

## المصفوفات والقوائم

- يجب التصريح عن المصفوفات. القوائم لا تفعل ذلك ، لأنها مدمجة في بايثون. في الأمثلة أعلاه ، رأيت أن القوائم يتم إنشاؤها ببساطة من خلال تضمين سلسلة من العناصر في أقواس مربعة. من ناحية أخرى ، يتطلب إنشاء مصفوفة وظيفة محددة من وحدة المصفوفة (على سبيل المثال ، `array.array()` أو حزمة NumPy (على سبيل المثال ، `numpy.array()`). لهذا السبب ، يتم استخدام القوائم أكثر من المصفوفات.
  - يمكن للمصفوفات تخزين البيانات بشكل مضغوط للغاية وتكون أكثر كفاءة لتخزين كميات كبيرة من البيانات.
  - المصفوفات رائعة للعمليات العددية ؛ لا يمكن للقوائم معالجة العمليات الحسابية بشكل مباشر. على سبيل المثال، يمكنك قسمة كل عنصر من عناصر المصفوفة على نفس الرقم بسطر واحد فقط من التعليمات البرمجية.
- إذا حاولت الشيء نفسه مع قائمة ، فستتلقى خطأ.
- وإذا كنت بحاجة إلى تخزين سلسلة قصيرة نسبيًا من العناصر ولا تخطط لإجراء أي عمليات حسابية بها ، فإن القائمة هي الخيار المفضل.
- إذا كان لديك تسلسل طويل جدًا من العناصر، ففكر في استخدام مصفوفة.

# التعامل مع المصفوفات في بايثون

نظم معلومات ادارية - المرحلة الثالثة  
م. زينب صبيح جمعة

## القوائم (lists)

القوائم - هذه هي بنية البيانات الأكثر تنوعًا في Python وتتم كتابتها كقائمة من العناصر المفصولة بفواصل داخل أقواس مربعة. يمكن أن تتكون القائمة من عناصر غير متجانسة ومتجانسة. بعض الطرق المطبقة في القائمة هي:

`index()`, `append()`, `extend()`, `insert()`, `remove()`, `pop`

القوائم قابلة للتغيير. أي أنه يمكن تغيير محتواها مع الحفاظ على الهوية سليمة.

```
fruit=['apple', 'orange', 'banana', 'banana', 100, 200, 300]
print(fruit)
a= fruit.count('banana')
print(a)
b= fruit.index('banana')
print(b)
fruit.reverse()
print(fruit)
x= fruit.pop()
print(x)
lesson=['math', 'science', 'art', 'lang', 'chimstry']
lesn= lesson.pop()
print(lesn)
lesson.append('space')
print(lesson)
```



