

SEPTEMBER 22 – 28, 2019



المبادرة السعودية للمطورين

تعلم .. فكر .. حاول .. أبداع

المبادرة السعودية للمطورين

مسار Python

مشرفي 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 الرابع والثلاثون

# الدوال في لغة البايثون

## Python Functions 2

## ➤ Passing a List as a Parameter

## تمرير القوائم كـ مُعامل للدالة

You can send any data types of parameter to a **function** (string, number, list, dictionary etc.), and it will be treated as the same data type inside the **function**.

المُعاملات التي نمررها للدالة عند استدعائها يمكن أن تكون أي نوع من أنواع البيانات (نصوص، أرقام، قوائم، قواميس، صفوف .. إلخ)

### Example

If you send a **List** as a parameter, it will still be a **List** when it reaches the function.

```
def my_function(food):
    for x in food:
        print(x)
```

هنا في هذا المثال

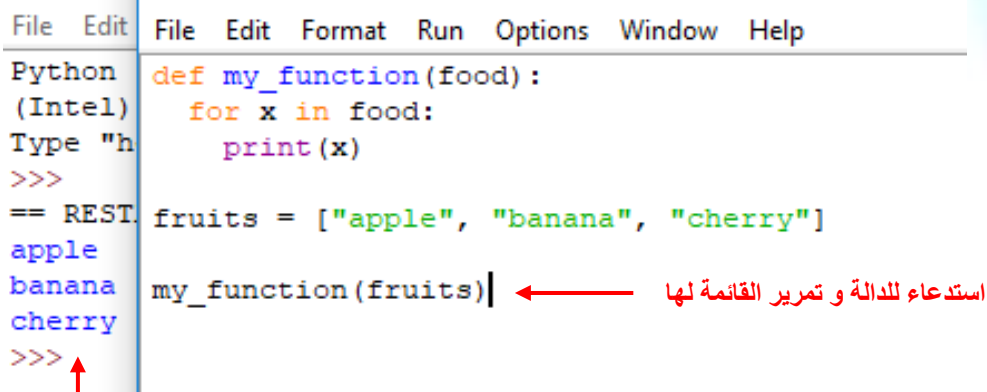
```
fruits = ["apple", "banana", "cherry"]
```

قمنا بتعريف دالة اسمها **my\_function**

وعند استدعائها مررنا لها القائمة **fruits**

```
my_function(fruits)
```

والدالة ستقوم بطباعة عناصر القائمة



```
File Edit Format Run Options Window Help
Python (Intel) Type "h" >>>
def my_function(food):
    for x in food:
        print(x)
fruits = ["apple", "banana", "cherry"]
my_function(fruits)
>>>
apple
banana
cherry
>>>
```

نتيجة تشغيل الكود

استدعاء للدالة و تمرير القائمة لها

## ➤ Return Values

## إرجاع القيم

لجعل الدالة تُعيد قيمة استخدم كلمة **return** (لإرجاع النتيجة)

To let a **function**, return a value, use the **return** statement.

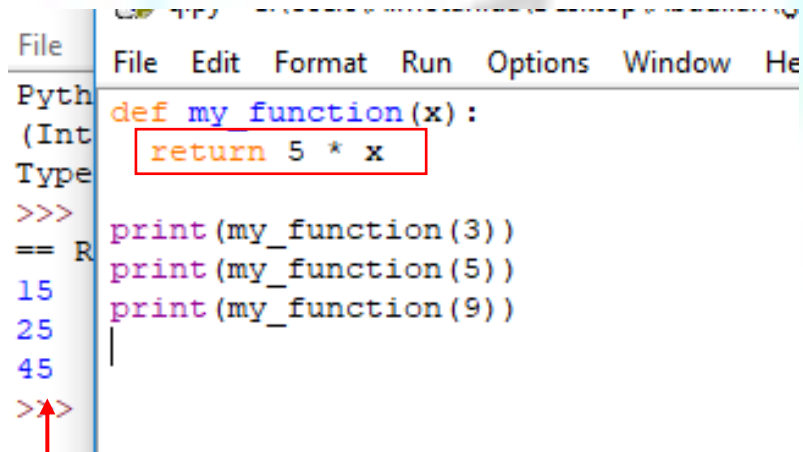
### Example

```
def my_function(x):  
    return 5 * x
```

هنا تُعيد الدالة **my\_function**

```
print(my_function(3))  
print(my_function(5))  
print(my_function(9))
```

ناتج ضرب المُعامل **x** (الذي تم تمريره إليها) في العدد 5



```
File Edit Format Run Options Window Help  
Python (Interpreter)  
Type  
>>> def my_function(x):  
      return 5 * x  
>>> print(my_function(3))  
15  
>>> print(my_function(5))  
25  
>>> print(my_function(9))  
45  
>>>
```

نتيجة تشغيل الكود

## ➤ Keyword Arguments

## مُعاملات الكلمة المفتاحية

You can also send arguments with the key = value syntax.

This way the order of the arguments does not matter.

