالات
البرمجة تدخل في الطب، الهندسة، التعليم، التجارة وغيرها.

تزيد من الإبداع
يمكنك إنشاء برامج وألعاب وتطبيقات خاصة بك.

كل ما تتعلمه اليوم في البرمجة،
هو **استثمار حقيقي** في مستقبلك.

ما هي البرمجة؟

البرمجة هي كتابة الأوامر والتعليمات بلغة يفهمها الحاسوب لتنفيذ مهام محددة وإنتاج نتائج معينة.

مكونات أي برنامج

1. مدخلات
البيانات التي ندخلها إلى البرنامج.

2. معالجة
البرنامج يعالج البيانات حسب الأوامر.

3. مخرجات
النتائج التي نحصل عليها بعد المعالجة.

لغات البرمجة

لغة البرمجة هي الوسيلة التي نستخدمها لكتابة الأوامر.
هناك لغات كثيرة، مثل:

Python | Java | C++ | JavaScript | Scratch

تذكّر

البرمجة ليست حفظ أكواد، بل
فهم طريقة التفكير وحل المشكلات بأسلوب منطقي.

ما هي Python ؟

Python هي لغة برمجة عالية المستوى، سهلة التعلم والاستخدام، تستخدم في مجالات كثيرة لحل المشكلات وبناء التطبيقات.

أهم مميزات Python

مجتمع كبير
لها مجتمع ضخم يدعمها ويوفر الكثير من المصادر والأدوات.

إنتاجية عالية
تساعد المبرمج على إنجاز مهامه بسرعة وكتابة كود أقل لتنفيذ نفس العمل.

متعددة الاستخدامات
تستخدم في تطوير الويب، تحليل البيانات، الذكاء الاصطناعي وغيرها الكثير.

قوية ومرنة
تستخدم في مجالات متعددة ويمكنها التعامل مع مشاريع كبيرة.

سهلة التعلم
تستخدم كلمات واضحة وبسيطة تشبه اللغة الإنجليزية.

أين تستخدم Python؟

تطوير الألعاب
بناء ألعاب ثنائية الأبعاد.

أتمتة المهام
أتمتة الأعمال المتكررة وتوفير الوقت.

الذكاء الاصطناعي
تطوير أنظمة ذكية وتعلم الآلة.

تحليل البيانات
جمع البيانات وتحليلها واستخراج النتائج.

تطوير الويب
بناء مواقع وتطبيقات الويب.

تذكّر

Python ليست فقط لغة برمجة، بل هي أداة قوية تساعدك على تحويل أفكارك إلى واقع ملموس.

أول برنامج بلغة Python

في هذا الدرس، سنكتب أول برنامج لنا بلغة Python ونتعلم كيفية إظهار رسائل على الشاشة.

1. دالة ()print

تُستخدم الدالة ()print لعرض النصوص (الرسائل) على الشاشة.
تكتب بين أقواس، ويكون النص بين علامتي اقتباس.

2. مثال عملي

الكود (Code)

```
1  print("مرحبًا بك في Python")
```

النتيجة (Output)

```
مرحبًا بك في Python
```

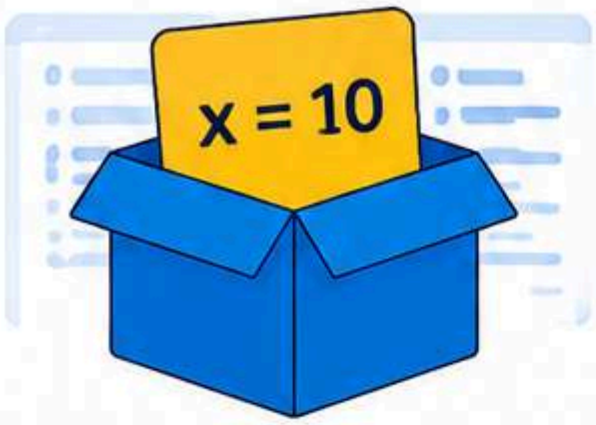
3. كيف يعمل البرنامج؟

- عند تشغيل البرنامج، ينفذ Python الأمر الموجود.
- تقوم الدالة ()print بإرسال النص الموجود بين الاقتباسات إلى الشاشة.
- تظهر الرسالة للمستخدم في نافذة الإخراج (Output).

جرب بنفسك

غيِّر النص داخل ()print واكتب رسالة ترحيب خاصة بك،
ثم شغِّل البرنامج وشاهد النتيجة.

المتغيرات Variables



المتغير هو مكان في الذاكرة يخزن قيمة يمكن استخدامها وتغييرها أثناء تنفيذ البرنامج.

1. قواعد تسمية المتغيرات

- لا يمكن استخدام الرموز الخاصة مثل @ #$ %.
- لا يمكن أن يبدأ الاسم برقم.
- يمكن أن يحتوي الاسم على حروف وأرقام و(_).
- يجب أن يبدأ الاسم بحرف أو شرطة سفلية (_).

2. أمثلة

أسماء غير صحيحة

- 1name
- age!
- total-score
- @student
- my name

أسماء صحيحة

- name
- age
- _value
- student1
- total_score

3. كيفية إنشاء متغير

يتم إنشاء المتغير باستخدام علامة المساواة (=) لإسناد قيمة له.

```
name = "Ali"
age = 16
height = 175.5
```

- متغير نصي (سلسلة نصية)
- متغير عددي صحيح
- متغير عددي عشري

تذكّر

المتغيرات تجعل برامجك أكثر مرونة، لأنها تسمح لك بتخزين البيانات واستخدامها أكثر من مرة.

أنواع البيانات
Data Types

تُستخدم أنواع البيانات لتحديد نوع القيمة المخزنة في المتغير،
وكل نوع له خصائص واستخدامات مختلفة.

1. أهم أنواع البيانات في Python

String	Int	Float	NoneType	Boolean
"abc"	10	3.14	None	✓
مثال: "Hello" 'Python'	مثال: 10 -25	مثال: 3.14 -0.5	مثال: None	مثال: True False
تخزن النصوص بين علامات اقتباس.	تخزن الأعداد الصحيحة.	تخزن الأعداد العشرية (ذات الجزء الكسري).	تمثل عدم وجود قيمة أو قيمة فارغة.	تخزن قيمتين فقط: صواب أو خطأ.

2. أمثلة على الاستخدام

```
1  name = "Ali"          # String
2  age = 16              # Int
3  height = 175.5        # Float
4  is_student = True     # Boolean
5  score = None          # NoneType
```

→ نص (سلسلة من الأحرف).
→ عدد صحيح.
→ عدد عشري.
→ قيمة منطقية (صواب / خطأ).
→ **لا توجد قيمة.**

3. معرفة نوع البيانات

يمكنك معرفة نوع أي متغير باستخدام الدالة type().

```
print(type(10))       # <class 'int'>
print(type(3.14))     # <class 'float'>
print(type("Ali"))    # <class 'str'>
print(type(True))     # <class 'bool'>
print(type(None))     # <class 'NoneType'>
```

النتيجة

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
<class 'NoneType'>
```

تذكّر

اختيار نوع البيانات المناسب يساعد على كتابة برامج أكثر كفاءة
وتجنب الأخطاء أثناء التنفيذ.

المعاملات Operators

المعاملاتُ هى رموز تُستخدم لإجراء عمليات على القيم والمتغيرات وإنتاج نتائج جديدة.

1. أنواع المعاملات في Python

المعاملات المنطقية

تُستخدم لدمج الشروط.

المعامل	المعنى
and	و (كل الشرطين صحيح)
or	أو أحد الشرطين صحيح)
not	ليس (يعكس النتيجة)

تُرجع نتيجة منطقية (True أو False)

المعاملات الحسابية

تُستخدم لإجراء العمليات الحسابية.

المعامل	المعامل
جمع	+
طرح	-
ضرب	*
قسمة	/
قسمة صحيحة (تجاهل الباقي)	//
باقي القسمة	%
أس (قوة)	**

معاملات المقارنة

تُستخدم لمفارنة قيمتين.

المعامل	المعامل
يساوي	==
لا يساوي	!=
أكبر من	>
أصغر من	<
أكبر من أو يساوي	>=
أصغر من أو يساوي	<=

تُرجع نتيجة منطقية (True) أو (False)

2. أمثلة عملية

```python
a = 10
b = 3
print(a + b)        # جمع
print(a - b)        # طرح
print(a * b)        # ضرب
print(a / b)        # قسمة
print(a // b)       # قسمة صحيحة
print(a % b)        # باقي القسمة
print(a ** b)       # أس
print(a > b)        # أكبر من
print(a == 10)      # يساوي
print(a != b)       # لا يساوي
print(not (a < b))  # ليس أصغر من
```

النتيجة (Output)

```
13
7
30
3.3333333333333335
3
1
1000
True
True
True
True
True
```

3. ترتيب تنفيذ المعاملات

يتم تنفيذ العمليات حسب الأولوية التالية:

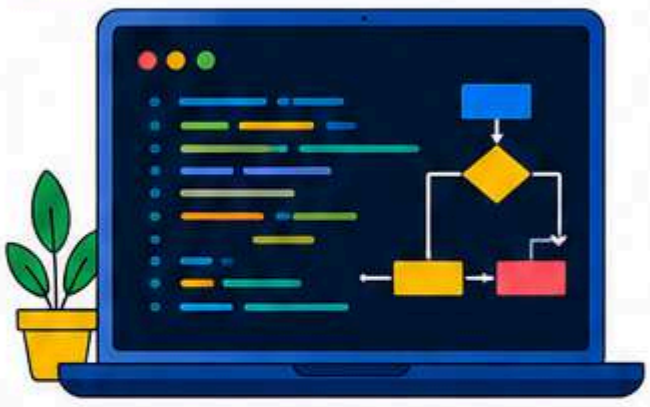
( ) أقواس > ** الأس > * % // الضرب والقسمة والباقي والقسمة الصحيحة > + - الجمع والطرح > > < >= <= المفارنة > not and or المنطقية

مثال:

3 + 5 * (2 - 1) ** 2

النتيجة: 8

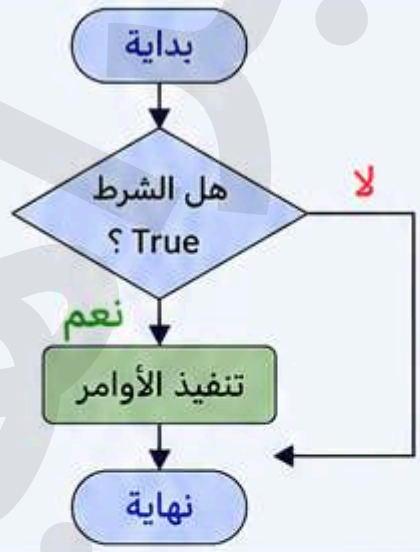
تذكَّر فهم المعاملاتَ جيدًا يساعدك على كتابة برامج دقيقة وصحيحة.



التحكم في التدفق
Control Flow

تُستخدم تراكيب التحكم لاتخاذ قرارات وتنفيذ أوامر معينة بناءً على شروط مُختلفة.

1. جملة الشرط if



تنفذ الأوامر داخل الكتلة إذا كان الشرط True.

النتيجة:

x أكبر من 10
انتهى

مثال:

```
x = 15
if x > 10:
    print("x أكبر 10")
print("انتهى")
```

2. جملة الشرط if-else

تنفذ الكتلة الاولى إذا كان الشرط True،
وإلا تنفذ الكتلة الثانية.

النتيجة:

x عدد زوجي

مثال:

```
x = 8
if x % 2 == 0:
    print("x عدد زوجي")
else:
    print("x عدد فردي")
```

3. جملة elif متعددة الشروط

تُستخدم لاختبار عدة شروط بالتتابع حتى يتحقق شرط واحد فقط.

الدرجة	التقدير
90 فأكثر	ممتاز
75 إلى 89	جيد جدًا
60 إلى 74	جيد
أقل من 60	مقبول

يتم تنفيذ أول شرط يكون True فقط، ثم يتم تجاوز باقي الشروط.

مثال:

```
mark = 85
if mark >= 90:
    print("ممتاز")
elif mark >= 75:
    print("جيد جدًا")
elif mark >= 60:
    print("جيد")
else:
    print("مقبول")
```

نصائح مهمة

- استخدم المسافات البادئة (Indentation) بشكل صحيح.
- تأكد من تغطية جميع الحالات باستخدام else عند الحاجة.

الدوال في Python

الدالة هي مجموعة من الأوامر تُنفَّذ عند استدعائها.
تساعد على تنظيم الكود وإعادة استخدامه.

1. تعريف دالة

الصيغة العامة:

```
def اسم_الدالة():
    # أوامر الدالة
```

طريقة العمل:

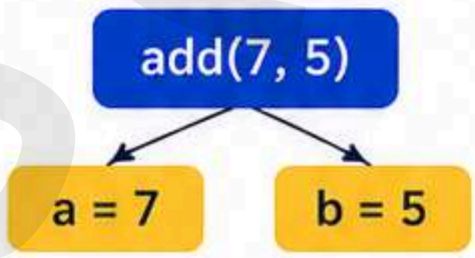
يتم تعريف الدالة مرة واحدة،
ثم يمكن استدعاؤها في أي مكان داخل البرنامج.

مثال:

```
1 def greet():
2     print("مرحبًا بك")
3
4 greet()  # استدعاء الدالة
```

2. دالة بالمعاملات

تستقبل الدالة قيمًا (معاملات) وتستخدمها داخل الأوامر.

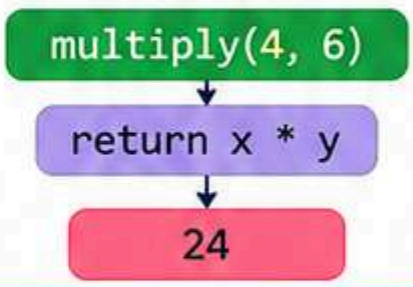


مثال:

```
1 def add(a, b):
2     result = a + b
3     print(result)
4
5 add(7, 5)    # الناتج :12
```

3. دالة تُرجع قيمة

تُعيد الدالة قيمة باستخدام return للاستفادة منها لاحقًا.



مثال:

```
1 def multiply(x, y):
2     return x * y
3
4 result = multiply(4, 6)
5 print(result) # الناتج :24
```

نصيحة

استخدام الدوال يجعل الكود مرتبًا ، سهل التعديل،
ويقلل من تكرار الأوامر.

القوائم في Python

القائمة هي مجموعة مرتبة من العناصر يمكن تغييرها.
تُكتب بين أقواس مربعة [ ] وتفصل العناصر بفاصلة.

1. إنشاء قائمة

```python
fruits = ["تفاح", "موز", "عنب", "مانجو"]
numbers = [5, 10, 15, 20]
mixed = [1, "Ali", 3.5, True]
empty = []  # قائمة فارغة
```

أمثلة على القوائم

- `fruits` — قائمة نص
- `numbers` — قائمة أرقام
- `mixed` — قائمة متعددة الأنواع
- `empty` — قائمة فارغة

2. الوصول إلى عناصر القائمة

```python
cities = ["القاهرة", "الإسكندرية", "أسوان", "طنطا"]

print(cities[0])    # القاهرة
print(cities[1])    # الإسكندرية
print(cities[-1])   # طنطا (آخر عنصر)
print(cities[-2])   # أسوان (قبل الأخير)
```

الفهارس (Indexes)

0	1	2	3
"القاهرة"	"الإسكندرية"	"أسوان"	"طنطا"
-4	-3	-2	-1

بيدأ الفهرس من 0
ويمكن استخدام الفهارس السالبة من نهاية القائمة.

3. تعديل عناصر القائمة

```python
items = [10, 20, 30, 40]

items[1] = 25    # تعديل العنصر
items[3] = 50    # تعديل العنصر
print(items)     # [10, 25, 30, 50]
```

	0	1	2	3
قبل التعديل	10	20	30	40
بعد التعديل	10	25	30	50

4. العمليات الأساسية على القوائم

العملية	الوصف	مثال
`len(list)`	يعيد عدد عناصر القائمة	`len([1, 2, 3, 4])` → 4
`list.append(x)`	يضيف عنصر x إلى نهاية القائمة	`[1, 2].append(3)` → [1, 2, 3]
`list.insert(i, x)`	يضيف عنصر x في الفهرس i	`[1, 3].insert(1, 2)` → [1, 2, 3]
`list.remove(x)`	يحذف أول عنصر قيمته x	`[1, 2, 3].remove(2)` → [1, 3]
`list.pop(i)`	يحذف ويرجع العنصر في الفهرس i	`[10, 20, 30].pop(1)` → 20
`list.clear()`	يحذف جميع العناصر	`a = [1, 2]  a.clear()` → []

مثال تطبيقي

```python
a = [100, 200]
a.append(300)
a.insert(1, 150)
a.remove(200)
x = a.pop(0)
a.clear()
```

القيمة النهائية لـ a:
[]
القيمة المحذوفة: 100

تذكّر: القوائم من أكثر الهياكل استخدامًا في البرمجة، تدرب عليها كثيرًا لتصبح أكثر إتقانًا.

التكرارات في Python

التكرارات تُستخدم لتنفيذ مجموعة من الأوامر
أكثر من مرة بدون تكرار كتابة الكود.

1. حلقة for

الصيغة العامة

```
for متغير in range (بداية , نهاية , خطوة)
    الأوامر
```

- بداية: القيمة التي يبدأ منها التكرار (تضمن).
- نهاية: القيمة التي يتوقف عندها (لا تُضمن).
- خطوة: مقدار الزيادة في كل تكرار (افتراضي 1).

```
for i in range(1, 6):
    print(i)
```

يطبع الأرقام من 1 إلى 5

مثال آخر

```
for x in range(2, 11, 2):
print(x)
```

النتيجة:
2 4 6 8 10

2. حلقة while

الصيغة العامة

```
while شرط
    الأوامر
```

- يتم تنفيذ الأوامر طالما الشرط True.
- يجب تحديث قيمة المتغير داخل الحلقة لتجنُّب التكرار اللانهائي.

```
count = 1
while count <= 5:
    print(count)
    count += 1
```

يطبع الأرقام من 1 إلى 5

مثال آخر

```
n = 10
while n > 0:
    print(n)
    n -= 2
```

النتيجة:
10 8 6 4 2

3. أوامر التحكم في التكرار

continue

يستخدم لتخطي التكرار الحالي والانتقال للتكرار التالي.

```
for i in range(1, 6):
  if i == 3:
      continue
  print(i)
```

النتيجة:
1 2 4 5

break

يستخدم لإيقاف الحلقة والخروج منها مباشرة.

```
for i in range(1, 10):
  if i == 5:
      break
  print(i)
```

النتيجة:
1 2 3 4

4. التكرارات المتداخلة

تتكرر الحلقة الداخلية بالكامل
في كل تكرار للحلقة الخارجية.

	j=1	j=2	j=3
i=1	●	●	●
i=2	●	●	●
i=3	●	●	●

```
for i in range(1, 4):
    for j in range(1, 4):
        print(f"i={i}, j={j}")
```

النتيجة:

```
i=1, j=1
i=1, j=2
i=1, j=3
i=2, j=1
i=2, j=2
i=2, j=3
i=3, j=1
i=3, j=2
i=3, j=3
```

نصائح

- استخدم التكرارات لتقليل عدد الأسطر وجعل الكود أكثر كفاءة.
- تأكد من وجود شرط إيقاف في حلقات while لتجنب التكرار اللانهائي.
- جرب أمثلة مختلفة لتفهم طريقة عمل التكرارات جيدًا.

السلاسل النصية في Python

السلسلة النصية (String) هي مجموعة من الحروف والأرقام والرموز تكتب بين علامات اقتباس مفردة ' ' أو مزدوجة " ".

1. إنشاء سلاسل نصية

```
name = "Omar"
city = 'Cairo'
message = "Hello, Python!"
multi_line = """
هذا نص متعدد الأسطر
    Python
"""
```

ملاحظات

- يمكن استخدام ' أو " .
- يمكن استخدام """ للنصوص متعددة الأسطر.
- السلسلة النصية غير قابلة للتعديل.

2. الوصول إلى الأحرف (الفهرسة والتقطيع)

مثال: `text = "Programming"`

الفهرس	0	1	2	3	4	5	6	7	8	9	10
الحرف	P	r	o	g	r	a	m	m	i	n	g
الفهرس السالب	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
text[0]      # 'P'
text[3]      # 'g'
text[-1]     # 'g'
text[1:5]    # 'rogr'      (من 1 إلى 4)
text[5:]     # 'amming'    بداية الكلمة من 5
text[:4]     # 'Prog'      نهاية الكلمة عند 4
text[::2]    # 'Pormn'     (كل حرفين)
text[::-1]   # 'gnimargorP' (عكس السلسلة)
```

3. أهم الدوال (الطرق) في السلاسل النصية

الدالة	الوصف	مثال	النتيجة
`len(s)`	إرجاع طول السلسلة	`len("Hello")`	5
`lower()`	تحويل جميع الأحرف إلى صغيرة	`"PYTHON".lower( )`	"python"
`upper()`	تحويل جميع الأحرف إلى كبيرة	`"python".upper( )`	"PYTHON"
`strip()`	إزالة الفراغات من البداية والنهاية	`"  hello  ".strip( )`	"hello"
`replace(a, b)`	استبدال جزء من النص	`"one two one".replace("one", "1")`	"1 two 1"
`split(sep)`	تقسيم السلسلة إلى قائمة	`"a,b,c".split(",")`	["a", "b", "c"]
`join(iterable)`	دمج عناصر قابلة للتكرار بسلسلة	`"-".join(["2026", "6", "15"])`	"2026-6-15"
`find(sub)`	إرجاع فهرس أول ظهور للجزء	`"hello world".find("world")`	6
`count(sub)`	عدد مرات تكرار الجزء	`"banana".count("a")`	3

4. تنسيق السلاسل (f-strings)

```
name = "Mona"
age = 18
score = 92.5

msg = f"""my name is {name},
 I am {age} years old
and my score is {score}"""

print(msg)
```

النتيجة:

اسمي Mona وعمري 18
حصلت على 92.5 من 100

مميزات f-strings

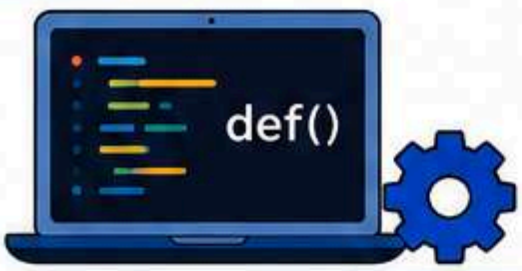
- سهلة القراءة والكتابة.
- تدعم المتغيرات والتعبيرات.
- تدعم تنسيق الأرقام والتواريخ.

مثال: `i=f"{score:.1f}"`
يظهر الرقم بمنزلة عشرية واحدة.

نصائح مهمة

- جرب الدوال المختلفة لتصبح أكثر كفاءة في معالجة النصوص.
- استخدم الفهرسة السالبة للوصول من نهاية السلسلة.
- السلاسل النصية لا يمكن تغييرها (immutable).

الدوال في Python



الدالة هي مجموعة من الأوامر تؤدي مهمة محددة عند استدعائها.
تساعد على تنظيم البرنامج وإعادة استخدام الكود.

1. تعريف دالة

```
1 def square(x):
2     result = x * x
3     return result
```

هذه الدالة تستقبل رقما وتعيد مربعه.

