

OCTOBER 6 – 12, 2019



المبادرة السعودية للمطورين

تعلم .. فكر .. حاول .. أبداع

المبادرة السعودية للمطورين

مسار Python

مشرقي المسار:

عبدالله عوده – انتصار النصار – رؤى كردي – ليلى المصعبي



## ملاحظات قبل بدء الدروس:

- على المتدربين نشر كل يوم الجزئية التي تم كتابتها من النص البرمجي في الـ **Github** تحت **Topic** بعنوان **saudidev.org** كما تم توضيحه في دروس الـ **Github** سابقاً

- على المتدربين نشر كل يوم مقدار التقدم وصورة لما تم تعلمه وتطبيقه على **Twitter** تحت الهاشتاقات:  
#المبادرة\_السعودية\_للمطورين  
\_100#يوم\_برمجة  
#100DaysOfCode

تمنياتنا لك بالتوفيق  
المبادرة السعودية للمطورين

# اليوم الثامن والأربعون

# المجال في لغة البايثون

## Python Scope

المجال هو المكان/المنطقة في البرنامج الذي يمكن استعمال المتغير فيها، يكون المتغير معروف في هذا المجال/المنطقة  
A variable is only available from inside the region it is created. This is called **scope**.

## ➤ Local Scope

### المجال المحلي

عند إنشاء متغير داخل دالة **function** ما، يكون مجال هذا المتغير محلي داخل هذه الدالة، واستعماله داخل الدالة فقط  
A variable created inside a function belongs to the local scope of that function, and can only be used inside that function.

**Example** المتغير الذي يتم إنشائه وتعريفه داخل الدالة يكون متاحا داخل الدالة ومجاله محلي (داخل الدالة)

A variable created inside a function is available inside that function.

```
def myfunc():
    x = 300
    print(x)
```

myfunc()

في هذا المثال قمنا بإنشاء دالة **myfunc**  
وتم تعريف المتغير **x** داخل الدالة  
هذا المتغير هو في المجال المحلي لأنه يقع ضمن الدالة

```
Python (Intel Type)
>>> def myfunc():
>>>     x = 300
>>>     print(x)
>>>
>>> == RE: myfunc()
>>> 300
>>>
```

المجال المحلي

نتيجة تشغيل الكود

## دالة بداخل دالة أخرى

### ➤ Function Inside Function

As explained in the example above, the variable `x` is not available outside the function, but it is available for any function inside the function.

كما هو موضح في المثال السابق المتغير `x` مجاله أو نطاقه يكون داخل الدالة فقط لا خارجها لكنه يكون متاحا للدالة التي تكون داخل الدالة التي تحتوي على هذا المتغير، لأنها تعتبر ضمن نطاقها انظر المثال التالي:

### Example

يمكنك الوصول إلى المتغير المحلي من دالة بداخل دالة أخرى

The local variable can be accessed from a function within the function.

```
def myfunc():
    x = 300
    def myinnerfunc():
        print(x)
    myinnerfunc()
```

myfunc()

في هذا المثال قمنا بإنشاء دالة `myfunc`

وتم تعريف المتغير `x` داخل الدالة

ثم قمنا بتعريف دالة `myinnerfunc` داخل الدالة `myfunc`

المتغير `x` هو في المجال المحلي لأنه يقع ضمن حدود الدالة `myfunc`

والدالة `myinnerfunc` أيضا تقع ضمن حدود الدالة `myfunc`

لذلك تستطيع الوصول إلى المتغير `x`

```
Python3
(Interactive Shell)
Type:
>>>
== RE
300
>>> myfunc()
```

نتيجة تشغيل الكود

## ➤ Global Scope

### المجال العام

المتغيرات التي يتم تعريفها وإنشائها داخل كود **البايثون** بشكل عام، تكون في المجال العام للكود وتسمى بالمتغيرات العامة أي يمكن الوصول إليها مباشرة في أي مكان في الكود.

A variable created in the main body of the **Python** code is a **global variable** and belongs to the **global scope**.

المتغيرات العامة تكون متاحة لجميع المجالات، العامة والمحلية

وهي المتغيرات التي يتم تعريفها خارج الدوال، ويمكنك الوصول إليها من أي مكان، حتى من داخل الدوال

**Global variables** are available from within any **scope**, global and local.

## Example

A variable created outside of a function is **global** and can be used by anyone.

```
x = 300
```

في هذا المثال تم تعريف المتغير **x** خارج الدالة

```
def myfunc():  
    print(x)
```

ثم قمنا بإنشاء دالة **myfunc** تقوم بطباعة قيمة المتغير **x** الذي تم تعريفه خارجها

```
myfunc()
```

قمنا باستدعاء الدالة وستقوم بطباعة قيمة المتغير **x**

```
print(x)
```

ثم قمنا بطباعة قيمة المتغير **x** الموجود خارج الدالة

المتغير **x** هو في المجال العام

لذلك تستطيع الوصول إلى المتغير **x** من أي مكان في الكود بطريقة مباشرة

```
Python x = 300
```

```
(Inter
```

```
Type
```

```
>>>
```

```
== RE
```

```
300
```

```
300
```

```
>>>
```

```
>>>
```

```
def myfunc():  
    print(x)
```

المجال المحلي

المجال العام

```
myfunc()
```

```
print(x)
```

محلي - local

عام - global

نتيجة تشغيل الكود

أتممت درسك بنجاح!

روابط قد تهمك

Useful links

- [Scope](#)
- [حياة المتغير Python tutorial Variables Scope](#)
- [Global and Local Variables in Python](#)

طبّق ما تعلمته في هذا الدرس

ولا تنسى مشاركتنا أكوادك



# اليوم التاسع والأربعون

# المجال في لغة البايثون

## **Python** Scope 2

## ➤ Naming Variables

## تسمية المتغيرات

If you operate with the same variable name inside and outside of a function, **Python** will treat them as two separate variables, one available in the **global scope** (outside the function) and one available in the **local scope** (inside the function).