يمكنك تمرير قيم للمُعاملات دون شرط الترتيب الذي تم وضعهم فيه وذلك من خلال ذكر اسم المُعاملات أو المفتاح key وإسناد قيمة له

### Example

```
def my_function(child3, child2, child1):
    print("The youngest child is " + child3)

my_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")
```

↑   ↑   ↑   ↑   ↑   ↑  
The key   value   The key   value   The key   value

عند الاستدعاء وبداخل أقواس الدالة قمنا بتمرير قيم للمُعاملات ولكن بدون التقيد بالترتيب شرط الترتيب غير مهم هنا وذلك لأننا قمنا بذكر أسماء المُعاملات وأسندنا إليها قيم المُعاملات التي نريد تمريرها

المفاتيح هنا هي أسماء المُعاملات، وقيم المفاتيح هي قيم المُعاملات

```
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3c4e3, Dec 23 2019) on win32
Type "help", "copyright", "credits()" or "quit()" for more
>>>
==
The youngest child is Linus
>>>
```

نتيجة تشغيل الكود

اختصار جملة مُعاملات الكلمة المفتاحية في بايثون هي **kwargs**

The phrase Keyword Arguments are often shortened to **kwargs** in Python documentations.

## ➤ Arbitrary Arguments

استقبال عدد غير محدد من المُعاملات

If you do not know how many arguments that will be passed into your **function**, add a **\*** before the parameter name in the **function** definition.

This way the **function** will receive a **tuple** of arguments and can access the items accordingly.

إذا كنت لا تعرف عدد المُعاملات التي سيتم تمريرها للدالة

عليك فقط كتابة مُعامل واحد وإضافة علامة النجمة \* قبل اسم المُعامل عند تعريفك وبنائك للدالة

هذه الطريقة ستجعل كل القيم التي ستُمرر للدالة يتم تجميعها داخل صف **tuple**

## Example

If the number of arguments is unknown, add a **\*** before the parameter name.

المُعامل سيكون عبارة عن صف **tuple**

```
def my_function(*kids):
    print("The youngest child is " + kids[2])
```

my\_function("Emil", "Tobias", "Linus") ← قمنا باستدعاء الدالة مع تمرير 3 قيم لها

في هذا المثال الدالة **my\_function** تقبل عدد غير محدد من القيم عند استدعائها

وتقوم بطباعة جملة نصية مع قيمة المُعامل الذي لديه الـ **index 2**

جميع القيم التي سيتم تمريرها للدالة يتم تخزينها على شكل صف في مُعامل واحد وهو هنا **kids**

```
File Edit Shell Debug Options Window File Edit Format Run Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ff... def my_function(*kids):
(Intel)] on win32                print("The youngest child is " + kids[2])
Type "help", "copyright", "cred my_function("Emil", "Tobias", "Linus")
>>>
==
The youngest child is Linus
>>>
```

نتيجة تشغيل الكود



## ➤ Recursion

## الاستدعاء الذاتي للدالة

بايثون توفر الدوال التي تقوم بمناداة/استدعاء نفسها

**Python** also accepts **function recursion**, which means a defined **function** can call itself.

الاستدعاء الذاتي أي أن الدالة تعيد استدعاء نفسها بنفسها و يعتبر من أشهر المفاهيم في علم الرياضيات والبرمجة، وتستخدمه أغلب لغات البرمجة، فتستدعي الدالة نفسها (تكرار تنفيذ كود معين) للوصول إلى نتيجة ما

**Recursion** is a common mathematical and programming concept. It means that a **function** calls itself. This has the benefit of meaning that you can loop through data to reach a result.

عليك أيها المبرمج/المطور أخذ الحيطة أثناء استخدام الاستدعاء الذاتي، فإذا قمت بجعل الدالة تستدعي نفسها بدون شرط للتوقف قد يؤدي ذلك إلى حدوث مشكلة في ذاكرة جهازك، لأنها ستأخذ مساحة وتمتلئ الذاكرة مما قد يبطئ نظام الجهاز .. إلى أن يتم توقف جهازك عن العمل

The developer should be very careful with **recursion** as it can be quite easy to slip into writing a **function** which never terminates, or one that uses excess amounts of memory or processor power. However, when written correctly recursion can be a very efficient and mathematically elegant approach to programming.

## Example

وضع مُعامل، يمثل كعداد لتكرار عملية الاستدعاء

```
def tri_recursion(k):
```

قمنا هنا بتعريف الدالة

```
    if(k>0):
```

وضع شرط لتحديد عدد مرات استدعاء الدالة لنفسها

```
        result = k+tri_recursion(k-1)
```

استدعاء الدالة لنفسها

```
        print(result)
```

```
    else:
```

```
        result = 0
```

ترجيع القيمة

```
    return result
```

```
print("\n\nRecursion Example Results")
tri_recursion(6)
```

In this example, `tri_recursion()` is a **function** that we have defined to call itself ("recurse"). We use the `k` variable as the data, which decrements (-1) every time we recurse.

The **recursion** ends when the condition is not greater than 0 (i.e. when it is 0).

To a new developer it can take some time to work out how exactly this works, best way to find out is by testing and modifying it.