## انشاء مصفوفة

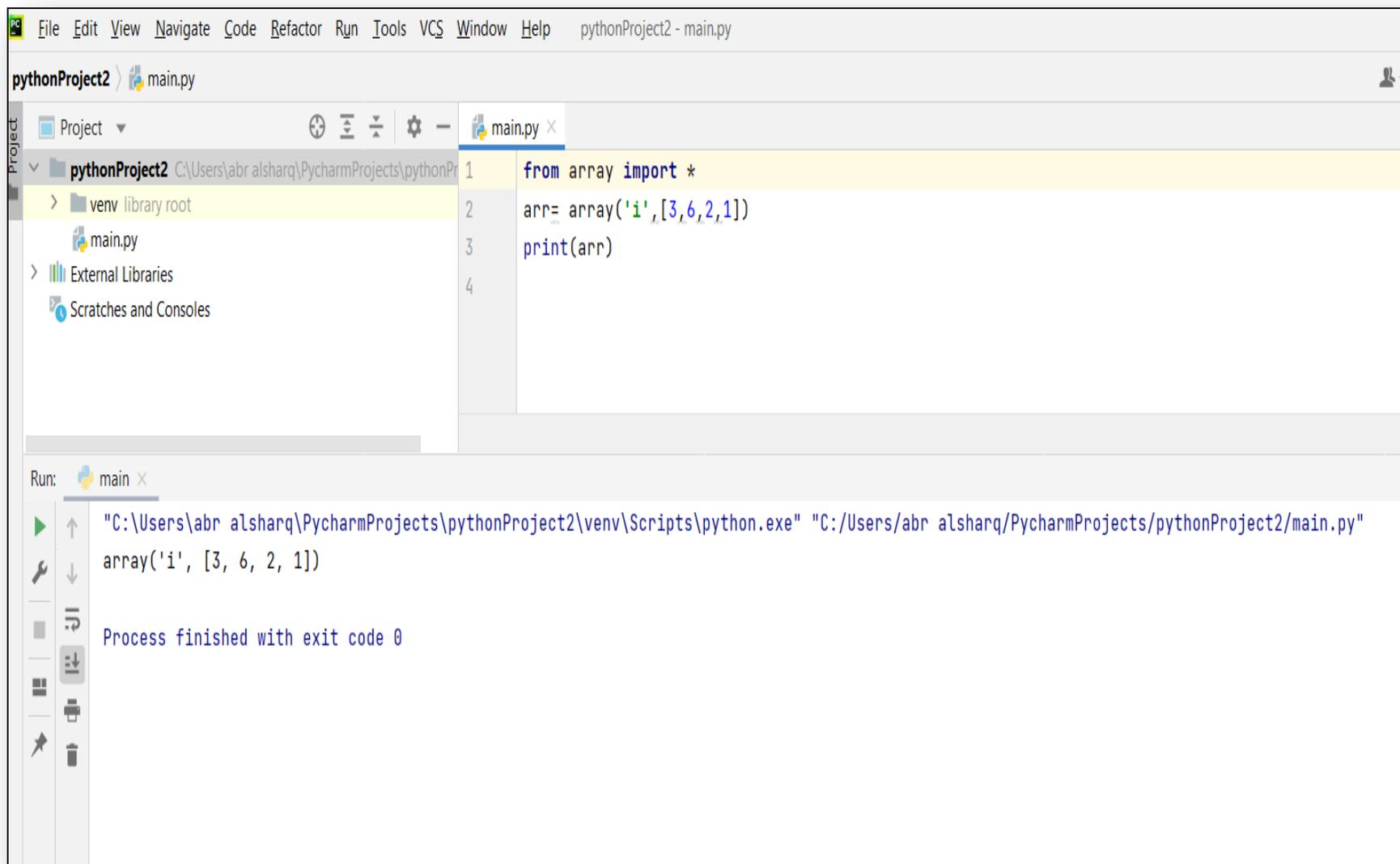
```
1 from array import   
2   
3 arr = array('i', [1,2,3,4,5])  
4 print(arr)
```

```
array('i', [1, 2, 3, 4, 5])
```

طباعة عنصر في مصفوفة باستخدام الدليل

```
1 print(arr[2])
```

```
3
```



The screenshot displays the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The current project is 'pythonProject2' and the active file is 'main.py'. The Project tool window on the left shows the project structure, including a virtual environment 'venv' and the 'main.py' file. The main editor window shows the following Python code:

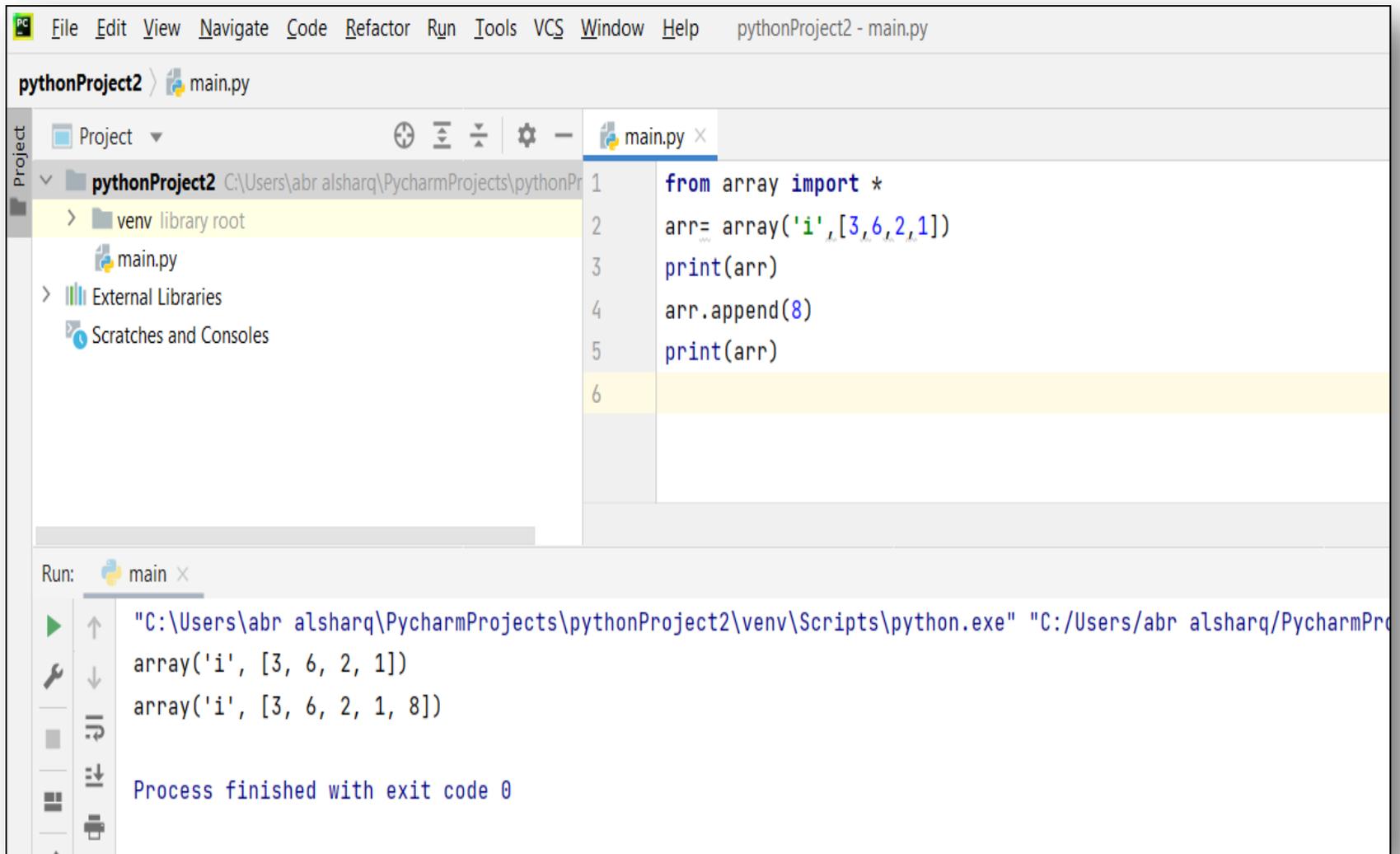
```
1 from array import *
2 arr= array('i',[3,6,2,1])
3 print(arr)
4
```

The Run tool window at the bottom shows the execution command and output:

```
Run: main x
"C:\Users\abr alsharq\PycharmProjects\pythonProject2\venv\Scripts\python.exe" "C:/Users/abr alsharq/PycharmProjects/pythonProject2/main.py"
array('i', [3, 6, 2, 1])
Process finished with exit code 0
```

# append

الماتريks في نهاية المصفوفة



The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The main editor window displays a Python file named main.py with the following code:

```
1 from array import *
2 arr = array('i', [3, 6, 2, 1])
3 print(arr)
4 arr.append(8)
5 print(arr)
6
```

The bottom panel shows the Run output for the main.py script:

```
Run: main x
"C:\Users\abr alsharq\PycharmProjects\pythonProject2\venv\Scripts\python.exe" "C:/Users/abr alsharq/PycharmPro
array('i', [3, 6, 2, 1])
array('i', [3, 6, 2, 1, 8])
Process finished with exit code 0
```