في حال قمت بتعريف متغير ويحمل نفس الاسم داخل وخارج الدالة فإن بايثون تستطيع التفريق بينهما، بأن المتغير الذي في المجال العام (هو الموجود خارج الدالة) والمتغير الذي في المجال المحلي ( هو الموجود داخل الدالة)

### Example

The function will print the **local x**, and then the code will print the **global x**.

```
x = 300 ← متغير عام - global

def myfunc():
    x = 200 ← متغير محلي - local
    print(x)
```

```
myfunc()
```

الدالة ستقوم بطباعة المتغير x الذي هو في المجال المحلي (داخل الدالة)

```
print(x)
```

والكود سيقوم بطباعة المتغير x الذي هو في المجال العام (خارج الدالة)

```
Python x = 300
(Intel
Type '
>>> def myfunc():
      x = 200
      print(x)
== RES myfunc()
200
300
>>> print(x)
```

نتيجة تشغيل الكود

## ➤ Global Keyword

الكلمة المفتاحية/المحجوزة **global**

يمكنك الوصول واستعمال المتغيرات المحلية لكن بطريقة غير مباشرة، ليس كما في المتغيرات العامة التي تستطيع الوصول لها مباشرة عليك باستخدام الكلمة المفتاحية **global** قبل اسم المتغير للوصول للمتغيرات المحلية في حال أردت التعامل معها

If you need to create a **global variable**, but are stuck in the **local scope**, you can use the **global** keyword. The **global** keyword makes the variable **global**.

### Example

If you use the **global** keyword, the variable belongs to the **global scope**.

```
def myfunc():
    global x
    x = 300
```

في المثال التالي استخدمنا الكلمة المفتاحية **global** قبل المتغير المحلي **x** وذلك لجعله في المجال العام ونتمكن من استعماله

```
myfunc()
```

```
print(x)
```

```
Python (Interpreter)
Type:
>>>
== RE myfunc()
300
>>> print(x)
```

نتيجة تشغيل الكود

جرب :

طبق هذا المثال ثم قم بحذف الكلمة المفتاحية **global** وانظر ماذا يحدث .. سيظهر لك **NameError** لأنك لا تستطيع الوصول إلى المتغير المحلي مباشرة

أيضا يمكنك استخدام الكلمة المفتاحية **global** لتغيير قيمة المتغير العام داخل دالة  
أي أن المتغير موجود أساسا خارج الدالة (متغير عام) لكنك في داخل الدالة تريد تغيير قيمته

Also, use the **global** keyword if you want to make a change to a **global variable** inside a function.

## Example

To change the value of a **global variable** inside a function, refer to the variable by using the **global** keyword.

لتغيير قيمة المتغير العام داخل دالة قم بكتابة الكلمة المفتاحية **global** ثم ألقها باسم المتغير

`x = 300` ← متغير عام - global

في المثال التالي:

```
def myfunc():
```

الدالة **myfunc** ستقوم بتغيير قيمة المتغير **x** الذي تم تعريفه خارج الدالة

```
    global x
```

`x = 200` ← تغيير قيمة المتغير العام x

```
myfunc()
```

```
print(x)
```

```
Python x = 300
(Intel
Type
>>> def myfunc():
== RE:     global x
200         x = 200
>>> myfunc()
>>> print(x)
```

نتيجة تشغيل الكود

قمنا بطباعة القيمة الجديدة للمتغير

الموجودة داخل الدالة

أتممتَ درسك بنجاح!

تابع التقدّم

روابط قد تساعدك

Check the links below

- [Global and Local variables](#)
- [Local and Global variable :بايثون 52](#)
- [Learn Python in Arabic #97 - global var inside class Python متغيرات عامة](#)
- [#36 Python Tutorial for Beginners | Global Keyword in Python | Global vs Local Variable](#)

طبّق ما تعلمته في هذا الدرس

ولا تنسى مشاركتنا أكوادك

# اليوم الخمسون

# الوحدات في لغة البايثون

## Python Modules



## ماهي الوحدة

### ➤ What is a Module?

الوحدة (الموديول) تشبه المكتبة، فهي تعتبر ملف بايثون عادي يحتوي على مجموعة من أكواد بايثون (دوال، أصناف، متغيرات ...) ويمكن إعادة استخدامها وتضمينها في برنامجك بمجرد قيامك باستيرادها

Consider a **module** to be the same as a code library. A file containing a set of functions you want to include in your application.

أي ملف يكون امتداده **.py** يمثل وحدة في بايثون

الوحدة تجعل الكود متاح لك لإعادة استخدامه ويمكنك كتابة وحدتك الخاصة، حيث أن الكود يتم وضعه في ملف معين ويمكنك استخدام هذا الكود بمجرد استيرادك للملف، فهذا يساعدك في بناء وترتيب برنامجك حيث يمكنك تقسيم مشروعك إلى وحدات ليسهل عليك

### ➤ Create a Module

To create a **module** just save the code you want in a file with the file extension **.py**

لإنشاء وحدة قم بحفظ الكود الذي تريد في ملف يكون امتداده **.py** حيث أن اسم الوحدة سيكون هو نفس اسم الملف الذي قمت بإنشائه

## Example

Save this code in a file named **mymodule.py**

```
def greeting(name):  
    print("Hello, " + name)
```