في هذا المثال قمنا بتعريف دالة `tri_recursion()` وتتوقف الدالة عن استدعاء نفسها عندما لا يتحقق الشرط  $k > 0$  لكن طالما أن الشرط متحقق سنقوم باستدعاء نفسها  
استخدمنا المتغير `k` الذي سيقوم بإنقاص 1 من قيمته في كل مرة عندما تستدعي الدالة نفسها

```
File Edit Shell Debug Options File Edit Format Run Options Window Help
Python 3.7.2 (tags/v3.7.2: (Intel)] on win32
Type "help", "copyright",
>>>
==

def tri_recursion(k):
    if(k>0):
        result = k+tri_recursion(k-1)
        print(result)
    else:
        result = 0
        return result

print("\n\nRecursion Example Results")
tri_recursion(6)
|

Recursion Example Results
1
3
6
10
15
21
>>>
```

نتيجة تشغيل الكود

أتممت درسك بنجاح!

روابط قد تهتمك

Useful links

- [Functions](#)
- [Functions in Python - Coderbyte](#)
- [Python Programming #12 - Defining and Creating Functions](#)
- [#32 Python Tutorial for Beginners | Functions in Python](#)
- [#33 Python Tutorial for Beginners | Function Arguments in Python](#)
- [#34 Python Tutorial for Beginners | Types of Arguments in Python](#)
- [#35 Python Tutorial for Beginners | Keyworded Variable Length Arguments in Python | \\*\\*kwargs](#)
- [#37 Python Tutorial for Beginners | Pass List to a Function in Python](#)
- [13 - Python - Beginners Tutorial - Functions - Part 1](#)
- [Function arguments in detail - Advanced Python 18 - Programming Tutorial](#)
- [Recursion](#)
- [Learn Python in Arabic #84 - الدالة المرجعية - Recursive Function Python](#)
- [#40 Python Tutorial for Beginners | Recursion](#)
- [Python: Recursion Explained](#)

طبق ما تعلمته في هذا الدرس

ولا تنسى مشاركتنا أكوادك

# اليوم الخامس والثلاثون

تعريف دالة لا تحمل اسمًا في لغة البايثون

**Python** Lambda

هناك طريقة أخرى لتعريف الدوال غير الكلمة المحجوزة **def** كما تعلمنا في الدروس السابقة يمكنك تعريف الدالة أيضا باستخدام الـ **lambda** وهذه الطريقة لا تحتاج منك أن تعطي اسمًا لهذه الدالة

**lambda** هي أيضا من الكلمات المحجوزة في لغة بايثون وهي أسلوب وطريقة لتعريف دوال صغيرة لا تحمل اسمًا، تُستخدم للمهام الضمنية تأخذ أي عدد من المُعاملات لكنها تحتوي على تعبير واحد

A **lambda function** is a small anonymous function.

A **lambda function** can take any number of arguments but can only have one expression.

## ➤ Syntax

**lambda arguments: expression**

**lambda** تعتبر صيغة مختصرة لتعريف الدوال

The expression is executed, and the result is returned. عندما يتم تنفيذ التعبير / الأمر، معناها أنه قام بترجيع قيمة

## Example

A **lambda** function that adds 10 to the number passed in as an argument and print the result.

كتابة الرمز إجباري      المعامل      تعريف دالة ليس لها اسم

الأمر (التعبير) الذي سيرجع القيمة      عند تنفيذ الدالة

اسم المتغير      الذي سيتم إسناد الدالة إليه

```
x = lambda a : a + 10
print(x(5))
```

قمنا بتعريف دالة ليس لها اسم

وعند استدعائها سنمرر لها العدد 5

ترجع لنا ناتج جمع العدد الذي تم تمريره مع العدد 10

وقمنا بإسناد الدالة إلى المتغير x حتى نتمكن من استدعاء الدالة من خلاله

```
File Edit Format Run Options
Python x = lambda a : a + 10
(Inte print(x(5))
Type |
>>>
== RE
15
>>>
```

نتيجة تشغيل الكود

لاحظ

عندما تقوم بتعريف دالة بصيغة **lambda** فالدالة ليس لها اسم كما تعلم

وقد تعلمت في الدوال عندما تريد استدعاء دالة ما، فإنك تقوم بمناداتها باسمها

لذلك نقوم بإنشاء متغير لإسناد الدالة إليه ليتم اعتبار اسم المتغير هو اسم الدالة، فمن خلاله يمكنك مناداة الدالة

**Lambda functions** can take any number of arguments.

تأخذ/تستقبل أي عدد من المُعاملات

## Example

A **lambda function** that multiplies argument **a** with argument **b** and print the result.

```
x = lambda a, b : a * b
print(x(5, 6))
```

قمنا بتعريف دالة ليس لها اسم

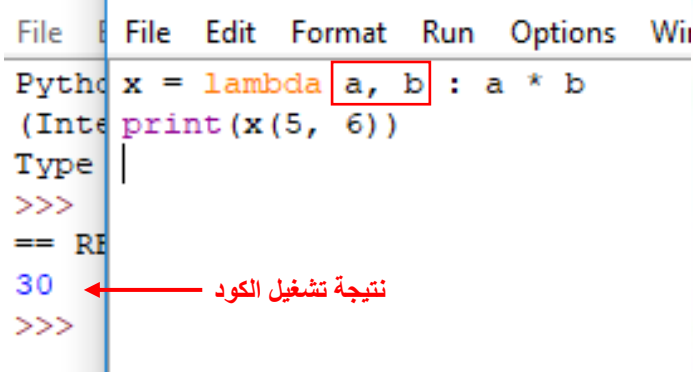
وعند استدعائها سنمرر لها العددين 5, 6

ترجع لنا ناتج ضرب العددين

وقمنا بإسناد الدالة إلى المتغير **x** حتى نتمكن من استدعاء الدالة من خلاله

في هذا المثال لدينا مُعاملين

يتم وضع الفاصلة بين كل مُعامل



```
File Edit Format Run Options Win
Python x = lambda a, b : a * b
(Inte print(x(5, 6))
Type |
>>>
== RE
30
>>>
```