كلمة تعريف الدالة : def
اسم يعبر عن اسم_الدالة:
المعاملات قُيم تُرسل للدالة (اختيارية):
تُعيد قيمة (اختيارية) :return

الصيغة العامة

```
def (اسم_الدالة) (المعاملات)
    أوامر الدالة
    return قيمة
```

2. استدعاء دالة

الصيغة العامة

```
result = (اسم_الدالة )
```

- تُنفذ الأوامر داخل الدالة.
- تُعيد الدالة القيمة الناتجة بـ return.
- يمكن استخدام القيمة في العمليات أو الإخراج.

مثال

```
1 def square(x):
2     return x * x
3
4 n = 7
5 s = square(n)
6 print(s)     # الناتج: 49
```

3. دوال بدون معاملات

مثال

```
1 def greet():
2     print("مرحبًا بك!")
3     print("نتمني لك يومًا سعيدًا.")
4
5 greet()
```

الشرح

- الدالة لا تستقبل أي قيم.
- تنفذ الأوامر وتقوم بطباعة الرسائل.
- لا يوجد return لأننا لا نعيد قيمة.

الناتج

```
مرحبًا بك!
نتمني لك يومًا سعيدًا.
```

4. دوال بمعاملات متعددة

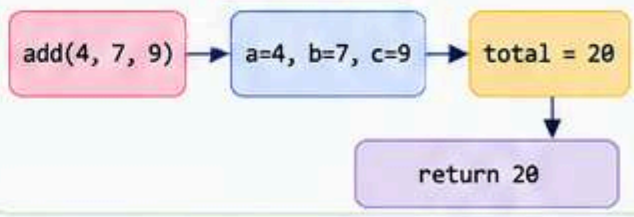
مثال

```
1 def add(a, b, c):
2     total = a + b + c
3     return total
4
5 x = add(4, 7, 9)
6 print(x)      # الناتج: 20
```

شرح

- الدالة تستقبل 3 قيم.
- تجمع القيم داخل الدالة.
- تعيد الناتج باستخدام return.

تدفق التنفيذ



5. نطاق المتغيرات (Scope)

خارج الدالة (عام)

المتغير العام يمكن استخدامه في كل البرنامج.

```
1 y = 20        عام
2 def show():
3     print(y)
4
5 show()
6 print(y)      # 20
```

داخل الدالة (محلي)

متغير محلي لا يمكن استخدامه خارج الدالة.

```
1 def test():
2     x = 10    محلي
3     print(x)
4
5 test()
6 # test (x) ✗ خطأ: x
```

ملاحظة مهمة

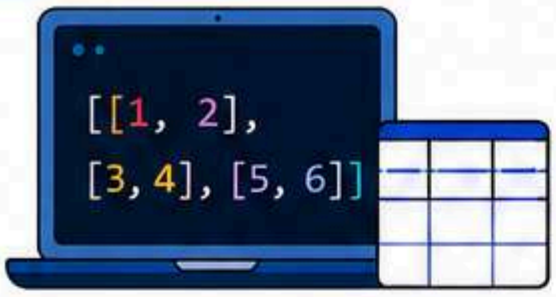
إذا كان هناك متغير داخل الدالة بنفس اسم متغير عام، يتم استخدام المتغير المحلي فقط داخل الدالة.

تحذير

لا يمكن تعديل المتغير العام داخل الدالة إلا باستخدام الكلمة global.

نصيحة

استخدم الدوال لجعل برنامجك منظما، سهل الفهم، وقابلاً لإعادة الاستخدام.
كلما كنت تقسم برنامجك إلى دوال صغيرة، كان أفضل وأسهل في الصيانة.



القوائم المتداخلة في Python

القائمة المتداخلة هي قائمة تحتوي على قوائم أخرى بداخلها.
تُستخدم لتنظيم البيانات في شكل جدول (صفوف وأعمدة).

1. إنشاء قائمة متداخلة

```python
data = [
    [11, 22, 33],
    [44, 55, 66],
    [77, 88, 99]
]
print(data)
```

مفهوم الصفوف والأعمدة

الصفوف \ الأعمدة	0	1	2
0	11	22	33
1	44	55	66
2	77	88	99

الوصول للعناصر

```
data[0][0] = 11   (الصف 0، العمود 0)
data[1][2] = 66   (الصف 1، العمود 2)
data[2][1] = 88   (الصف 2، العمود 1)
```

2. عمليات شائعة على القوائم المتداخلة

```python
data = [
    [2, 4, 6],
    [8, 10, 12],
    [14, 16, 18]
]
print(data[1][1])      # 10
print(len(data))       # 3 (عدد الصفوف)
print(len(data[0]))    # 3 (عدد الأعمدة)
data[0][2] = 100
print(data[0])         # [2, 4, 100]
```

العملية	النتيجة
data[1][1]	10
len(data) (عدد الصفوف)	3
len(data[0]) (عدد الأعمدة)	3
data[0][2] إلى 100	[2, 4, 100]

3. المرور على عناصر القائمة المتداخلة

```python
data = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
for row in data:
    for item in row:
        print(item, end=' ')
    print()
```

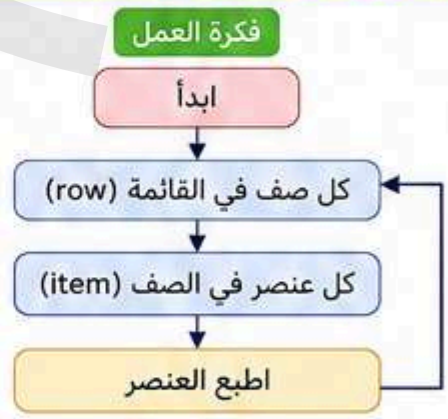
الشرح

نتنقل على كل صف (row) ثم على كل عنصر (item) داخل الصف.

النتيجة على الشاشة:

```
1 2 3
4 5 6
7 8 9
```

فكرة العمل



4. مثال تطبيقي

```python
students = [
    ['Ali', 85, 90],
    ['Mona', 78, 92],
    ['Omar', 88, 76]
]
for s in students:
    name = s[0]
    math = s[1]
    eng  = s[2]
    total = math + eng
    print(name, '->', total)
```

شرح البيانات

الفكرة: لدينا قائمة تحتوي على بيانات كل طالب: الاسم، درجة الرياضيات، ودرجة الإنجليزي. سنقوم بحساب المجموع (الرياضيات + الإنجليزي) لكل طالب.

الصف	الاسم	الرياضيات	الإنجليزي	المجموع
0	Ali	85	90	175 (85 + 90)
1	Mona	78	92	170 (78 + 92)
2	Omar	88	76	164 (88 + 76)

ملاحظة: المجموع = درجة الرياضيات + درجة الإنجليزي

النتيجة

```
Ali  -> 175
Mona -> 170
Omar -> 164
```

نصيحة

تدرّب على بناء جداول ومعالجتها بطرق مختلفة.

- القائمة المتداخلة تشبه الجدول (صفوف × أعمدة).
- الوصول لأي عنصر يكون باستخدام [رقم الصف][رقم العمود].
- يمكن التعديل على العناصر كما في القوائم العادية.

الدوال في Python

الدالة هي مجموعة أوامر تؤدي مهمة محددة عند استدعائها.
تساعد على تنظيم البرنامج وإعادة استخدام الكود وتقليل التكرار.

1. تعريف الدالة

- الكلمة المحجوزة لتعريف دالة: `def`
- اسم الدالة: `اسم_الدالة`
- قيم تُرسل للدالة (اختيارية): `(معاملات)`
- تعيد قيمة من الدالة (اختياري): `return`

الصيغة العامة

```
def (اسم_الدالة) ( المعاملات) :
    أوامر الدالة
    return قيمة
```

```python
def add(a, b):
    result = a + b
    return result

x = add(5, 7)
print(x)    # النانج : 12
```

2. أنواع المعاملات

النوع	الوصف	مثال	الكود	النتيجة

بدون معاملات	دالة لا تستقبل أي معاملات.	تحية بسيطة	`def hi():`
			`  print("مرحبا")`
			`hi()`
				مرحباً

بمعاملات شرطية (مطلوبة)	يجب تمرير قيمة لكل معامل.	جمع رقمين	`def add(a, b):`
			`  return a + b`
			`print(add(4, 6))`
				10

بمعاملات افتراضية (اختيارية)	للمعامل قيمة افتراضية تُستخدم إذا لم تُمرَّر قيمة.	عرض رسالة	`def show(msg="مرحباً"):`
			`  print(msg)`
			`show()  # بدون معامل`
			`show("أهلاً")`
				مرحباً
				أهلاً بك

3. الدوال التي تُعيد قيمة والتي لاتُعيد

دالة تُعيد قيمة (return موجود)

```python
def square(x):
    return x * x

n = square(6)
print(n)      # النانج : 36
```

هذه الدالة تُعيد القيمة إلى المكان الذي استُدعيت منه.

دالة لا تُعيد قيمة (return غير موجود)

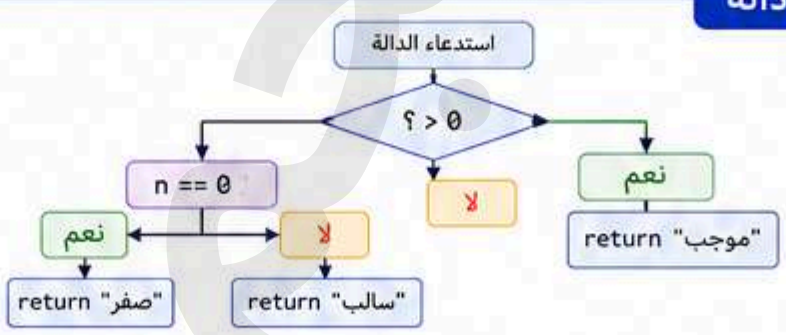
```python
def greet(name):
    print("أهلاً ", name)

greet("Ali")    # النانج : أهلاً Ali
```

هذه الدالة تُنفذ الأوامر فقط ولا تُعيد قيمة.

4. أكثر من return في دالة

```python
def check(n):
    if n > 0:
        return "موجب"
    elif n == 0:
        return "صفر"
    else:
        return "سالب"
print(check(5))      # موجب
print(check(0))      # صفر
print(check(-3))     # سالب
```



النتيجة
موجب
صفر
سالب

5. المتغيرات المحلية والعامة

متغير عام (خارج الدالة)

يُستخدم في جميع أنحاء البرنامج.

```python
x = 20          # عام
def fun():
    print(x)    # يقرأ القيمة العامة
fun()
print(x)    # 20
```

متغير محلي (داخل الدالة)

يُستخدم داخل الدالة فقط.

```python
def fun():
    x = 10   # محلي
    print(x)
fun()
# print(x)    # خطأ : x
```

ملاحظة

- الدالة تتعامل مع المتغيرات المحلية الخاصة بها.
- بمكن للدالة قراءة المتغيرات العامة وليس تعديلها إلا باستخدام `global`.

نصائح مهمة

- اختر أسماء دوال واضحة تعبر عن وظيفتها.
- استخدم الدوال لتقسيم البرنامج إلى أجزاء صغيرة.
- جرب دوالاً مختلفة لتفهم طريقة عملها بشكل أفضل.

القواميس في Python

القاموس (Dictionary) هو مجموعة غير مرتبة من العناصر،
تتكون من أزواج (مفتاح : قيمة) حيث يكون كل مفتاح فريد.

1. إنشاء قاموس

```python
student = {
    "name": "Ali",
    "age": 17,
    "grade": 92
}
print(student)
```

مثال

```python
student = {
    "name": "Ali",
    "age": 17,
    "grade": 92
}
```

النتيجة:

```
{'name': 'Ali', 'age': 17, 'grade': 92}
```

ملاحظات مهمة

- المفاتيح تكون فريدة (لا تتكرر).
- المفاتيح تكون من نوع غير قابل للتغيير (مثل: str, int, float, tuple).
- القيم يمكن أن تكون من أي نوع.
- القاموس غير مرتب (قبل Python 3.7) أما من Python 3.7 فأصبح يحتفظ بترتيب الإدخال.

2. الوصول إلى العناصر

```python
student = {"name": "Mona", "age": 16, "city": '
print(student["name"])    # Mona
print(student["age"])     # 16
print(student["city"])    # Cairo
```

طرق الوصول

```python
student["name"]     # باستخدام المفتاح
student.get("age")  # طريقة آمنة
```

الاستخدام ()get

تُستخدم ()get لتجنب الخطأ إذا كان المفتاح غير موجود.

```python
student.get("phone")  # None
student.get("phone", " غير موجود ")
# غير موجود
```

3. تعديل البيانات

```python
student = {"name": "Ali", "age": 17}
student["age"] = 18          # تعديل قيمة موجودة
student["grade"] = 95        # إضافة زوج جديد
del student["name"]          # حذف زوج
print(student)
```

النتيجة:

```
{'age': 18, 'grade': 95}
```

أهم العمليات

dict[key] = new_value	تعديل قيمة.
dict[new_key] = value	إضافة عنصر.
del dict[key]	حذف عنصر.

4. دوال وقطع مهمة في القواميس

الدالة	الوصف	مثال	النتيجة
keys()	ترجع جميع المفاتيح.	student.keys()	dict_keys(['name', 'age', 'grade'])
values()	ترجع جميع القيم.	student.values()	dict_values(['Ali', 17, 92])
items()	ترجع جميع الأزواج (مفتاح : قيمة).	student.items()	dict_items([('name', 'Ali'), ('age', 17), ('grade', 92)])
get(key)	ترجع قيمة المفتاح. وإذا لم يوجد ترجع None.	student.get("age")	17
update(dict2)	تضيف عناصر من قاموس آخر.	student.update({"age": 20})	يتم تحديث age إلى 20
clear()	يمسح جميع العناصر.	student.clear()	{} (قاموس فارغ)

5. مثال على قاموس متداخل

```python
students = {
    1: {"name": "Ali", "age": 17},
    2: {"name": "Mona", "age": 16}
}
print(students[1]["name"])   # Ali
print(students[2]["age"])    # 16
```

الشكل التوضيحي

المفتاح (ID)	القيمة (القاموس)
1	{"name": "Ali", "age": 17}
2	{"name": "Mona", "age": 16}

ملاحظة

يمكن أن تكون القيمة نفسها قاموساً آخر، أو قائمة، أو أي نوع.

نصائح

- استخدم القواميس عندما لديك بيانات مرتبطة بمفتاح واضح.
- تأكد من عدم تكرار المفاتيح.
- استخدم ()get للتعامل الآمن مع المفاتيح.

المجموعات في Python

المجموعة (**Set**) هي مجموعة غير مرتبة من عناصر **فريدة** (**لا تتكرر**).
تُستخدم لتخزين قيم مختلفة فقط.

1. إنشاء مجموعة

```python
fruits = {"apple", "banana", "orange"}
numbers = set([1, 2, 3, 4])
empty_set = set()
print(fruits)
print(numbers)
print(empty_set)
```

مثال

```
{'banana', 'apple', 'orange'}
{1, 2, 3, 4}
set()
```

ملاحظات مهمة

- العناصر داخل المجموعة لا تتكرر.
- ترتيب العناصر غير ثابت وقد يتغير.
- يمكن أن تحتوي على أنواع مختلفة.
- لإنشاء مجموعة فارغة نستخدم ()set وليس { }.

2. إضافة عنصر

```python
fruits = {"apple", "banana"}
fruits.add("orange")
fruits.add("banana")    # لن يتم إضافتها
print(fruits)
```

النتيجة

```
{'banana', 'orange', 'apple'}
```

لم يتم إضافة "banana" مرة أخري
لأنها موجودة بالفعل.

add( )

تضيف عنصر واحد فقط إلى المجموعة.
إذا كان العنصر موجوداً
فلن يتم تكراره.

3. حذف عنصر

```python
fruits = {"apple", "banana", "orange"}
fruits.remove("banana")    # إذا لم يكن موجوداً يحدث خطأ #
fruits.discard("apple")    # إذا لم يكن موجوداً لا يحدث خطأ #
print(fruits)
fruits.pop()               # حذف عنصر عشوائي
print(fruits)              #
fruits.clear()             # حذف كل العناصر
print(fruits)
```

النتيجة خطوة بخطوة

بعد remove:
`{'orange', 'apple'}`
بعد discard:
`{'orange'}`
بعد pop (يحذف عنصر عشوائي):
`{}` أو `set()`
بعد clear:
`set()`

طرق الحذف

remove(x)
يحذف العنصر x.
إذا لم يكن موجوداً يحدث خطأ.

discard(x)
يحذف العنصر x.
إذا لم يكن موجوداً لا يحدث خطأ.

pop()
يحذف عنصر عشوائي
(ولا يمكن تحديد العنصر).

clear()
يحذف جميع العناصر.

4. العمليات على المجموعات

العملية	الوصف	مثال	النتيجة
الاتحاد (Union)	يجمع كل العناصر من المجموعتين.	A.union(B) أو A | B	{1, 2, 3, 4, 5}
التقاطع (Intersection)	العناصر المشتركة بين المجموعتين.	A.intersection(B) أو A & B	{3}
الفرق (Difference)	العناصر الموجودة في A فقط.	A - B	{1, 2}
الفرق المتناظر (Symmetric Difference)	العناصر الموجودة في واحدة فقط من المجموعتين.	A ^ B	{1, 2, 4, 5}
الانتماء (Membership)	التحقق إذا كان عنصر موجود في المجموعة.	x in A	True أو False
عدم الانتماء	التحقق إذا كان عنصر غير موجود في المجموعة.	x not in A	True أو False

أمثلة سريعة:

```
A = {1, 2, 3, 4}
B = {3, 4, 5}
```

A | B → {1, 2, 3, 4, 5}
A & B → {3, 4}
A - B → {1, 2}
A ^ B → {1, 2, 5}
3 in A → True
6 not in B → True

5. خصائص مهمة للمجموعات

لا تدعم الفهرسة
لا يمكن الوصول للعناصر
بواسطة الفهرس.
`s = {1, 2, 3}`
`s[0]` ✗

تحتوي على أي نوع
يمكن أن تحتوي على
أرقام، نصوص، وغيرها.
`{1, "Ali", 3.5}`

قابلة للتغيير
يمكن إضافة أو حذف
العناصر.

غير مرتبة
لا تعتمد على ترتيب معين.
`{1, 2, 3}`
قد تظهر بهذا الشكل.
`{3, 1, 2}`

فريدة
لا تقبل التكرار.
`{1, 2, 2, 3}` ✗
`{1, 2, 3}` ✓

التعامل مع الملفات في Python

تستخدم الملفات لحفظ البيانات واسترجاعها بشكل دائم حتى بعد إغلاق البرنامج.
في Python نستخدم الدالة open() للتعامل مع الملفات.

أوضاع الفتح (Modes)

الوضع	المعنى
'r'	فتح الملف للقراءة فقط (الملف يجب أن يكون موجود)
'w'	فتح الملف للكتابة (يمسح المحتوى القديم إن وجد)
'a'	فتح الملف للإضافة (يضيف في نهاية الملف)
'r+'	فتح الملف للقراءة والكتابة معاً
'w+'	فتح الملف للقراءة والكتابة (يمسح المحتوى)
'a+'	فتح الملف للقراءة والإضافة

1. فتح ملف

```
f = open("example.txt", "r")
  # "r" للقراءة #
g = open("data.txt", "w")
  "w" للكتابة (يُنشئ ملف جديد #
    أو يمسح المحتوى القديم إن وجد #
h = open("info.txt", "a")
  "a" للإضافة إلى نهاية الملف #
f.close()   # غلق الملف
```

2. قراءة من ملف

```
f = open("example.txt", "r")
data = f.read()       # قراءة كل المحتوى
print(data)
f.close()
------------------------------
f = open("example.txt", "r")
line = f.readline()    # قراءة سطر واحد
print(line)
lines = f.readlines()  # قراءة كل الأسطر
print(lines)
f.close()
```

3. كتابة في ملف

```
f = open("output.txt", "w")
f.write("أهلاً بك في Python.\n")
f.write("هذا سطر جديد \n")
f.close()
--------------------------
f = open("notes.txt", "a")
f.write("سطر إضافي في نهاية الملف.\n")
f.close()
```

4. الطريقة المفضلة باستخدام with open

```
with open("example.txt", "r") as f:
    data = f.read()
    print(data)
# هنا يتم غلق الملف تلقائياً بعد الانتهاء
```

لماذا نستخدم with open ؟

- ✔ يُغلق الملف تلقائياً بعد الانتهاء.
- ✔ أكثر أماناً وأقل عرضة للأخطاء.
- ✔ يُفضل استخدامها دائماً.

مثال عملي

```
with open("students.txt", "w") as f:
    f.write("Ali\n")
    f.write("Mona\n")
    f.write("Omar\n")

with open("students.txt", "r") as f:
    for line in f:
        print(line.strip())
```

النتيجة:

```
Ali
Mona
Omar
```

ملاحظات مهمة

- إذا حاولت فتح ملف للقراءة وهو غير موجود سيظهر خطأ.
- عند الكتابة بوضع "w" سيتم حذف المحتوى القديم.
- إستخدم "a" إذا أردت إضافة بيانات بدون حذف القديم.
- تأكد دائماً من غلق الملف أو استخدام with open.

نصائح ذهبية

- ✔ استخدم أسماء واضحة ومنظمة.
- ✔ تأكد من مسار الملف صحيح قبل الفتح.
- ✔ احتفظ بنسخة احتياطية للملفات المهمة.
- ✔ جرب البرنامج بملفات صغيرة أولاً.

معالجة الأخطاء Python

الأخطاء تحدث أثناء تنفيذ البرنامج وتتسبب في توقفه.
باستخدام **try** و **except** نستطيع التعامل مع الأخطاء ومنع توقف البرنامج.

1. أنواع الأخطاء الشائعة

نوع الخطأ	الوصف
Syntax Error	خطأ في كتابة الكود (قواعد اللغة).
Runtime Error	يحدث أثناء تشغيل البرنامج.
Logical Error	البرنامج يعمل بدون أخطاء لكن النتيجة غير صحيحة.

أمثلة

Syntax Error:

```
print("Hello"
```
خطأ: قوس ناقص

Runtime Error:

```
x = 10 / 0
```
خطأ: قسمة على صفر

Logical Error:

```
x = 5 + 5
print(x)
```
الناتج 10 ولكن المطلوب ربما 15

2. البنية الأساسية (try - except)

```
try:
    الكود الذي قد يسبب خطأ #
except:
    الكود الذي ينفذ عند حدوث خطأ
else:
    الكود الذي ينفذ إذا لم يحدث خطأ
finally:
    الكود الذي ينفذ دائما في النهاية
```

3. مثال عملي بسيط

```
1  try:
2      x = int(input("إدخل رقما."))
3      y = 10 / x
4      print("الناتج" =", y)
5  except ZeroDivisionError:
6   print("خطأ : لا يمكن القسمة على صفر.")
7  except ValueError:
8      print("خطأ: يجب إدخال رقم.")
9  finally:
       print("تم إنهاء إلبرنامج.")
```

تجربة البرنامج

إذا أدخلنا: 2
الناتج:
الناتج = 5.0
تم إنهاء البرنامج.

إذا أدخلنا: 0
الناتج:
خطأ: لا يمكن القسمة على صفر.
تم انهاء البرنامج.