في مثالنا هنا

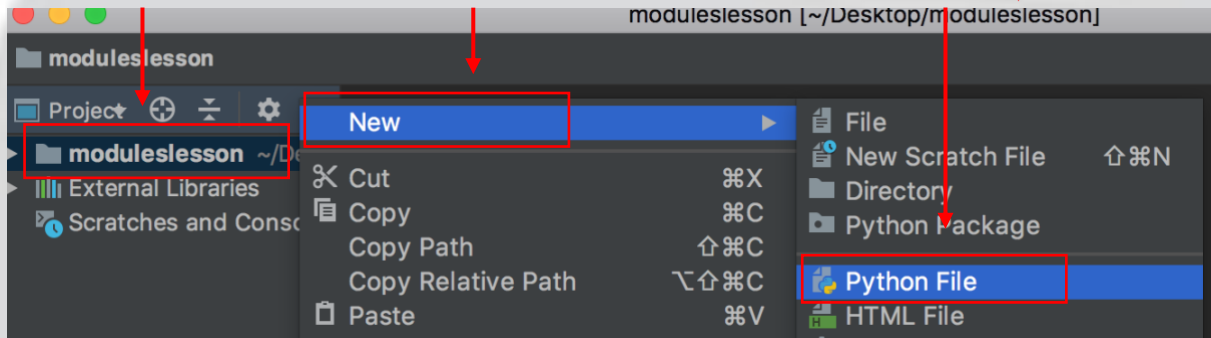
قمنا بحفظ هذا الكود والذي يحتوي فقط على دالة **greeting** في ملف امتداده **.py** باسم **mymodule**

( تم التطبيق هنا على برنامج PyCharm ) إليك الخطوات :

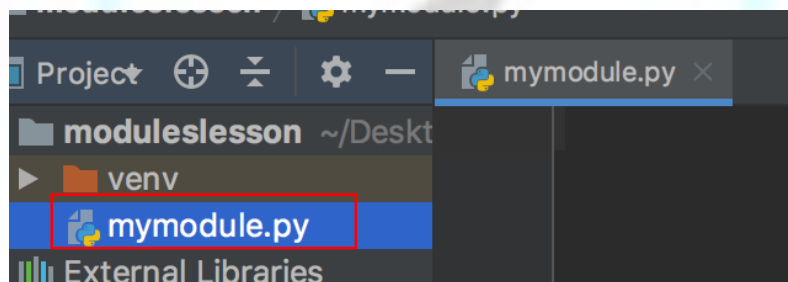
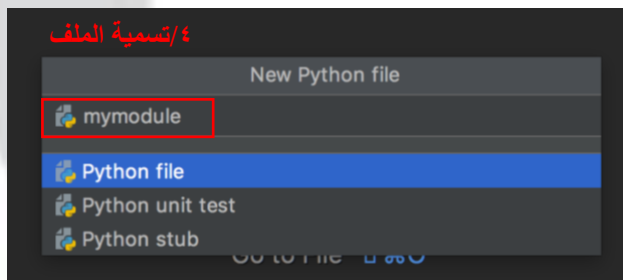
١/ انقر على المشروع الذي أنشأته بالزر الأيمن

٢/ ثم اختر جديد

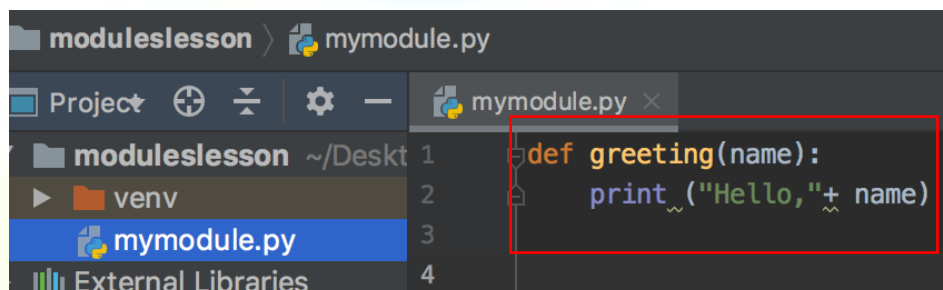
٣/ ثم اختر ملف بايثون



٤/ تسمية الملف



تم إنشاء وحدة جديدة باسم **mymodule.py**  
داخل المشروع الذي تعمل عليه



الآن قم بكتابة كود الدالة  
داخل هذا الملف/الوحدة

## استخدام الوحدة

### ➤ Use a Module

Now we can use the **module** we just created, by using the **import** statement.

لاستخدام الوحدة التي قمنا بإنشائها استخدم الكلمة المحجوزة **import** والتي معناها هو الاستيراد ثم ألحقها باسم الوحدة تستخدم هذه الكلمة المحجوزة لتضمين وحدة في وحدة أخرى

### Example

نقوم باستيراد الوحدة التي قمنا بإنشائها في المثال السابق باستخدام **import**

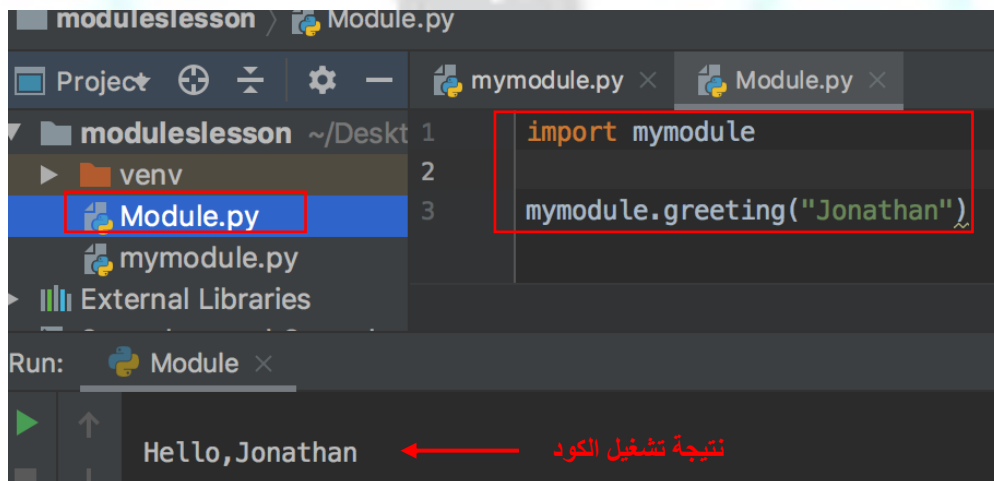
Import the **module** named **mymodule**, and call the greeting function.