استمر .. إنك تحرز تقدّمًا



## Example

A **lambda** function that sums argument **a**, **b**, and **c** and print the result.

```
x = lambda a, b, c : a + b + c  
print(x(5, 6, 2))
```

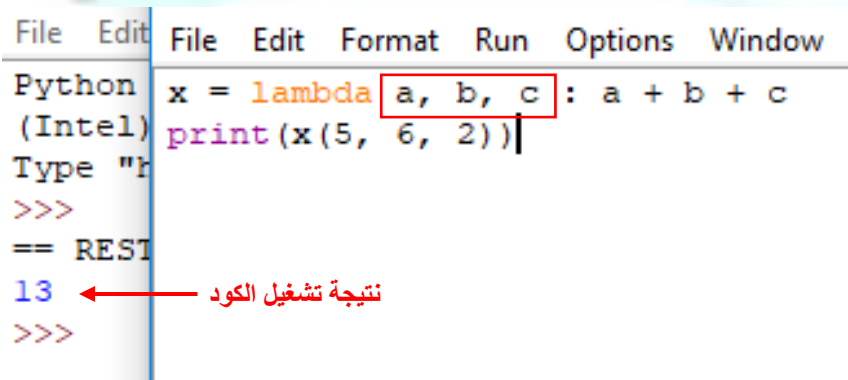
قمنا بتعريف دالة ليس لها اسم

وعند استدعائها سنمرر لها الأعداد 5, 6, 2

ترجع لنا ناتج جمعهم

وقمنا بإسناد الدالة إلى المتغير **x** حتى نتمكن من استدعاء الدالة من خلاله

في هذا المثال لدينا أكثر من مُعامل



```
File Edit File Edit Format Run Options Window  
Python x = lambda a, b, c : a + b + c  
(Intel) print(x(5, 6, 2))  
Type "r"  
>>>  
== REST  
13  
>>>
```

نتيجة تشغيل الكود

أتممتَ درسك بنجاح!  
تابع التقدم

روابط قد تساعدك

Check the links below

- [Python lambda](#)
- [Python tutorial - Lambda Function](#)
- [#42 Python Tutorial for Beginners | Anonymous Functions | Lambda](#)
- [شرح - lambda 28 - تعلم البرمجة بلغة بايثون](#)

طبق ما تعلمته في هذا الدرس  
ولا تنسى مشاركتنا أكوادك

# اليوم السادس والثلاثون

تعريف دالة لا تحمل اسمًا في لغة البايثون

**Python** Lambda 2

## ➤ Why Use **Lambda** Functions?

لماذا **lambda**

تكمُن قوة الـ **lambda** عند استخدامها داخل دالة أخرى

The power of **lambda** is better shown when you use them as an **anonymous function** inside another **function**.

Say you have a **function definition** that takes one argument, and that argument will be multiplied with an unknown number.

```
def myfunc(n):
    return lambda a : a * n
```

لديك هنا دالة معرفة اسمها **myfunc** تستقبل فقط مُعامل واحد  
وقيمة هذا المُعامل سيتم ضربها بعدد مجهول

Use that **function definition** to make a **function** that always doubles the number you send in.

### Example

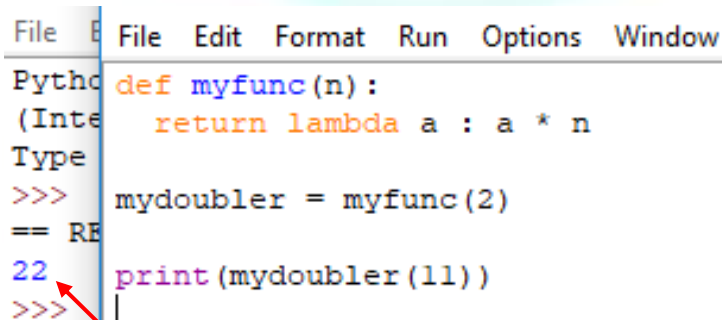
في هذا المثال تقوم الدالة بمضاعفة العدد الذي يتم إرساله

```
def myfunc(n):
    return lambda a : a * n
```

قمنا بتعريف دالة **myfunc** والتي تستقبل مُعاملاً واحداً  
وتقوم بإرجاع قيمة الدالة المكتوبة داخلها بصيغة **lambda**

`mydoubler = myfunc(2)` ← استدعاء الدالة مع تمرير القيمة 2 لـ **n** وإسناد الدالة لمتغير

`print(mydoubler(11))` ← استدعاء الدالة وتمرير القيمة 11 لـ **a** ومن ثم الطباعة



```
File Edit Format Run Options Window
Python3 def myfunc(n):
(Inte    return lambda a : a * n
Type
>>> mydoubler = myfunc(2)
== RE
22 print(mydoubler(11))
>>>
```

نتيجة تشغيل الكود

Or, use the same **function definition** to make a **function** that always triples the number you send in.

