

🌟 كتاب ملوّن خطوة بخطوة للمبتدئين

برمجة بايثون

كأن لم تعرفها من قبل

من أول سطر برمجي إلى أول مشروع حقيقي، بلغة بسيطة وأمثلة ملوّنة



```
hello.py

def ابدأ():
    print("مرحباً بايثون")
    return "🚀"
```

🔧 مشروع حقيقي

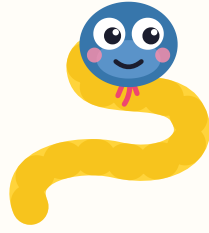
🏭 أمثلة من الصناعة

📖 تمارين وتحديات

🎯 من الصفر

Dr Merabti

تأليف د. سفيان مرابطي



برمجة بايثون

كأن لم تعرفها من قبل

دليلك المصوّر خطوة بخطوة، من الصفر إلى أوّل مشروع حقيقي

تأليف

د. سفيان مرابطي

برمجة بايثون كأن لم تعرفها من قبل — د. سفيان مرابطي

الطبعة الأولى، 2026

جميع الحقوق محفوظة للمؤلف. لا يجوز نسخ هذا الكتاب أو توزيعه أو بيعه، كليًا أو جزئيًا، بأي وسيلة دون إذن كتابي من المؤلف.

Python اسم وعلامة مسجلة لمؤسسة Python Software Foundation، ولا علاقة لها بهذا الكتاب.

merabti.com

مرحبًا بك يا صديقي



هل سمعت كلمة «برمجة» فتخيلت شاشة سوداء مليئة بالرموز الغامضة، وشخصًا عبقريًا لا ينام؟ انتس هذه الصورة تمامًا. البرمجة ببساطة هي أن **تتحدث مع الحاسوب بلغة يفهمها**، فتطلب منه أن يحسب أو يرتب أو يرسم أو يكرر عملاً مملًا بدلًا منك. ولغة بايثون هي أسهل هذه اللغات وأقربها إلى لغة البشر.

هذا الكتاب كُتب لمن **لم يكتب سطرًا برمجيًا واحدًا في حياته**: الطالب والمهندس والتقني والأستاذ وصاحب المشروع، بل وحتى من جرّب البرمجة من قبل فتعقّدت أمامه، فأراد أن يبدأ من جديد «كأن لم يعرفها من قبل».

الفهرس

1 كيف صمّم هذا الكتاب؟

كل فصل يبدأ بصفحة ملوّنة تخبرك بما ستتعلمه، ثم يأتي الشرح خطوة خطوة، ومع كل فكرة مثال برمجي تستطيع كتابته وتجربته فورًا، ثم ترى **الناتج** الذي سيظهر على شاشتك. وفي نهاية كل فصل خلاصة سريعة وتحذّر صغير تجد حله في آخر الكتاب.

الأمثلة البرمجية تظهر هكذا، في صندوق داكن مثل محرّر الأكواد الحقيقي، مع أرقام الأسطر:

Python

<> python

```
1 name = "سفيان"
2 print("أهلاً", name)
3 print("مرحبًا بك في عالم بايثون")
```

أما ما يظهر على الشاشة بعد تشغيل البرنامج فيكون في صندوق أخضر:

> الناتج

أهلاً سفيان
مرحبًا بك في عالم بايثون!

الفهرس

2 رموز ستلازمك في الطريق

نصيحة ذهبية



حيلة أو عادة جيدة توقّر عليك الوقت وتجعل برامجك أجمل.

انتبه!

خطأ شائع يقع فيه المبتدئون، فانتبه له قبل أن تقع فيه.

معلومة

معلومة إضافية لطيفة تفهم بها ما يجري «خلف الكواليس».

جرب بنفسك

تمرين قصير: أغلق الكتاب وجربه بنفسك على الحاسوب قبل أن تكمل.

من أرض المصنع

أمثلة من عالم الصناعة والطاقة: محرّكات، حسّاسات، استهلاك كهرباء... لترى أن بايثون أداة عمل حقيقية.

تحدي الفصل

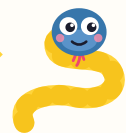
تحدي أكبر في آخر كل فصل، وحله في ملحق الحلول.

3 تعرّف على «بايثو»

سيرافك في هذه الرحلة صديق صغير اسمه **بايثو**، ثعبان لطيف لا يعصّ أبدًا (إلا الأخطاء البرمجية!). يظهر بين الحين والآخر ليعلّق أو ينصح أو يمزح قليلًا.

بايثو يقول:

مرحبًا! أنا بايثو. سمعت أنك تخاف من البرمجة؟ لا تقلق، أنا أيضًا كنت أخاف من الفئران... ثم أصبحت ثعبانًا. كل شيء ممكن!



4 نصيحتي الوحيدة قبل أن نبدأ

لا تقرأ هذا الكتاب كما تقرأ رواية. البرمجة مثل السباحة وقيادة الدراجة: لا تتعلّم بالقراءة وحدها. افتح الحاسوب بجانب الكتاب، واكتب كل مثال بيدك (لا تنسخه!)، وغيّر فيه، وأخطئ عمدًا لترى ماذا يحدث. فالخطأ هنا لا يكسر شيئًا، بل هو أفضل معلّم.

اضغط على أي عنوان في الفهرس لتنتقل إلى صفحته، واضغط على رأس أي صفحة (الشريط العلوي) أو رقمها في الأسفل لتعود إلى الفهرس.

أتمنى لك رحلة ممتعة، وإلى اللقاء في السطر الأول!

الفهرس

اضغط على أي عنوان لتنتقل إلى صفحته، واضغط على رأس أي صفحة لتعود إلى هنا.

3

مرحبًا بك يا صديقي



10

الفصل 1

مرحبًا بايثون

1

- 11 ما هي البرمجة أصلاً؟
- 11 لماذا بايثون بالذات؟
- 12 تثبيت بايثون على حاسوبك
- 12 أين نكتب البرامج؟
- 13 أول برنامج لك
- 13 البرامج في ملفات
- 14 التعليقات: ملاحظات لك لا للحاسوب
- 14 عندما يحدث خطأ

16

الفصل 2

المتغيرات وأنواع البيانات

2

- 17 ما هو المتغير؟
- 17 المتغير يتغير!
- 18 كيف نسمي المتغيرات؟
- 18 أنواع البيانات الأساسية
- 19 الدالة type: ما نوع هذه القيمة؟
- 20 التحويل بين الأنواع

22

الفصل 3

العمليات والحساب

3

- 23 العمليات الحسابية
- 24 ترتيب العمليات
- 24 الاختصارات الذكية
- 25 المقارنة: صح أم خطأ؟
- 25 العمليات المنطقية
- 26 مكتبة math

28

الفصل 4

النصوص

4

- 29 إنشاء النصوص
- 29 اللصق والتكرار
- 30 سلاسل f: أجمل طريقة لتركيب الجمل
- 30 الفهرسة: كل حرف له رقم
- 31 القص: أخذ جزء من النص
- 32 دوال النصوص المفيدة

34

الفصل 5

الحوار مع المستخدم

5

- 35 الدالة input
- 35 انتبه: input تعيد نصًا دائمًا!
- 36 أسرار الدالة print
- 37 تنسيق الأرقام والجداول
- 37 مشروع صغير: حاسبة قانون أوم

40

الفصل 6

الشروط واتخاذ القرار

6

- 41 الجملة if
- 41 الإزاحة: قانون بايثون الذهبي
- 42 else: وإلا...
- 42 elif: عندما تكثر الخيارات
- 43 الشروط المركبة
- 44 الشرط في سطر واحد
- 44 الجملة match