في المثال التالي قمنا باستيراد الوحدة **mymodule** التي تحتوي على دالة واحدة وهي دالة الترحيب لاحظ أن اسم الوحدة هو نفس اسم الملف

```
import mymodule
```

```
mymodule.greeting("Jonathan")
```

نقوم بإنشاء ملف آخر ليتم فيه تنفيذ الأوامر وتضمين/استيراد الوحدة **mymodule** فيه سيتم تضمين محتوى الوحدة **mymodule** في الوحدة الأخرى **Module** التي تم فيه الاستيراد



**Note:** When using a function from a **module**, use the **syntax: module\_name.function\_name**.

عند استخدامك لدالة موجودة داخل وحدة، تكون طريقة استدعائها كالتالي: اسم الوحدة متبوعة بنقطة ثم اسم الدالة

## ➤ Variables in Module

## المتغيرات في الوحدة

The **module** can contain functions, as already described, but also variables of all types (arrays, dictionaries, objects etc).

الوحدة تحتوي على مجموعة من أكواد بايثون (دوال، أصناف، كل أنواع المتغيرات، القواميس ...)

### Example

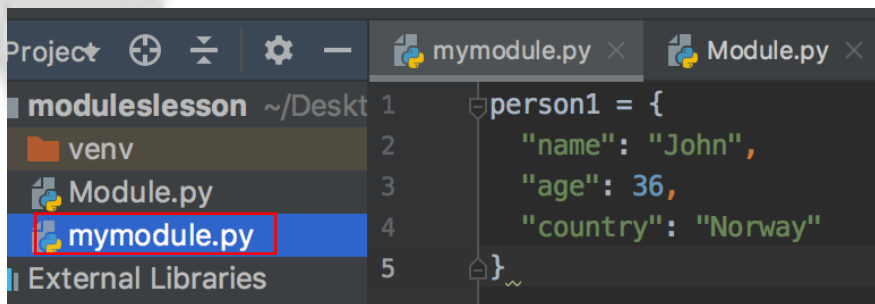
Save this code in the file **mymodule.py**

```
person1 = {
    "name": "John",
    "age": 36,
    "country": "Norway"
}
```

في هذا المثال

قمنا بإنشاء ملف باسم **mymodul**

وتمت فيه كتابة هذا الكود الذي يحتوي على القاموس **person1**



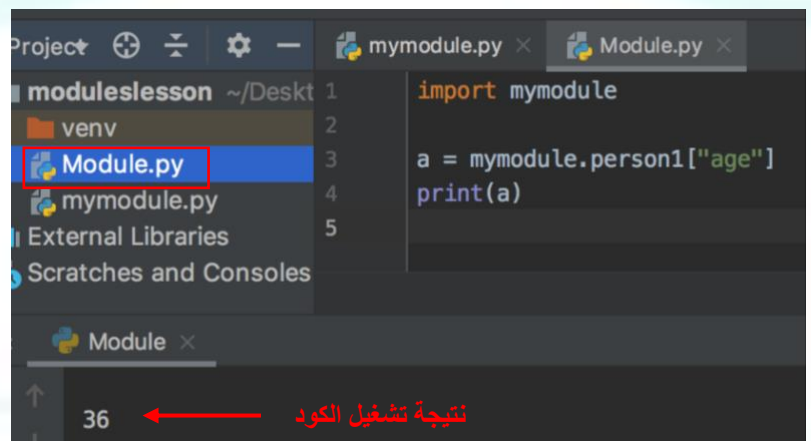
### Example

Import the **module** named **mymodule**, and access the **person1** dictionary.

في هذا المثال: قمنا باستيراد محتوى الوحدة **mymodul** في الوحدة التي يتم فيها تنفيذ الأوامر أو الـ **Run** وهي **Module**

```
import mymodule

a = mymodule.person1["age"]
print(a)
```



مبرمج الغد!  
أتممت درسك

روابط قد تهتمك

Useful links

- [Python modules](#)
- [الوحدات - Modules 6 - تعلم البرمجة بلغة بايثون](#)
- [موديل 35- Python OOP|| Create new Model](#)

طبّق ما تعلمته في هذا الدرس  
ولا تنسى مشاركتنا أكوادك

# اليوم الواحد والخمسون

## الوحدات في لغة البايثون

### **Python** Modules 2

## ➤ Naming a Module

### تسمية الوحدة

You can name the **module file** whatever you like, but it must have the file extension **.py**

يمكنك تسمية ملف الوحدة بأي اسم تريد، لكن يجب أن يكون امتداده **.py**

## ➤ Re-naming a Module

### إعادة تسمية الوحدة

You can create an **alias** when you import a **module**, by using the **as** keyword.