4. أكثر الاستثناءات شيوعاً

الاستثناء	السبب	مثال
ZeroDivisionError	محاولة القسمة على صفر.	`10 / 0`
ValueError	قيمة غير صحيحة لنوع البيانات.	`int("abc")`
TypeError	نوع بيانات غير مناسب للعملية.	`"5" + 3`
FileNotFoundError	ملف غير موجود.	`open("file.txt")`
IndexError	الوصول إلى فهرس غير موجود في قائمة.	`my_list[10]`

5. مثال مع أكثر من except

```
1  try:
2      n = int(input("إدخل رقما."))
3      result = 100 / n
4      print("النانج" =", result)
5  except ZeroDivisionError:
6      print("خطأ : لا يمكن القسمة على صفر.")
7  except ValueError:
8      print("خطأ: يجب إدخال رقم صحيح.")
9  except Exception as e:
10     print("حدث خطأ أخر ", e)
11 finally:
11     print("تم إنهاء البرنامج .")
```

نصائح ذهبية

- توقع حدوث الأخطاء وتجنب توقف البرنامج المفاجئ.
- لا تتجاهل الأخطاء، بل تعامل معها بطريقة مناسبة.
- اجعل رسائل الخطأ مفهومة للمستخدم.

ملاحظات مهمة

- يجب وضع الكود المحتمل حدوث خطأ داخل **try** فقط.
- يمكن كتابة أكثر من **except** للتعامل مع أكثر من نوع من الأخطاء.
- يُفضل دائماً استخدام **finally** لإغلاق الملفات أو تنفيذ مهام أخيرة.
- استخدم رسائل واضحة للمستخدم عند حدوث خطأ.

مشروع تطبيقي صغير Python

سنقوم ببناء برنامج بسيط وهو "ألة حاسبة" يمكنه إجراء العمليات الحسابية الأساسية.
الجمع + ، الطرح - ، الضرب * ، والقسمة /.

1. فكرة المشروع

- يطلب البرنامج من المستخدم إدخال رقمين.
- يختار المستخدم العملية الحسابية.
- يقوم البرنامج بإجراء العملية.
- يعرض النتيجة.

2. خطوات بناء البرنامج

1. طلب إدخال الرقم الأول من المستخدم.
2. طلب إدخال الرقم الثاني من المستخدم.
3. طلب اختيار العملية (+) , - , * , أو /).
4. إجراء العملية حسب اختيار المستخدم.
5. عرض النتيجة بشكل واضح.

3. الكود الكامل للبرنامج

```python
print("=== ألة حاسبة ===")
print()

# إدخال الرقمين
num1 = float(input("أدخل الرقم الأول: "))
num2 = float(input("أدخل الرقم الثاني: "))

# اختيار العملية
print("\nاختار العملية:")
print("1- جمع (+)")
print("2- طرح (-)")
print("3- ضرب (*)")
print("4- قسمة (/)")
choice = input("أدخل رقم العملية (1/2/3/4): ")

# إجراء العملية
if choice == '1':
    result = num1 + num2
    operation = "جمع"
elif choice == '2':
    result = num1 - num2
    operation = "طرح"
elif choice == '3':
    result = num1 * num2
    operation = "ضرب"
elif choice == '4':
    if num2 != 0:
        result = num1 / num2
        operation = "قسمة"
    else:
        print("خطأ : القسمة على الصفر غير مسموح")
        exit()
else:
    print("الرجاء اختيار رقم العملية صحيح")
    exit()

# عرض النتيجة
print("----------------------")
print(f"العملية: {operation}")
print(f"النتيجة: {result}")
print("----------------------")
```

4. شرح الكود

- السطران 5 و 6: نستخدم float لتحويل الإدخال إلى رقم عشري.
- السطر 9-13: عرض قائمة العمليات للمستخدم.
- السطر 14: حفظ اختيار المستخدم في المتغير choice.
- السطر 17-35: تنفيذ العملية المناسبة بناء على اختيار المستخدم.
- تتحقق من أن المقسوم عليه لا يساوي صفر.
- السطر 37-41: عرض اسم العملية والنتيجة النهائية.

5. مثال تشغيل البرنامج

```
=== ألة حاسبة بسيطة ===

أدخل الرقم الأول: 15
أدخل الرقم الثاني: 3

اختر العملية:
1- جمع (+)
2- طرح (-)
3- ضرب (*)
4- قسمة (/)
أدخل رقم العملية (1/2/3/4): 3

----------------------
العملية: ضرب
النتيجة: 45.0
----------------------
```

تجربة أخرى

ادخل الأرقام التالية:
الرقم الأول: 20
الرقم الثاني: 4
العملية: 1

النتيجة
24.0

6. ملاحظات مهمة

- استخدمنا if / elif / else لاتخاذ القرارات.
- استخدمنا exit() لإيقاف البرنامج عند حدوث خطأ.
- يمكن تحسين البرنامج بإضافة المزيد من العمليات.
- تأكد دائماً من إدخال أرقام صحيحة.

7. تطوير المشروع (اختياري)

- إضافة عملية (%): باقي القسمة.
- إضافة إمكانية إجراء عدة عمليات متتالية.
- حفظ التاريخ في ملف نصي (log).
- تصميم واجهة رسومية باستخدام Tkinter.

البرمجة الكائنية Python

البرمجة الكائنية (OOP) هي نمط برمجي يعتمد على "الكائنات" التي تحتوي على بيانات (خصائص) ودوال (طرق) للتعامل مع هذه البيانات.

1. الكائن (Object)

- هو مثيل من صنف (Class).
- يمتلك خصائص وطرق.

مثال:

```
student1 = Student()
student2 = Student()
```

2. الطرق (Methods)

- هي الدوال داخل الكائن.
- تستخدم للتعامل مع الخصائص.

مثال:

```
def show():
    print(name)
```

3. الخصائص (Attributes)

- هي المتغيرات داخل الكائن.
- تخزن البيانات الخاصة بالكائن.

مثال:

```
name = "Ali"
age = 18
```

تعريف صنف (Class)

تستخدم الكلمة المفتاحية `class` لتعريف صنف جديد.

```
class Student:
    def __init__(self, name, age):
        self.name = name      # خاصية (attribute)
        self.age = age        # خاصية (attribute)

    def show_info(self):      # طريقة (method)
        print(f"الاسم: {self.name}")
        print(f"العمر: {self.age}")
```

(Constructor) : دالة منشئ __init__().

- تستدعى تلقائيا عند إنشاء كائن جديد.
- تستخدم لتهيئة خصائص الكائن.

إنشاء كائنات (Objects) واستخدامها

```
# إنشاء كائنات من الصنف Student
s1 = Student("Ali", 18)
s2 = Student("Mona", 17)
# عرض معلومات الكائن الأول
s1.show_info()
print()
# عرض معلومات الكائن الثاني
s2.show_info()
```

الناتج:

```
الاسم: Ali
العمر: 18
----------------------------------------
الاسم: Mona
العمر: 17
```

مميزات البرمجة الكائنية (OOP)

1. التنظيم

تجميع البيانات والوظائف ذات الصلة في كائن واحد، مما يسهل إدارة البرامج الكبيرة.

2. إعادة الاستخدام

يمكن استخدام نفس الكود (الصنف) لإنشاء عدة كائنات مختلفة.

3. الحماية

يمكن إخفاء البيانات وحمايتها من التعديل المباشر.

4. التوسع

يمكن إضافة وظائف جديدة أو تعديلها دون التأثير على باقي أجزاء البرنامج.

مثال تطبيقي بسيط

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

r = Rectangle(5, 8)
print("المساحة =", r.area())
```

الناتج:

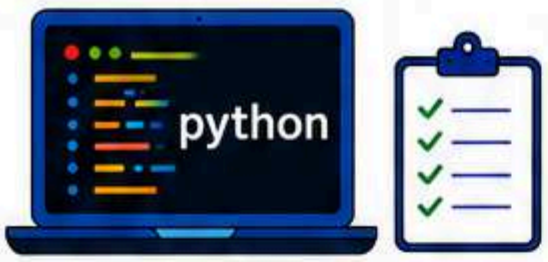
المساحة = 40

ملاحظات مهمة

- اسم الصنف يبدأ بحرف كبير (مثل: Student).
- تستخدم self للإشارة إلي الكائن الحالي.
- يمكن إنشاء عدة كائنات من نفس الصنف.
- كل كائن له بياناته الخاصة المستقلة عن الأخرين.
- البرمجة الكائنية تُستخدم في معظم البرامج الحديثة.

نصائح ذهبية

- تعلم OOP هو خطوة كبيرة نحو الاحتراف.
- اختبر برنامجك خطوة بخطوة.
- استخدم أسماء واضحة للكائنات والطرق.
- إجعل كل كائن يقوم بمهمة محددة.
- إبدأ بتصميم الصنف بشكل جيد.



ملخص Python الشامل

(Cheat Sheet)

1. أنواع البيانات الأساسية

النوع	مثال
int (عدد صحيح)	10 , -5 , 0
float (عدد عشري)	3.14 , -2.5
str (نص)	"Hello"
bool (منطقي)	True , False
list (قائمة)	[1, 2, 3]
tuple (مجموعة ثابتة)	(1, 2, 3)
set (مجموعة)	{1, 2, 3}
dict (قاموس)	{"name": "Ali"}

2. العمليات الحسابية

+	الجمع
-	الطرح
*	الضرب
/	القسمة
//	القسمة الصحيحة
%	باقي القسمة
**	الأس (قوة)

مثال:

```
a = 10
b = 3
print(a + b)    # 13
print(a // b)   # 3
print(a % b)    # 1
print(a ** b)   # 1000
```

3. المقارنات المنطقية

==	يساوي
!=	لا يساوي
>	أكبر من
<	أصغر من
>=	أكبر من أو يساوي
<=	أصغر من أو يساوي

مثال:

```
a = 5
b = 3
print(a > b)  # True
print(a == 5) # True
print(a != b) # True
```

4. القيم المنطقية

and	(و)	True إذا كان الشرطان True
or	(أو)	True إذا كان أحد الشرطين True
not	(ليس)	يعكس القيمة

مثال:

```
x = True
y = False
print(x and y)  # False
print(x or y)   # True
print(not x)    # False
```

5. البنى الشرطية

```
if شرط:
    # نفذ إذا كان الشرط True
elif شرط:
    # نفذ إذا كان الشرط السابق False
else:
    # نفذ إذا لم يتحقق أي شرط
```

مثال:

```
x = 10
if x > 0:
    print("موجب")
elif x == 0:
    print("صفر")
else:
    print("سالب")
```

6. الحلقات التكرارية

```
for حلقة:
 for i in range(5):
     print(i)    # 0 1 2 3 4
```

```
while حلقة:
 i = 0
 while i < 5:
     print(i)
     i += 1
```

أوامر التحكم في الحلقات:

break	خروج من الحلقة
continue	# تخطي التكرار الحالي
pass	(لا يفعل شيء مؤقتا)

7. أهم الدوال الجاهزة

print()	طباعة أي شيء
input()	إدخال من المستخدم
len()	طول (عدد عناصر)
type()	نوع البيانات
int()	تحويل إلى عدد صحيح
float()	تحويل إلى عدد عشري
str()	تحويل إلى نص
max()	أكبر قيمة
min()	أصغر قيمة
sum()	مجموع العناصر
sorted()	ترتيب العناصر

8. العمل مع السلاسل النصية

```
s = "Hello"
s[0]          # H
s[1]          # e
s[-1]         # o
s[1:4]        # "ell"
len(s)        # 5
s.upper()     # "HELLO"
s.lower()     # "hello"
s.strip()     # حذف الفراغات
s.replace(a, b)  استبدال
s.split()     # تقسيم النص
```

مثال:

```
name = "  Ali  "
print(name.strip())  # "Ali"
```

9. الفرق بين هياكل البيانات

النوع	قابل للتعديل؟	مكرر؟	مرتب؟	مثال
(قائمة)	نعم	نعم	نعم	[1, 2, 3]
Tuple (مجموعة ثابتة)	لا	نعم	نعم	(1, 2, 3)
Set (مجموعة)	نعم	لا	لا	{1, 2, 3}
Dictionary (قاموس)	نعم	لا	(مفاتيح)	{"a": 1}

10. القوائم (List)

```
L =[1, 2, 3, 4]
L.append(5)          # إضافة عنصر
L.insert(1, 10)      # إدراج عنصر في موقع
L.remove(3)          # حذف أول عنصر قيمته
L.pop()              # حذف أخر عنصر وإرجاعه
L.pop(1)             # حذف عنصر من موقع محدد
L.sort()             # ترتيب القائمة تصاعديا
L.reverse()          # عكس ترتيب القائمة
```

11. القواميس (Dictionary)

```
D = {"name": "Ali", "age": 18}
D["city" ] = "Cairo"     # إضافة عنصر
D["name"]                # الوصول إلى
D.keys()                 # المفاتيح
D.values()               # القيم
D.items()                # الأزواج (مفتاح:قيمة)
D.pop("age")             # حذف عنصر
```

12. المجموعات (Set)

```
S = {1, 2, 3}
S.add(4)              # إضافة عنصر
S.remove(2)           # حذف عنصر
S.discard(5)          # حذف عنصر
S.union({3,4})        # دمج مجموعتين
S.intersection({2,3,4}) # العناصر المشتركة
```

13. المجموعات الثابتة (Tuple)

```
T = (1, 2, 3)
T[0]          # الوصول إلى عنصر
len(T)        # طول المجموعة
T.count(2)    # عدد ظهور العنصر
T.index(3)    # موقع العنصر
```

لا يمكن تعديل العناصر بعد الإنشاء

اختصارات مفيدة

Ctrl + S	حفظ الملف
Ctrl + F	بحث في الملف
Ctrl + Z	تراجع
Ctrl + Y	إعادة

تذكّر دائما

- ✓ المسافة البادئة مهمة في Python.
- ✓ الكود حساس لحالة الأحرف (Ali # ali).
- ✓ كل شيء في Python هو كائن (Object).
- ✓ الدوال الجاهزة توفر عليك وقتاً وجهداً كبيرين.

نصائح مهمة

- استخدم أسماء واضحة للمتغيرات والدوال.
- اكتب تعليقات لشرح الكود: # تعليق
- اختبر الكود على أمثلة مختلفة.
- حل الكثير من الأسئلة والتمارين.

تدريبات شاملة على Python

اختبر نفسك وتأكد من فهمك لكل ما تعلمته في Python

السؤال الأول: أختر الإجابة الصحيحة

1 ما نوع البيانات الناتج عن: `type(3.14)` ؟

a) int **b) float**
c) str d) bool

2 ما الناتج: `print(2 ** 3)` ؟

a) 5 b) 6
c) 8 d) 9

3 أي معامل بعطي باقي القسمة؟

a) / b) //
c) % d) **

4 ما الناتج: `print("Hello" * 3)` ؟

a) HelloHelloHello
b) Hello * 3
c) Hello3
d) خطأ

5 ما الناتج: `print(10 > 5 and 3 < 2)`

a) True **b) False**
c) 10 d) 3

6 ما الناتج: `len([1, 2, 3, 4])` ؟

a) 3 **b) 4**
c) 5 d) خطأ

7 ما الذي يستخدم للوصول إلى العنصر الثاني في القائمة `L = ["a", "b", "c"]` ؟

a) L[1]
b) L[2] c) L(1)
d) L[3]

8 ما الناتج: `bool("")` ؟

a) True
b) False
c) خطأ
d) 0

9 ما الناتج: `print(type({1,2,3}))` ؟

a) <class 'list'>
b) <class 'tuple'>
c) <class 'set'>
d) <class 'dict'>

10 ما الناتج: `"abc"[1]` ؟

a) a
b) b
c) c
d) خطأ

السؤال الثاني: ما ناتج الكود التالي؟

1
```
x = 5
y = 2
print(x // y)
print(x % y)
```
الناتج:
2
1

2
```
s = "Python"
print(s[0])
print(s[-1])
print(s[2:5])
```
الناتج:
P
n
tho

3
```
L = [10, 20, 30]
L.append(40)
L.pop(1)
print(L)
```
الناتج:
[10, 30, 40]

4
```
d = {"a": 1, "b": 2}
d["c"] = 3
d.update({"a": 10})
print(d["a"])
```
الناتج:
10

5
```
x = True
y = False
print(not x or y
  and not y)
```
الناتج:
False

السؤال الثالث: مسائل برمجية

1 اكتب برنامجًا يطلب من المستخدم إدخال رقم ثم يطبع إن كان زوجيًا أو فرديًا.

مثال:
```
أدخل رقمًا: 7
النتيجة: الرقم فردي
أدخل رقمًا: 10
النتيجة: الرقم زوجي
```

2 اكتب برنامجًا لحساب مجموع عناصر قائمة أرقام يقوم المستخدم بإدخالها حتى يكتب 0 (الصفر) لإنهاء الإدخال.

مثال:
```
أدخل رقمًا: 5
أدخل رقمًا: 10
أدخل رقمًا: 15
أدخل رقمًا: 0
مجموع الأرقام = 30
```

3 اكتب برنامجًا يطلب من المستخدم إدخال كلمة ثم بطبعها معكوسة.

مثال:
```
أدخل كلمة: python
النتيجة: nohtyp
أدخل كلمة: hello
النتيجة: olleh
```

السؤال الرابع: تحدي إضافي

1 اكتب برنامجًا يحسب عدد تكرار كل حرف في كلمة يدخلها المستخدم.

مثال:
```
أدخل كلمة: banana
{'b': 1, 'a': 3, 'n': 2}
```

2 اكتب برنامجًا يطبع جميع الأعداد الأولية من 1 إلى حيث N يحدده المستخدم.

مثال:
```
أدخل العدد: 20
2 3 5 7 11 13 17 17 19
```

نصائح قبل الاختبار

- اقرأ السؤال جيداً قبل الإجابة.
- اختبر جميع الحالات الممكنة.
- تأكد من صحة بناء الجمل البرمجية.
- راجع أهم الدوال والعمليات.
- تمرن أكثر... فالتمرين يصنع المبدع! 💪

ما هي علوم الحاسب؟

علوم الحاسب (Computer Science) هي العلم الذي يهتم بدراسة الحاسبات، والخوارزميات، والبرمجيات، وكيفية حل المشكلات باستخدام التكنولوجيا.

مكونات نظام الحاسب

1. المكونات المادية (Hardware)

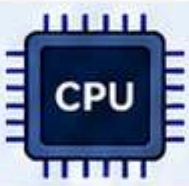
هي الأجزاء التي يمكن رؤيتها ولمسها مثل:

الفأرة (Mouse)

لوحة المفاتيح (Keyboard)

الشاشة (Monitor)

الطابعة (Printer)



وحدة المعالجة المركزية (CPU)

2. البرمجيات (Software)

هي البرامج التي تجعل الحاسب يؤدي المهام المختلفة، وتنقسم إلى:

أ) برمجيات النظام (System Software)

برامج تشغيل وإدارة الحاسب.
مثال: نظام التشغيل، Windows.

ب) البرمجيات التطبيقية (Application)

برامج يستخدمها المستخدم لإنجاز أعماله.

أمثلة:

- Microsoft Word
- PowerPoint
- Web Browser

دورة معالجة البيانات (IPO Cycle)

Input (إدخال)

إدخال البيانات.

Processing (معالجة)

معالجة البيانات.

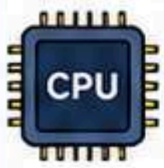
Output (إخراج)

إظهار النتائج.

مثال:
إدخال درجتين لطالب
↓
حساب المتوسط
↓
عرض النتيجة على الشاشة.

المكونات الأساسية داخل الحاسب

CPU
Central Processing Unit



وتسمى عقل الحاسب لأنها مسؤولة عن تنفيذ الأوامر وإجراء العمليات الحسابية والمنطقية.

Memory (الذاكرة)

تستخدم لتخزين البيانات والتعليمات.

ومن أشهر أنواعها:

RAM

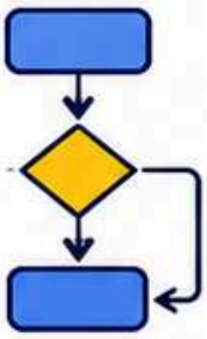
ذاكرة موقتة تفقد محتوياتها عند إيقاف تشغيل الجهاز.

ROM

ذاكرة دائمة تحتوي على تعليمات أساسية لتشغيل الحاسب.

تذكر

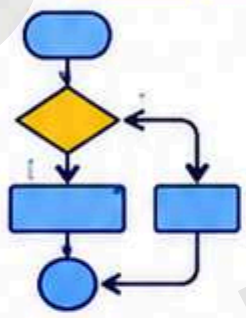
- Hardware + Software
- CPU هو عقل الحاسب
- دورة العمل الأساسية: Input → Processing →



ما هي الخوارزميات والمخططات الانسيابية؟

الخوارزمية هي مجموعة من الخطوات المرتبة والمنطقية لحل مشكلة أو لإنجاز مهمة معينة.
أما **المخطط الإنسيابي** فهو تمثيل رسومي لهذه الخطوات باستخدام رموز وأشكال قياسية مرتبطة بأسهم.

المخطط الانسيابي



- هو تمثيل رسومي للخوارزمية باستخدام رموز قياسية.
- يسهل فهم الحل وتنفيذه.
- يستخدم في تحليل وتصميم البرامج.

مثال:

المخطط الانسيابي لحساب متوسط درجتين:

الخوارزمية

- هي وصف كتابي للخطوات اللازمة لحل مشكلة ما.
- تتكون من خطوات مرتبة ومنطقية.
- كل خطوة تؤدي إلى التالية.

مثال:

خوارزمية لحساب متوسط درجتين:

1. أدخل الدرجة الأولى.
2. أدخل الدرجة الثانية.
3. احسب المتوسط.
4. اعرض النتيجة.

أهم رموز المخطط الانسيابي

الرمز	الاسم	الوظيفة
	بداية / نهاية	تحدد بداية أو نهاية المخطط.
	إدخال / إخراج	تستخدم لإدخال البيانات أو إخراج النتائج.
	عملية	تستخدم لإجراء عمليات حسابية أو تعليمات عامة.
	قرار	تستخدم لاتخاذ قرار (نعم / لا).
	خط انسيابي	يوضح اتجاه سير العمليات.

اختر الإجابة الصحيحة:

1. تمثيل رسومي للخوارزمية باستخدام رموز قياسية هو:
   - ○ الخوارزمية ○ البرمجة
   - ● المخطط الانسيابي ○ التجميع

2. الرمز المستخدم لعملية حسابية في المخطط الانسيابي هو:
   - ○ ◇ ○ ▱ ● ▭ ○ ⬭

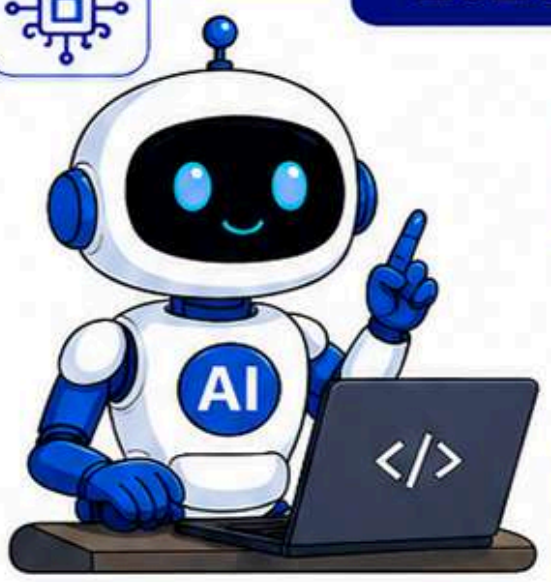
3. أي مما يلي يعتبر خوارزمية صحيحة؟
   - ○ شكل رسومي للبرنامج.
   - ● مجموعة خطوات مرتبة لحل مشكلة.
   - ○ لغة برمجة عالية المستوى.
   - ○ جهاز لإدخال البيانات.

الفرق بين الخوارزمية والمخطط الانسيابي

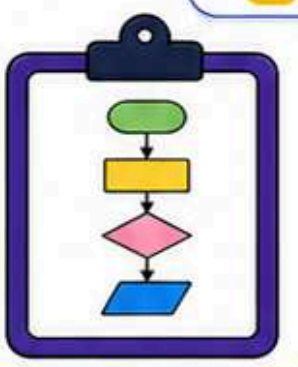
الخوارزمية		المخطط الانسيابي
• وصف كتابي للخطوات.	VS	• تمثيل رسومي بالخطوات.
• يستخدم اللغة الطبيعية.		• يستخدم رموزاً وأشكالاً.
• أسهل في الكتابة والتعديل.		• أسهل في الفهم.
• مناسب للمشكلات البسيطة.		• مناسب لتصميم البرامج المعقدة.

تذكر

- ✓ الخوارزمية = خطوات مرتبة ومحددة لحل مشكلة.
- ✓ المخطط الانسيابي = تمثيل رسومي للخوارزمية.
- ✓ المخطط الانسيابي يسهل فهم حل المشكلة وتنفيذه.



pseudocode و flowchart



pseudocode هو وصف خطوات الحل بلغة شبه برمجية سهلة.

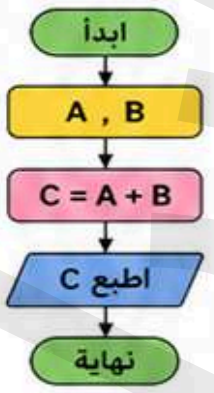
flowchart هو تمثيل بصري للخوارزمية باستخدام **الأشكال** والأسهم.

مراجعة سريعة: أهم ما تعلمناه

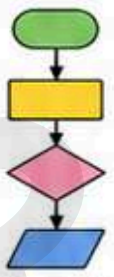
أهميتهم

- يساعدان على فهم المشكلة جيداً.
- يسهلان تصميم الحل.
- يقللان من الأخطاء قبل كتابة الكود.

flowchart



ما هو flowchart ؟



هو تمثيل بصري واضح للخوارزمية باستخدام **الأشكال** الهندسية والأسهم.

مثال pseudocode

