

Intro To Python -

عن لغة البايثون (Python)

لغة البايثون (Python) تعتبر لغة عالية المستوى High-level ، عامة الغرض general-purpose . اكتسبت شعبية كبيرة بين المطورين لعدة أسباب ، أهمها :

- سهولة التعلم.
- تعمل تقريباً في كل مكان مع جميع أنظمة التشغيل.
- دعم كبير من مجتمع المطورين.
- مفتوحة المصدر.
- استعمالاتها في الذكاء الاصطناعي وعلم البيانات.
- استعمالاتها في تطوير الأنظمة الخلفية لمشاريع الشبكة العنكبوتية مع ظهور أطر عمل قوية مثل , Django , Flask FastAPI و غيرها.

هي إحدى اللغات الديناميكية (Dynamically Typed)، التي لا تتطلب تحديد النوع و إنما توفره كجزء اختياري . على الأقل حتى الوقت الراهن .

تعتمد على جامع النفايات (Garbage Collector) في إدارة الذاكرة العشوائية و لا تتطلب من المبرمج أي تدخل يدوي لإدارتها بأغلب الحالات.

مع دعم لعدة أنماط من البرمجة ، التي تتضمن البرمجة الإجرائية (Procedural Programming) ، البرمجة الوظيفية (Functional Programming) و البرمجة المعتمدة على الأجسام (Object Oriented Programming). و مع وجود مكتبة أساسية ضخمة ، فهي تتيح للمبرمج أغلب الأدوات التي يحتاجها المشروع الذي يعمل عليه ، لتنفيذه بسرعة و بكفاءة عالية .

تم اطلاق النسخة 0.9.0 من اللغة في عام 1991 من قبل المهندس Guido Van Rossum كخليفة أو لغة مطورة من لغة الـ ABC . و في عام 2000 تم اطلاق النسخة الثانية ، وأخيراً في عام 2008 تم إطلاق النسخة الثالثة و التي تعتبر أكبر تحديث للغة منذ نشأتها الأولى .

لغة البايثون تم تصميمها بفلسفة معينة ، أبرز نقاطها :

- الجميل أفضل من القبيح.
- الصريح أفضل من الضمني.
- البسيط أفضل من المعقد.
- قابلية القراءة مهمة.

لذا فاللغة تعتمد على فكرة الإضافة عبر الوحدات الخارجية . ففي جوهرها ، هي لغة صغيرة ، بسيطة قابلة للإمتداد والإضافة عبر الوحدات الخارجية حسب احتياج البرنامج أو المشروع . كما أنها لغة سهلة القراءة ، تكتبها و كأنك تعبر عن برنامجك باللغة الإنجليزية باستخدام كلماتها المعتادة مع تقليل الاعتماد على الرموز قدر الإمكان واستبدالها بكلمات .

من هنا نجد أنها تُشجع و تحفز المطور على أن يكتب كود مُرتب و سهل القراءة . فبدلاً من استخدام الأقواس المعكوفة لبدء كتلة من الكود ، فهي تستخدم المسافات ، فمثلاً ثلاث مسافات تعني بدأ كتلة من الكود ، و الرجوع لبداية السطر تعني انتهاء تلك الكتلة .

لغة البايثون لغة مُترجمة (Interpreted) على عكس اللغات المجموعة (Compiled) . فعند تشغيل البرنامج ، يقوم المترجم بمعالجة الكود و تنفيذه. تتم هذه العملية من خلال قراءة الكود و تحويله الى bytecode و من ثم تنفيذه بواسطة آلة البايثون الافتراضية Virtual Machine ، مما يجعل من تطوير برامج البايثون أسرع و أسهل في اكتشاف أخطاء البرنامج أثناء تشغيله ، ولكن أيضاً يجعل منها أبطأ من اللغات المجموعة (compiled) مثل C, C++ و غيرها.