يمكنك إعادة تسمية الوحدة عند تضمينها في بايثون، وذلك بإعطاء الوحدة اسم آخر عند تضمينها لمناداته به بدل مناداته بالاسم الأصلي، بواسطة الكلمة المحجوزة **as** هذه الطريقة تعتبر كأنك قمت باختصار أسماء الوحدات عند استيرادها/تضمينها

## Example

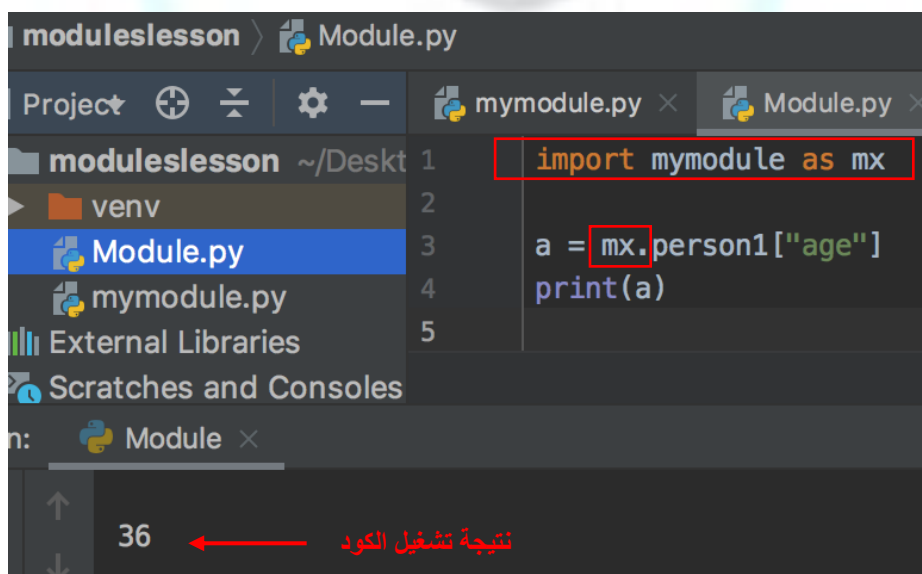
Create an **alias** for **mymodule** called **mx**.

في هذا المثال

```
import mymodule as mx

a = mx.person1["age"]
print(a)
```

قمنا بتضمين الوحدة **mymodule** مع إعطاء الوحدة اسم آخر مختصر وهو **mx**



```
moduleslesson > Module.py
Project: moduleslesson ~/Desktop
venv
Module.py
mymodule.py
External Libraries
Scratches and Consoles
n: Module
36 ← نتيجة تشغيل الكود
```



## الوحدات الجاهزة

### ➤ Built-in Modules

There are several **built-in modules** in **Python**, which you can import whenever you like.

في بايثون يوجد عدة وحدات جاهزة للاستخدام يمكنك الاستفادة منها وذلك باستيرادها وتضمينها في كودك

### Example

Import and use the **platform** module.

في هذا المثال سنقوم باستخدام الوحدة الجاهزة **platform**

```
import platform

x = platform.system()
print(x)
```

قمنا بتضمين الوحدة الجاهزة **platform**

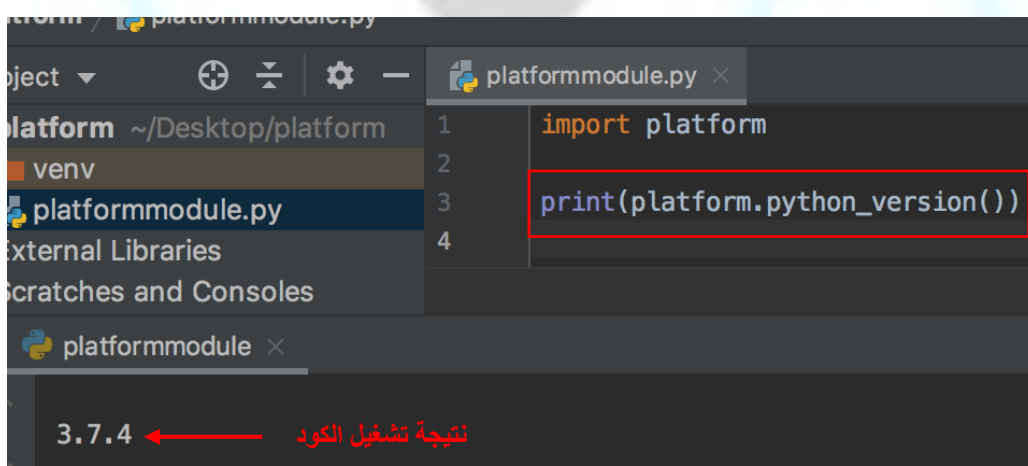
ومن خلالها استطعنا معرفة خصائص الجهاز الذي نستخدمه

هنا أردنا معرفة نوع نظام التشغيل بالتالي قمنا باستدعاء الدالة **system()**

```
C:\Users\My Name>python demo_module4.py
Windows
```

نتيجة تشغيل الكود

وهنا أردنا معرفة إصدار البايثون المثبت على الجهاز



```
platformmodule.py
1 import platform
2
3 print(platform.python_version())
4
```

3.7.4

نتيجة تشغيل الكود

يمكنك الاطلاع أكثر على وحدة **platform** من خلال هذا الرابط

[platform — Access to underlying platform's identifying data](#)

## ➤ Using the dir() Function

استخدام الدالة الجاهزة **dir()**

There is a **built-in function** to list all the function names (or variable names) in a **module**.

The **dir()** function.

دالة الاستخراج **dir()** تُستخدم لمعرفة أسماء الدوال أو المتغيرات الموجودة في الوحدة

## Example

List all the defined names belonging to the **platform module**.

```
import platform  
  
x = dir(platform)  
print(x)
```

في المثال التالي

قمنا بطباعة واستخراج جميع الأشياء الموجودة في الوحدة **platform**

باستخدام الدالة **dir()**



**Note:** The **dir()** function can be used on *all* **modules**, also the ones you create yourself.

دالة الاستخراج **dir()** تُستخدم على جميع الوحدات، حتى الوحدات التي تقوم بتعريفها بنفسك

## ➤ Import From Module

استيراد جزء من الوحدة باستخدام **From**

You can choose to import only parts from a **module**, by using the **from** keyword.