```
ابدأ
اقرأ عددين A , B
اجمعهما في C
اطبع C
نهاية
```

pseudocode ؟

هو وصف خطوات الحل بلغة **شبه برمجية** سهلة ومفهومة قبل كتابة الكود الفعلي.

مقارنة بين pseudocode و flowchart

flowchart	وجه المقارنة	pseudocode
يستخدم الأشكال الهندسية والأسهم.	طريقة التمثيل	يستخدم اللغة الطبيعية وشبه البرمجية.
رسومي (مخطط).	الشكل	نصي (كتابة).
يحتاج إلى رسم وترتيب الأشكال.	الكتابة	أسهل وأسرع في الكتابة.
تعديله يحتاج إلى تغيير الرسم.	التعديل	مرن ويمكن تغييره بسهولة.
مناسب لعرض الحل بشكل واضح ومنظم.	الاستخدام	مناسب للتخطيط السريع للحل.

السؤال الأول: اختر الإجابة الصحيحة

1. pseudocode هو :
   رسم بياني | لغة برمجة | ✓ وصف خطوات الحل | أداة تصميم
2. يستخدم flowchart :
   الأرقام فقط | ✓ الأشكال والأسهم | الكلمات فقط | الصور
3. أي من التالية يمثل pseudocode ؟
   مستطيل وأسهم | أشكال هندسية | ✓ كتابة خطوات الحل | دوائر فقط
4. من فوائد استخدام pseudocode و flowchart :
   زيادة الأخطاء | ✓ تسهيل الفهم | تعقيد الحل | إضاعة الوقت
5. الرمز الذي يدل على قرار في flowchart هو :
   مستطيل | معين | معين | متوازي الأضلاع

السؤال الثاني: أكمل العبارات التالية

1. ..... هو وصف خطوات الحل بلغة شبه برمجية سهلة.
2. ..... هو تمثيل بصري للخوارزمية باستخدام الأشكال والأسهم.
3. في flowchart الشكل المستخدم للبداية والنهاية هو الشكل .....
4. يساعد كل من pseudocode و flowchart على .....
5. الرمز المستخدم لاتخاذ قرار في flowchart هو الشكل .....

السؤال الثالث: صل بين العمودين

(أ)	(ب)
1 pseudocode	وصف خطوات الحل
2 flowchart	تمثيل بصري بالأشكال
3 البداية/ النهاية	بيضاوي (Oval)
4 اتخاذ قرار	معين (Diamond)
5 معالجة البيانات	مستطيل (Rectangle)

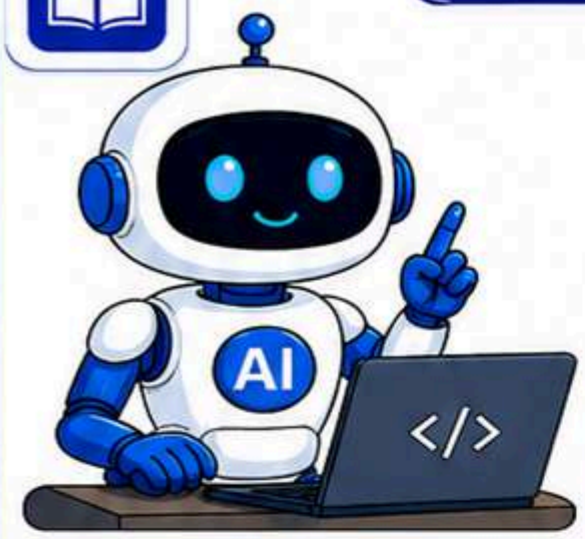
السؤال الرابع: فكر

- اذكر مميزتين لكل من pseudocode و flowchart.

- متى تفضل استخدام flowchart بدلاً من pseudocode؟ ولماذا؟

تذكّر: استخدم **pseudocode** لتخطيط خطوات الحل **كتابةً** قبل البرمجة، واستخدم **flowchart** لتمثيلها **بصريًا** وفهمها بسهولة.

تعلم بذكاء ... برمج مستقبلك.

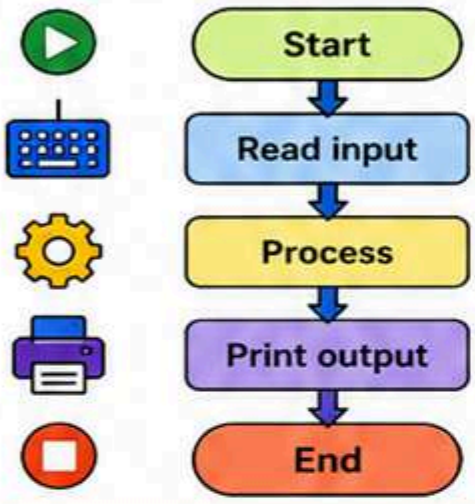


Pseudocode

الـ Pseudocode هو طريقة بسيطة لكتابة خطوات الحل في شكل كلمات وجمل واضحة يفهمها الإنسان قبل كتابة البرنامج الحقيقي.

مراجعة سريعة: أهم ما تعلمناه

خطوات الـ Pseudocode



المميزات

- سهل الفهم والكتابة.
- غير مرتبط بلغة برمجة معينة.
- يساعد في تخطيط الحل وتنظيم الأفكار.
- يُستخدم قبل رسم المخطط الانسيابي وكتابة الكود.

مثال بسيط