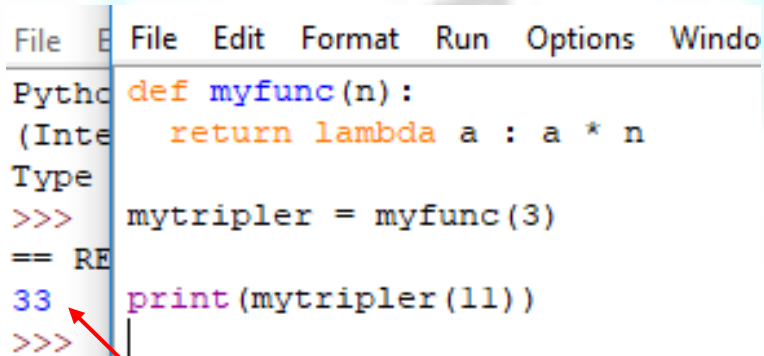
## Example

في هذا المثال تقوم الدالة بمضاعفة العدد الذي يتم تمريره، ثلاث مرات

```
def myfunc(n):  
    return lambda a : a * n
```

استدعاء الدالة مع تمرير القيمة 3 لـ  $n$  وإسناد الدالة لمتغير `mytripler = myfunc(3)` ←

تمرير القيمة 11 لـ  $a$  أثناء الاستدعاء ومن ثم الطباعة `print(mytripler(11))` ←



```
File Edit Format Run Options Windo  
Python3 def myfunc(n):  
(Inter return lambda a : a * n  
Type  
>>> mytripler = myfunc(3)  
== RE  
33 print(mytripler(11))  
>>>
```

نتيجة تشغيل الكود

Or, use the same **function definition** to make both **functions**, in the same program.

## Example

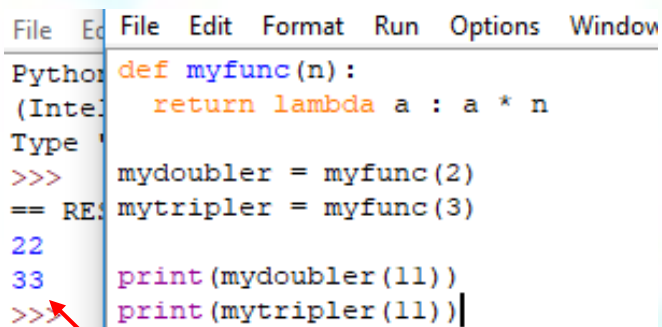
```
def myfunc(n):  
    return lambda a : a * n
```

```
mydoubler = myfunc(2)  
mytripler = myfunc(3)
```

```
print(mydoubler(11))  
print(mytripler(11))
```

في هذا المثال

قمنا بوضع الدالتين في المثالين السابقين في برنامج واحد



```
File Edit Format Run Options Window  
Python def myfunc(n):  
(Intel return lambda a : a * n  
Type  
>>> mydoubler = myfunc(2)  
== RE mytripler = myfunc(3)  
22  
33 print(mydoubler(11))  
>>> print(mytripler(11))|
```

نتيجة تشغيل الكود

Use **lambda functions** when an **anonymous function** is required for a short period of time

مبرمج الغد!  
أتممت درسك

روابط قد تهتمك

Useful links

- [في بايثون lambda تعابير](#)
- [Lambda Expressions & Anonymous Functions || Python Tutorial || Learn Python Programming](#)
- [Python Lambda | Advanced Python | Tutorial 18](#)

طبق ما تعلمته في هذا الدرس  
ولا تنسى مشاركتنا أكوادك



# اليوم السابع والثلاثون

## المصفوفات في لغة البايثون

### Python Arrays

## ➤ Arrays

### المصفوفات

تُستخدم المصفوفة كمتغير واحد تستخدم لتخزين عدة قيم كأنها متغير واحد

Arrays are used to store multiple values in one single variable.

### Example

Create an **array** containing car names

مصفوفة تحتوي على أسماء سيارات

```
cars = ["Ford", "Volvo", "BMW"]  
print(cars)
```