يمكنك استيراد جزء معين من الوحدة وليس كامل الوحدة، كـ استيراد دالة محددة بمفردها فقط

وذلك باستخدام الكلمة المحجوزة **from** في بايثون

## Example

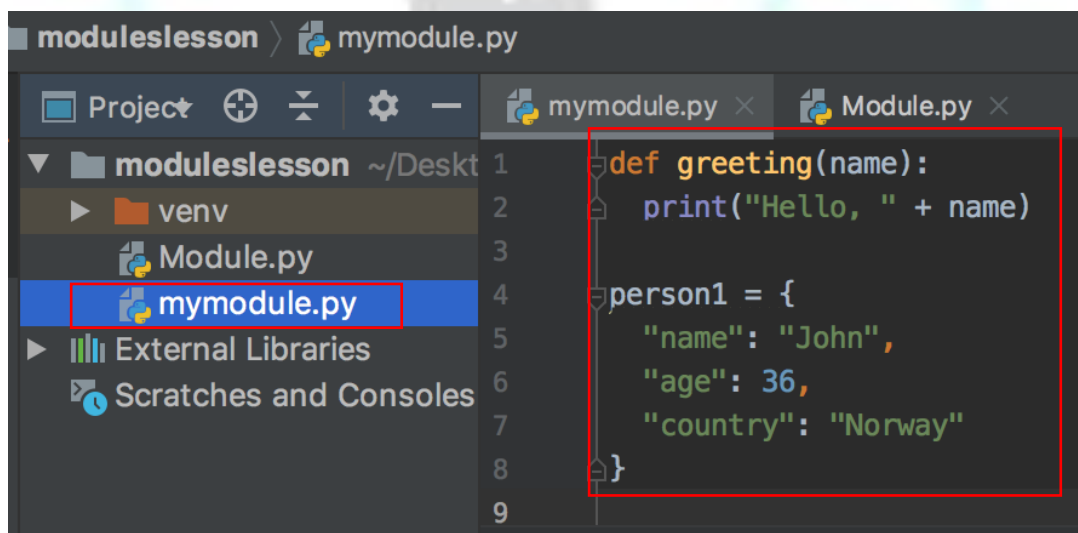
The **module** named **mymodule** has one **function** and one **dictionary**.

```
def greeting(name):  
    print("Hello, " + name)  
  
person1 = {  
    "name": "John",  
    "age": 36,  
    "country": "Norway"  
}
```

في المثال التالي

قمنا بإنشاء وحدة باسم **mymodule**

تحتوي على دالة **greeting** وقاموس **person1**



```
moduleslesson > mymodule.py  
Project  
moduleslesson ~/Desktop  
venv  
Module.py  
mymodule.py  
External Libraries  
Scratches and Consoles  
1 def greeting(name):  
2     print("Hello, " + name)  
3  
4 person1 = {  
5     "name": "John",  
6     "age": 36,  
7     "country": "Norway"  
8 }  
9
```

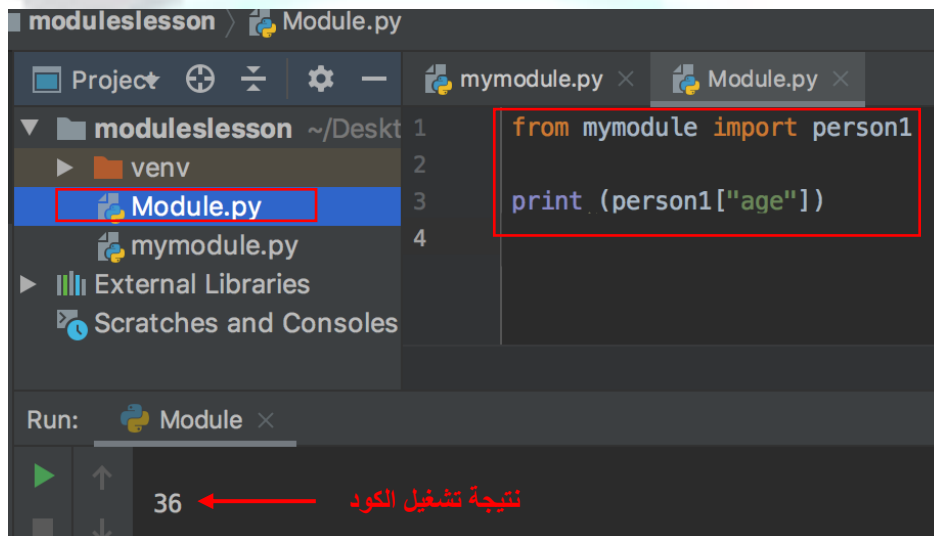
نريد فقط استيراد القاموس `person1` من الوحدة `mymodule`

Import only the `person1` dictionary from the `module`.

```
from mymodule import person1
print(person1["age"])
```

نضع الكلمة المحجوزة `from` ثم بعدها اسم الوحدة `mymodule` ثم كلمة الاستيراد المحجوزة `import` بعدها نذكر اسم ما نريد استيراده من الوحدة وهو هنا القاموس `person1`

وإذا أردت استيراد أكثر من دالة أو أكثر من شيء، افصل بينهم بفواصل



**Note:** When importing using the `from` keyword, do not use the `module` name when referring to elements in the `module`. Example: `person1["age"]`, **not** `mymodule.person1["age"]`

عند استيراد عناصر من الوحدة باستخدام `from` فإننا نستخدمها بشكل مباشر دون الحاجة لذكر اسم الوحدة التي تم الاستيراد منها

أتممت درسك بنجاح!  
واصل التعلم

روابط قد تهمك

Useful links

- [17 - Python - Beginners Tutorial - Modules](#)
- [التعرف علي الموديول Python in Arabic شرح بايثون بالعربي - 81 Learn Python in Arabic Module Python](#)
- [#45 Python Tutorial for Beginners | Modules](#)
- [مفهوم الـ Python tutorial import](#)
- [Modules - تعلم بايثون 3 13 #](#)
- [Python Module Index](#)
- [Python Modules Part 1 | Arabic](#)

طبق ما تعلمته في هذا الدرس  
ولا تنسى مشاركتنا أكوادك

# اليوم الثاني والخمسون

## التاريخ والوقت في لغة البايثون

### **Python** Datetime

## ➤ Python Dates

## التواريخ في بايثون

لمعرفة الوقت فإن بايثون تحتوي على وحدة جاهزة وهي **datetime** للتعامل مع التاريخ بكل سهولة

A **date** in **Python** is not a data type of its own, but we can import a **module** named **datetime** to work with **dates** as **date objects**.