46

الفصل 7

الحلقات والتكرار

7

- 47 لماذا نحتاج الحلقات؟
- 47 الحلقة for مع range
- 48 المرور على النصوص
- 49 الحلقة while
- 49 continue و break
- 50 الحلقات المتداخلة
- 50 مشروع: لعبة «خمن الرقم»

53

الفصل 8

8

القوائم والصفوف

- 54 ما هي القائمة؟
- 54 الوصول إلى العناصر
- 55 الإضافة والحذف
- 56 الترتيب والبحث والإحصاء
- 56 المرور على القوائم
- 57 فهم القوائم: سطر واحد يغني عن أربعة
- 58 tuple الصفوف

60

الفصل 9

9

القواميس والمجموعات

- 61 ما هو القاموس؟
- 61 القراءة بأمان باستعمال get
- 62 الإضافة والتعديل والحذف
- 62 المرور على القواميس
- 63 القواميس المتداخلة
- 64 المجموعات set

66

الفصل 10

10

الدوال

- 67 لماذا الدوال؟
- 67 تعريف الدالة واستدعاؤها
- 68 المعاملات: نعطى الدالة معلومات
- 68 return: الدالة تعيد نتيجة
- 69 القيم الافتراضية والمعاملات المسماة
- 69 المجال: أين يعيش المتغير؟
- 70 التوثيق و lambda

72

الفصل 11

11

الوحدات والمكتبات

- 73 ما هي الوحدة؟
- 73 طرق الاستيراد
- 74 جولة في المكتبات المدمجة
- 75 صنع وحدتك الخاصة
- 76 تثبيت المكتبات الخارجية بـ pip
- 77 لمحة: ارسم أول منحني لك

78

الفصل 12

12

الملفات ومعالجة الأخطاء

- 79 لماذا الملفات؟
- 79 الكتابة في ملف
- 80 القراءة من ملف
- 80 ملفات CSV
- 81 الأخطاء الشائعة
- 81 try و except: شبكة الأمان
- 82 إدخال لا يتعطل أبدًا

85

الفصل 13

13

البرمجة الكائنية

- 86 الفكرة: الصنف والكائن
- 86 أول صنف لك
- 87 التتابع: ماذا يستطيع الكائن أن يفعل؟
- 89 الطباعة الجميلة بـ `__str__`
- 89 الوراثة

92

الفصل 14

14

المشروع النهائي: مراقب الطاقة

- 93 فكرة المشروع
- 93 الخطوة 1: ملف البيانات
- 94 الخطوة 2: الصنف Motor
- 94 الخطوة 3: قراءة الملف بأمان
- 95 الخطوة 4: التقرير وفرص التوفير
- 96 الخطوة 5: الحفظ والقائمة الرئيسية
- 97 لحظة الحقيقة: التشغيل

99

الورقة المرجعية السريعة



103

حلول التحديات



112

ماذا بعد؟





01

مرحبًا بايثون

ما البرمجة؟ ولماذا بايثون؟ وكيف تكتب أول برنامج لك في أقل من خمس دقائق

🎯 في هذا الفصل ستتعلم

- ✓ ما معنى البرمجة ولماذا نحتاجها
- ✓ تثبيت بايثون على حاسوبك خطوة خطوة
- ✓ الفرق بين الوضع التفاعلي والملفات
- ✓ لماذا اختار العالم لغة بايثون
- ✓ كتابة أول برنامج وتشغيله
- ✓ كتابة التعليقات داخل البرنامج

كل مبرمج عظيم بدأ بسطر واحد: `print("Hello")`. اليوم دورك!



1.1 ما هي البرمجة أصلاً؟

الحاسوب آلة سريعة جدًا لكنها **لا تفكر**. إنها تنفذ الأوامر حرفيًا، واحدًا بعد الآخر، بلا ملل ولا تعب. البرنامج هو مجموعة من هذه الأوامر مكتوبة بترتيب معيّن، مثل وصفة طبخ:

- 1 اغسل الطماطم.
- 2 قطّعها إلى مكعبات.
- 3 أضف الملح.
- 4 إذا كانت حامضة، أضف قليلًا من السكر.

لاحظ أن الوصفة فيها **خطوات متتالية وشرط** («إذا كانت حامضة»). البرامج كلها مبنية من هذه الأفكار البسيطة نفسها، وستتعلمها كلها في هذا الكتاب.

1.2 لماذا بايثون بالذات؟

ظهرت لغة بايثون سنة 1991 على يد المبرمج الهولندي **غيدو فان روسم** (Guido van Rossum)، وأخذت اسمها من برنامج كوميدي بريطاني اسمه Monty Python، لا من الثعبان! واليوم هي من أكثر لغات البرمجة استعمالًا في العالم، وذلك لأسباب كثيرة:

- **سهولة القراءة:** تكاد تُقرأ كأنها جمل إنجليزية بسيطة.
- **مجانية** ومفتوحة المصدر، وتعمل على ويندوز وماك ولينكس.
- **متعددة الاستعمالات:** الذكاء الاصطناعي، تحليل البيانات، المواقع، الأتمتة، العلوم والهندسة، التحكم في الأجهزة...
- **مجتمع ضخم:** أي سؤال يخطر ببالك، ستجد أن آلاف الناس سألوه قبلك وأجيبوا عنه.

قارن بين برنامج يطبع جملة على الشاشة بلغة جافا وبلغة بايثون:

اللغة	البرنامج
جافا	<pre>public class A { public static void main(String[] a) { System.out.println("Hi"); } }</pre>
بايثون	<pre>print("Hi")</pre>

سطر واحد فقط! هذا هو سحر بايثون.

من أرض المصنع

في المصانع الحديثة تُستعمل بايثون لجمع قراءات الحساسات، ورسم منحنيات استهلاك الطاقة، وكشف الأعطال قبل وقوعها بالذكاء الاصطناعي (الصيانة التنبؤية)، وكتابة التقارير اليومية آليًا بدل نسخها يدويًا في إكسل.

الفهرس

1.3 تثبيت بايثون على حاسوبك

على ويندوز

- 1 ادخل إلى الموقع الرسمي python.org واضغط على **Downloads**.
- 2 حمّل آخر إصدار (يبدأ بالرقم 3، مثل Python 3.13).
- 3 شغل الملف الذي حمّلته.
- 4 **مهم جدًا:** فَعِّل الخيار **Add python.exe to PATH** في أسفل النافذة الأولى.
- 5 اضغط **Install Now** وانتظر حتى ينتهي التثبيت.

انتبه!

إذا نسيت تفعيل خيار **Add to PATH** فلن يتعرّف الحاسوب على الأمر **python** في نافذة الأوامر. الحل: أعد تشغيل ملف التثبيت واختر **Modify** ثم فَعِّل الخيار.

هل نجح التثبيت؟

افتح نافذة الأوامر: اضغط **Win** **R** ثم اكتب **cmd** واضغط **Enter**، ثم اكتب:

```
Terminal
python --version

> الناتج
Python 3.13.0
```

إذا ظهر رقم الإصدار فمبروك، بايثون موجودة على حاسوبك! (على ماك ولينكس اكتب **python3** بدل **python**).

الفهرس

1.4 أين نكتب البرامج؟

مع بايثون يأتي برنامج صغير اسمه **IDLE**، وهو كافٍ تمامًا للبداية. ابحث عنه في قائمة ابدأ. وعندما تتقدّم قليلًا أنصحك بمحرّر **Visual Studio Code** المجاني، فهو يلوّن الأكواد ويكتشف الأخطاء ويكمل الكلمات بدلًا منك.