إعداد بيئة العمل

أولاً: تثبيت الـ Python على جهازك

بالأعم الأغلب ، جهازك يأتي مع نسخة مثبتة من البايثون . للتأكد من وجود نسخة من البايثون على جهازك ، استخدم الأمر التالي على الـ Terminal , PowerShell, Bash :


```
python --version
Or
python3 --version
Or
py --version
```

من المفترض أن تحصل على المخرج التالي (أو مماثل له) :

Python 3.8.9

في حال كانت لديك نسخة قديمة ، أو لا يوجد لديك أي نسخة يمكنك تثبيت آخر إصدار من البايثون على جهازك من خلال زيارة الرابط التالي و اتباع التعليمات كُل حسب نظام التشغيل الذي يستخدمه .

<https://www.python.org/downloads/>


 Download Python • www.python.org

ثانياً : تثبيت الـ IDE (بيئة العمل المدمجة)

لتطوير برامج البايثون على أجهزتنا ، سنستخدم الـ Visual Studio Code لكتابة الكود و ادارة مشاريعنا . و تم اختياره لشعبيته الكبيرة والدعم الكبير على مختلف الأجهزة .

من أجل تثبيت برنامج Visual Studio Code قم بالتوجه الى صفحة التحميل :

<https://code.visualstudio.com/download>

 Download Visual Studio Code - Mac, Linux, Windows • code.visualstudio.com

بعد إكمال تثبيت المحرر ، قم بتثبيت إضافة (Extension) البايثون IntelliSense المقدمة من Microsoft و ذلك من خلال قائمة الإضافات في المحرر على يسار الشاشة ، ابحث عن Python و ثبت الإضافة المقدم من Microsoft . هذه الإضافة ستوفر لنا الأدوات اللازمة للتعامل مع لغة البايثون في هذا المحرر ك تلوين الجمل ، تصحيح الأخطاء ، مستكشف الكود، و غيرها.

مقدمة في لغة البايثون (Python)

مترجم البايثون التفاعلي (Python Interactive Interpreter)

يمكنك كتابة اكواد البايثون و العمل مباشرة على الـ terminal/PowerShell/Bash من خلال استخدام مترجم البايثون التفاعلي وذلك من خلال طباعة الأمر python او python3 و الذي سيشغله لك . بعدها تستطيع كتابة أكواد البايثون كما و لو أنك تكتبها في ملف .

للخروج من المترجم التفاعلي ، اضغط CTRL-D

ملفات البايثون

لكتابة كود البايثون وتنفيذ البرنامج، نحتاج أن نضعه داخل ملف ينتهي بامتداد .py ، بعدها نستطيع تنفيذ الكود من داخل المحرر بالضغط على زر التشغيل في أعلى يمين المحرر . كما هو موضح لك .

أو من خلال الدخول على الـ Termina/PowerShell/Bash و طباعة الأمر python3 متبوعا بإسم الملف ، كما التالي (يجب أن تكون في نفس المجلد أو أن تكتب المسار الكامل للملف) :

```
python3 my_app.py
```

البيانات و أنواعها

أي برنامج على حاسوبك عبارة عن مجموعة من البيانات المخزنة في الذاكرة ، بيانات من عدة أنواع و أحجام مختلفة كالأرقام ، النصوص ، المصفوفات ، و غيرها التي تحتل مكانا في الذاكرة . فكر في الذاكرة و كأنها دولا ب كبير ، في داخل هذا الدولا ب بإمكاننا تخزين صناديق تحتوي على بيانات مختلفة . هذه الصناديق يمكن أن تكون لها أنواع و أحجام مختلفة . و لكل صندوق معرف يسمح للبرنامج بالوصول الى مكانه في الذاكرة بسرعة و قراءة البيانات التي بداخله .

فجميع البيانات في البايتون عبارة عن أجسام (objects) مخزنة في الذاكرة ، و كل جسم على الأقل يحتوي على التالي:

- نوع يعرف لنا ما يمكن له أن يفعله (مثل رقم ، نص، مصفوفة ، الخ)
- مُعرف فريد يميزه عن باقي الأجسام .
- قيمة متوافقة مع نوعه .
- عدد مرجعي لمعرفة كم مرة تم استخدام الجسم هذا.