الوحدة **datetime** تحتوي على مجموعة من الأصناف **classes** فيها دوال جاهزة وذلك للتعامل مع التاريخ والوقت

الأصناف هي :

الصنف **date** ويحتوي على مجموعة دوال للتعامل مع التاريخ

الصنف **time** ويحتوي على مجموعة دوال للتعامل مع الوقت

الصنف **datetime** و يحتوي على مجموعة دوال للتعامل مع التاريخ والوقت

الصنف **timedelta** ويحتوي على مجموعة دوال لحساب الفرق بين تاريخ وتاريخ آخر بدقة

الصنف **timezone** ويحتوي على مجموعة دوال لحساب فرق التوقيت بين تاريخ وتاريخ آخر حسب المنطقة الزمنية لكل تاريخ



## Example

Import the **datetime** module and display the current **date**.

في المثال التالي

قمنا باستيراد الوحدة **datetime** لعرض التاريخ والوقت الحالي

وذلك باستخدام الدالة **now()** الموجودة في الصنف **datetime**

الموجود بداخل الوحدة **datetime**

**datetime** هو صنف موجود في الوحدة **datetime** وهذا الصنف يحتوي على عدة دوال

اسم الوحدة

```
import datetime
```

```
x = datetime.datetime.now()
```

```
print(x)
```

سترجع الدالة **now()** التاريخ والوقت الحالي

ككائن من الصنف **datetime** ويتم تخزينه في **x**

اسم الوحدة

```
datetime x
2019-10-03 23:26:04.077070
```

نتيجة تشغيل الكود

**ملاحظة:** في بعض إصدارات بايثون الجديدة قد تُكتب الوحدة بهذا الشكل **\_datetime**

## عرض التاريخ

### ➤ Date Output

When we execute the code from the example above the result will be:

**2019-10-03 23:26:04.077070**

The **date** contains year, month, day, hour, minute, second, and microsecond.

The **datetime** module has many methods to return information about the **date** object.

التاريخ شمل المعلومات التالية مرتبة: السنة، الشهر، اليوم، الساعة، الدقيقة، الثانية، أجزاء الثانية القادمة

بالتالي الصنف **datetime** أتاح لنا الوقت والتاريخ بدقة عالية

والوحدة **datetime** تحتوي على عدة دوال جاهزة لإرجاع المعلومات والتعامل مع الكائن

### Example

Return the **year** and **name** of weekday.

```
import datetime
```

```
x = datetime.datetime.now()
```

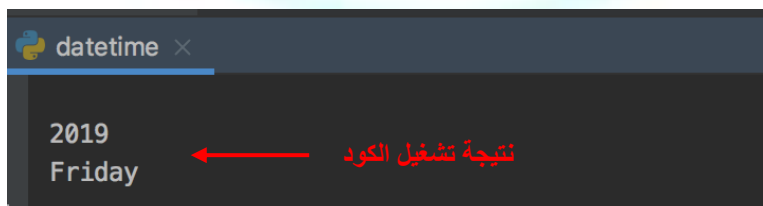
```
print(x.year)
```

```
print(x.strftime("%A"))
```

في هذا المثال قمنا بعرض السنة واليوم

الدالة **strftime()** موجودة في الصنف **datetime**

ستتعرف عليها لاحقاً في هذا الدرس



```
datetime x
2019
Friday
```

نتيجة تشغيل الكود

## إنشاء كائن

### ➤ Creating Date Objects

To create a **date**, we can use the **datetime()** class (constructor) of the **datetime** module.

The **datetime()** class requires three parameters to create a **date**: year, month, day.

وضحنا سابقاً أن الوحدة **datetime** تحتوي على الصنف **datetime** بالتالي يمكنك إنشاء كائن من هذا الصنف مع تحديد التاريخ الذي تريد مباشرةً عند إنشائه

الصنف **datetime** يحتوي على الـ constructor دالة البناء **datetime()**

القيم المسموح تمريرها للـ parameters هي :

**year** رقم السنة وتكون قيمته عدد صحيح

**month** رقم الشهر وتكون قيمته عدد صحيح

**day** رقم اليوم وتكون قيمته عدد صحيح

**hour** رقم الساعة وتكون قيمته عدد صحيح

**minute** رقم الدقائق وتكون قيمته عدد صحيح

**second** رقم الثواني وتكون قيمته عدد صحيح

**microsecond** رقم أجزاء الثواني وتكون قيمته عدد صحيح

**fold** وتكون قيمته عدد صحيح

وعند إنشائك للكائن يجب عليك إدخال ثلاث قيم (إجبارية) وهي **year – month – day** مكان المُعاملات لهذا الـ constructor لأنه لم يتم إعطائها قيم افتراضية، أما بقية الـ parameters الموجودة في دالة البناء constructor فتحدد قيمهم أمر اختياري لأنه قد تم إعطائهم قيماً افتراضية بـ 0 أو None

The **datetime()** class also takes parameters for **time** and **timezone** (hour, minute, second, microsecond, tzzone), but they are optional, and has a default value of 0, (None for timezone).