معلومة

يمكنك أيضًا البرمجة دون تثبيت أي شيء عبر مواقع مثل **Google Colab** أو **replit.com**، تكتب فيها الكود في المتصفح وتشغله مباشرة. مفيد جدًا إذا كنت تتعلم من الهاتف أو من حاسوب العمل.

الفهرس

1.5 أول برنامج لك

افتح IDLE، ستظهر نافذة فيها ثلاث علامات **>>>** هذا هو **الوضع التفاعلي**: تكتب أمرًا فتتقّده بايثون فورًا. اكتب:

```
Python Shell
<> python

>>> print("مرحبًا يا عالم")
مرحبًا يا عالم
```

ها قد كتبت أول برنامج لك! الأمر **print** معناه «اطبع»، وكل ما تضعه بين القوسين وعلامتي التنصيص يظهر على الشاشة كما هو.

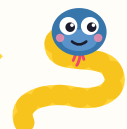
جرب الآن أن تستعمل بايثون كآلة حاسبة:

```
Python Shell
<> python

>>> 7 + 5
12
>>> 230 * 15
3450
>>> 100 / 3
33.333333333333336
```

بايثو يقول:

لاحظ أن بايثون لا تتعب من الحساب أبدًا. اطلب منها **1000 ** 2** وانظر إلى الرقم العملاق! (العلامة ****** تعني «أس»).



الفهرس

1.6 البرامج في ملفات

الوضع التفاعلي رائع للتجربة، لكن ما تكتبه فيه يضيع عند إغلاق النافذة. البرامج الحقيقية تُكتب في **ملفات** تنتهي بـ **.py**.

1 في IDLE اختر **New File** ← **File**.

2 اكتب البرنامج التالي.

3 احفظه باسم `hello.py` عبر `Ctrl` `S`.

4 شغله بالضغط على `F5`.

```
hello.py python
1 print("السلام عليكم")
2 print("اسمي بايثون، وعمري أكثر من 30 سنة")
3 print("وأنا هنا لأساعدك!")
```

> الناتج

السلام عليكم
اسمي بايثون، وعمري أكثر من 30 سنة
وأنا هنا لأساعدك!

لاحظ أن بايثون تنفذ الأسطر بالترتيب من الأعلى إلى الأسفل، سطرًا بعد سطر.

1.7 التعليقات: ملاحظات لك لا للحاسوب

كل ما يأتي بعد العلامة `#` يتجاهله الحاسوب تمامًا. نستعمل ذلك لكتابة **تعليقات** تشرح الكود لنا ولغيرنا:

```
Python python
1 # هذا البرنامج يحسب مساحة غرفة
2 print(4 * 5) # الطول * العرض
3 # print("هذا السطر لن يُنفَّذ")
```

> الناتج

20

نصيحة ذهبية

اكتب التعليقات لتشرح **لماذا** كتبت الكود، لا **ماذا** يفعل. وصدّقي، بعد ستة أشهر ستنسى كل شيء، وستشكر نفسك على كل تعليق كتبتّه!

1.8 عندما يحدث خطأ

جرب أن تكتب `Print` بحرف كبير، أو أن تنسى قوسًا:

```
Python Shell <> python

>>> Print("مرحبًا")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Print' is not defined
```

لا تخف من هذه الرسالة الحمراء! إنها ليست عقابًا، بل **مساعدة**: بايثون تخبرك بالسطر الذي فيه الخطأ (**line 1**) وبنوعه (**NameError** : اسم غير معروف). بايثون **تفرّق بين الحروف الكبيرة والصغيرة**، ف **print** غير **Print**.

ملف

- البرنامج مجموعة أوامر يُنفّذها الحاسوب بالترتيب.
- **print()** تطبع على الشاشة، وبايثون تحسب مثل الآلة الحاسبة.
- الوضع التفاعلي >>> لتجربة، والملفات **.py** للبرامج الحقيقية.
- **#** لكتابة التعليقات، والحروف الكبيرة والصغيرة مختلفة.
- رسائل الخطأ صديقتك: اقرأ آخر سطر منها دائمًا.

ملف

اكتب برنامجًا في ملف اسمه **card.py** يطبع بطاقة تعريف لك من خمسة أسطر: اسمك، مهنتك، مدينتك، هوايتك، وسطرًا أخيرًا من النجوم *********.



02

المتغيّرات وأنواع البيانات

صناديق صغيرة نحفظ فيها المعلومات لنستعملها متى شئنا

🎯 في هذا الفصل ستتعلم

- ✓ ما هو المتغيّر وكيف نُنشئه
- ✓ الأعداد الصحيحة والعشرية
- ✓ معرفة نوع أيّ قيمة بالدالة type
- ✓ قواعد تسمية المتغيّرات
- ✓ النصوص والقيم المنطقية
- ✓ التحويل بين الأنواع

المتغيّر مثل علبة عليها اسم. ضع فيها ما تشاء، ثم نادها باسمها!



2.1 ما هو المتغير؟

تخيل أن أمامك علبة صغيرة، تكتب على كل علبة اسمًا، وتضع داخلها شيئًا ما. هذا بالضبط هو **المتغير**: اسم نربطه بقيمة لنستعملها لاحقًا.

```
Python <> python
1 age = 25
2 name = "أمينة"
3 print(name)
4 print(age)
```

> الناتج

أمينة

25

العلامة **=** في البرمجة **لا تعني «يساوي»** كما في الرياضيات، بل تعني **«ضع»** القيمة التي على اليمين في الصندوق الذي على اليسار». لذلك نقرأ `age = 25` هكذا: «ضع 25 في المتغير age».

2.2 المتغير يتغير!

سُمي متغيرًا لأن قيمته يمكن أن تتغير في أي لحظة:

```
Python <> python
1 score = 10
2 print(score)
3 score = 15
4 print(score)
5 score = score + 5
6 print(score)
```

> الناتج

10

15

20

السطر الخامس قد يبدو غريبًا: `score = score + 5`. بايثون تحسب الجهة اليمنى أولاً ($20 = 5 + 15$) ثم تضع النتيجة في `score`. القيمة القديمة تضيع وتحلّ محلها الجديدة.

معلومة ⓘ

يمكنك إعطاء عدّة متغيرات قيمًا في سطر واحد: `x, y, z = 1, 2, 3` وكذلك تبديل قيمتين بسهولة مذهلة: `x, y = y, x`. في أغلب اللغات الأخرى تحتاج متغيرًا ثالثًا لتفعل ذلك!

☰ الفهرس

2.3 كيف نسمّي المتغيرات؟

للتسمية قواعد لا بدّ منها، وعادات حسنة يُستحسن اتباعها:

القاعدة	صحيح	خطأ
يبدأ بحرف أو <code>_</code>	<code>name</code> و <code>_count</code>	<code>2name</code>
بلا مسافات	<code>motor_speed</code>	<code>motor speed</code>
حروف وأرقام و <code>_</code> فقط	<code>temp2</code>	<code>temp-2</code> و <code>price\$</code>
ليس كلمة محجوزة	<code>my_class</code>	<code>for</code> و <code>if</code> و <code>class</code>

نصيحة ذهبية ⓘ

اختر أسماء تشرح نفسها: `temperature` أفضل بكثير من `t`، و `total_price` أفضل من `tp`. والعادة في بايثون أن نكتب الأسماء بحروف صغيرة ونفصل الكلمات بشرطة سفلية، مثل `max_power`، ويسمّى هذا الأسلوب `snake_case`، أي «أسلوب الثعبان»... طبقًا!

انتبه! ⚠

بايثون تسمح بالأسماء العربية مثل `العمر = 30`، لكن لا أنصحك بذلك أبدًا، فهي تُربك المحرّرات وتختلط فيها الاتجاهات. اكتب الأسماء بالإنجليزية، والنصوص والتعليقات بالعربية كما تشاء.