```
START
INPUT number
IF number > 0 THEN
    PRINT "Positive"
ENDIF
END
```

السؤال الأول: اختر الإجابة الصحيحة

1. ما هو الـ Pseudocode؟
   - لغة برمجة
   - ✓ طريقة لكطوات خطول بكلمات مفهومة
   - مخطط انسيابي
   - كود جاهز للتنفيذ
2. يُستخدم الـ Pseudocode قبل:
   - تشغيل البرنامج
   - ✓ كتابة الكود
   - ترجمة البرنامج
   - تصميم الواجهة
3. أي مما يلي من مميزات الـ Pseudocode؟
   - صعب الفهم
   - مرتب بالرسوم فقط
   - ✓ غير مرتبط بلغة برمجة
   - يعمل تلقائياً
4. خطوة قراءة البيانات في الـ Pseudocode تسمي:
   - End
   - ✓ Read input
   - Process
   - Print output
5. أحد رموز النهاية في الـ Pseudocode هو:
   - ✓ End
   - Start
   - Read
   - Process

السؤال الثاني: أكمل

1. الـ Pseudocode هو طريقة بسيطة لكتابة خطوات الحل في شكل ..... .
2. يُستخدم الـ Pseudocode قبل رسم المخطط ..... وكتابة ..... .
3. من خطوات الـ Pseudocode: ..... ثم Process . ثم ..... ثم End.
4. الـ Pseudocode يساعد في ..... وتنظيم ..... .
5. مثال على أمر إدخال البيانات في الـ Pseudocode هو ..... .

السؤال الثالث: صل بين العمودين

(أ)		(ب)
1 Start	•	• قراءة البيانات من المستخدم
2 Read input	•	• بداية تنفيذ البرنامج
3 Process	•	• معالجة البيانات
4 Print output	•	• عرض النتائج
5 End	•	• نهاية تنفيذ البرنامج

السؤال الرابع: ضع علامة (✓) أو (✗)

1. الـ Pseudocode هو نفس الكود البرمجي. ( )
2. يمكن فهم الـ Pseudocode من قبل الإنسان بسهولة. ( )
3. الـ Pseudocode لا يساعد في تخطيط الحل. ( )
4. يُستخدم الـ Pseudocode قبل كتابة البرنامج. ( )
5. الـ Pseudocode يعتمد على رسومات فقط. ( )

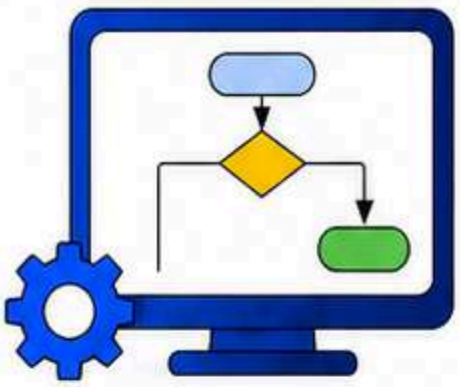
السؤال الخامس: فكر

لماذا من المهم كتابة الـ Pseudocode قبل كتابة الكود البرمجي؟ اذكر فائدتين.

.....

.....

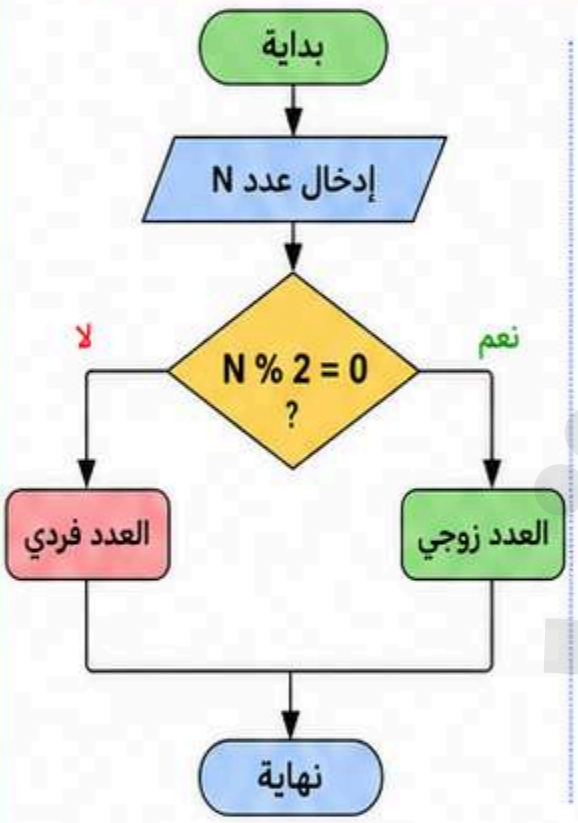
تذكّر: الـ Pseudocode يساعد على تنظيم الأفكار وتحديد خطوات الحل قبل كتابة الكود البرمجي.



تطبيقات عملية على المخططات الانسيابية

المخطط الانسيابي يُستخدم لتمثيل خطوات حل المشكلات بشكل منظم قبل تحويلها إلى برنامج. لنفهم أكثر، لنطبق معًا بعض الأمثلة البسيطة.

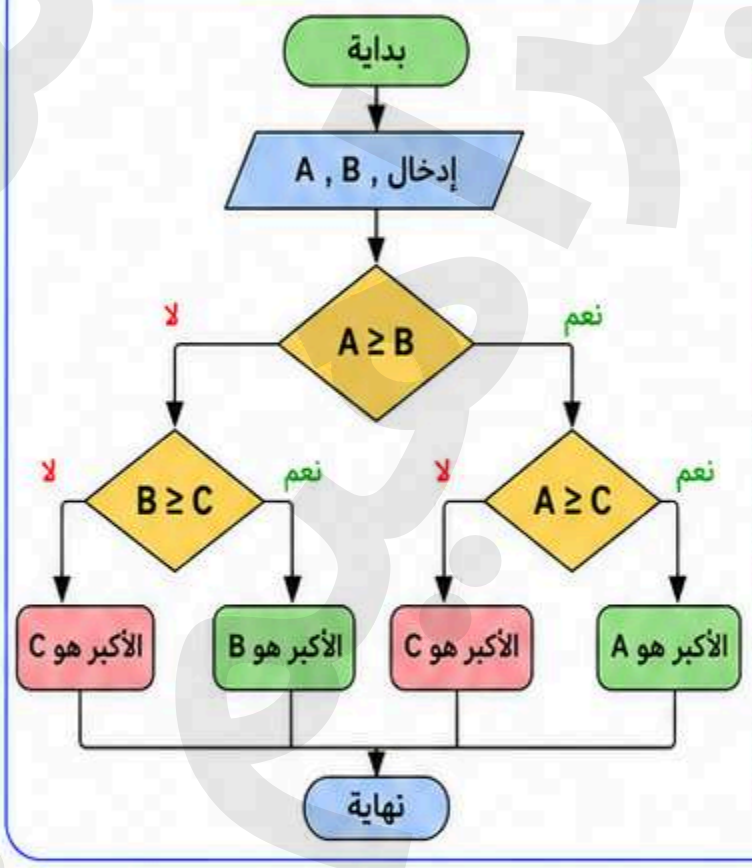
مثال 1: التحقق من عدد زوجي أو فردي



الخطوات:

1. إدخال عدد.
2. تحقق هل باقي قسمة العدد على 2 يساوي صفر أم لا.
3. إذا كان نعم فالعدد زوجي.
4. إذا كان لا فالعدد فردي.
5. نهاية البرنامج.

مثال 2: إيجاد أكبر عدد من ثلاثة أعداد



الخطوات:

1. إدخال ثلاثة أعداد.
2. نقارن A مع B.
3. ثم نقارن العدد الأكبر مع C.
4. نطبع أكبر عدد.
5. نهاية البرنامج.

أهم رموز المخطط الانسيابي

الرمز	الاسم	الوظيفة
	بداية / نهاية	تحديد نقطة البداية أو النهاية.
	إدخال / إخراج	إدخال البيانات أو عرض النتائج.
	عملية	تنفيذ عملية حسابية أو تعيين قيمة.
	قرار	اتخاذ قرار (نعم / لا).
	خط اتجاه	يوضح تسلسل خطوات البرنامج.

تصائح ذهبية عند رسم المخطط الانسيابي

- راجع المخطط لفهمه والتأكد من صحته.
- اجعل كل خطوة واضحة ومحددة بلا غموض.
- استخدم رموز المخطط الصحيحة لكل خطوة.
- اكتب الخطوات بترتيب منطقي ومسلسل.

تذكر

التخطيط الجيد قبل البرمجة يوفر الوقت والجهد ويقلل الأخطاء.

فكر جيدًا... ارسم خطواتك... ثم برمج بثقة!

النظام الثنائي والعشري والسادس عشر

تستخدم الحاسبات الأنظمة الرقمية لتمثيل البيانات. من أهم هذه الأنظمة: النظام الثنائي (2)، النظام العشري (10)، والنظام السادس عشر (16).

مقارنة بين الأنظمة

	النظام الثنائي (Base 2)	النظام العشري (Base 10)	النظام السادس عشر (Base 16)
التعريف	يستخدم رقمين فقط (0 و 1) وهو لغة الحاسوب الأساسية.	يستخدم عشرة أرقام (0) إلى (9) وهو النظام الذي نستخدمه يومياً.	يستخدم ستة عشر رقماً (0 إلى 9) والحروف(A إلى F) لتمثيل البيانات بشكل مختصر.
الأرقام المستخدمة:	0 , 1	0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9	0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , A , B , C , D , E , F
الأساس:	2	10	16
أمثلة:	0 , 1 , 10 , 101 , 1101 , 10101	0 , 7 , 25 , 125 , 1000	0 , A , 1F , 2B , 7F , FF

التحويل بين الأنظمة

من عشري إلى سداسي عشر

مثال: حول العدد 255 إلى سداسي عشر

القسمة على 16	الباقي
255 ÷ 16 = 15	15 (F)
15 ÷ 16 = 0	15 (F)

نقرأ البواقي من الأسفل إلى الأعلى

$255_{10} = FF_{16}$

من ثنائي إلى عشري

مثال: حول العدد 1101_2 إلى عشري

1	1	0	1
2^3	2^2	2^1	2^0

$= (1\times2^3) + (1\times2^2) + (0\times2^1) + (1\times2^0)$

$= 8 + 4 + 0 + 1$

$= 13_{10}$

من عشري إلى ثنائي

مثال: حول العدد 13 إلى ثنائي

القسمة على 2	الباقي
13 ÷ 2 = 6	1
6 ÷ 2 = 3	0
3 ÷ 2 = 1	1
1 ÷ 2 = 0	1

نقرأ البواقي من الأسفل إلى الأعلى

$13_{10} = 1101_2$

أمثلة سريعة

من سداسي عشر إلى ثنائي

مثال: حول العدد $3F_{16}$ إلى ثنائي

3	F
0011	1111

$3F_{16} = 0011\ 1111_2$

من ثنائي إلى سداسي عشر

مثال: حول العدد 1111_2 إلى سداسي عشر

1111 → F

$1111_2 = F_{16}$

من سداسي عشر إلى عشري

مثال: حول العدد $2A_{16}$ إلى عشري

2	A

$2\times16^1 + 10\times16^0$

$= 32 + 10 = 42_{10}$

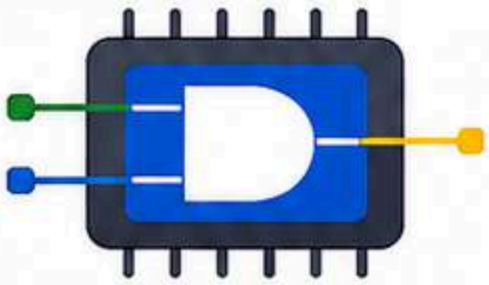
مفتاح سريع

- 0–9 = أرقام عادية في جميع الأنظمة.
- A = 10 , B = 11, C = 12 , D = 13 , E = 14 , F = 15.
- كل 4 بت = رقم سداسي عشر واحد.

تذكر

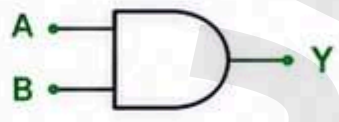
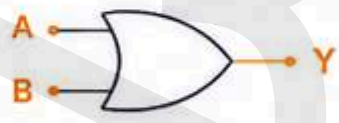
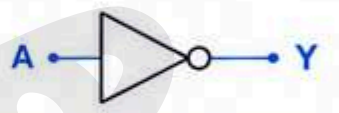
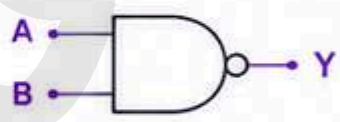
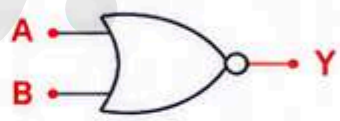
- ✓ النظام العشري سهل للبشر.
- ✓ الحاسب يفهم فقط 0 و 1.
- ✓ النظام السادس عشر مختصر ومفيد للمبرمجين.
- ✓ النظام الثنائي هو الأساس، والباقي صور مختلفة لتمثيل البيانات.

البوابات المنطقية والدوائر المنطقية



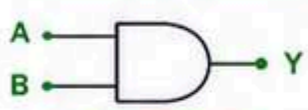
البوابات المنطقية هي اللبنات الأساسية للدوائر الرقمية، وتستخدم لتنفيذ العمليات المنطقية على القيم الثنائية (0 و 1)، وتُعد أساس تصميم جميع الأجهزة والحاسبات.

1. البوابات المنطقية الأساسية

البوابة	الرمز	المعادلة المنطقية	جدول الحقيقة	الوظيفة
AND (بوابة و)		$Y = A \cdot B$	A B Y 0 0 0 0 1 0 1 0 0 1 1 1	تعطي 1 فقط إذا كان كلا المدخلين 1.
OR (بوابة أو)		$Y = A + B$	A B Y 0 0 0 0 1 1 1 0 1 1 1 1	تعطي 1 إذا كان أحد المدخلين على الأقل 1.
NOT (بوابة نفي)		$Y = \overline{A}$	A Y 0 1 1 0	تعكس قيمة المدخل. (0 تصبح 1 , 1 تصبح 0)
NAND (بوابة و-نفي)		$Y = \overline{A \cdot B}$	A B Y 0 0 1 0 1 1 1 0 1 1 1 0	تعكس ناتج بوابة AND. تعطي 0 فقط إذا كان كلا المدخلين 1.
NOR (بوابة أو-نفي)		$Y = \overline{A + B}$	A B Y 0 0 1 0 1 0 1 0 0 1 1 0	تعكس ناتج بوابة OR. تعطي 1 فقط إذا كان كلا المدخلين .0.

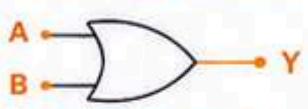
2. الدوائر المنطقية (أمثلة)

دائرة AND



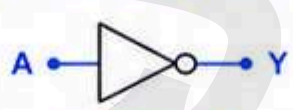
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

دائرة OR



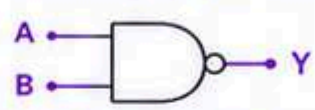
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

دائرة NOT



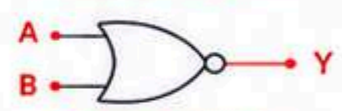
A	Y
0	1
1	0

دائرة NAND



A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

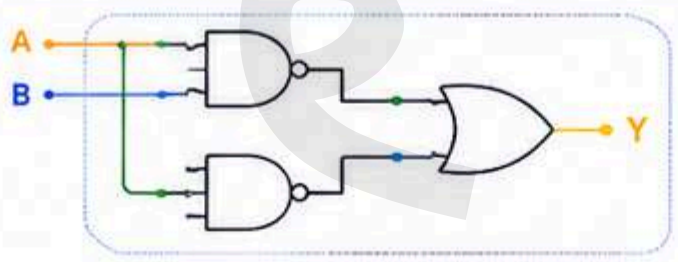
دائرة NOR



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

معلومة مهمة

يمكن بناء أي دارة منطقية باستخدام بوابة واحدة فقط هي NOR أو NAND وهذا ما يُعرف بـ Universal Gates



A	B	Y (XOR)
0	0	0
0	1	1
1	0	1
1	1	0

تركيب البوابات لبناء دوائر أكبر

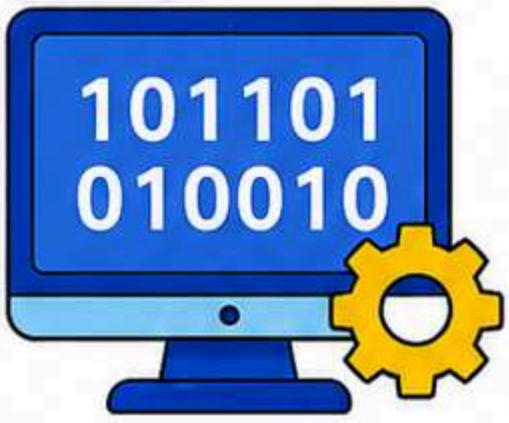
يمكن دمج البوابات المنطقية لبناء دوائر تنفذ عمليات معقدة.

مثال: دائرة XOR باستخدام البوابات الأساسية

Y ✔

- ✔ البوابات هي أساس بناء المعالجات والذاكرة والأجهزة الرقمية.
- ✔ فهم البوابات ضروري لتعلم الدوائر الرقمية والبرمجة المتقدمة.
- ✔ البوابات تعمل على القيم الثنائية (0) و (1).
- ✔ جدول الحقيقة يوضح جميع الحالات الممكنة للمدخلات.

كيف يفهم الحاسب البيانات؟



الحاسب **لا يفهم** الحروف أو الصور أو الأصوات مباشرة،
بل يحول كل شيء إلى أرقام ثنائية تتكون من:

0 و 1

وتسمى هذه اللغة **اللغة الثنائية** (Binary Language).

1. وحدة القياس الأساسية: Bit

Bit

هي أصغر وحدة بيانات داخل الحاسب.

يمكن أن تحمل قيمة واحدة فقط:

1 أو **0**

2 Byte

Byte

8 Bits = 1 Byte

مثال:

01000001

تمثل حرفاً واحداً داخل الحاسب.

3. وحدات قياس البيانات

الوحدة	القيمة
Byte	8 Bits
KB	1024 Bytes
MB	1024 KB
GB	1024 MB
TB	1024 GB

4. كيف يخزن الحاسب البيانات؟

الصوت

يتم تحويل الموجات الصوتية إلى أرقام يمكن للحاسب حفظها ومعالجتها.

الصور

تتكون من آلاف أو ملايين النقاط (Pixels).
كل نقطة لها لون يتم تخزينه كرقم.

النصوص

مثال:

A → 65

B → 66

يتم تحويل كل حرف إلى رقم.

5. مثال سريع

ملف حجمه:

500 KB أكبر أم. **5 MB** ؟

✅ الإجابة: **5 MB** أكبر.

6. تذكر

- ✅ **Bit** أصغر وحدة بيانات.
- ✅ **8 Bits = 1 Byte**
- ✅ الحاسب يخزن كل شيء في صورة أرقام.
- ✅ الصور والصوت والنصوص كلها تتحول إلى بيانات رقمية.

مكونات الحاسب الاساسية

يتكون الحاسب من أجزاء تعمل معاً مثل فريق واحد لكي ينفذ الأوامر ويخزن البيانات ويعرض النتائج.

1 وحدة المعالجة المركزية CPU

عقل الحاسب

- تنفذ الأوامر.
- تجري العمليات الحسابية.
- تتحكم في عمل الجهاز.

2 الذاكرة RAM

ذاكرة موقتة

- تخزن البيانات أثناء التشغيل.
- سريعة جدًا.
- تفقد محتوياتها عند إيقاف الجهاز.

مثال: فتح برنامج Word أو لعبة.

3 الذاكرة ROM

ذاكرة دائمة

- تحتوي تعليمات بدء التشغيل.
- لا تفقد البيانات عند إيقاف الجهاز.

4 وحدة التخزين Storage

مكان حفظ الملفات

أمثلة:

- HDD
- SSD
- Flash Memory

5 اللوحة الأم Motherboard

تربط جميع أجزاء الحاسب معاً.

تسمح بتبادل البيانات بين:

CPU ↔ RAM ↔ Storage

RAM ↕ Motherboard

Storage ↔ Motherboard

مقارنة سريعة

الجزء	الوظيفة
CPU	المعالجة
RAM	تخزين مؤقت
ROM	تعليمات التشغيل
SSD / HDD	تخزين دائم للملفات
Motherboard	الربط بين المكونات

معلومة مهمة

اذا كان

CPU = عقل الإنسان

RAM = الذاكرة قصيرة المدى

Storage = المكتبة التي تحفظ المعلومات

تذكر

- CPU هو أهم جزء في الحاسب.
- RAM أسرع من وحدات التخزين.
- SSD أسرع من HDD.
- اللوحة الأم تربط كل المكونات معاً.

كيف تتصل الأجهزة ببعضها؟

ما هي الشبكة (Network) ؟

شبكة الحاسب هي مجموعة من الأجهزة المتصلة ببعضها لتبادل البيانات والموارد.

أمثلة:

 أجهزة معمل المدرسة

 أجهزة المنزل

 الإنترنت

أجهزة الشيكات

1 الراوتر Router

- يوزع الإنترنت على الأجهزة.
- يربط الشبكات ببعضها.

2 السويتش Switch

- يربط عدة أجهزة داخل نفس الشبكة.

3 Access Point

نقطة وصول لاسلكية

- توفر اتصال Wi-Fi.

أنواع الشيكات

WAN

Wide Area Network

شبكة كبيرة تربط مدناً ودولاً مختلفة.

مثال:

الإنترنت.

LAN

Local Area Network

شبكة صغيرة داخل:

- منزل
- مدرسة
- شركة

الإنترنت

أكبر شبكة في العالم.

تسمح لنا بـ:

- تصفح المواقع.
- إرسال الرسائل.
- مشاهدة الفيديوهات.
- التعلم عن بعد.

عنوان IP

IP Address

هو العنوان الذي يميز كل جهاز على الشبكة.

مثال:

192.168.1.1

مثل رقم المنزل لكنه خاص بالحاسب.

كيف يتم الاتصال؟

تذكر

- LAN: شبكة صغيرة.
- WAN: شبكة كبيرة.
- Router: يوزع الإنترنت.
- كل جهاز يمتلك IP Address.
- الإنترنت: أكبر شبكة في العالم.

كيف نحمي بياناتنا؟

ما هو الأمن السيبراني؟

الأمن السبيراني هو حماية:


الأجهزة


الحسابات

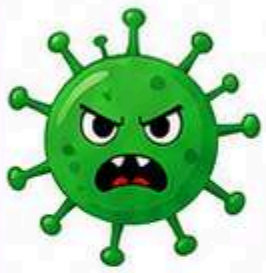

البيانات


الشبكات

من **الاختراق** أو **السرقة** أو **التلف**.

أهم التهديدات الإلكترونية

1 الفيروسات (Viruses)



برامج ضارة تدخل إلى الجهاز وتسبب مشكلات.

قد:

- تحذف الملفات.
- تبطئ الجهاز.
- تسرق البيانات.

2 التصيد الاحتيالي (Phishing)

رسائل أو مواقع مزيفة تحاول سرقة:

- كلمات المرور
- الحسابات
- البيانات الشخصية

3 كلمات المرور الضعيفة

- ~~123456~~
- ~~password~~
- ~~111111~~

تعتبر **سهلة** الاختراق.

نصائح الحماية


احتفظ بنسخة احتياطية من ملفاتك.


استخدم برنامج مكافحة الفيروسات.


حدث برامجك باستمرار.


لا تفتح روابط مجهولة.


لا تشارك كلمة المرور.

كيف تتشئ كلمة مرور قوية؟

استخدم:

#@!	123	A
رموز خاصة	أرقام	حروف كبيرة وصغيرة

Aa#2026Programming

تذكر

- الأمن السيبراني يحمي البيانات.
- لا تثق في أي رسالة مجهولة.
- كلمة المرور القوية هي خط الدفاع الأول.
- التحديثات تساعد في سد الثغرات الأمنية.

مثال من الحياة اليومية

فإن الحاسب يحتاج إلى:


تشفير


حماية


كلمة مرور

إذا كان منزلك يحتاج إلى:


كاميرات


مفتاح


باب قوي

تعلم بذكاء... برمج مستقبلك.

ماذا يمكن أن تصبح في المستقبل؟

علوم الحاسب ليست مجرد برمجة، بل تضم العديد من المجالات المهمة التي تبني عالم التكنولوجيا الحديث.

1 مبرمج Software Developer

يقوم بإنشاء:
- البرامج
- التطبيقات
- المواقع الإلكترونية:

WhatsApp Instagram Google Chrome

2 مطور ألعاب Games Developer

يصمم ويطور الالعاب.

أمثلة:

FIFA MINECRAFT PUBG

3 متخصص ذكاء اصطناعي AI Engineer

يطور أنظمة تستطيع:
- التعلم
- التحليل
- اتخاذ القرارات

ChatGPT السيارات ذاتية القيادة

4 خبير أمن سيبراني Cybersecurity

يحمي:
- الشبكات
- الأجهزة
- البيانات

من الاختراق والهجمات الإلكترونية.

5 عالم بيانات Data Scientist

يحلل البيانات الضخمة لاستخراج المعلومات المهمة واتخاذ القرارات.

6 مطور تطبيقات الهواتف Mobile Developer

يصمم تطبيقات:
- Android
- iPhone

تذكر

- علوم الحاسب تضم مجالات كثيرة.
- البرمجة هي الأساس لمعظم هذه المجالات.
- يمكنك اختيار المجال الذي يناسب اهتماماتك.
- التعلم المستمر هو سر النجاح في التكنولوجيا.

مهارات مشتركة

العمل الجماعي | التعلم المستمر | الإبداع | حل المشكلات | التفكير المنطقي

تعلم بذكاء... برمج مستقبلك.

ما هي تكنولوجيا المعلومات والاتصالات (ICT) ؟

تكنولوجيا المعلومات والاتصالات **(ICT)** هي استخدام أجهزة الحاسب والبرمجيات وشبكات الاتصال لإدارة المعلومات وتبادلها وتخزينها ونقلها.

تشمل ICT:

الشبكات

ربط الأجهزة معاً لمشاركة المعلومات والموارد.

الاتصالات

وسائل لنقل المعلومات بين الأجهزة.

البرمجيات

برامج تساعدنا على تَشغيل الأجهزة.

الحاسبات

أجهزة تُستخدم لتخزين ومعالجة المعلومات.

المعلومات

البيانات التي نحتاجها ونستخدمها.

أهمية ICT في حياتنا

تطوير الابتكار

تواصل سريع

رعاية صحية عن بعد

خدمات بنكية

تسوق إلكتروني

تعليم أفضل

تذكر

- ICT تساعدنا على الوصول إلى المعلومات بسهولة.
- تجعل التواصل أسرع وأفضل.
- تدعم التعليم والعمل والخدمات اليومية.
- تعتبر أساس التطور التكنولوجي في العالم.
- كل يوم نستخدم ICT في حياتنا دون أن نشعر.

أمثلة على استخدام ICT