في هذا المثال

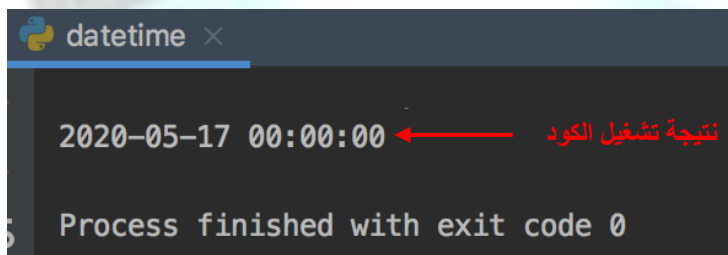
قمنا بإنشاء كائن من الصنف `datetime`

مع تحديد التاريخ معين نريد تخزينه فيه مباشرة عند إنشائه

## Example

Create a **date** object

```
import datetime  
x = datetime.datetime(2020, 5, 17)  
print(x)
```



```
datetime ×  
2020-05-17 00:00:00 ← نتيجة تشغيل الكود  
Process finished with exit code 0
```

## ➤ The strftime() Method

دالة لتحديد فورمات للتاريخ والوقت

The **datetime** object has a method for formatting **date objects** into readable strings.

The method is called **strftime()**, and takes one parameter, **format**, to specify the format of the returned string.

الكائن **datetime** لديه دالة جاهزة لتحديد التاريخ والوقت وعرضه وتخزينه بأشكال مختلفة

دالة الفورمات **strftime()** نقوم باستدعائها من الكائن **datetime** ثم نمرر لها مُعاملاً واحداً وهو عبارة عن رمز لتحديد الأشياء التي نريد إرجاعها كنص

## Example

في هذا المثال

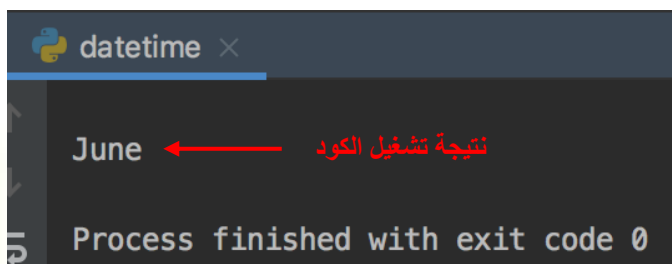
Display the name of the month.

١/ قمنا باستيراد الوحدة **datetime**

٢/ إنشاء كائن من الصنف **datetime** يمثل تاريخ معين وتم تخزينه في **x**

٣/ قمنا باستدعاء الدالة **strftime()** وتمرير الرمز **%B** لها كنص حتى تقوم بترجيع اسم الشهر الذي تم تخزينه في **x**

```
import datetime ← ١
x = datetime.datetime(2018, 6, 1) ← ٢
print(x.strftime("%B")) ← ٣
```



الرموز التي يمكنك استخدامها في الدالة **strftime()** لتحديد شكل التاريخ والوقت

A reference of all the legal format codes

Directive	Description	Example
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%d	Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12
%y	Year, short version, without century	18
%Y	Year, full version	2018
%H	Hour 00-23	17
%I	Hour 00-12	05
%p	AM/PM	PM
%M	Minute 00-59	41
%S	Second 00-59	08
%f	Microsecond 000000-999999	548513
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00-53	52
%W	Week number of year, Monday as the first day of week, 00-53	52
%c	Local version of date and time	Mon Dec 31 17:41:00 2018
%x	Local version of date	12/31/18
%X	Local version of time	17:41:00
%%	A % character	%

رائع!  
أتممت درسك الأخير لهذا الأسبوع

روابط قد تهتمك للاستفادة فقط

### Useful links

- [datetime — Basic date and time types](#)
- [الوقت والتاريخ في بايثون](#)
- [Learn Python in Arabic #65 - طباعة التاريخ و الوقت - print Date And Time Python](#)
- [Learn Python in Arabic #66 - تخصيص التاريخ و الوقت - custom Date And Time Python](#)
- [Learn Python in Arabic #67 - تهيئة التاريخ و الوقت - format Date And Time Python](#)
- [Python Date&Time 131 التاريخ والوقت](#)
- [مقدمه عن التعامل مع الوقت والتاريخ في بايثون 16-01](#)
- [Datetime Module \(Dates and Times\) || Python Tutorial || Learn Python Programming](#)
- [strftime and strptime | Python Datetime | Python30 | Day 3](#)
- [Python Tutorial: Datetime Module - How to work with Dates, Times, Timedeltas, and Timezones](#)
- [Datetime Python Programming Tutorial](#)

طبّق ما تعلمته في هذا الدرس  
ولا تنسى مشاركتنا أكوادك

# اليوم الثالث والخمسون & اليوم الرابع والخمسون



## تحدي الأسبوع (يتم حله ورفعته على Github)

### التحدي الأول

قُم بإنشاء وحدة (**module**) تتضمن العمليات الأساسية للحساب: الجمع، الطرح، الضرب، القسمة  
ثم قُم باستيراد الوحدة السابقة وقُم بالعمليات التالية، ثم اطبع النواتج :

- $8 + 1$  جمع
- $4 - 2$  طرح
- $6 * 6$  ضرب
- $2 / 8$  قسمة

### التحدي الثاني

باستعمال (**datetime module**) اطبع الآتي:

- السنة
- الوقت والتاريخ لليوم
- الشهر
- اليوم

### تحدي إضافي - لست مُلزمًا بحله -

باستعمال (**datetime module**) اطبع تاريخ اليوم السابق وتاريخ الغد .. يا مبرمج الغد!

## موفق دومًا

انتظرنا في دروس الأسبوع القادم