☰ الفهرس

2.4 أنواع البيانات الأساسية

ليست كل القيم من نوع واحد. فالعدد 25 ليس مثل الكلمة "مرحبًا". هذه الأنواع الأربعة الأساسية:

النوع	الاسم في بايثون	أمثلة
عدد صحيح	int	7 و -3 و 2026
عدد عشري	float	3.14 و 0.5 و -12.75
نص	str	"سلام" و 'Python' و "123"
منطقي	bool	True و False فقط

```
Python
1 voltage = 380          # int
2 current = 12.5         # float
3 motor = "مضخة الماء"   # str
4 running = True         # bool
```

انتبه!

العدد العشري يُكتب **بنقطة** لا بفاصلة: 12.5 صحيحة، أما 12,5 فبايثون تفهمها شيئاً آخر تماماً (صفاً من عددين، ستعرّف عليه في الفصل الثامن).

2.5 الدالة type: ما نوع هذه القيمة؟

إذا شككت في نوع قيمة ما، اسأل بايثون مباشرة:

```
Python Shell
>>> type(25)
<class 'int'>
>>> type(25.0)
<class 'float'>
>>> type("25")
<class 'str'>
>>> type(True)
<class 'bool'>
```

لاحظ الفرق المهم: 25 عدد، أما "25" بين علامتي تنصيص فهي نص مكوّن من حرفين، حتى لو كانا رقمين!

2.6 التحويل بين الأنواع

كثيرًا ما نحتاج إلى تحويل قيمة من نوع إلى آخر، وذلك باستعمال اسم النوع كدالة:

```
Python
1 a = "50"
2 b = int(a)      # من نص إلى عدد صحيح
3 c = float(a)    # من نص إلى عدد عشري
4 d = str(3.7)    # من عدد إلى نص
5 print(b + 10)
6 print(c)
7 print(d + " كيلوواط")
```

الناتج

```
60
50.0
3.7 كيلوواط
```

انتبه!

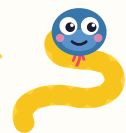
لا يمكن تحويل أي نص إلى عدد: `int("مرحبًا")` يعطي خطأ `ValueError`، وكذلك `int("3.5")`. لتحويل "3.5" استعمال `float` أولاً.

من أرض المصنع

لوحات التحكم وأجهزة القياس ترسل القراءات غالبًا كنصوص، مثل "231.4". قبل أن تحسب بها المتوسط أو تقارنها بحدّ الإنذار، يجب أن تحوّلها إلى أعداد باستعمال `float()`.

بايثو يقول:

سؤال سريع: ما نوع `True`؟ وما نوع `"True"`؟ إذا أجبت «`bool` ثم `str`» فأنت جاهز للفصل القادم!



ملخص الفصل

- المتغير اسم يُشير إلى قيمة، ونعطيه القيمة بالعلامة = .
- الأسماء بالإنجليزية، بحروف صغيرة، وتشرح نفسها: `motor_speed` .
- الأنواع الأساسية: `int` و `float` و `str` و `bool` .
- `type()` تخبرك بالنوع، و `int()` و `float()` و `str()` تحوّل بين الأنواع.

تحدي الفصل

أنشئ متغيرات تصف محرّكًا كهربائيًا: اسمه، قدرته بالكيلوواط (عدد عشري)، عدد أقطابه (عدد صحيح)، وهل هو يعمل الآن أم لا. ثم اطبع كل متغير مع نوعه باستعمال `type()` .



03

العمليات والحساب

بايثون آلة حاسبة خارقة، وأكثر من ذلك بكثير

🎯 في هذا الفصل ستتعلم

- ✓ العمليات الحسابية السبع
- ✓ الاختصارات مثل $+$ و $-$
- ✓ العمليات المنطقية and و or و not
- ✓ ترتيب العمليات والأقواس
- ✓ عمليات المقارنة
- ✓ مكتبة math للعمليات المتقدمة

هل تعلم أن $7 // 2$ تساوي 3؟ لا تقلق، سأشرح لك السبب بعد قليل.



3.1 العمليات الحسابية

العملية	الرمز	المثال	النتيجة
الجمع	+	$7 + 2$	9
الطرح	-	$7 - 2$	5
الضرب	*	$7 * 2$	14
القسمة	/	$7 / 2$	3.5
القسمة الصحيحة	//	$7 // 2$	3
باقي القسمة	%	$7 \% 2$	1
الأس (القوة)	**	$7 ** 2$	49

العمليات الثلاث الأخيرة تحتاج إلى شرح قصير:

- **القسمة الصحيحة //** تعطي عدد المرات الكاملة: في 7 دنانير كم قطعة من فئة 2 دينار؟ 3 قطع.
- **الباقى %** يعطي ما تبقى بعد القسمة: يبقى دينار واحد.
- **الأس **: 3 ** 2** تعني $2 \times 2 \times 2 = 8$.

```

Python
python
1 minutes = 135
2 hours = minutes // 60
3 rest = minutes % 60
4 print(hours, "ساعة و", rest, "دقيقة")

```

الناتج

2 ساعة و 15 دقيقة

نصيحة ذهبية

الباقى % مفيد جداً لمعرفة إن كان العدد زوجياً: إذا كان $n \% 2$ يساوي 0 فالعدد زوجي. ستستعمل هذه الحيلة كثيراً.

3.2 ترتيب العمليات

تتبع بايثون الترتيب نفسه الذي تعلّمته في المدرسة: الأقواس أولاً، ثم الأس، ثم الضرب والقسمة، ثم الجمع والطرح.

```
Python Shell
>>> 2 + 3 * 4
14
>>> (2 + 3) * 4
20
>>> 2 ** 3 ** 2
512
```

نصيحة ذهبية



عندما تشكّ في الترتيب، ضع أقواسًا. الأقواس لا تكلف شيئًا، وتجعل الكود أوضح لك وللمن يقرأه بعدك.

3.3 الاختصارات الذكية

تغيير قيمة متغير بناءً على قيمته الحالية أمر شائع جدًا، لذلك توجد له اختصارات:

المختصر	الطويل
<code>x += 5</code>	<code>x = x + 5</code>
<code>x -= 2</code>	<code>x = x - 2</code>
<code>x *= 3</code>	<code>x = x * 3</code>
<code>x /= 2</code>	<code>x = x / 2</code>

```
Python
1 counter = 0
2 counter += 1
3 counter += 1
4 print(counter)

> الناتج
2
```

3.4 المقارنة: صح أم خطأ؟

عمليات المقارنة لا تعطي عددًا، بل تعطي قيمة منطقية: **True** (صح) أو **False** (خطأ).

الرمز	المعنى	المثال	النتيجة
=	يساوي	5 = 5	True
≠	لا يساوي	5 ≠ 3	True
>	أكبر من	5 > 8	False
<	أصغر من	5 < 8	True
≥	أكبر من أو يساوي	5 ≥ 5	True
≤	أصغر من أو يساوي	9 ≤ 4	False

انتبه!