# insert

اضافة عنصر بين عناصر المصفوفة حيث تتم الاضافة قبل العنصر المحدد

The screenshot shows the PyCharm IDE interface. The top part displays the project structure on the left and the code editor on the right. The code editor shows the following Python code:

```
2 arr = array('i', [3, 6, 2, 1])
3 print(arr)
4 arr.append(8)
5 print(arr)
6 arr.insert(1, 10)
7 print(arr)
8
```

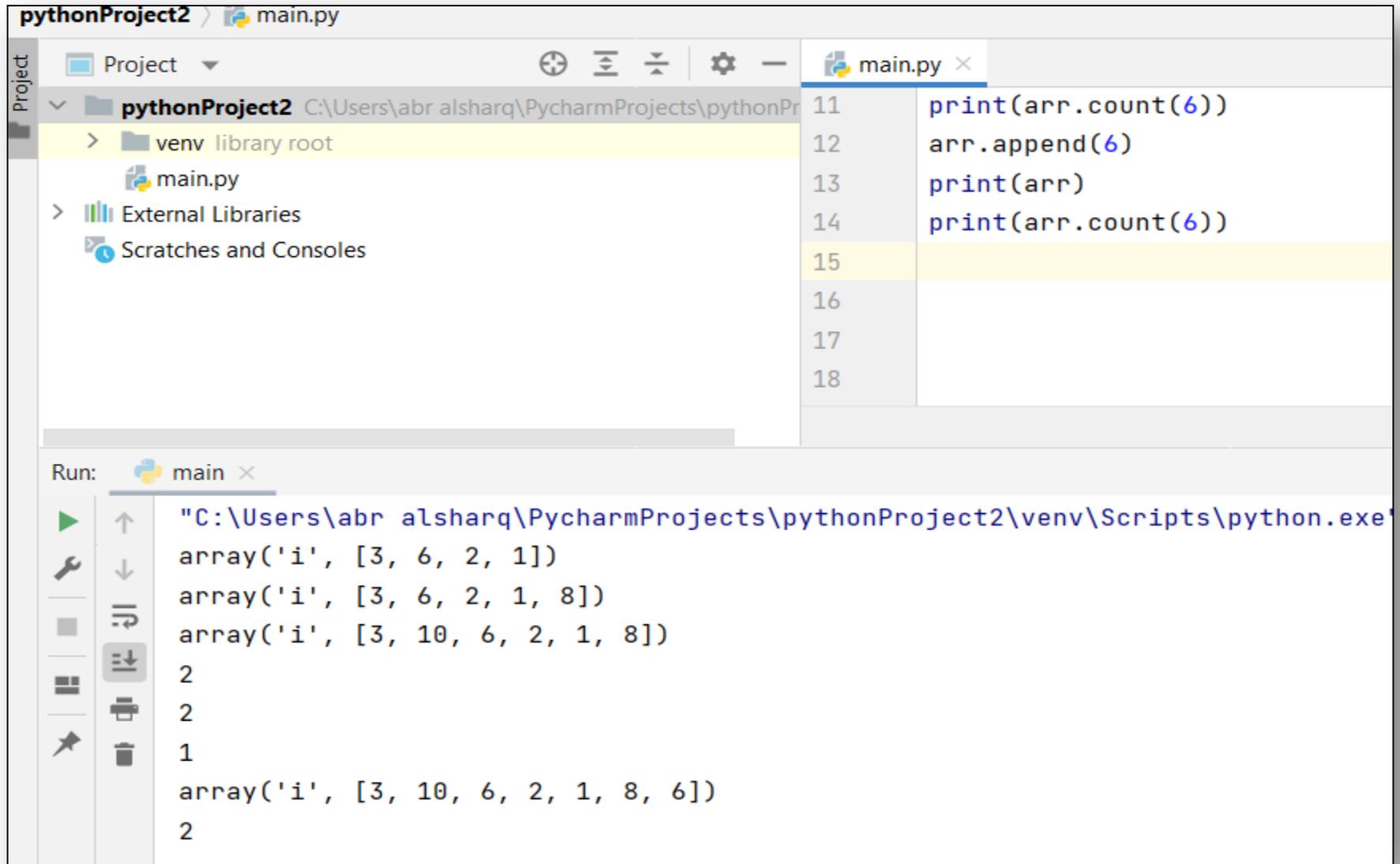
The bottom part of the screenshot shows the Run console output for the code:

```
Run: main x
"C:\Users\abr alsharq\PycharmProjects\pythonProject2\venv\Scripts\python.exe" "C:/Users/abr alsharq/Pycha
array('i', [3, 6, 2, 1])
array('i', [3, 6, 2, 1, 8])
array('i', [3, 10, 6, 2, 1, 8])
Process finished with exit code 0
```