- تصفح الإتترنت للبحث عن المعلومات.
- إرسال واستقبال الرسائل الإلكترونية.
- المشاركة في الاجتماعات عبر الإتترنت.
- حفظ الملفات على السحابة ومشاركتها.
- التعلم عن بُعد من أي مكَان.

كيف تنتقل المعلومات؟

ما هو نظام الاتصالات؟

نظام الاتصالات هو مجموعة عناصر تعمل معاً لإرسال المعلومات من شخص إلى آخر أو من جهاز إلى جهاز.

المكونات الأساسية

1 المرسل

الشخص أو الجهاز الذي يرسل الرسالة.

مثال:
هاتفك المحمول.

2 الرسالة

المعلومات التي يتم إرسالها.

أمثلة:

3 وسيلة الاتصال

الطريق الذي تسلكه الرسالة.

أمثلة:
- Wi-Fi
- الإنترنت
- كابل
- شبكة الهاتف

4 المستقبل

الشخص أو الجهاز الذي يستقبل الرسالة.

مثال من حياتنا اليومية

محمد يرسل صورة إلى أحمد عبر WhatsApp

لماذا تعد الاتصالات مهمة؟

- التواصل بسرعة.
- مشاركة المعلومات.
- التعليم عن بعد.
- التجارة الإلكترونية.
- الخدمات الحكومية الرقمية.

اتصالات لاسلكية

Wireless

- Wi-Fi
- شبكات المحمول
- الأقمار الصناعية

اتصالات سلكية

Wired

- كابلات الإنترنت
- الألياف الضوئية

تذكر

كل عملية اتصال تحتوي على:

يمكن أن تكون الاتصالات سلكية أو لاسلكية.

الإنترنت من أهم وسائل الاتصال الحديثة.

أين تُحفظ ملفاتك على الإنترنت؟

ما هي الحوسبة السحابية؟

Cloud Computing

هي تخزين الملفات والبيانات على خوادم عبر الإنترنت بدلاً من حفظها على جهازك فقط.

ما هي الخدمات الرقمية؟

خدمات تقدم عبر الإنترنت لتسهيل حياتنا اليومية.

- الخدمات البنكية
- التعليم الإلكتروني
- التسوق الإلكتروني
- البريد الإلكتروني

كيف تعمل؟

يمكنك الوصول إلى نفس الملفات من أي جهاز.

استخدامات يومية

- حفظ الصور.
- تخزين الملفات الدراسية.
- مشاهدة الفيديوهات عبر الإنترنت.
- إرسال واستقبال البريد الإلكتروني.

مميزات الحوسبة السحابية

- الوصول للملفات من أي مكان.
- توفير مساحة على الجهاز.
- مشاركة الملفات بسهولة.
- النسخ الاحتياطي للبيانات.

أمثلة على الخدمات السحابية

- **Google Drive** حفظ الملفات.
- **OneDrive** مزامنة الملفات.
- **Dropbox** مشاركة الملفات.

تذكر

- الخدمات الرقمية جزء أساسي من حياتنا.
- الحوسبة السحابية تعتمد على الإنترنت.
- يمكن الوصول للملفات من أي جهاز متصل.
- النسخ الاحتياطي يحمي بياناتك.

كيف تكون مواطنًا رقميًا مسؤولاً ؟

ما هي المواطنة الرقمية؟

هي استخدام التكنولوجيا والإنترنت بطريقة:

مسؤولة أخلاقية آمنة

قواعد المواطن الرقمي

1 الاحترام

احترم الآخرين عند التواصل عبر الإنترنت.

2 الخصوصية

لا تشارك:
- كلمات المرور
- البيانات الشخصية
- الصور الخاصة

مع أشخاص غير موثوقين.

3 التحقق من المعلومات

لا تصدِّق كل ما تراه على الإنترنت.

تحقق من المصدر أولًا.

4 الاستخدام المتوازن

خصص وقتاً مناسباً للتكنولوجيا.
ولا تجعلها تؤثر على:
- الدراسة
- النوم
- الصحة

بصمتك الرقمية

كل ما تنشره على الإنترنت يترك أثراً يسمى:

Digital Footprint

لذلك فكر جيدًا قبل النشر.

السلوك الصحيح والسلوك الخاطئ

سلوك خاطئ

- التنمر الإلكتروني.
- نشر الشائعات.
- مشاركة معلومات خاصة.

سلوك صحيح

- احترام الآخرين.
- استخدام كلمات مناسبة.
- حماية الحسابات.

مواطن رقمي ناجح

يستخدم التكنولوجيا بشكل إيجابي.

يتحقق من المعلومات.

يحمي بياناته.

يحترم الآخرين.

تذكر

الإنترنت أداة قوية إذا استخدمناها بشكل صحيح.

فكر قبل أن تنشر.

خصوصيتك مسؤوليتك.

كن مواطنًا رقميا إيجابيًا.

كيف نتواصل مع العالم ؟

ما هو الإنترنت؟

الإنترنت هو أكبر شبكة تربط ملايين الأجهزة حول العالم.

يسمح لنا بـ:

- التسوق
- التعلم
- مشاهدة الفيدوهات
- إرسال الرسائل

وسائل الاتصال الحديثة

شبكات التواصل الاجتماعي Social Media

مثل:

Facebook Instagram X

تطبيقات المراسلة Messaging Apps

مثل:

WhatsApp Telegram

مكالمات الفيديو Video Calls

مثل:

Zoom Google Meet

البريد الإلكتروني Email

إرسال واستقبال الرسائل الإلكترونية بسرعة.

كيف تتنقل البيانات؟

استخدام الإتترنت بأمان

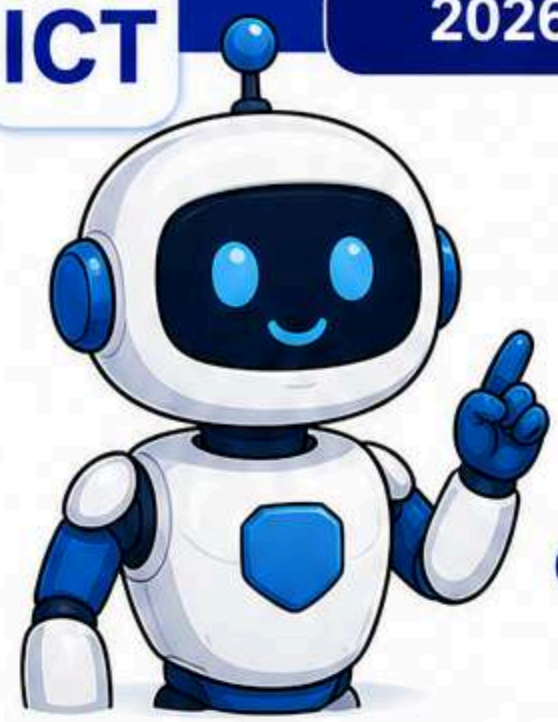
- استخدم كلمات مرور قوية.
- لا تضغط على الروابط المشبوهة.
- تحقق من المواقع قبل إدخال بياناتك.
- لا تشارك كلمات المرور .

فوائد الإتترنت

- التجارة الإلكترونية.
- الخدمات الحكومية.
- التعليم عن بعد.
- التواصل السريع.
- الوصول إلى المعلومات.

تذكر

- الاستخدام الآمن يحمي بياناتك. ✓
- الإنترنت يساعد في التعليم والعمل والتواصل. ✓
- تطبيقات التواصل تعتمد على الإنترنت. ✓
- الإنترنت أكبر شبكة في العالم. ✓



كيف تشعر الأجهزة بما حولها؟

ما هو السنسور؟

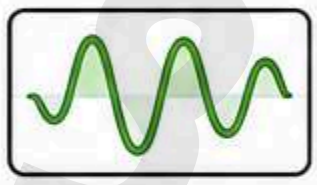
السنسور (Sensor) هو جهاز يكتشف التغيرات في البيئة المحيطة ويحولها إلى بيانات يمكن للحاسب أو المتحكم فهمها.

كيف يعمل السنسور؟

1 يكتشف التغير

حركة مساقة ضوء حرارة

2 يحولها إلى بيانات



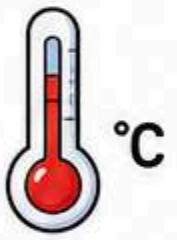
السنسور يحول التغير إلى إشارات رقمية.

3 يرسلها للحاسب

أو المتحكم (Controller) ليتخذ قراراً مناسباً.

أمثلة على السنسورات

1 Sensor الحرارة



يقيس درجة الحرارة.

2 Sensor الضوء

يقيس شدة الإضاءة.

3 Motion Sensor

يكتشف الحركة.

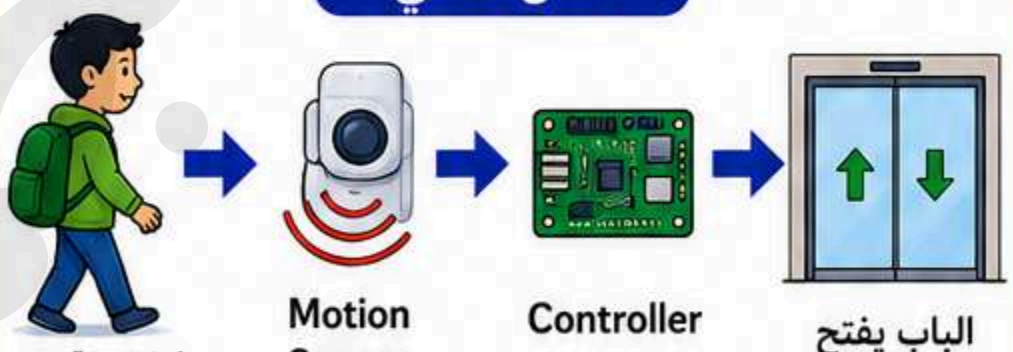
4 Ultrasonic Sensor

يقيس المسافة.

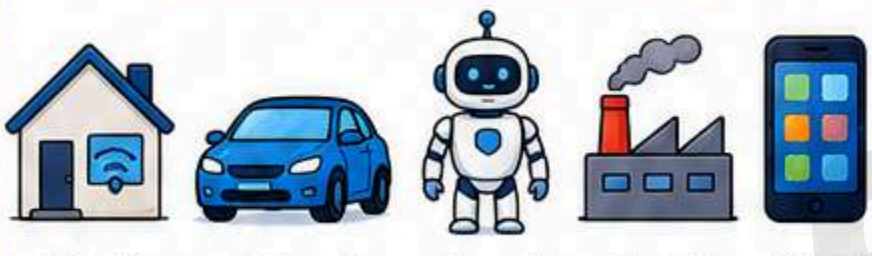
5 Water Sensor

يكتشف وجود الماء.

مثال عملي



أين نستخدم السنسورات؟



تذكر

- تستخدم في الروبوتات والذكاء الاصطناعي وإنترنت الأشياء.
- السنسورات هي "حواس" الأجهزة الذكية.
- يحول البيانات إلى إشارات يفهمها الحاسب.
- السنسور يجمع البيانات من البيئة.

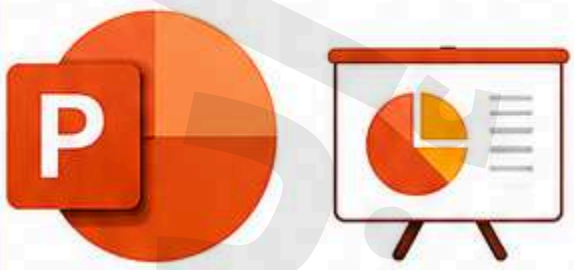
أشهر برامج الأوفيس

Microsoft Office ؟

مجموعة برامج تساعدنا على:

- كتابة المستندات
- إجراء الحسابات
- إنشاء العروض التقديمية

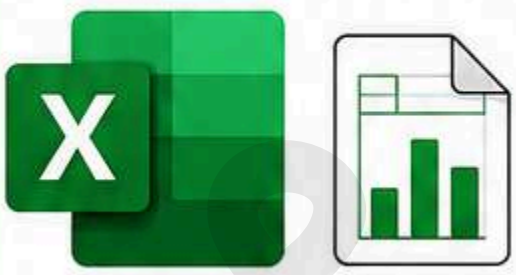
Microsoft PowerPoint



يستخدم لإنشاء:

- العروض التقديمية
- الشروحات
- المشاريع

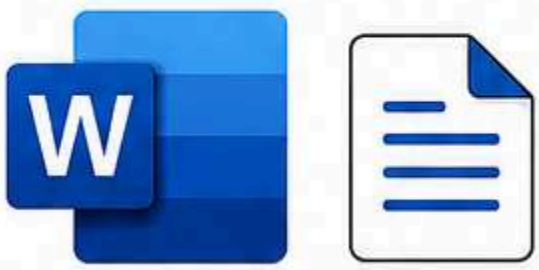
Microsoft Excel



يستخدم لـ:

- الجداول
- الحسابات
- الرسوم البيانية

Microsoft Word



يستخدم لكتابة:

- الأبحاث
- التقارير
- المستندات

OneNote

لتدوين وتنظيم الملاحظات.

Outlook

لإرسال واستقبال البريد الإلكتروني.

مقارنة سريعة

البرنامج	الاستخدام
Word	كتابة المستندات
Excel	الجداول والحسابات
PowerPoint	العروض التقديمية
Outlook	البريد الإلكتروني
OneNote	الملاحظات

تذكر

- Word للكتابة.
- Excel للحسابات.
- PowerPoint للعروض.
- Outlook للبريد الإلكتروني.

اختبر معلوماتك

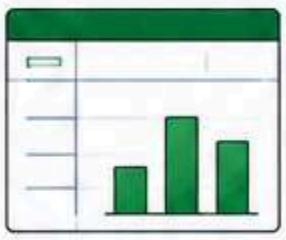
تدريبيات على برامج Microsoft Office

1 اختر البرنامج المناسب

1 كتابة بحث مدرسي

- ☐ Excel
- ☐ PowerPoint
- ☑ Word

2 إنشاء جدول درجات الطلاب



- ☐ Outlook
- ☑ Excel
- ☐ OneNote

3 تصميم عرض تقديمي لمشروع

- ☐ Word
- ☑ PowerPoint
- ☐ Outlook

4 إرسال رسالة إلكترونية

- ☐ Excel
- ☑ Outlook
- ☐ Word

5 تدوين الملاحظات الدراسية

- ☑ OneNote
- ☐ PowerPoint
- ☐ Excel

2 صل بين البرنامج ووظيفته

Word	كتابة المستندات والأبحاث
Excel	الجداول والحسابات والرسوم البيانية
PowerPoint	العروض التقديمية والشرائح
Outlook	البريد الإلكتروني وإدارة المواعيد
OneNote	تنظيم وتدوين الملاحظات

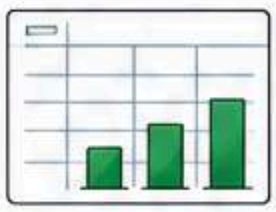
3 أكمل الجمل التالية

1. برنامج _____ يستخدم لإنشاء العروض التقديمية.
2. برنامج _____ يستخدم للحسابات والجداول.
3. برنامج _____ يستخدم لإرسال البريد الإلكتروني.
4. برنامج _____ يستخدم لكتابة المستندات.
5. برنامج _____ يستخدم لتنظيم الملاحظات.

4 سؤال تفكير

إذا طلب منك المعلم:

رسم جدول درجات



فأي برنامج ستستخدم؟

_____

5 تحدي سريع

ضع علامة ✓ أمام البرنامج المناسب:

المهمة	Word	Excel	PowerPoint
1 كتابة تقرير	☐	☐	☐
2 إنشاء رسم بياني	☐	☐	☐
3 عرض تقديمي	☐	☐	☐

دائماً داذكر

- **Word** للكتابة.
- **Excel** للحسابات والجداول.
- **PowerPoint** للعروض.
- **Outlook** للبريد الإلكتروني.
- **OneNote** للملاحظات.

اختبر مهاراتك الرقمية

1 يريد أحمد كتابة تقرير عن الطاقة المتجددة.
ما البرنامج الأنسب؟

- ☐ Excel
- ☐ PowerPoint
- ☑ Word
- ☐ Outlook

2 تريد المدرسة إعداد عرض تقديمي لليوم العلمي.
ما البرنامج المناسب؟

- ☐ Word
- ☐ Excel
- ☑ PowerPoint
- ☐ OneNote

3 يريد المعلم إنشاء جدول درجات الطلاب وحساب المتوسط.
ما البرنامج المناسب؟

- ☐ Outlook
- ☑ Excel
- ☐ Word
- ☐ PowerPoint

4 أي خدمة سحابية تساعدك على حفظ الملفات والوصول إليها من أي مكان؟

- ☐ Word
- ☐ PowerPoint
- ☑ Google Drive
- ☐ Outlook

5 أي من التالي يُستخدم لإرسال واستقبال البريد الإلكتروني؟

- ☐ Excel
- ☐ Word
- ☑ Outlook
- ☐ OneNote

6 ضع ✗ أو

	الإجابة	العبـارة
1	✓	الإنترنت أكبر شبكة في العالم.
2	✗	Excel يستخدم للعروض التقديمية.
3	✓	Outlook يستخدم للبريد الإلكتروني.
4	✓	الحوسبة السحابية تعتمد على الإنترنت.
5	✗	Word يستخدم لإنشاء الجداول فقط.

7 أكمل العيارات التالية

- ..... يستخدم للحسابات والرسوم البيانية.
- ..... يستخـدم لإنشـاء العروض التقديمية.
- ..... هـو أكبر شـبكة في العالم.
- ..... تسـاعد علـى تخزين الملفات عبر الإنترنت.

8 سؤال تفكير

إذا كنت تعمل على مشروع جماعي مع زملائك وتريد مشاركة الملفات بينهم باستمرار:
ما الخدمة التي ستستخدمها؟

.....

9 صل بين العنصر ووظيفته

Word	كتابة المستندات والتقارير.
Excel	الجداول والحسابات والرسوم البيانية.
PowerPoint	إنشاء العروض التقديمية.
Outlook	البريد الإلكتروني والتقويم والاتصالات.
Google Drive	تخزين الملفات ومشاركتها عبر الإنترنت.

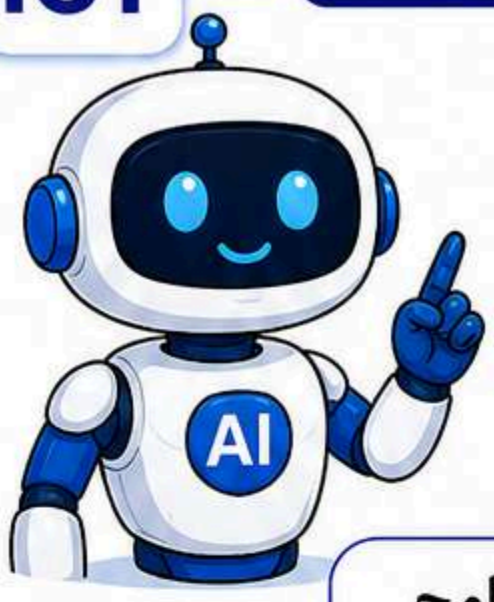
تحدي النجوم

رتب الخطوات التالية لإرسال بريد إلكتروني:

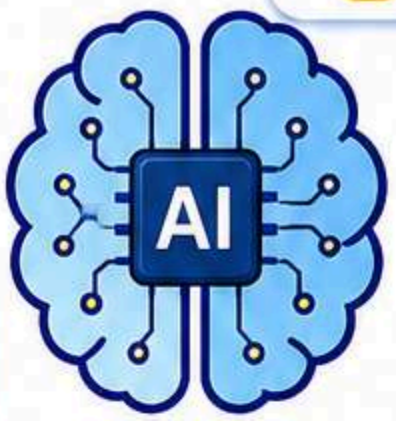
- ☐ كتابة الرسالة
- ☐ الضغط على إرسال
- ☐ كتابة عنوان البريد
- ☐ إرفاق الملف

تذكر

- مهارات ICT تُساعدك في الدراسة والعمل.
- التطبيق العملي أهم من الحفظ.
- كل برنامج له وظيفة محددة.
- استخدم الإنترنت بأمان ومسؤولية.
- استمر في التعلم لتصبح مبدعًا رقميًا.



ما هو الذكاء الاصطناعي؟



الذكاء الاصطناعي (AI) هو قدرة الأجهزة والبرامج على التفكير والتعلم واتخاذ القرارات مثل الإنسان.

أمثلة على الذكاء الاصطناعي في حياتنا اليومية

التوصيات الذكية

مثل اقتراح الفيديوهات والأغاني التي تحبها

السيارات الذكية

سيارات تقود نفسها وتتجنب الحوادث

التعرف على الوجه

فتح الهاتف بالبصمة أو الوجه

الترجمة الفورية

ترجمة النصوص والكلام تلقائيًا

المساعدات الذكية

مثل سيري وجوجل مساعد

فوائد الذكاء الاصطناعي

يوفر الوقت والجهد
ويقوم بالمهام بسرعة ودقة.

يحسن جودة الحياة
في الصحة، التعليم، النقل وغيرها.

يقلل الأخطاء البشرية
لأنه يعمل بدقة عالية.

يساعد في حل المشكلات المعقدة
التي يصعب على الإنسان حلها.

كيف يعمل الذكاء الاصطناعي؟

1 يجمع البيانات
من الصور، النصوص، الأصوات والتجارب السابقة.

2 يتعلم من البيانات
يكتشف الأنماط والعلاقات المفيدة.

3 يتخذ القرار
يستخدم ما تعلمه لحل مشكلة أو اتخاذ قرار.

معلومات سريعة

يجب استخدامه بمسؤولية وأمان واحترام خصوصيتك.

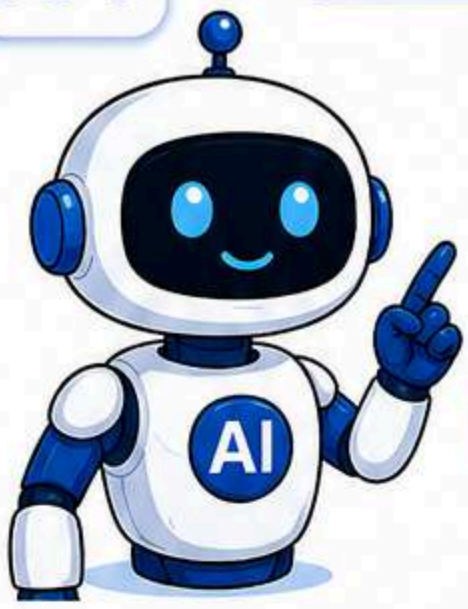
تعلّم الذكاء الاصطناعي مهارة مهمة لمستقبل العمل والدراسة.

الذكاء الاصطناعي موجود اليوم ويطور مستقبلنا كل يوم.

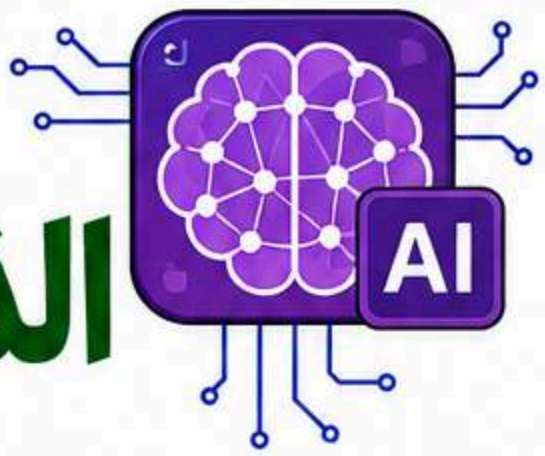
الذكاء الاصطناعي ليس بديلاً للإنسان، بل مساعد قوي له.

تذكر دائماً

الذكاء الاصطناعي مجال رائع ومتطور، وفهمه اليوم هو مفتاح المستقبل.



تطبيقات الذكاء الاصطناعي



يُستخدم الذكاء الاصطناعي في **الكثير** من **المجالات** لتسهيل حياتنا.

أهم تطبيقات الذكاء الاصطناعي في حياتنا اليومية

1 الطب والصحة

يساعد في تشخيص الأمراض، تحليل الأشعة، واكتشاف الأدوية.

2 التعليم

يوفر تعلمًا مخصصًا لكل طالب، ويساعد في تصحيح الاختبارات.

3 النقل والمواصلات

يُستخدم في السيارات ذاتية القيادة، وتحسين حركة المرور.

4 الترفيه والإعلام

يوصي بمقاطع الفيديو والأغاني التي تناسب اهتماماتك.

5 التسوق والتجارة

يوصي بالمنتجات المناسبة، ويتنبأ باحتياجات العملاء.

6 الخدمات الذكية

مثل المساعدات الذكية (سيري، أليكسا) للرد على أسئلتك.

فوائد تطبيقات الذكاء الاصطناعي

تحسين الجودة
يرفع مستوى الخدمات والمنتجات.

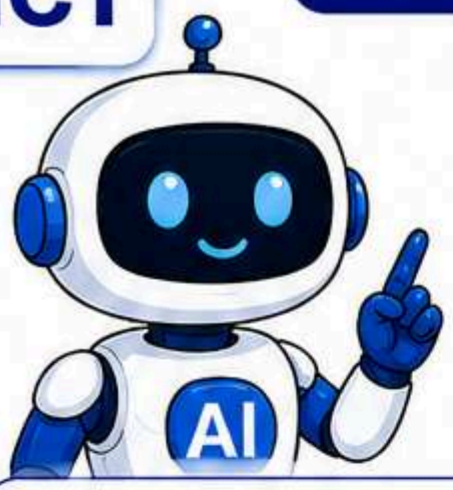
توفير الوقت
يوفر وقت الإنسان وجهده.

دقة عالية
يقلل الأخطاء ويزيد الدقة.

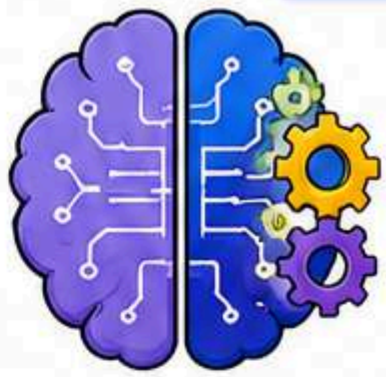
زيادة السرعة
ينجز المهام بسرعة كبيرة.

تـذكـر

الذكاء الاصطناعي موجود من حولنا في كل مكان، وهو مستمر في التطور ليجعل حياتنا أسهل وأفضل.



تعلم الآلة
(Machine Learning)



تعلم الألة هو فرع من الذكاء الاصطناعي يتيح للأنظمة **التعلم من البيانات** والتجارب لتحسين أدائها **واتخاذ قرارات دقيقة** دون برمجة صريحة.

كيف يعمل تعلم الألة؟

1 جمع البيانات

يتم جمع كميات كبيرة من البيانات.

2 تحليل البيانات

يتم تحليل البيانات واكتشاف الأنماط.

3 التعلم من البيانات

يتعلم النموذج من البيانات والأنماط.

4 التنبؤ أو اتخاذ القرار

يستخدم ما تعلمه للتنبؤ أو اتخاذ القرار.

5 تحسين الأداء

يتحسن النموذج كلما زادت البيانات والتجارب.

أمثلة على تعلم الألة

1. **التوصية بالمحتوى**
مثل اقتراح الفيديوهات أو الأفلام التي قد تهمك.

2. **التعرف على الصور**
مثل التعرف على الوجوه أو الأشياء في الصور.

3. **الكشف عن البريد المزعج**
يتعلم النظام تمييز الرسائل غير المرغوب فيها من الرسائل العادية.

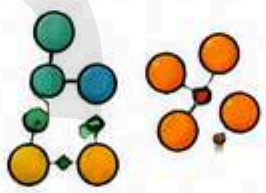
4. **المساعدات الذكية**
تفهم الأوامر الصوتية وتتعلم من تفاعلك معها.

5. **السيارات ذاتية القيادة**
تتعلم من البيئة المحيطة لأتخاذ أفضل القرارات أثناء القيادة.

أنواع تعلم الألة

1 التعلم المُشرف (Supervised Learning)
يتعلم النموذج من بيانات مُصنفة وبها إجابات صحيحة.

2 التعلم غير المُشرف (Unsupervised Learning)
يكتشف النموذج الأنماط والعلاقات من بيانات غير مُصنفة.



3 التعلم بالتعزيز (Reinforcement Learning)
يتعلم النموذج من خلال التجربة والخطأ ويحصل على مكافأة.

فوائد تعلم الألة

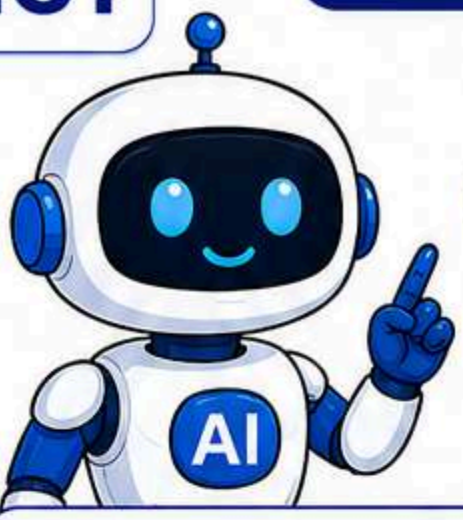
- يتعلم من البيانات ويتحسن باستمرار.
- يقلل من الأخطاء البشرية.
- يتخذ قرارات أسرع وأكثر دقة.
- يعمل على كميات كبيرة من البيانات.
- يُستخدم في مجالات متعددة وواسعة.

تذكر دائماً

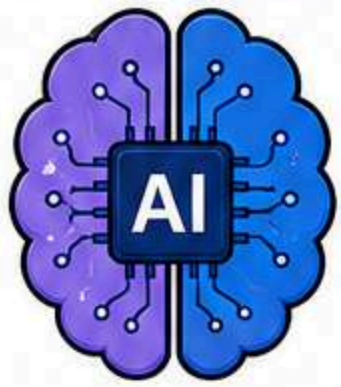
كلما زادت البيانات وجودتها كلما أصبح النموذج أكثر ذكاءً ودقةً في قراراته.

معلومات سريعة

- **هدفه الأساسي** تحسين الأداء واتخاذ القرار.
- يُستخدم في حياتنا اليومية.
- **يحتاج إلي** بيانات كثيرة للتعلم.
- **تعلم الألة** يجعل الألات أكثر ذكاء.
- يساعد في حل المشكلات المعقدة بكفاءة عالية.



الذكاء الاصطناعي في حياتنا اليومية



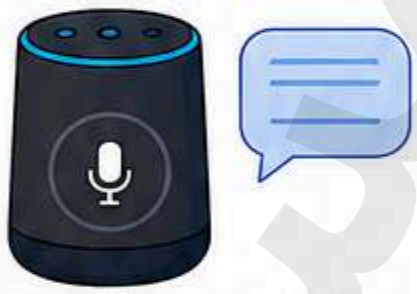
أصبح الذكاء الاصطناعي جزءًا مهمًا من حياتنا اليومية، ويساعدنا في إنجاز الكثير من الأمور بسهولة وسرعة.

أمثلة على استخدام الذكاء الاصطناعي في حياتنا اليومية