أشهر خطأ عند المبتدئين: الخلط بين = و ==. العلامة الواحدة = تضع قيمة في متغير، والعلامة == تساؤل: هل القيمتان متساويتان؟

3.5 العمليات المنطقية

نربط بها بين عدّة شروط:

- **and** (و): النتيجة صحيحة إذا كان الشرطان معًا صحيحين.
- **or** (أو): النتيجة صحيحة إذا كان أحدهما على الأقل صحيحًا.
- **not** (ليس): قلب الصحيح خطأ والخطأ صحيحًا.

```

Python
1 temperature = 85
2 pressure = 4.2
3 print(temperature > 80 and pressure > 5)
4 print(temperature > 80 or pressure > 5)
5 print(not temperature > 80)

```

> الناتج

False
True
False

معلومة ⓘ

بايثون تسمح بكتابة المقارنات المتسلسلة كما في الرياضيات تمامًا: $18 \leq \text{age} \leq 60$ ، وهذا أجمل من $\text{age} \geq 18 \text{ and } \text{age} \leq 60$.

الفهرس

3.6 مكتبة math

لعمليات أكثر تقدّمًا، تأتي بايثون بمكتبة جاهزة اسمها `math` . نستدعيها بالكلمة `import` (وستحدّث عن المكتبات بالتفصيل في الفصل الحادي عشر):

```
Python <> python
1 import math
2
3 print(math.sqrt(144))      # الجذر التربيعي
4 print(math.pi)           # العدد π
5 print(round(math.pi, 2))   # تقريب إلى رقمين بعد الفاصلة
6 print(math.cos(0))        # جيب التمام
```

> الناتج

12.0
3.141592653589793
3.14
1.0

من أرض المصنع

القدرة الفعلية لمحرك ثلاثي الطور تُحسب بالعلاقة: $P = \sqrt{3} \times U \times I \times \cos \varphi$. لنحسبها لمحرك يعمل تحت 400 فولط ويسحب 32 أمبير بمعامل قدرة 0.85:

```
Python
1 import math
2 U = 400
3 I = 32
4 cos_phi = 0.85
5 P = math.sqrt(3) * U * I * cos_phi
6 print("القدرة =", round(P / 1000, 2), "kW")
```

الناتج

القدرة = 18.84 kW

خلاصة الفصل

- $+$ $-$ $*$ $/$ للحساب العادي، و $//$ للقسمة الصحيحة، و $\%$ للباقي، و $**$ للأس.
- الأقواس أولاً، ثم الأس، ثم الضرب والقسمة، ثم الجمع والطرح.
- $+=$ و $-=$ اختصارات مفيدة.
- المقارنات $<=$ $<=>$ $>$ $!=$ تعطي `True` أو `False`.
- `and` و `or` و `not` لربط الشروط، و `math` للعمليات المتقدمة.

تحدي الفصل

مصباح قدرته 60 واط يعمل 5 ساعات يومياً. احسب استهلاكه الشهري (30 يوماً) بالكيلوواط ساعي، ثم تكلفته إذا كان سعر الكيلوواط ساعي 4.2 دينار. اطبع النتيجة مقربتين إلى رقمين بعد الفاصلة.

T

04

النصوص

الحروف والكلمات والجمل: كيف نصنعها ونقّصّها ونلصقها ونحوّلها

🎯 في هذا الفصل ستتعلم

- ✓ إنشاء النصوص بعلامات التنصيص
- ✓ سلاسل f السحرية لتنسيق الجمل
- ✓ قصّ أجزاء من النص
- ✓ لصق النصوص وتكرارها
- ✓ الوصول إلى الحروف بالفهرس
- ✓ أهم دوال النصوص

النص بالنسبة لي قطار من الحروف، وكل عربة لها رقم يبدأ من الصفر!



4.1 إنشاء النصوص

النص (`str`) هو أي سلسلة من الحروف بين علامتي تنصيص مزدوجتين " أو مفردتين ' ، والاثنتان سواء:

```
Python
1 city = "الجزائر"
2 lang = 'Python'
3 message = "قال المهندس: 'المحرك يعمل'"
```

وللنص الطويل الممتد على عدة أسطر نستعمل ثلاث علامات تنصيص:

```
Python
1 report = """تقرير الصيانة
2 المحرك: M-12
3 الحالة: جيدة"""
4 print(report)
```

> الناتج

تقرير الصيانة
المحرك: M-12
الحالة: جيدة

4.2 اللصق والتكرار

العلامة + تلصق النصوص ببعضها، والعلامة * تكررهما:

```
Python
1 first = "سفيان"
2 last = "مرابطي"
3 full = first + " " + last
4 print(full)
5 print("=" * 20)
6 print("ها" * 3)
```

> الناتج

سفيان مرابطي
=====

هاهاها

انتبه!

لا يمكنك لصق نص بعدد مباشرة: `العمر: " + 30` تعطي خطأ `TypeError`. حوّل العدد أولاً: `str(30) + " :العمر"`، أو استعمل الطريقة الأجمل في القسم التالي.

4.3 سلاسل f: أجمل طريقة لتركيب الجمل

ضع الحرف `f` قبل علامة التنصيص، ثم اكتب أيّ متغيّر بين قوسين معقوفين `{}` داخل النص:

```
Python
1 name = "خالد"
2 age = 30
3 salary = 85000.456
4 print(f"سنة {age} وعمري {name} اسمي")
5 print(f"بعد 5 سنوات سيكون عمري {age + 5}")
6 print(f"الراتب: {salary:.2f} دينار")
7 print(f"الراتب: {salary:,.0f} دينار")
```

الناتج

اسمي خالد وعمري 30 سنة
بعد 5 سنوات سيكون عمري 35
الراتب: 85000.46 دينار
الراتب: 85,000 دينار

التنسيق `:.2f` معناه «رقمان بعد الفاصلة»، و `:` يضع فاصلاً بين كل ثلاثة أرقام لتسهيل القراءة.

نصيحة ذهبية

اعتمد سلاسل `f` في كل برامجك، فهي أوضح وأسرع من الطرق القديمة. يمكنك أيضاً أن تكتب `{x=}` لترى اسم المتغيّر مع قيمته، وهذا مفيد جداً عند البحث عن الأخطاء: `print(f"{age=}")` تطبع `age=30`.

4.4 الفهرسة: كل حرف له رقم

كل حرف في النص له رقم يسمّى الفهرس (index)، والعَدّ يبدأ من الصفر لا من الواحد:

الحرف	P	y	t	h	o	n
الفهرس الموجب	0	1	2	3	4	5
الفهرس السالب	6-	5-	4-	3-	2-	1-

```
Python
1 word = "Python"
2 print(word[0])
3 print(word[3])
4 print(word[-1])
5 print(len(word))
```

> الناتج

```
P
h
n
6
```

الفهرس السالب يعدّ من النهاية: -1 هو آخر حرف دائمًا مهما كان طول النص. والدالة `len()` تعطي عدد الحروف.

4.5 القصّ: أخذ جزء من النص

نكتب [البداية: النهاية] لأخذ قطعة من النص، والنهاية غير محسوبة:

```
Python
1 code = "MOTOR-2026-A"
2 print(code[0:5])
3 print(code[6:10])
4 print(code[:5])
5 print(code[-1:])
6 print(code[::-1])
```

> الناتج

```
MOTOR
2026
MOTOR
A
A-6202-ROTOM
```

معلومة



[-1::] حيلة شهيرة تقلب النص كله. وصيغة القصّ الكاملة هي [البداية:النهاية:الخطوة] ، ف "123456"[:2] تعطي "135" .