## ➤ What is an Array?

### ماذا تعني المصفوفة؟

An **array** is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
car1 = "Ford"  
car2 = "Volvo"  
car3 = "BMW"
```

ماذا لو كان لديك قائمة كبيرة من العناصر، وأردت البحث عن عنصر معين في هذه القائمة من الصعب جدا قيامك بتخزين كل عنصر لوحده في متغير خاص

However, what if you want to loop through the cars and find a specific one?

And what if you had not 3 cars, but 300?

الأفضل هو قيامك باستخدام مصفوفة فهي تستطيع أن تخزن بداخلها عددا من العناصر مما سيؤدي إلى تقليل حجم الكود وإمكانية الوصول لقيم العناصر بطريقة سهلة وسريعة وذلك من خلال الـ **index** الخاص بكل عنصر.

The solution is an **array**!

An **array** can hold many values under a single name, and you can access the values by referring to an **index** number.

## ➤ Access the Elements of an Array

## الوصول لعناصر المصفوفة

يتم الإشارة لعناصر المصفوفة من خلال الـ **index** الخاص بكل عنصر.

You refer to an **array** element by referring to the **index** number.

### Example

للوصول لعناصر المصفوفة، يجب عليك كتابة اسم المصفوفة أولاً ثم تحديد رقم الـ **index** الخاص بالعنصر الذي نريد الوصول إليه داخل أقواس مربعة []

Get the value of the first **array** item

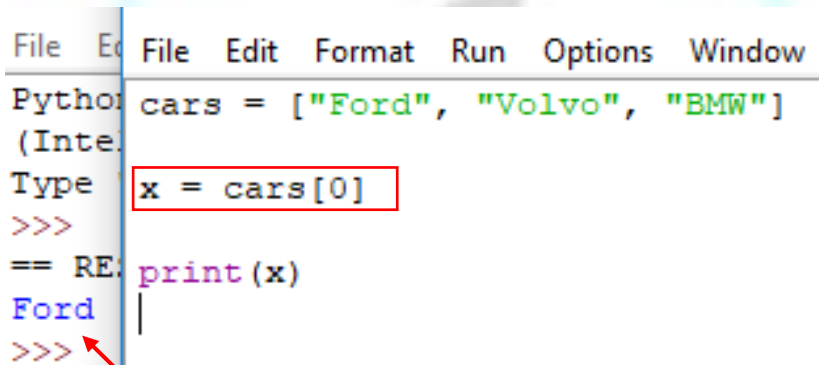
```
cars = ["Ford", "Volvo", "BMW"]
```

```
x = cars[0] ←
```

```
print(x)
```

في هذا المثال

قمنا بطباعة قيمة العنصر الأول في المصفوفة



```
File Edit Format Run Options Window
Python cars = ["Ford", "Volvo", "BMW"]
(Interp
Type x = cars[0]
>>>
== RE: print(x)
Ford
>>>
```

نتيجة تشغيل الكود

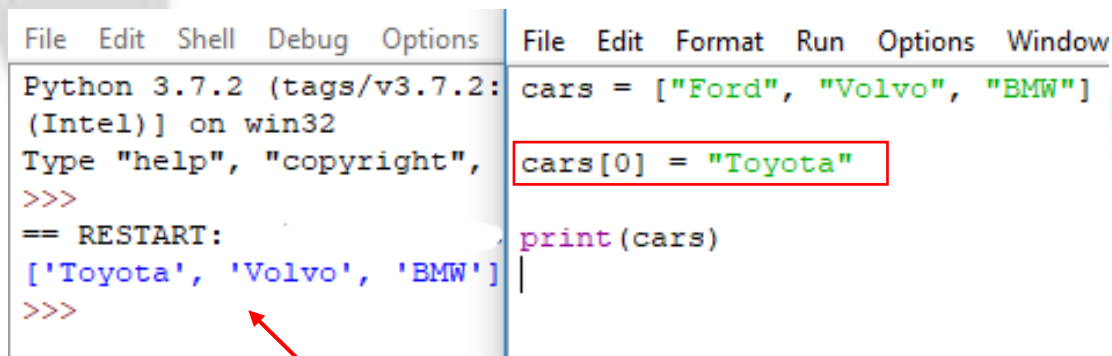
## Example

في هذا المثال

قمنا بتغيير/تعديل قيمة العنصر الأول في المصفوفة، ومن ثم طباعة القيمة

Modify the value of the first **array** item

```
cars = ["Ford", "Volvo", "BMW"]  
cars[0] = "Toyota" ←  
print(cars)
```



```
File Edit Shell Debug Options | File Edit Format Run Options Window  
Python 3.7.2 (tags/v3.7.2: (Intel)] on win32  
Type "help", "copyright",  
>>>  
== RESTART:  
['Toyota', 'Volvo', 'BMW']  
>>>  
cars = ["Ford", "Volvo", "BMW"]  
cars[0] = "Toyota"  
print(cars)
```

نتيجة تشغيل الكود

## ➤ The Length of an Array

### حجم المصفوفة

استخدم الدالة **len()** لمعرفة عدد عناصر المصفوفة

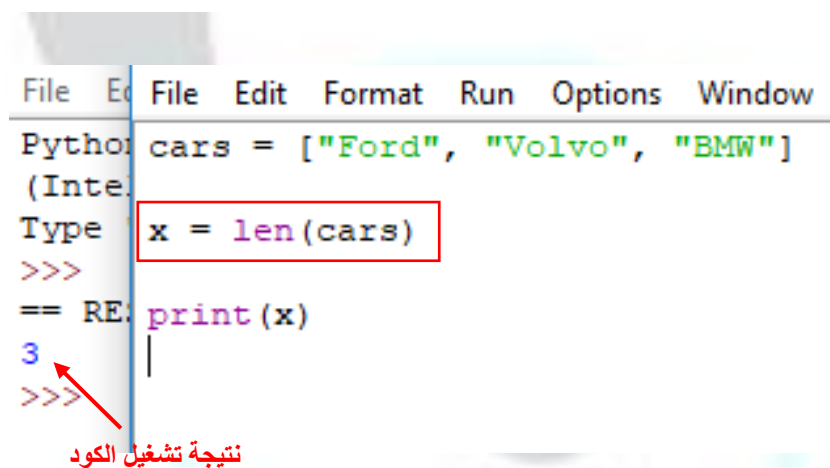
Use the **len()** method to return the length of an **array** (the number of elements in an **array**).

### Example

Return the number of elements in the **cars** array