1 الهواتف الذكية

فتح الهاتف بالتعرف على الوجه.

2 المساعدات الذكية



مثل (سيري – أليكسا) تجيب على أسئلتك.

3 التوصيات الذكية

توصي بمقاطع فيديو وأغاني قد تعجبك.

4 التسوق الإلكتروني

يقترح منتجات تناسب اهتماماتك واحتياجاتك.

5 الخرائط والملاحة

تساعدك على اختيار أسرع طريق وتجنب الزحام.

6 الترجمة الفورية

ترجمة النصوص والمحادثات بين اللغات بسرعة.

7 البريد الإلكتروني

يكتشف الرسائل المزعجة ويحميك من الرسائل الضارة.

8 الطب والصحة

يساعد الأطباء في التشخيص السريع وتحليل الأشعة.

تحديات الذكاء الاصطناعي

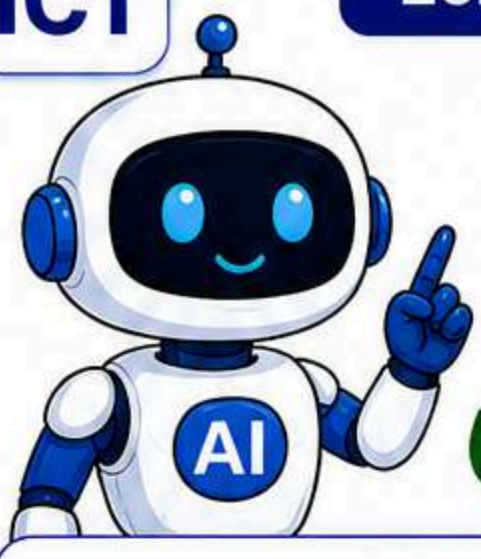
- الخصوصية وحماية البيانات.
- فقدان بعض فرص العمل.
- استخدامه بشكل غير أخلاقي.
- الاعتماد الزائد على التكنولوجيا.
- اتخاذ قرارات خاطئة أحيانًا.

فوائد الذكاء الاصطناعي

- يوفر الوقت والجهد.
- يعمل بدقة عالية.
- يحسن جودة الخدمات والمنتجات.
- يساعد الإنسان في حل المشكلات.
- يزيد من الأمان والسلامة.

تذكر دائمًا

الذكاء الاصطناعي أداة قوية، وعندما نستخدمه بشكل صحيح فإنه يساعدنا على بناء مستقبل أفضل.



أخلاقيات الذكاء الاصطناعي

أخلاقيات الذكاء الاصطناعي هي **القواعد والمبادئ** التي تضمن استخدام الذكاء الاصطناعي بطريقة **عادلة** و**آمنة** و**مسؤولة**.

أهم مبادئ أخلاقيات الذكاء الاصطناعي

1 العدالة والمساواة

يجب أن يتعامل الذكاء الاصطناعي مع جميع الأشخاص بعدالة دون تمييز.

2 الشفافية

يجب أن تكون قرارات الذكاء الاصطناعي واضحة ومفهومة.

3 الخصوصية

يجب حماية بيانات الأشخاص وعدم استخدامها بدون موافقتهم.

4 المسؤولية

يجب تحديد من هو المسؤول عن قرارات الذكاء الاصطناعي وتحمل النتائج.

5 السلامة والأمان

يجب أن تكون أنظمة الذكاء الاصطناعي آمنة ولا تسبب ضرراً للأشخاص.

تحديات أخلاقيات الذكاء الاصطناعي

عدم الشفافية

صعوبة فهم كيف يتخذ النموذج قراراته أحيانًا.

انتهاك الخصوصية

جمع البيانات بدون إذن قد يؤدي إلى انتهاك خصوصية الأفراد.

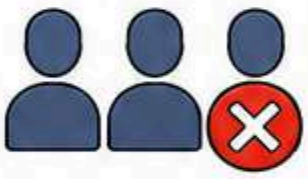
سوء الاستخدام

قد يتم استخدام الذكاء الاصطناعي في أعمال ضارة أو غير قانونية.

فقدان الوظائف

قد يؤدي الذكاء الاصطناعي إلى تقليل بعض فرص العمل.

التحيز



قد يحتوي النموذج على تحيز ضد فئة معينة.

ماذا يجب علينا فعل ؟

الإبلاغ عن المشاكل

أبلغ إذا لاحظت أي استخدام غير آمن أو غير أخلاقي للذكاء الاصطناعي.

التعلم المستمر

تعلم المزيد عن الذكاء الاصطناعي وكيفية استخدامه بشكل صحيح.

احترام الأخرين

تأكد من أن قرارات الذكاء الاصطناعي لا تضر أو تظلم الأخرين.

حماية البيانات

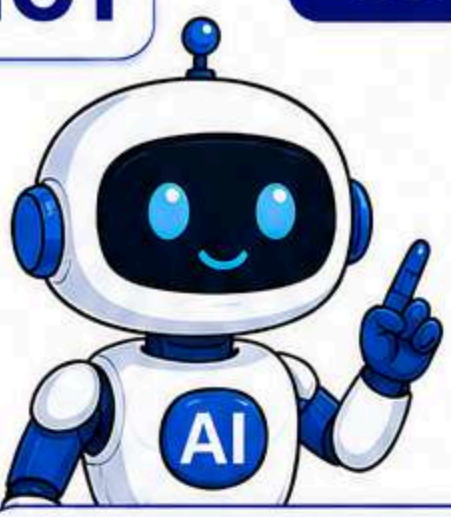
لا تشارك بياناتك الشخصية إلا عند الضرورة.

استخدام مسؤول

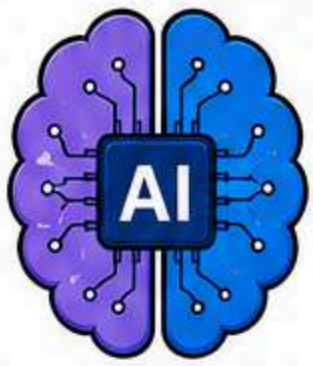
استخدم الذكاء الاصطناعي بطريقة مفيدة وآمنة.

تذكر دائمًا:

الذكاء الاصطناعي **أداة قوية**، واستخدامه **الأخلاقي** يضمن مستقبلاً **أفضل** للجميع.



تدريب ومراجعة على الذكاء الاصطناعي



اختبر فهمك لما تعلمته عن الذكاء الاصطناعي من خلال الأسئلة والتدريبات التالية.

السؤال الأول: اختر الإجابة الصحيحة

1 ما هو الذكاء الاصطناعي؟
- أ. جهاز ذكي
- ب. برنامج فقط
- ✓ ج. قدرة الأجهزة على التفكير والتعلم
- د. نوع من الألعاب

2 أي من هذه تطبيقات الذكاء الاصطناعي؟
- أ. الآلة الحاسبة
- ✓ ب. الترجمة الفورية
- ج. الورقة والقلم
- د. الساعة العادية

3 ما الذي يساعد الذكاء الاصطناعي على التعلم؟
- أ. حفظ الصور
- ✓ ب. البيانات والخبرات
- ج. الكهرباء فقط
- د. اللعب فقط

4 أي من التالي يعتبر ذكاءً اصطناعيا؟
- ✓ أ. سيارة ذاتية القيادة
- ب. المروحة
- ج. التلفاز العادي
- د. الكرسي

5 ما الهدف الأساسي من الذكاء الاصطناعي؟
- أ. زيادة المال
- ✓ ب. تسهيل الحياة وحل المشكلات
- ج. إضاعة الوقت
- د. تعقيد الأمور

السؤال الثاني: ضع ✓ أو ✗

	العبارة	✓	✗
1	الذكاء الاصطناعي يمكنه التعلم من البيانات.	✓	
2	الروبوتات لا تستخدم الذكاء الاصطناعي.		✗
3	الترجمة الفورية من تطبيقات الذكاء الاصطناعي.	✓	
4	الذكاء الاصطناعي لا يتخذ قرارات.		✗
5	السيارات ذاتية القيادة تعتمد على الذكاء الاصطناعي.	✓	

السؤال الثالث: صل بين العمودين

العمود (أ)	العمود (ب)
1 المساعدات الذكية	تحليل الصور والفيديو
2 التعلم الآلي	أجهزة تساعدك في المهام
3 الروبوتات	تعلم من البيانات دون برمجة مباشرة
4 الرؤية الحاسوبية	آلات تقوم بمهام بدلاً من الإنسان
5 الذكاء الاصطناعي	قدرة الأجهزة على التفكير واتخاذ القرارات

السؤال الرابع: أجب بإيجاز

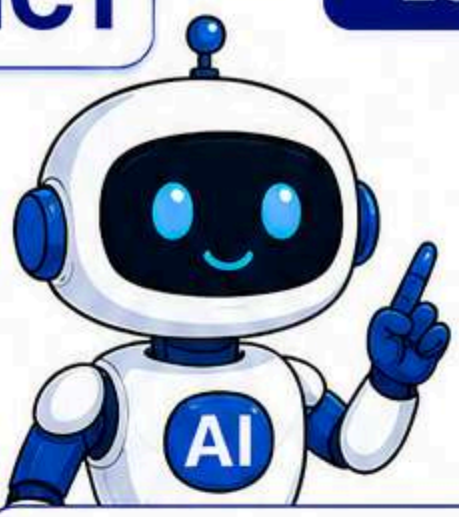
1. اذكر مثالين من تطبيقات الذكاء الاصطناعي في حياتنا اليومية.
.....
2. كيف يساعد الذكاء الاصطناعي في المجال الطبي؟
.....
3. ما أهمية البيانات في الذكاء الاصطناعي؟
.....

السؤال الخامس: فكر وأجب

إذا كان لديك فكرة لاستخدام الذكاء الاصطناعي لحل مشكلة في مدرستك أو مجتمعك، اكتب فكرتك باختصار.
.....
.....

تذكر دائمًا.

- مستقبلنا أفضل مع استخدام الذكاء الاصطناعي بشكل صحيح.
- الذكاء الاصطناعي هدفه تسهيل الحياة وحل المشكلات.
- كلما زادت بيانات جيدة كلما كان الذكاء الاصطناعي أذكى.
- استخدم الذكاء الاصطناعي بطريقة أمنة ومسؤولة.



تدريب ومراجعة على الذكاء الاصطناعي

اختبر نفسك فيما تعلمته عن الذكاء الاصطناعي من خلال الأسئلة والتدريبات التالية.

مراجعة سريعة: أهم ما تعلمناه

أخلاقيات الذكاء الاصطناعي

الخصوصية، العدالة، السلامة، الشفافية، والمسؤولية من أهم المبادئ.

فوائد الذكاء الاصطناعي

يوفر الوقت والجهد، يحسن الدقة، يساعد في حل المشكلات، ويزيد الإنتاجية.

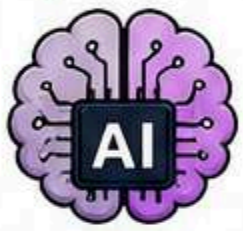
تطبيقات الذكاء الاصطناعي

يُستخدم في الطب، التعليم، النقل، التجارة، الترفيه، الأمان، وغيرها.

تعلم الآلة (Machine Learning)

فرع من الذكاء الاصطناعي يتعلّم من البيانات ويتطور بمرور الوقت دون برمجة مباشرة.

ما هو الذكاء الاصطناعي؟



تقنية تجعل الأجهزة تتفكر وتتعلّم وتُتخذ قرارات ذكية مثل الإنسان.

السؤال الأول: اختر الإجابة الصحيحة

1. الذكاء الاصطناعي يعني جعل الآلات ..... مثل الإنسان.
   تنسى | **تفكر** | تتوقف | تتعب
2. أي من التالي مثال على تطبيق من تطبيقات الذكاء الاصطناعي؟
   المسطرة الذكية | **سيارة ذاتية القيادة** | الآلة الحاسبة | المروحة
3. فرع من الذكاء الاصطناعي يتعلم من البيانات يسمي:
   البرمجة العادية | **تعلم الآلة** | الإنترنت الأشياء | شبكات الاتصال
4. أي من التالي ليس من فواند الذكاء الاصطناعي؟
   إضاعة الوقت | زيادة الدقة | حل المشكلات | تقليل التكاليف
5. مبدأ من مبادئ أخلاقيات الذكاء الاصطناعي يتعلق بحماية بيانات الأشخاص:
   العدالة | **الخصوصية** | السرعة | الترفيه

السؤال الثاني: أكمل العبارات التالية

1. ..... هو فرع من الذكاء الاصطناعي يتعلم من البيانات ويتطور ذاتياً.
2. تُستخدم تقنيات الذكاء الاصطناعي في مجال ..... و ..... و .....
3. من المبادئ الأخلاقية للذكاء الاصطناعي: ..... و .....
4. تساعد السيارات ذاتية القيادة في تحسين ..... و تقليل .....
5. كلما زادت البيانات، كلما كان نموذج الذكاء الاصطناعي ..... ودقة.

السؤال الثالث: صل بين العمودين

(أ)	(ب)
1 روبوت الدردشة	تحسين تجربة التسوق
2 التعرف على الوجه	اكتشاف الأمراض
3 التوصيات الذكية	فهم اللغة والترجمة مباشرة
4 تحليل الصور الطبية	التعرف على الأشخاص
5 الترجمة الفورية	الإجابة على الأسئلة

السؤال الرابع: تفكير نقدي

اذكر تطبيقاً واحداً للذكاء الاصطناعي تستخدمه في حياتك اليومية.
.....

ما أهم فائدة تتوقع أن يحققها الذكاء الاصطناعي في المستقبل؟
.....

ما التحديات التي قد يسببها الذكاء الاصطناعي إذا أُسيء أستخدامه؟
.....

تذكّر: الذكاء الاصطناعي فرصة رائعة لنبني مستقبلاً أفضل، لكن يجب أن نستخدمه بطريقة ذكية وأخلاقية وآمنة.

لغات البَرمَجة عالية المستوَى

High Level Programming Languages

تعرف على لغات البرمجة عالية المستوى، ولماذا تعد الخيار الأفضل للمبرمجين لبناء التطبيقات والبرامج بسهولة وسرعة.

مراجعة سريعة: أهم ما تعلمناه

الاستخدامات

- تطوير البرامج.
- تطبيقات الويب.
- تطبيقات الهاتف.
- الذكاء الاصطناعي.

أمثلة

- Python
- Java
- C#
- JavaScript

كيف تعمل؟

- يكتب المبرمج الكود.
- يقوم Compiler أو Interpre بترجمته.
- ينفذه الحاسب.

كود المصدر ← Compiler أو Interpreter ← لغة الآلة

المميزات

- سهلة التعلم.
- سهلة كتابة الأكواد.
- سهلة اكتشاف الأخطاء.
- مناسبة للمبتدئين.

ما هي؟

- لغات قريبة من لغة الإنسان.
- سهلة القراءة والكتابة.

السؤال الأول: اختر الإجابة الصحيحة

1. لغة Python تعتبر من:
   Low Level | ✓ High Level | Machine Language | Assembly
2. من مميزات High Level:
   صعبة التعلم | تعتمد على 0 و1 | ✓ سهلة القراءة | لا تحتاج مترجمًا
3. لتحويل الكود إلى لغة الآلة نستخدم:
   Hard Disk | ✓ Compiler أو Interpreter | RAM | CPU
4. أي اللغات التالية High Level؟
   Binary | ✓ Java | Assembly | Machine Language
5. High Level مناسبة لـ:
   تطوير التطبيقات | كتابة الأكواد بسهولة | تتعلم للمبتدئين | ✓ جميع ما سبق

السؤال الثاني: أكمل

1. لغات البرمجة عالية المستوى قريبة من لغة ..... .
2. يتم تحويلها إلى لغة الآلة باستخدام ..... أو ..... .
3. لغة Python تعتبر لغة ..... المستوى.
4. من أشهر لغات البرمجة عالية المستوى ..... و
5. تستخدم هذه اللغات في تطوير ..... و ..... .

السؤال الثالث: صل بين العمودين

(أ)		(ب)
1 Python	• — •	ترجمة الكود
2 Java	• — •	تطوير تطبيقات الويب
3 JavaScript	• — •	لغة عالية المستوى
4 Compiler	• — •	تنفيذ الأوامر
5 Interpreter	• — •	لغة برمجة

السؤال الرابع: فكر

لماذا يفضل معظم المبرمجين استخدام لغات البرمجة عالية المستوى بدلاً من لغة الآلة؟

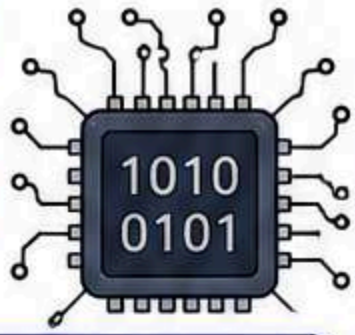
.....

.....

تذكّر: لغات البرمجة عالية المستوى سهلة للمبرمج، لكنها تحتاج إلى مترجم حتى يفهمها الحاسب.

لغات البرمجة منخفضة المستوى

Low Level Programming Languages



لغات قريبة جداً من لغة الحاسب، تتحكم في العتاد مباشرة وتعمل بسرعة عالية، لكنها أصعب في التعلم والاستخدام.

مراجعة سريعة: أهم ما تعلمناه

تستخدم في

- أنظمة التشغيل.
- الروبوتات والأنظمة المدمجة.
- التطبيقات التي تحتاج سرعة عالية.

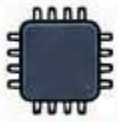
العيوب

- صعبة التعلم.
- كثيرة الأخطاء.
- تستغرق وقتاً أطول في البرمجة.
- صعبة القراءة والفهم.

المميزات

- سرعة عالية جداً.
- استهلاك قليل للذاكرة.
- تحكم مباشر في مكونات الحاسب.
- كفاءة عالية في الأداء.

أنواعها



1. Machine Language
   لغة الآلة
   تتكون من
   0 و 1 فقط.
2. Assembly Language
   لغة التجميع
   تستخدم أوامر
   مختصرة مثل:

MOV ADD SUB JMP

ما هي؟

- لغات قريبة جداً من لغة الحاسب.
- تستخدم أوامر يفهمها المعالج مباشرة.

السؤال الأول: اختر الإجابة الصحيحة

1. تتكون لغة الألة من:
   أوامر إنجليزية | ✓ 0 و1 فقط | كلمات بشرية | أكواد خاصة
2. لغة Assembly تعتبر من:
   JavaScript | Machine Language | ✓ Low Level | High Level
3. من مميزات لغات البرمجة منخفضة المستوى:
   صعبة التعلم | ✓ تتحكم في العتاد مباشرة | غير سريعة | لا تحتاج ذاكرة
4. أي من التالي مثال على لغة منخفضة المستوى؟
   C# | ✓ Assembly | Java | Python
5. تستخدم لغات المستوى المنخفض في:
   الألعاب فقط | ✓ الأنظمة المدمجة | تطبيقات التواصل | كتابة المقالات

السؤال الثاني: أكمل

1. تتكون لغة الألة من الأرقام: ..... و ..... فقط.
2. تستخدم لغة التجميع أوامر مختصرة مثل: ..... و ..... .
3. لغات البرمجة منخفضة المستوى تتحكم في ..... مباشرة.
4. من عيوب هذه اللغات أنها ..... و ..... .
5. تستخدم هذه اللغات في أنظمة التشغيل و ..... .

السؤال الثالث: صل بين العمودين

(أ)	(ب)
1 Machine Language	لغة الألة
2 Assembly Language	لغة التجميع
3 MOV	نقل محتوى
4 ADD	جمع قيم
5 أنظمة التشغيل	تستخدم لغات منخفضة المستوى

لماذا تعتبر لغات البرمجة منخفضة المستوى سريعة جداً؟
وما هي التحديات التي قد تواجه المبرمج عند استخدامها؟

.....

.....

تذكّر: لغات البرمجة منخفضة المستوى قوية وسريعة لكنها تحتاج إلى دقة عالية ومعرفة كبيرة بالحاسب.

مقارنة بين لغات البرمجة
High Level & Low Level

لكل نوع من لغات البرمجة مميزاته واستخداماته،
والاختيار الصحيح يعتمد على الهدف من البرنامج.

مراجعة سريعة: أهم ما تعلمناه

وجه المقارنة	High Level	Low Level
سهولة التعلم	✓ سهلة التعلم ومناسبة للمبتدئين.	✗ صعبة التعلم وتحتاج لخبرة كبيرة.
القرب من	✓ قريبة من لغة الإنسان.	✗ قريبة جدًا من لغة الحاسب.
الحاجة إلى مترجم	✓ تحتاج إلى Compiler أو Interpreter.	✗ تعمل مباشرة أو باستخدام Assembler.
سرعة التنفيذ	✗ أبطأ من اللغات منخفضة المستوى.	✓ سريعة جدًا.
استهلاك الذاكرة	✗ استهلاك أعلى للذاكرة.	✓ استهلاك أقل للذاكرة.
اكتشاف الأخطاء	✓ سهل ومريح.	✗ صعب ويستغرق وقتًا أطول.
الاستخدامات	✓ تطوير التطبيقات، الويب، الذكاء الاصطناعي، التطبيقات العامة.	✓ أنظمة التشغيل، البرمجيات المدمجة، التحكم في العتاد.

السؤال الأول: اختر الإجابة الصحيحة

1. اللغة التي تعتبر الأقرب إلى لغة الإنسان هي:
   Assembly | Machine Language | **High Level ✓** | Low Level
2. من عيوب لغات البرمجة منخفضة المستوى:
   صعبة التعلم ✓ | قريبة من الإنسان | سريعة الإنسان | سهلة التعلم
3. لغة Assembly تحتاج إلى:
   CPU | **Assembler ✓** | Interpreter | Compiler
4. أي اللغات التالية من High Level؟
   Assembly | **C# ✓** | Binary | Machine Language
5. تستخدم لغات البرمجة العالية المستوى في:
   أنظمة التشغيل | **تطوير التطبيقات ✓** | **جميع ما سبق ✓** | جميع ما سبق

السؤال الثاني: أكمل العبارات التالية

1. لغات البرمجة العالية المستوى أسهل في القراءة والكتابة من اللغات ..... .
2. لغة الألة تعتمد على الرمزين ..... و ..... فقط.
3. يتم تحويل لغات Assembly إلى لغة الألة باستخدام ..... .
4. تتميز لغات Low Level بـ ..... و ..... .
5. تستخدم لغات High Level في تطوير ..... و ..... .

السؤال الثالث: صل بين العمودين

(أ)	(ب)
1 High Level	قريبة من لغة الإنسان
2 Low Level	قريبة من لغة الحاسب
3 Machine Language	تعتمد على 0 و 1 فقط
4 Assembly Language	تستخدم أوامر مختصرة
5 Compiler/Interpreter	يحول الكود إلى لغة الألة

السؤال الرابع: ضع علامة (✓) أو (✗)

1. لغات High Level تحتاج إلى مترجم حتى يفهمها الحاسب. ( )
2. لغة الألة تعتمد على أوامر مثل MOV و ADD. ( )
3. لغات Low Level أبطأ من لغات High Level. ( )
4. Python تعتبر من لغات البرمجة عالية المستوى. ( )
5. لغة Assembly قريبة جدًا من لغة الإنسان. ( )

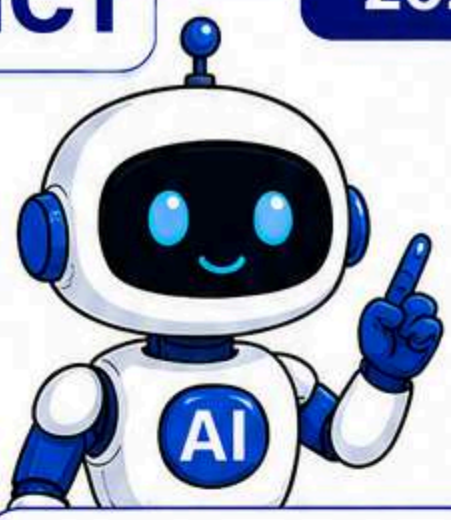
السؤال الخامس: فكر

متى تستخدم لغات البرمجة منخفضة المستوى (Low Level) ومتى تستخدم لغات البرمجة عالية المستوى (High Level)؟ أعطِ أمثلة لكل حالة.

.....

.....

تذكَّر: اختيار لغة البرمجة يعتمد على الهدف من البرنامج، فـ**High Level** سهلة ومناسبة للتطوير العام، بينما **Low Level** سريعة ومناسبة للتحكم في العتاد والأنظمة.