4.6 دوال النصوص المفيدة

لنصوص «دوال خاصة» (تسمى توابع أو methods) نستدعيها بنقطة بعد النص:

الدالة	ماذا تفعل	المثال	النتيجة
<code>upper()</code>	حروف كبيرة	<code>"abc".upper()</code>	<code>"ABC"</code>
<code>lower()</code>	حروف صغيرة	<code>"ABC".lower()</code>	<code>"abc"</code>
<code>strip()</code>	تُحذف المسافات من الطرفين	<code>" hi ".strip()</code>	<code>"hi"</code>
<code>replace(a, b)</code>	تستبدل a بـ b	<code>"1,5".replace(", ", ".")</code>	<code>"1.5"</code>
<code>split(s)</code>	تقسم النص إلى قائمة	<code>"a-b".split("-")</code>	<code>['a', 'b']</code>
<code>startswith(s)</code>	هل يبدأ بـ s؟	<code>"M12".startswith("M")</code>	<code>True</code>
<code>count(s)</code>	كم مرة تكرر s؟	<code>"banana".count("a")</code>	<code>3</code>
<code>find(s)</code>	أين يوجد s؟	<code>"hello".find("l")</code>	<code>2</code>

وللبحث عن كلمة داخل نص نستعمل الكلمة `in` :

Python

<> python

```
1 alarm = "خطأ: ارتفاع حرارة المحرك"
2 print("حرارة" in alarm)
3 print("ضغط" in alarm)
```

> الناتج

True
False

من أرض المصنع

تحويل قراءة مكتوبة بفاصلة أوروبية إلى عدد صالح للحساب، وهي مشكلة تتكرر مع ملفات الأجهزة:

```
Python
1 reading = " 231,7 V "
2 value = float(reading.strip().replace(" V", "").replace(",", "."))
3 print(value * 2)
```

> الناتج

463.4

انتبه!

النصوص في بايثون لا تتغير (immutable). الدوال مثل `upper()` لا تغير النص الأصلي بل تعطيك نصًا جديدًا. لذا اكتب: `name = name.upper()` إن أردت حفظ النتيجة.

خلاصة الفصل

- النص بين " " أو ' ' ، والنص متعدد الأسطر بين """ """ .
- + تلصق و * تكرر، و `len()` تعطي الطول.
- f"...{x}..." أفضل طريقة لتركيب الجمل، و `:.2f` لتحديد الأرقام العشرية.
- الفهرس يبدأ من 0، و -1 آخر حرف، و `[a:b]` للقص.
- `upper` و `strip` و `replace` و `split` و `in` من أكثر الأدوات استعمالاً.

تحدي الفصل

لديك رمز قطعة غيار "pump-0457-dz". اكتب برنامجًا يطبع: نوع القطعة بحروف كبيرة (PUMP)، ورقمها كعدد صحيح (457)، ورمز البلد (DZ)، ثم جملة كاملة بسلسلة f: «القطعة PUMP رقم 457 من DZ».

05

الحوار مع المستخدم

برامج تسألك وتنتظر جوابك، ثم ترد عليك

🎯 في هذا الفصل ستتعلم

- ✓ قراءة ما يكتبه المستخدم بالدالة input
- ✓ خيارات print الخفية sep و end
- ✓ بناء أول برنامج تفاعلي حقيقي
- ✓ تحويل المدخلات إلى أعداد
- ✓ تنسيق الجداول والأرقام

حتى الآن كنت أتكلّم وحدي... الآن جاء دورك لتكلّمني!



5.1 الدالة input

تعرض `input()` سؤالاً على الشاشة، ثم **تنتظر** حتى يكتب المستخدم شيئاً ويضغط `Enter`، وتعيد ما كتبه:

```
Python <> python
name = input("ما اسمك؟ ")
print(f"أهلاً وسهلاً يا {name}!")
```

> الناتج

ما اسمك؟ مريم
أهلاً وسهلاً يا مريم!

نصيحة ذهبية

اترك مسافة في آخر نص السؤال: " ما اسمك؟"، حتى لا يلتصق جواب المستخدم بعلامة الاستفهام.

5.2 انتبه: input تعيد نصّاً دائماً!

هذا أشهر فخّ في بايثون. حتى لو كتب المستخدم رقماً، فإن `input` تعيده نصّاً:

```
Python <> python
1 a = input("العدد الأول: ")
2 b = input("العدد الثاني: ")
3 print(a + b)
```

> الناتج

العدد الأول: 5
العدد الثاني: 3

53

النتيجة 53 لا 8، لأن بايثون لصقت النصّين "5" و "3" بدل أن تجمع العددين! الحل: حوّل المدخلات إلى أعداد فوراً:

```
Python <> python
1 a = float(input("العدد الأول: "))
2 b = float(input("العدد الثاني: "))
3 print(f"المجموع = {a + b}")
```

الناتج

العدد الأول: 5
العدد الثاني: 3
المجموع = 8.0

انتبه!

إذا كتب المستخدم حروفاً بدل الأرقام، فإن `float()` تعطي خطأ `ValueError` ويتوقف البرنامج. في الفصل الثاني عشر ستتعلم كيف تلتقط هذا الخطأ بلطف وتطلب منه إعادة الإدخال.

الفهرس

5.3 أسرار الدالة print

للدالة `print` خياران مفيدان: `sep` لتحديد الفاصل بين القيم (الافتراضي مسافة)، و `end` لتحديد ما يُطبع في النهاية (الافتراضي سطر جديد):

```
1 print("2026", "10", "26", sep="/")
2 print("جارٍ التحميل", end="")
3 print("...", end="")
4 print("تم!")
```

الناتج

2026/10/26

جارٍ التحميل .. تم!

الرموز الخاصة

الرمز	معناه
<code>\n</code>	سطر جديد
<code>\t</code>	مسافة جدولة (Tab)
<code>\\</code>	الخط المائل نفسه
<code>\"</code>	علامة تنقيص داخل النص

```
print("السطر الثاني\nالسطر الأول")
```

> الناتج

السطر الأول
السطر الثاني

الفهرس

5.4 تنسيق الأرقام والجداول

داخل سلاسل f يمكنك التحكم في عرض كل قيمة ومحاذاتها:

Python

<> python

```
1 print(f"{'المعدّة':<12}{'القدرة':>8}")
2 print(f"{'ضاغط هواء':<12}{75.5:>8.1f}")
3 print(f"{'مروحة':<12}{11:>8.1f}")
4 print(f"{'كثّارة':<12}{250:>8.1f}")
5 print(f"المردود: {0.873:.1%}")
```

> الناتج

المعدّة	القدرة
ضاغط هواء	75.5
مروحة	11.0
كثّارة	250.0
المردود: 87.3%	

الصيغة	المعنى
:<12	محاذاة لليسار في عرض 12 خانة
:>8	محاذاة لليمين في عرض 8 خانات
:^10	في الوسط
:.1f	رقم واحد بعد الفاصلة
:.1%	نسبة مئوية

الفهرس

5.5 مشروع صغير: حاسبة قانون أوم

لنجمع كل ما تعلّمناه في برنامج حقيقي يحسب التيار والقدرة في مقاومة كهربائية:

```
ohm.py python
1 print("⚡ حاسبة قانون أوم ⚡")
2 print("-" * 24)
3
4 U = float(input("الجهد (فولط): "))
5 R = float(input("المقاومة (أوم): "))
6
7 I = U / R          # قانون أوم
8 P = U * I          # القدرة
9
10 print("-" * 24)
11 print(f"التيار I = {I:.3f} A")
12 print(f"القدرة P = {P:.1f} W")
```

> الناتج

⚡ حاسبة قانون أوم ⚡

الجهد (فولط): 220

المقاومة (أوم): 47

I = 4.681 A التيار

P = 1029.8 W القدرة

بايثو يقول:

مبروك! لقد كتبت برنامجًا يتحاور مع البشر. لكن جرّب أن تُدخل مقاومة تساوي صفرًا... ستري خطأ `ZeroDivisionError`. سنتعلّم كيف نمنعه في الفصل القادم!