جدول بأنواع البيانات الأساسية في البايتون

مثال	قابل للتعديل ؟	النوع	الإسم
True, False	لا	bool	قيمة منطقية
50, 24, 105	لا	int	رقم
3.14, 0.45	لا	float	رقم عائ (فاصلة)
3j , 5 + 9j	لا	complex	معقد
'Ahmed', "Mohammed"	لا	str	نص
[1,2,3,4,5]	نعم	list	قائمة
(1,2,3,4)	لا	tuple	جدول
set(["kiwi", "apple"])	نعم	set	مجموعة
{"Riyadh" : 34, "Medina" : 56, "Dammam" : 32}	نعم	dict	قاموس

المتغيرات Variables

هناك طريقتين لكتابة قيم البيانات في لغة البايثون :

- بكتابتها حرفياً .
 - بكتابتها و تعيينها لـ متغير .
- المتغيرات عبارة عن أسماء تدل على قيم معينة في مسير برنامجك. لكتابة المتغير نحتاج إسم للمتغير ، و قيمة لتعيينها لذلك الإسم .

مثال للمتغير:

```
my_var = 27
```

ففي المثال السابق ، اسم المتغير هو my_var والقيمة هي 27 . لاحظ استخدامنا لرمز التعيين = ، و ذلك لتعيين القيمة للمتغير . كما يمكننا أيضاً اختيارياً أن نحدد نوع المتغير وذلك من خلال استخدام نقطتين فوق بعض بعد اسم المتغير و من ثم نحدد النوع ، كالتالي :

```
my_var:int = 27
```

قواعد كتابة المتغير في البايثون

- يجب أن يبدأ المتغير بحرف أو شرطة تحتية .
- يمكن أن يحتوي على حروف كبيرة .
- يمكن أن يحتوي على حروف صغيرة .
- يمكن أن يحتوي على أرقام من 0 حتى الـ 9 .
- الأحرف الصغيرة و الكبيرة معتبرة فـ T غير t .
- لا يمكن أن يكون المتغير أحد الكلمات المفتاحية (قائمة لها بالأسفل)

قابلية التعديل (Mutability)

نوع البيانات المخزنة تنقسم إلى قسمين :

- بيانات قابلة للتعديل (Mutable)

- بيانات غير قابلة للتعديل (Immutable) كما يوحي الاسم ، البيانات ذات النوع القابل للتعديل ، بالإمكان التعديل على محتوى المتغير . بينما العكس غير صحيح ، فلا يمكن التعديل على محتوى البيانات ذات النوع الغير قابل للتعديل .

الكلمات المفتاحية Keywords

هي كلمات خاصة باللغة يتم استخدامه لغرض معين . هذه الكلمات لا يمكن استخدامها كأسماء للمتغيرات .
قائمة الكلمات المفتاحية في البايثون :

```
False  await  else  import  pass
None   break  except  in      raise
True   class  finally  is      return
and     continue  for    lambda  try
as      def     from    nonlocal  while
assert  del     global  not      with
async   elif    if      or       yield
```

تعيين قيمة لأكثر من متغير

كما تعلمنا آنفاً ، لتعيين قيمة لمتغير نستخدم علامة التعيين = . و من أجل تعيين قيمة واحدة لأكثر من متغير في البايثون استخدم التركيب التالي :

```
var1 = var2 = var3 = 25
```

فالآن جميع المتغيرات var1, var2, var3 جميعها أصبحت قيمتها 25 .
و لتعيين أكثر من قيمة لأكثر من متغير على نفس السطر ، نستخدم التركيب التالي:

```
var1, var2 = 35, 40
```

فالآن المتغير var1 اصبحت قيمته 35 ، و var2 أصبحت قيمته 40 .

تحويل نوع من البيانات إلى نوع آخر

تسمح لك لغة البايثون بالتحويل اغلب انواع البيانات من نوع الى آخر . فمثلا يمكننا التحويل من نص str الى رقم من خلال استخدام التركيبة التالية :

```
int("64")
```

وبطريقة مشابهة يمكنك التحويل بين العديد من أنواع البيانات و مماثلها كـ bool ، float ، وغيرها .

التعليقات (comments)

لعمل تعليقات على داخل الكود في البايثون ، نستخدم الرمز # قبل السطر المراد جعله كتعليق . و لتعليم عدة أسطر كتعليق ، نستخدم ثلاث علامات تنصيص بداية التعليق و ثلاث علامات تنصيص نهاية التعليق .

```
# One line comment
```

```
'''
```

```
Multiline
```

```
comment
```

```
'''
```