```
cars = ["Ford", "Volvo", "BMW"]
x = len(cars)
print(x)
```

هنا قمنا بطباعة عدد عناصر المصفوفة **cars**



**Note:** The length of an **array** is always one more than the highest **array index**

Element	1	2	3	4	5
Index	0	1	2	3	4

عدد عناصر المصفوفة دائما أكبر برقم واحد من **index**

أتممت درسك بنجاح!  
واصل التعلم

طبق ما تعلمته في هذا الدرس  
ولا تنسى مشاركتنا أكوادك

# اليوم الثامن والثلاثون



## المصفوفات في لغة البايثون

### **Python** Arrays 2

## عرض قيم عناصر المصفوفة

### ➤ Looping Array Elements

للتعامل والوصول إلى جميع عناصر المصفوفة استخدم الحلقة **for**

You can use the **for in loop** to loop through all the elements of an **array**.

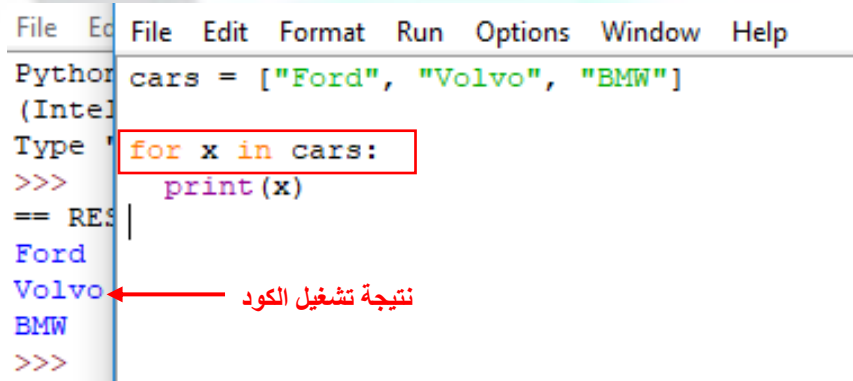
### Example

Print each item in the **cars array**

```
cars = ["Ford", "Volvo", "BMW"]  
  
for x in cars:  
    print(x)
```

هنا في هذا المثال

استخدمنا الحلقة **for** لعرض جميع قيم المصفوفة دفعة واحدة



```
File Edit Format Run Options Window Help  
Python cars = ["Ford", "Volvo", "BMW"]  
(Intel  
Type 'for x in cars:  
>>>     print(x)  
== RES |  
Ford  
Volvo  
BMW  
>>>
```

نتيجة تشغيل الكود

## إضافة عناصر للمصفوفة

### ➤ Adding Array Elements

لإضافة عناصر للمصفوفة استخدم الدالة **append()**

You can use the **append()** method to add an element to an **array**.

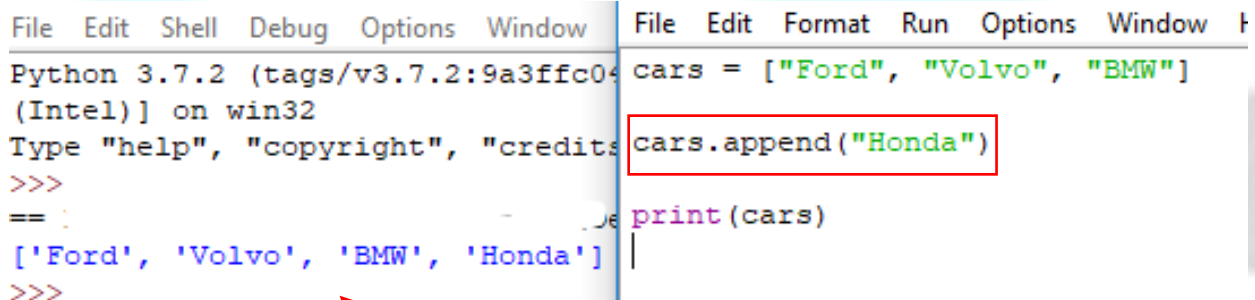
### Example

Add one more element to the **cars** array

```
cars = ["Ford", "Volvo", "BMW"]  
cars.append("Honda")  
print(cars)
```

في المثال هنا

قمنا بإضافة عنصر جديد للمصفوفة



```
File Edit Shell Debug Options Window File Edit Format Run Options Window  
Python 3.7.2 (tags/v3.7.2:9a3ffc04c, Dec 10 2019, [AMD64]) on win32  
Type "help", "copyright", "credits()" or "quit()" to interact with the interpreter  
>>>  
==> cars = ["Ford", "Volvo", "BMW"]  
cars.append("Honda")  
print(cars)  
['Ford', 'Volvo', 'BMW', 'Honda']  
>>>
```

نتيجة تشغيل الكود

طبّق لتتأكد من فهمك

## ➤ Removing Array Elements

## حذف عناصر المصفوفة

لحذف عنصر من المصفوفة استخدم الدالة **pop()**

You can use the **pop()** method to remove an element from the **array**.

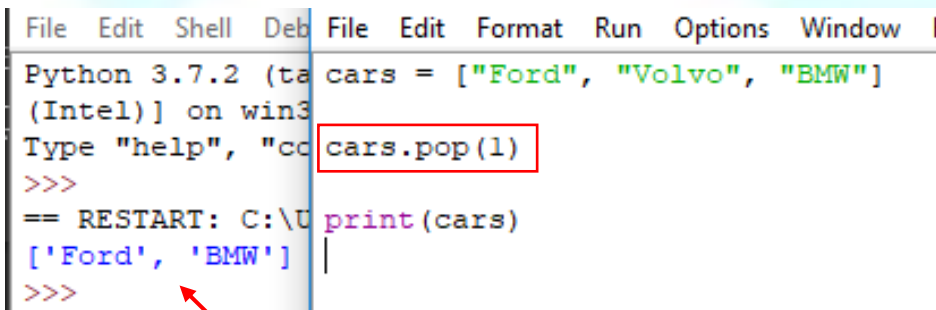
### Example

Delete the second element of the **cars** array