خلاصة الفصل

- `input("سؤال")` تقرأ ما يكتبه المستخدم، وتعيده نصًا دائمًا.
- حوّل الأعداد فورًا: `float(input(...))` أو `int(input(...))`.
- `sep` و `end` تتحكّمان في شكل الطباعة، و `\n` سطر جديد.
- `<` و `>` و `:.2f` و `:.1%` لتنسيق الجداول والأرقام.

تحدي الفصل

اكتب برنامجًا يسأل عن طول غرفة وعرضها بالمتر، وعن سعر المتر المربع من البلاط، ثم يطبع فاتورة صغيرة منسقة فيها: المساحة، والسعر الإجمالي، والسعر مع إضافة 10% احتياطيًا للكسر.

06

الشروط واتخاذ القرار

كيف يختار البرنامج طريقه: إذا... وإلا...

🎯 في هذا الفصل ستتعلم

- ✓ أهمية الإزاحة في بايثون
- ✓ الجملة الشرطية if
- ✓ الشروط المركبة والمتداخلة
- ✓ else و elif لتعدد الخيارات
- ✓ الجملة match للحالات المتعددة
- ✓ الشرط في سطر واحد

البرنامج بلا شروط مثل سيارة بلا مقود: تمشي... لكن إلى الأمام فقط!



6.1 الجملة if

نستعمل `if` (إذا) لنقول للبرنامج: «نقذ هذه الأسطر فقط إذا كان الشرط صحيحًا»:

```
Python
1 temperature = 92
2
3 if temperature > 85:
4     print("إنذار: الحرارة مرتفعة!")
5     print("أوقف المحرك فورًا")
6
7 print("انتهى الفحص")
```

الناتج

إنذار: الحرارة مرتفعة!
أوقف المحرك فورًا
انتهى الفحص

لو كانت الحرارة 70 مثلاً، لتجاوزت بايثون السطرين المُزاحين، وطبعت «انتهى الفحص» فقط.

6.2 الإزاحة: قانون بايثون الذهبي

لاحظ أمرين في المثال السابق:

1 سطر `if` ينتهي بنقطتين `:`.

2 الأسطر التابعة للشرط **مُزاحة** إلى الداخل بأربع مسافات.

في أغلب اللغات الأخرى تُحدّد الكتل بالأقواس `{ }`، أما في بايثون فالإزاحة **إجبارية** وهي التي تحدّد ما ينتمي إلى الشرط وما لا ينتمي إليه.

انتبه!

لا تخلط أبداً بين المسافات وزرّ `Tab` في الملف نفسه، وإلا ظهر لك الخطأ `IndentationError`. لحسن الحظ، VS Code و IDLE يحوّلان زرّ `Tab` إلى أربع مسافات تلقائياً.

6.3 else: وإلّا...

Python

<> python

```
1 age = int(input("كم عمرك؟ "))
2
3 if age ≥ 18:
4     print("يمكنك الحصول على رخصة السياقة")
5 else:
6     print(f"انتظر {18 - age} سنوات أخرى")
```

> الناتج

كم عمرك؟ 15
انتظر 3 سنوات أخرى

6.4 elif: عندما تكثر الخيارات

`elif` اختصار لـ «else if»، أي «وإلّا إذا». تفحص بايثون الشروط بالترتيب، وتنقذ أول شرط صحيح فقط ثم تتجاهل الباقي:

Python

<> python

```
1 grade = float(input("أدخل المعدّل: "))
2
3 if grade ≥ 16:
4     print("ممتاز ★")
5 elif grade ≥ 14:
6     print("جيد جدًا")
7 elif grade ≥ 12:
8     print("جيد")
9 elif grade ≥ 10:
10    print("مقبول")
11 else:
12    print("راسب، لا تيأس")
```

> الناتج

أدخل المعدّل: 14.5
جيد جدًا

معلومة



لماذا لم تطبع بايثون «جيد» و«مقبول» أيضًا مع أن 14.5 أكبر من 12 و 10؟ لأنها توقفت عند أول شرط صحيح. لذلك ترتيب الشروط مهم: ابدأ دائمًا بالأكثر تشددًا.

6.5 الشروط المركبة

نجمع عدّة شروط بـ `and` و `or` التي تعرّفنا عليها في الفصل الثالث:

```
Python
1 voltage = 395
2 current = 48
3
4 if voltage < 360 or voltage > 440:
5     print("الجهد خارج المجال المسموح")
6 elif current > 45 and voltage > 380:
7     print("حمولة زائدة")
8 else:
9     print("كل شيء على ما يرام")
```

> الناتج

حمولة زائدة

الشروط المتداخلة

يمكن أن نضع `if` داخل `if` ، مع زيادة الإزاحة في كل مستوى:

```
Python
1 running = True
2 vibration = 7.8 # مم/ثانية
3
4 if running:
5     if vibration > 7.1:
6         print("اهتزاز خطير: برمج صيانة عاجلة")
7     else:
8         print("الاهتزاز طبيعي")
9 else:
10    print("المعدّة متوقفة")
```

الناتج

اهتزاز خطير: برمج صيانة عاجلة

نصيحة ذهبية

إذا وجدت نفسك تكتب ثلاثة مستويات من `if` المتداخلة أو أكثر، فتوقّف وفكّر: غالبًا يمكنك تبسيطها بـ `and` أو `elif`. الكود «المسطح» أسهل قراءة.

الفهرس

6.6 الشرط في سطر واحد

للقرارات البسيطة يمكن كتابة الشرط في سطر واحد:

Python

<> python

```
1 n = 7
2 kind = "زوجي" if n % 2 == 0 else "فردى"
3 print(f"العدد {n} {kind}")
```

الناتج

العدد 7 فردى

الفهرس

6.7 الجملة match

عندما نقارن متغيّرًا واحدًا بقيم كثيرة، تكون `match` أنظف من سلسلة طويلة من `elif` (متوقّرة منذ بايثون 3.10):

Python

<> python

```
1 code = input("رمز الحالة: ")
2
3 match code:
4     case "R":
5         print("● يعمل")
6     case "S":
7         print("● متوقّف")
8     case "M":
9         print("🔧 فى الصيانة")
10    case _:
11        print("رمز غير معروف")
```

> الناتج

رمز الحالة: M
في الصيانة

الحالة _ تعني «أي شيء آخر»، وهي تشبه `else`.

من أرض المصنع

نظام إنذار بسيط لفرن إسمنت، بثلاثة مستويات حسب حرارة الغلاف الخارجي:

```
Python
1 shell_temp = float(input("حرارة غلاف الفرن (°C): "))
2
3 if shell_temp >= 400:
4     level = "● خطر: احتمال سقوط البطانة الحرارية"
5 elif shell_temp >= 330:
6     level = "● تحذير: راقب المنطقة عن قرب"
7 else:
8     level = "● عادي"
9
10 print(level)
```

> الناتج

حرارة غلاف الفرن (°C): 352
● تحذير: راقب المنطقة عن قرب

ملخص الفصل

- شرط `if`: ثم أسطر مُزاحة بأربع مسافات.
- `elif` لخيارات إضافية، و `else` لكل ما تبقى، ونُنفذ أول حالة صحيحة فقط.
- الإزاحة إجبارية في بايثون وهي التي تحدّد الكتل.
- `else y if x` للقرار في سطر واحد، و `match` للحالات الكثيرة.

تحدي الفصل

اكتب برنامجًا يحسب فاتورة الكهرباء بنظام الشرائح: أول 125 كيلوواط ساعي بسعر 1.8 دينار، وما بين 126 و 250 بسعر 4.2، وما زاد على 250 بسعر 5.5. اطلب الاستهلاك من المستخدم واطبع المبلغ الإجمالي.