كيف تذاكر البرمجة بذكاء؟

المذاكرة الذكيةِ توفر وقتك وجهدك وتساعدك على الفهم والتذكر وتحقق أعلى الدرجات.

استراتيجيات المذاكرة الذكية

1. **ضع هُدفًا واضحًا**

حدد ما تريد تحقيقه في كل جلسة مذاكرة.

2. **افهم قبل الحفظ**

حاول فهم الفكرة بعمق، ثم انتقل إلى التطبيق.

3. **طبق ما تتعلمه**

تدرب على حل التمارين أثناء المذاكرة.

4. **راجع بانتظام**

راجع ما تعلمته باستمرار لتثبيت المعلومة.

5. **حل أسئلة السنوات السابقة**

تدرب على نمط الامتحان وتعرف على نوع الأسئلة.

6. **لا تذاكر لساعات طويلة**

خذ فترات راحة منتظمة لزيادة تركيزك.

تهيئة بيئة مذاكرة مناسبة

- اختر مكانًا هادئا ومضاءة جيدة.
- ابتعد عن الهاتف ومشتتات الانتباه.
- اجلس بشكل مريح على كرسي مناسب.
- رتب مكانك وأدواتك قبل البدء.
- اشرب ماء وتناول وجبات خفيفة صحية.
- استخدم مؤقت (Timer) لتنظيم وقتك.

نموذج لخطة جلسة مذاكرة فعالة

مراجعة سريعة راجع ما تعلمته في الجلسة السابقة.	10 دقائق
تعلم وفهم اقرأ وتعلم مفهوماً جديداً بتركيز.	30 دقيقة
تطبيق وحل تدريبات حل أمثلة وتمارين على ما تعلمته.	30 دقيقة
تلخيص وملاحظات اكتب ملخصًا مختصراً للمفاهيم المهمة.	10 دقائق
راحة قصيرة استرح قليلا ثُم عد بنشاط.	10 دقائق

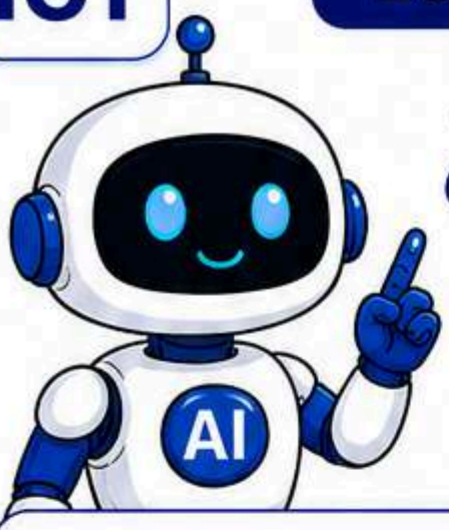
أدوات تساعدك على المذاكرة

فيديوهات تعليمية موثوقة.	جهاز كمبيوتر للتطبيق والبرمجة.	دفتر ملاحظات لتلخيص الدروس.
قوائم مهام لتنظيم وقتك.	خرائط ذهنية لتنظيم المعلومات	بطاقات مراجعة (Flashcards)

أخطاء شائعة يجب تجنبها

- مذاكرة بدون خطة أو هدف محدد.
- الحفظ بدون فهم أو تطبيق.
- تأجيل المذاكرة لأخر لحظة.
- عُدم مراجعة الدروس بشكل دوري.
- الإرهاق وقلة النوم وعدم أخذ راحة.

المذاكرة الذكيّة ليست في عدد الساعات، بل في جودة الفهم والتطبيق والمراجعة.
كن منظمًا، واثقاً، ومثابراً... والنجاح سيكون حليفك!



أهم الأخطاء الشائعة في الامتحان

تجنّب هذه الأخطاء لتضمن أعلى الدرجات

1 عدم قزاءة السؤال جيداً

أسرع خطأ ! اقرأ السؤال بعناية وحدد المطلوب بالضبط قبل الحل.

2 سوء إدارة الوقت

تضييع وقت طويل في سؤال واحد يسبب عدم إكمال بقية الأسئلة.

3 أخطاء في الكود

مثل (فواصل، مسافات، أقواس) تؤدي إلى خطأ في الناتج.

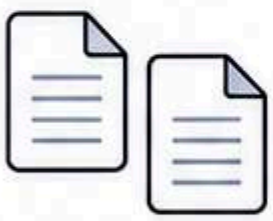
4 عدم اختبار البرنامج

عدم تشغيل البرنامج ومراجعة النتائج قبل التسليم.

5 إهمال كتابة التعليقات

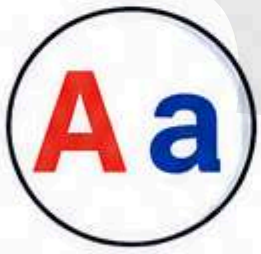
التعليقات توضح فكرتك وتحسن من فهم المصحح لإجابتك.

6 نسخ الكود بدون فهم



قد يختلف السؤال قليلاً ويؤدي النسخ إلى خطأ في الإجابة.

7 إهمال حالة الحروف



في البرمجة حالة الحروف فرق كبير مثل (name) مختلفة عن (Name).

8 عدم الانتباه للشروط

أخطاء في الشروط تؤدي إلى نتائج غير صحيحة للبرنامج.

9 إهمال تنسيق الإجابة

الإجابة غير منظمة تجعل من الصعب فهمها وإعطاء الدرجة كاملة.

10 عدم مراجعة الإجابة النهائية

مراجعة سريعة في النهاية تساعدك على اكتشاف الأخطاء البسيطة.

تذكّر دائما

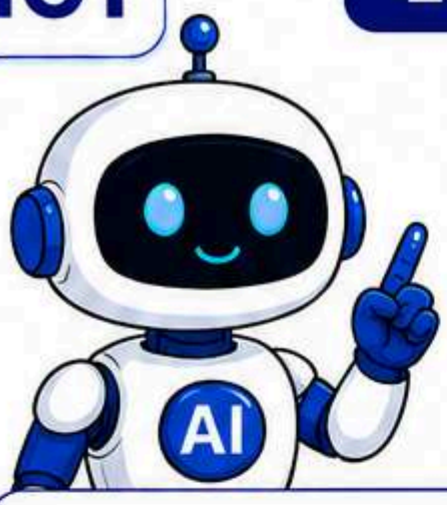
- النجاح في الامتحان ليس بالذكاء فقط بل بالتركيز والتنظيم والمراجعة.
- كل نقطة صغيرة تفرق في المجموع النهائي.
- الثقة بالنفس + التحضير الجيد = أعلى الدرجات.

نصائح ذهبية لتجنب الأخطاء

- اقرأ السؤال مرتين على الأقل.
- خطط لحلك قبل كتابة الكود.
- استخدم وقتك على جميع الأسئلة.
- راجع إجابتك قبل تسليم الورقة.
- حافظ على هدوئك وثقتك بنفسك.

تجنب الأخطاء ... وركز في الحل ... وثق في نفسك

النجاح يبدأ من الآن !



مراجعة شاملة ونهايـة

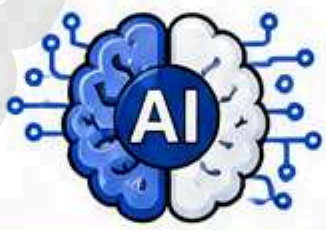
هنا ملخص كل ما تعلمته في البرمجة والذكاء الاصطناعي.
اقرأها جيداً قبل الامتحان وتأكد أنك مستعد لتحقيق أعلى الدرجات!

1 البرمجة الأساسية

- خوارزميات وحل المشكلات
- المتغيرات وأنواع البيانات
- العمليات الحسابية والمنطقية
- الجمل الشرطية والتكرار
- الدوال (الإجراءات)
- القوائم والمصفوفات
- التعامل مع النصوص
- البرمجة الكائنية (OOP)

2 الذكاء الاصطناعي

- ما هو الذكاء الاصطناعي
- تطبيقات الذكاء الاصطناعي
- تعلم الآلة (Machine Learning)
- تعلم عميق (Deep Learning)
- معالجة اللغة الطبيعية (NLP)
- الرؤية الحاسوبية (Computer Vision)
- أمثلة عملية في حياتنا اليومية

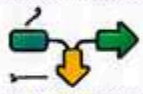


3 أخلاقيات الذكاء الاصطناعي

- العدالة وعدم التحيز
- الشفافية وقابلية التفسير
- الخصوصية وحماية البيانات
- الأمان والسلامة
- المسؤولية والمساءلة
- المنفعة العامة واحترام الإنسان

خطوات حل أي سؤال برمجي


اقرأ السؤال جيداً وافهم المطلوب.


حلل المعطيات وحدد المدخلات والمخرجات.


ضع خطة الحل (خوارزمية أو خطوات).

اكتب الكود بعناية وتأكد من صحته.


اختبر الكود بحالات مختلفة.


راجع الإجابة وحسّن الحل إن لزم.

نقاط مهمة تذكرها دائمًا

- افهم الفكرة قبل حفظ الكود.
- حل الكثير من التمارين والتطبيقات.
- راجع أخطاءك وتعلم منها.
- قسّم وقتك بين الدراسة والمراجعة.
- نم جيدًا وأكل صحي يحسن التركيز.
- ثق بقدراتك ولا تستسلم.
- الممارسة اليومية سر التفوق.

قبل الامتحان


راجع الملخصات والملاحظات.


حل نماذج امتحانات سابقة.

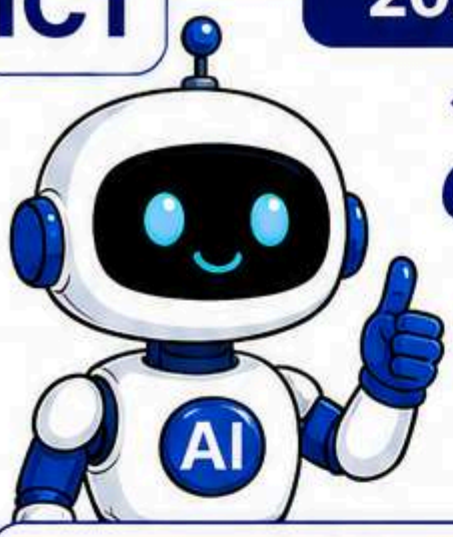

تأكد من فهمك للأفكار الأساسية.


كن هادئاً وواثقاً في نفسك.

أخطاء شائعة تجنبها

- عدم قراءة السؤال جيداً.
- إهمال حالة الأحرف (a ≠ A).
- نسيان نقطتين (:) في التكرار أو الشرط.
- خلط بين (=) و (==).
- نسيان إرجاع القيم من الدوال.
- اختبار الكود بحالة واحدة فقط.

أنت قادر على النجاح والتفوق !
لا تتوقف عن التعلم... فالمستقبل ينتظرك



صفحة ختامية وتحفيزية

رحلة التعلُّم لا تنتهي هنا... بل هي مجرد بداية جميلة لمستقبل رائع تنتظرك.
أنت الآن تملك المعرفة، والآن دورك لتطبق وتبدع وتحقّق أحلَامك!

مهارات المستقبل

- التفكير المنطقي
- الإبداع والابتكار
- حل المشكلات المعقدة
- العمل الجماعي
- التعلم المستمر
- الذكاء الرقمي

كيف تستمر في التطور؟

- مارس البرمجة يومياً ولو قليلاً.
- اقرأ الكثير واكتشف الجديد.
- حل تحديات ومسائل برمجية.
- شارك في مشاريع صغيرة.
- تعلم من الأخطاء ولا تستسلم.
- كن فضولياً... السؤال بداية الإبداع.

ماذا تعلمت؟

- أساسيات البرمجة
- حل المشكلات والخوارزميات
- البرمجة الكائنية (OOP)
- القوائم والمصفوفات
- التعامل مع النصوص
- الذكاء الاصطناعي
- أخلاقيات الذكاء الاصطناعي

قبل الامتحان

1. راجع الملخصات والملاحظات.
2. حل نماذج امتحانات سابقة.
3. أعد حل الأخطاء حتى تتقنها.
4. نظم وقتك أثناء الامتحان.
5. اقرأ السؤال جيداً قبل الإجابة.
6. كن هادئاً وواثقاً من نفسك.

نصائح ذهبية للنجاح

1. ضع هدفاً واضحاً واعمل من أجله.
2. نظم وقتك واختر بيئة مناسبة للدراسة.
3. أبدأ بالأسهل ثم انتقل للأصعب.
4. لا تترك شيئاً غامضاً، اسأل وتأكد.
5. اختبر نفسك باستمرار.
6. ثق بنفسك... أنت تستطيع!

تذكر دائماً


المستقبل يصنعه من يتعلم اليوم ويبدع غداً.


أنت أفضل مما تظن، فقط استمر!


التعلم عادة... عندما تتعلم أكثر تربح أكثر.


المثابرة أقوى من الذكاء، لا تيأس وأنت على الطريق الصحيح.

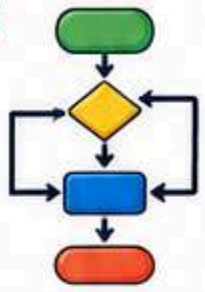

كل خطوة صغيرة اليوم تقربك من النجاح الكبير غداً.

أنت الآن جاهز
للإبداع والتفوق
مستقبلك في انتظارك... اذهب واصنعه !

قاموس المصطلحات البرمجية

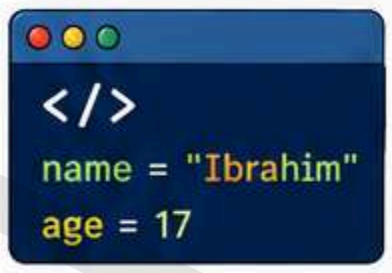
📖 أهم الكلمات التي يجب أن تعرفها في البرمجة وعلوم الحاسب

1 Algorithm
الخوارزمية



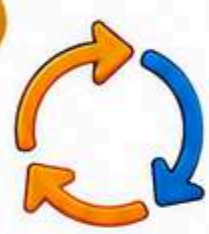
مجموعة خطوات مرتبة
لحل مشكلة أو تنفيذ مهمة.

2 Variable
المتغير



مكان في الذاكرة يستخدم
لتخزين البيانات.

3 Loop
الحلقة التكرارية



تعيد تنفيذ الأوامر
أكثر من مرة.

4 Function
الدالة

مجموعة أوامر تؤدي مهمة
محددة ويمكن إعادة استخدامها.

5 Hardware
المكونات المادية

الأجزاء التي يمكن لمسها من الحاسب مثل:

6 Software
البرمجيات

البرامج التي تعمل على الحاسب.
مثل:

7 Network
الشبكة

مجموعة أجهزة متصلة لتبادل البيانات.

8 Sensor
المستشعر

جهاز يكتشف التغيرات في البيئة
ويرسل بيانات للحاسب.

9 Artificial Intelligence (AI)
الذكاء الاصطناعي

تقنية تجعل الأجهزة تتعلم
وتتخذ قرارات ذكية.

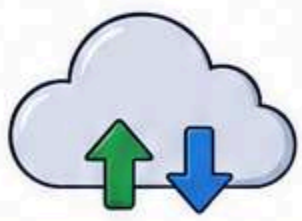
10 Data
البيانات

حقائق ومعلومات يتم إدخالها
للحاسب لمعالجتها.

11 Binary System
النظام الثنائي

نظام عد يعتمد على:
0 و 1 فقط.

12 Cloud Computing
الحوسبة السحابية



استخدام خدمات التخزين
والبرامج عبر الإنترنت.

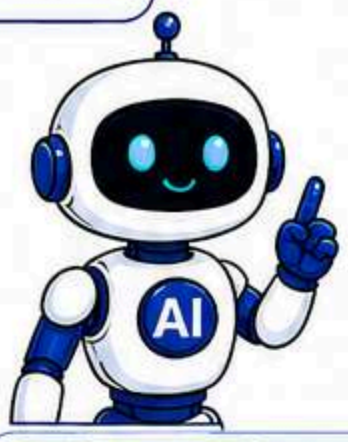
13 Cyber Security
الأمن السيبراني

حماية الأجهزة والشبكات والبيانات
من الهجمات الإلكترونية.

14 ICT
تكنولوجيا المعلومات والاتصالات

استخدام التكنولوجيا لتخزين المعلومات
ومعالجتها ونقلها.

**تذكّر: كلما فهمت المصطلحات الأساسية جيداً،
أصبح تعلم البرمجة أسهل وأسرع.**



اختبر نفسك في دقيقة

مراجعة سريعة لأهم مفاهيم الملزمة قبل الامتحان

البرمجة

1. ما هي البرمجة؟
2. ما هي الخوارزمية؟
3. ما وظيفة مخطط الانسياب؟
4. ما الفرق بين الإدخال والإخراج؟
5. ما فائدة المتغير؟

2 'Python

6. ما وظيفة ()print ؟
7. ما وظيفة ()input ؟
8. ما الفرق بين int و st ؟
9. ما هي الحلقة التكرارية؟
10. ما هي الدالة Function ؟

3 Computer Science

11. ما هو Hardware ؟
12. ما هو Software ؟
13. ما وظيفة CPU ؟
14. ما وظيفة RAM ؟
15. ما وظيفة نظام التشغيل؟

4 أنظمة العد

16. النظام الثنائي يعتمد على؟
    - ☐ 0 و 1
    - ☐ 1 و 2
    - ☐ 0 و 9
17. النظام العشري يعتمد على؟
    - ☐ 10 رمـوز
    - ☐ 2 رمز
    - ☐ 16 رمــز
18. النظام السادس عشر يسمي؟
    - Hexadecimal ☐
    - Binary ☐
    - Decimal ☐

5 Logic Gates

19. بوابة AND تعطي 1 عندما؟
20. بوابة OR تعطي 1 عندما؟
21. بوابة NOT تقوم بـــ؟

6 الشبكات والأمن السيبراني

22. ما هي الشبكة؟
23. ما هو الإنترنت؟
24. ما هو الأمن السيبراني؟
25. لماذا نستخدم كلمات مرور قوية؟

7 ICT

26. ماذا تعني ICT ؟
27. ما هي الحوسبة السحابية؟
28. ما هو البريد الإلكتروني؟
29. ما هو Sensor ؟
30. اذكر مثالاً على Sensor.

8 Microsoft Office

31. Word يستخدم في؟
32. Excel يستخدم في؟
33. PowerPoint يستخدم في؟
34. Outlook يستخدم في؟
35. OneNote يستخدم في؟

الذكاء الاصطناعي

36. ما هو الذكاء الاصطناعي؟
37. ما هو Machine Learning ؟
38. اذكر تطبيقاً للذكاء الاصطناعي.
39. ما فائدة الذكاء الاصطناعي في الطب؟
40. ما فائدة الذكاء الاصطناعي في التعليم؟

10 أخلاقيات الذكاء الاصطناعي

41. ما المقصود بالخصوصية؟
42. ما المقصود بالعدالة؟
43. ما المقصود بالشفافية؟
44. لماذا يجب استخدام الذكاء الاصطناعي بشكل مسؤول؟

أسئلة التميز

45. كيف يساعد الذكاء الاصطناعي في حل المشكلات؟
46. كيف تحمي بياناتك على الإنترنت؟
47. لماذا نتعلم البرمجة؟
48. ما المهارات التي اكتسبتها من الملزمة؟
49. ما المجال الذي أعجبك أكثر؟
50. ما هدفك القادم في تعلم البرمجة؟

تـذكـر

أنت تستطيع
ثق بقدرتك ولا تستسلم، المثابرة طريق النجاح.

التدريب المستمر
حل الأسئلة والتمارين يجعلك أكثر تميزاً.

الفهم أهم من الحفظ
حاول أن تفهم الفكرة جيداً وليس فقط حفظها.

مستر إبراهيم نجم
Ibrahim Negm

نموذج امتحان نهائي شامل

اختبر معلوماتك قبل الامتحان الحقيقي

1 السؤال الأول
اختر الإجابة الصحيحة

5	4	3	2	1
أي برنامج يستخدم للعروض التقديمية؟	النظام الثنائي يعتمد على:	CPU تعني:	أي لغة برمجة تعلمناها في الملزمة؟	البرمجة هي:
☐ Word	☑ 0 و 1	☐ وحدة التخزين	☐ Java	☐ لعبة إلكترونية
☐ Excel	☐ 1 و 2	☐ وحدة المعالجة المركزية ☑	☐ C++	☐ كتابة أوامر
☐ PowerPoint	☐ 0 و 9	☐ الشاشة	☐ Python ☑	☑ للحاسب
☐ Outlook	☐ A و B	☐ لوحة المفاتيح	☐ PHP	☐ نوع من الأجهزة
				☐ شبكة إنترنت

2 السؤال الثاني
ضع علامة ✔ أو ✘

البرمجة تساعد في حل المشكلات.	✔
Word يستخدم للحسابات.	✘
الإنترنت أكبر شبكة في العالم.	✔
Sensor يجمع بيانات من البيئة.	✔
الذكاء الاصطناعي يستخدم في الطب.	✔

3 السؤال الثالث
أكمل الفراغات بالكلمة المناسبة

1. _____ هو عقل الحاسب.
2. _____ تستخدم لتخزين البيانات.
3. _____ يستخدم لإنشاء الجداول والحسابات.
4. _____ هو فرع من الذكاء الاصطناعي يتعلم من البيانات.
5. _____ هي مجموعة أجهزة متصلة لتبادل البيانات.

4 السؤال الرابع
صل بين العمودين

العمود (أ)	العمود (ب)
1 Word	العروض التقديمية
2 Excel	المستندات
3 PowerPoint	جمع البيانات
4 AI	الذكاء الاصطناعي
5 Sensor	الجداول والحسابات

5 السؤال الخامس
سؤال التفكير

كيف يمكن للذكاء الاصطناعي أن يساعد في حياتك اليومية؟

.....

.....

.....

6 السؤال السادس
تحدي الأبطال ★

رتب خطوات حل أي مشكلة برمجية:

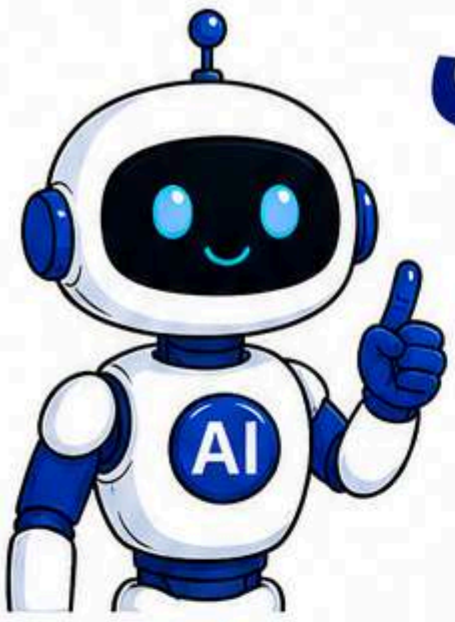
- ☐ كتابة الكود
- ☐ فهم المشكلة
- ☐ اختبار الحل
- ☐ وضع الخوارزمية

رسالة أخيرة

- ✔ لقد أنهيت رحلة التأسيس بنجاح.
- البرمجة ممارة تبنى بالممارسة.
- استمر في التعلم والتطوير.
- المستقبل للمبدعين في التكنولوجيا.

الدرجة النهائية

يحاج مراجعة	جيد	جيد جداً	ممتاز
أقل من 12	14 – 12	17 – 15	20 – 18
★★	★★★	★★★★	★★★★★



الملزمة انتهت
والبداية الحقيقية الآن

لقد أنهيت رحلة التأسيس بنجاح

الآن أنت **جاهز** للمستقبل

ماذا بعد؟

1 استمر في التعلم

التعلم لا يتوقف، كل يوم جديد يعطيك فرصة لتصبح أفضل.

2 مارس البرمجة

الممارسة اليومية تبني مهارتك وتحول المعرفة إلى إبداع.

3 حل المشكلات

كل مشكلة هي فرصة لتطوير عقلك وتفكيرك المنطقي.

4 اصنع مشروعك

ابدأ بمشاريع صغيرة واجعلها خطوة نحو إنجازات كبيرة في المستقبل.

أنت بطل رحلتك

ثق بنفسك... فكّر بإبداع ... ابني مستقبلك

المستقبل للمبدعين في التكنولوجيا!

المهارات التي اكتسبتها


الأمن السيبراني

الذ كاء الاصطناعي


تقنية المعلومات والاتصالات


علوم الحاسب


لغة البايثون

أساسيات البرمجة

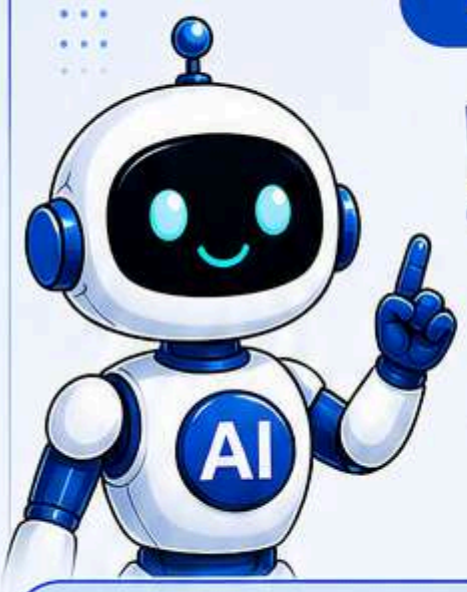
مستر إبراهيم نجم
Ibrahim Negm

نتمنى لك التوفيق والنجاح

في امتحانك وفي حياتك المستقبلية

تذكر دائمًا: التعلم هو مفتاح النجاح

ملزمة التأسيس لبرنامج البكالوريا المصرية 2026



لقد وصلت للنهاية!

مبروك إنجازك

لكن رحلتك الحقيقية تبدأ الآن...

سنتر مستر إبراهيم نجم

لشرح مادة البرمجة والذكاء الاصطناعي

لطلبة مسار الهندسة

في نظام البكالوريا المصرية

ماذا ستحصل معنا؟

- هدفنا أعلى الدرجات والتميز
- دعم كامل للإجابة عن جميع الاستفسارات
- مراجعات واختبارات دورية
- متابعة مستمرة للطلاب
- تدريبات عملية وأسئلة متنوعة
- شرح مبسط ومنظم خطوة بخطوة

لا تكتفي بالقراءة... ابدأ رحلتك نحو التميز الآن !

للحجز والاستفسار

01224194220

تواصل معنا الآن
واحجز مكانك !

الموقع الرسمي
mribrahim.in-masr.com

Facebook
Mr Ibrahim Negm

Instagram
@mribrahimnegm

المقـر

126 شارع المطرية
بجوار مطعم رضا حلمي
محافظة الاسماعيلية

تعلم بذكاء ... برمج مستقبلك

معك من البداية حتى الإبداع

أ / إبـراهيـم نجم

تعلم بذكاء ... برمج مستقبلك.