```
cars = ["Ford", "Volvo", "BMW"]  
cars.pop(1)  
print(cars)
```

في المثال هنا

قمنا بحذف العنصر الثاني من المصفوفة



```
File Edit Shell Deb File Edit Format Run Options Window  
Python 3.7.2 (tags/bpo-42032, Jul 8 2020, 14:32:08) on win32  
Type "help()", "copyright()", "credits()", "license()", "sys.version()",  
>>> cars = ["Ford", "Volvo", "BMW"]  
>>> cars.pop(1)  
>>> print(cars)  
['Ford', 'BMW']  
>>>
```

نتيجة تشغيل الكود

You can also use the `remove()` method to remove an element from the **array**.

## Example

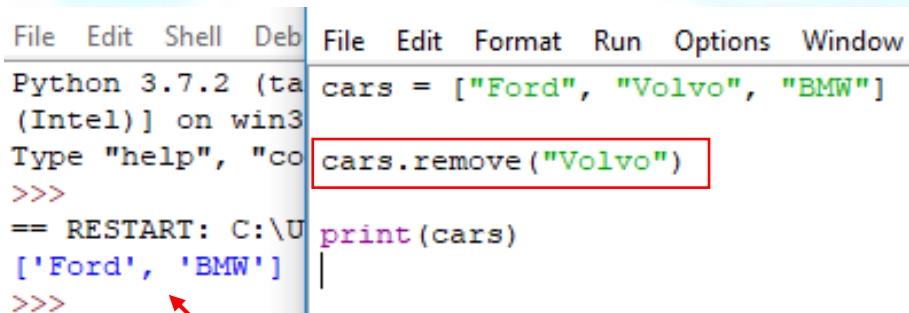
يمكنك أيضاً استخدام الدالة `remove()` لحذف عنصر من المصفوفة

Delete the element that has the value "Volvo"

```
cars = ["Ford", "Volvo", "BMW"]  
cars.remove("Volvo")  
print(cars)
```

انظر المثال التالي

قمنا بحذف العنصر الثاني من المصفوفة



```
File Edit Shell Deb File Edit Format Run Options Window  
Python 3.7.2 (ta cars = ["Ford", "Volvo", "BMW"]  
(Intel)] on win3 cars.remove("Volvo")  
Type "help", "co print(cars)  
>>> == RESTART: C:\U  
['Ford', 'BMW']  
>>>
```

نتيجة تشغيل الكود

**Note:** The list's `remove()` method only removes the first occurrence of the specified value.

## ➤ Array Methods

## دوال المصفوفة في لغة بايثون

بايثون تحتوي على دوال جاهزة يمكنك استخدامها للتعامل مع المصفوفات /القوائم وهي ما ناقشنا أغلبها في هذا الدرس مع المصفوفات ومع القوائم في دروس سابقة

**Python** has a set of built-in methods that you can use on **lists/arrays**.

Method	Description
<u>append()</u>	Adds an element at the end of the list
<u>clear()</u>	Removes all the elements from the list
<u>copy()</u>	Returns a copy of the list
<u>count()</u>	Returns the number of elements with the specified value
<u>extend()</u>	Add the elements of a list (or any iterable), to the end of the current list
<u>index()</u>	Returns the index of the first element with the specified value
<u>insert()</u>	Adds an element at the specified position
<u>pop()</u>	Removes the element at the specified position
<u>remove()</u>	Removes the first item with the specified value
<u>reverse()</u>	Reverses the order of the list
<u>sort()</u>	Sorts the list

**Note:** **Python** does not have built-in support for **Arrays**, but **Python Lists** can be used instead.

رائع!  
أتممت درسك الأخير لهذا الأسبوع

روابط قد تهتمك

Useful links

- [Python/array](#)
- [Arrays تعلم بايثون 3 - المصفوفات #6](#)

طبّق ما تعلمته في هذا الدرس  
ولا تنسى مشاركتنا أكوادك

# اليوم التاسع والثلاثون & اليوم الأربعون



## تحدي الأسبوع (يتم حله ورفعته على Github)

### السؤال الأول:

استعمل الاستدعاء الذاتي **Recursion** لحساب  $5^3$  واطبع النتيجة

### السؤال الثاني:

قم بإنشاء قائمة **List** تحتوي على الأرقام التالية: 88, 9, 7, 3, 2, -1, -5, -6, -4.  
ثم باستعمال **Lambda** قم بكتابة برنامج يطبع فقط الأرقام الموجبة من القائمة

انتظرونا في دروس الأسبوع القادم