07

الحلقات والتكرار

دع الحاسوب يكرّر العمل الممل ألف مرة دون أن يشتكي

🎯 في هذا الفصل ستتعلم

- ✓ الحلقة for والدالة range
- ✓ الحلقة while وشرط التوقّف
- ✓ الحلقات المتداخلة
- ✓ المرور على حروف نص
- ✓ التحكم بـ break و continue
- ✓ لعبة «خمن الرقم»

أنا أحب الحلقات! أدور وأدور وأدور... حتى تقول لي: توقّف!



7.1 لماذا نحتاج الحلقات؟

تخيل أنك تريد طباعة الأعداد من 1 إلى 100. هل ستكتب `print` مئة مرة؟ بالطبع لا! الحلقة (loop) تكرر مجموعة من الأوامر عددًا من المرات نحدده نحن.

7.2 الحلقة for مع range

```
Python
for i in range(5):
    print("مرحبًا", i)
```

> الناتج

```
مرحبًا 0
مرحبًا 1
مرحبًا 2
مرحبًا 3
مرحبًا 4
```

`range(5)` تولّد الأعداد من 0 إلى 4 (خمس أعداد، والخمسة نفسها غير محسوبة). وفي كل دورة، يأخذ المتغيّر `i` العدد التالي ثم تُنفّذ الأسطر المُزاحة.

الصيغة	الأعداد المولّدة
<code>range(5)</code>	4, 3, 2, 1, 0
<code>range(1, 6)</code>	5, 4, 3, 2, 1
<code>range(0, 20, 5)</code>	15, 10, 5, 0
<code>range(10, 0, -2)</code>	2, 4, 6, 8, 10

مثال: جدول الضرب

```
Python
1 n = 7
2 for i in range(1, 11):
3     print(f"{n} × {i:>2} = {n * i:>3}")
```

الناتج >

```
7 × 1 = 7
7 × 2 = 14
7 × 3 = 21
7 × 4 = 28
7 × 5 = 35
7 × 6 = 42
7 × 7 = 49
7 × 8 = 56
7 × 9 = 63
7 × 10 = 70
```

مثال: المجموع التراكمي

نمط برمجي ستستعمله كثيرًا: نبدأ بمتغير قيمته صفر، ونضيف إليه في كل دورة:

Python

<> python

```
1 total = 0
2 for k in range(1, 101):
3     total += k
4 print("مجموع الأعداد من 1 إلى 100 =", total)
```

الناتج >

مجموع الأعداد من 1 إلى 100 = 5050

الفهرس

7.3 المرور على النصوص

يمكن لـ `for` أن تمرّ على أي «مجموعة» من العناصر، لا الأعداد فقط. مع النص، تمرّ على حروفه واحدًا واحدًا:

Python

<> python

```
for letter in "Python":
    print(letter, end=" ")
```

الناتج >

P y t h o n

7.4 الحلقة while

تستمر `while` (ما دام) في التكرار ما دام الشرط صحيحًا. نستعملها حين لا نعرف مسبقًا عدد الدورات:

```
Python
1 level = 100 # مستوى الخزّان
2 hours = 0
3 while level > 20:
4     level -= 15
5     hours += 1
6 print(f"فنعيد الملاء {level}% ساعات يصل المستوى إلى {hours} بعد")
```

> الناتج

بعد 6 ساعات يصل المستوى إلى 10% فنعيد الملاء

انتبه!

إذا لم يتغيّر شيء داخل `while` يجعل الشرط خاطئًا يومًا ما، فستدور الحلقة إلى الأبد! وإذا حدث لك ذلك، اضغط `Ctrl` `C` لإيقاف البرنامج.

7.5 continue و break

- `break` : اخرج من الحلقة فورًا.
- `continue` : تجاوز بقية هذه الدورة وانتقل إلى الدورة التالية.

```
Python
1 for n in range(1, 11):
2     if n % 2 == 0:
3         continue # تجاهل الأعداد الزوجية
4     if n > 7:
5         break # توقّف بعد 7
6     print(n, end=" ")
```

> الناتج

1 3 5 7

نمط مفيد جدًا: حلقة «لا نهائية» `while True` نخرج منها بـ `break` حين يكتب المستخدم كلمة معيّنة:

```
Python
1 while True:
2     cmd = input("أمر (اكتب خروج للإنتهاء) ")
3     if cmd == "خروج":
4         break
5     print("نقّذت:", cmd)
6 print("مع السلامة 🙌")
```

الفهرس

7.6 الحلقات المتداخلة

حلقة داخل حلقة: الداخلية تكتمل كلها في كل دورة من الخارجية:

```
Python
1 for row in range(1, 5):
2     for col in range(row):
3         print("★", end=" ")
4     print()
```

الناتج >

```
★
★ ★
★ ★ ★
★ ★ ★ ★
```

الفهرس

7.7 مشروع: لعبة «خمن الرقم»

يختار الحاسوب رقمًا عشوائيًا، وعليك أن تخمّنه بأقل عدد من المحاولات. نستعمل المكتبة `random`:

```

guess.py python
1 import random
2
3 secret = random.randint(1, 100)
4 tries = 0
5 print("اخترت رقمًا بين 1 و 100، هل تستطيع تخمينه؟")
6
7 while True:
8     guess = int(input("تخمينك: "))
9     tries += 1
10    if guess < secret:
11        print("↑ أكبر من ذلك")
12    elif guess > secret:
13        print("↓ أصغر من ذلك")
14    else:
15        print(f"🎉 أحسنت! الرقم هو {secret}، وجدته في {tries} محاولات")
16        break

```

➤ الناتج

🎲 اخترت رقمًا بين 1 و 100، هل تستطيع تخمينه؟
 تخمينك: 50
 ↑ أكبر من ذلك
 تخمينك: 75
 ↓ أصغر من ذلك
 تخمينك: 62
 🎉 أحسنت! الرقم هو 62، وجدته في 3 محاولات

بايثو يقول:

سرّ صغير: إذا بدأت دائمًا من المنتصف ثم قسمت المجال إلى نصفين في كل مرة، فستجد أي رقم بين 1 و 100 في 7 محاولات على الأكثر. هذه فكرة «البحث الثنائي» الشهيرة في علوم الحاسوب!



❏ خلاصة الفصل

- `for x in range(...)` عندما نعرف عدد الدورات، و `range(a, b)` لا تشمل `b`.
- `for` تمرّ على أي مجموعة عناصر: أعداد، حروف، قوائم...
- **شرط `while`**: تكرر ما دام الشرط صحيحًا، فاحذر الحلقة اللانهائية.
- `break` للخروج فورًا، و `continue` للانتقال إلى الدورة التالية.

تحدي الفصل

اكتب برنامجًا يطلب من المستخدم إدخال قراءات حرارة واحدة تلو الأخرى، ويتوقف عندما يكتب 0 . في النهاية يطبع:
عدد القراءات، ومتوسطها، وأعلى قراءة.



لم تكتب سطرًا برمجياً في حياتك؟ ممتاز!

هذا الكتاب كُتب لك أنت. سنبدأ معًا من تثبيت بايثون على جهازك، ثم نتعلّم المتغيرات والشروط والحلقات والقوائم والدوال والملفات والبرمجة الكائنية، خطوة خطوة، بأمثلة ملوّنة وتمارين وتحديات، حتى تبني بيدك مشروعًا حقيقيًا من عالم الصناعة.

✓ أكثر من 100 مثال جاهز للتجربة

✓ شرح بسيط بلا تعقيد

✓ ورقة مرجعية سريعة

✓ تمارين وحلول وتحديات

Dr Merabti

تأليف د. سفيان مرابطي