The output shows the array after each operation: first the initial array [3, 6, 2, 1], then after append(8) it becomes [3, 6, 2, 1, 8], and finally after insert(1, 10) it becomes [3, 10, 6, 2, 1, 8].

# count

لمعرفة عدد مرات تكرار عنصر في مصفوفة



The screenshot shows the PyCharm IDE interface. The top-left pane displays the project structure for 'pythonProject2', including a 'venv' directory and a 'main.py' file. The top-right pane shows the code in 'main.py' with line numbers 11 through 18. The bottom pane shows the execution output for 'main.py'.

```
11 print(arr.count(6))
12 arr.append(6)
13 print(arr)
14 print(arr.count(6))
15
16
17
18
```

Run: main ×

```
"C:\Users\abr alsharq\PycharmProjects\pythonProject2\venv\Scripts\python.exe"
array('i', [3, 6, 2, 1])
array('i', [3, 6, 2, 1, 8])
array('i', [3, 10, 6, 2, 1, 8])
2
2
1
array('i', [3, 10, 6, 2, 1, 8, 6])
2
```

## reverse

```
Scratches and Consoles
```

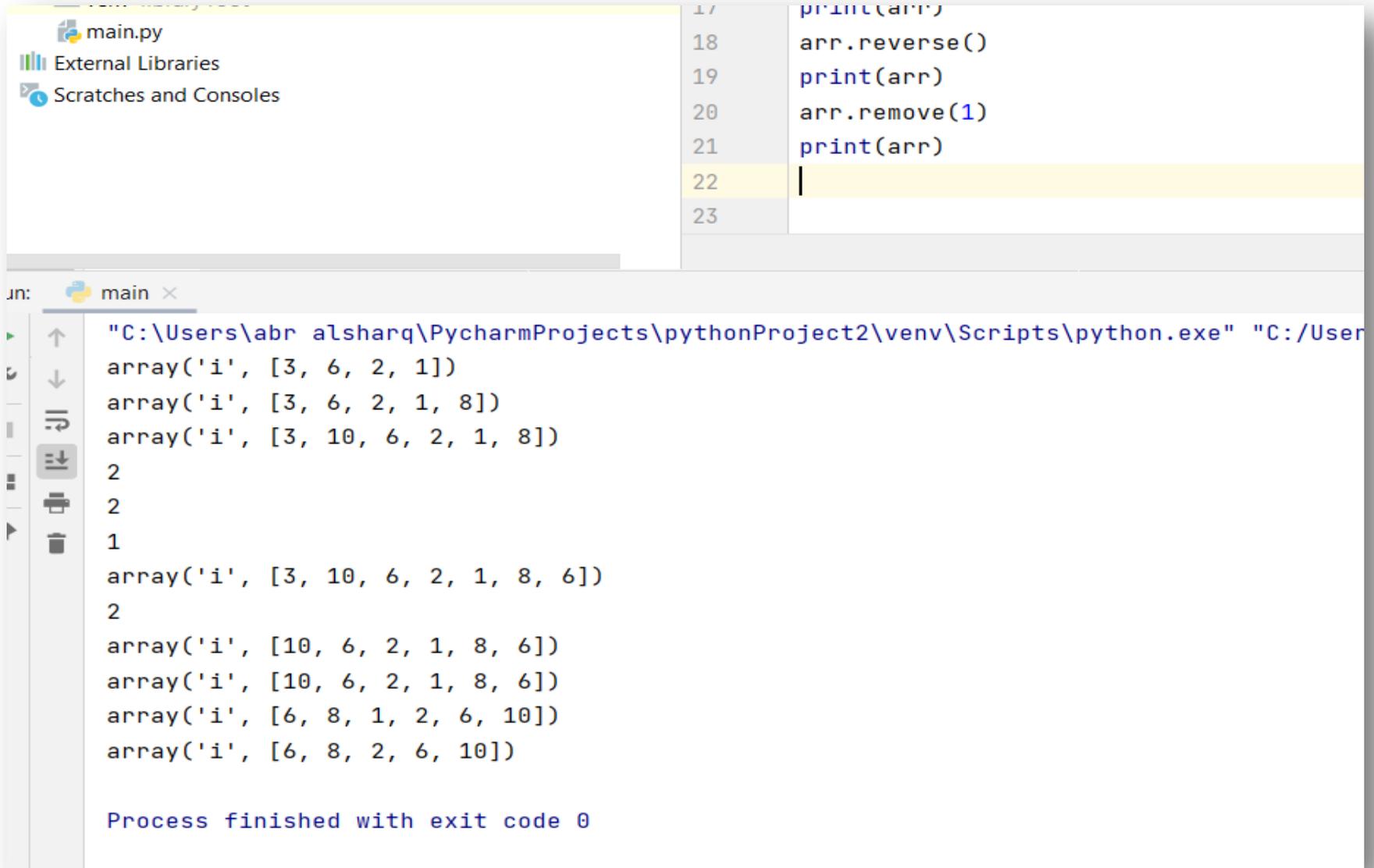
18	<code>arr.reverse()</code>
19	<code>print(arr)</code>
20	
21	

```
main x
"C:\Users\abr alsharq\PycharmProjects\pythonProject2\venv\Scripts\python.exe
array('i', [3, 6, 2, 1])
array('i', [3, 6, 2, 1, 8])
array('i', [3, 10, 6, 2, 1, 8])
2
2
1
array('i', [3, 10, 6, 2, 1, 8, 6])
2
array('i', [10, 6, 2, 1, 8, 6])
array('i', [10, 6, 2, 1, 8, 6])
array('i', [6, 8, 1, 2, 6, 10])

Process finished with exit code 0
```

# remove

حذف عنصر من مصفوفة



The screenshot shows a Python IDE with a file named 'main.py' open. The code in the editor is as follows:

```
17 print(arr)
18 arr.reverse()
19 print(arr)
20 arr.remove(1)
21 print(arr)
22 |
23
```

The output console shows the following execution results:

```
array('i', [3, 6, 2, 1])
array('i', [3, 6, 2, 1, 8])
array('i', [3, 10, 6, 2, 1, 8])
2
2
1
array('i', [3, 10, 6, 2, 1, 8, 6])
2
array('i', [10, 6, 2, 1, 8, 6])
array('i', [10, 6, 2, 1, 8, 6])
array('i', [6, 8, 1, 2, 6, 10])
array('i', [6, 8, 2, 6, 10])

Process finished with exit code 0
```

sorted

الترتيب

The image shows a Python IDE window with a code editor and a console. The code editor contains the following code:

```
34 arr=sorted(arr,reverse=True)
35 print(arr)
36 arr=sorted(arr,reverse=False)
37 print(arr)
```

The console output shows the following array states:

```
[99, 12, 10, 8, 6, 2, 2]
[2, 2, 6, 8, 10, 12, 99]
```

Annotations in Arabic explain the code:

- A speech bubble labeled "تنازلي" (Descending) points to the `reverse=True` parameter in line 34.
- A speech bubble labeled "تصاعدي" (Ascending) points to the `reverse=False` parameter in line 36.

The IDE interface includes a "main" tab and a vertical toolbar on the left side.