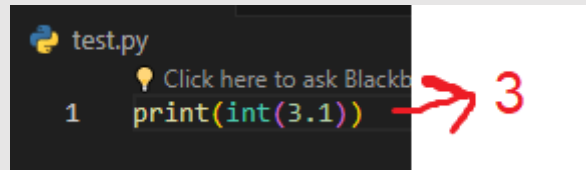


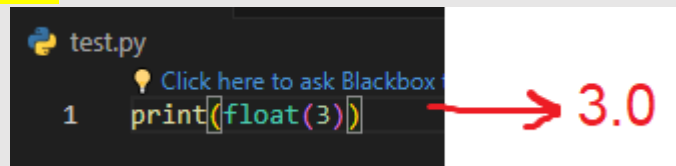
Arrays in python

در جلسات قبلی گفته شد که تبدیل فلوت به اینتیجر ممکن نیست ولی این چنین نیست :



The screenshot shows a code editor with a file named 'test.py'. The code is: `1 print(int(3.1))`. A red arrow points from the code to the number '3', indicating the output of the program.

اگر یک عدد اعشاری به عدد صحیح تبدیل شود بخش اعشاری آن حذف میشود و اگر عدد صحیح به عدد اعشاری تبدیل شود بخش اعشاری به آن اضافه میشود




The screenshot shows a code editor with a file named 'test.py'. The code is: `1 print(float(3))`. A red arrow points from the code to the number '3.0', indicating the output of the program.

در پایتون آرایه ها وجود ندارند و بجای آنها از سیستم های سازمان دهی شده دیگر استفاده شده است :

Type	List[]	Tuple()	Set{}	Dic{key:value}
Order	Y	Y	N	Y
Changable	Y	N	N	Y
Duplicate	Y	Y	N	Not in keys

```
Click here to ask blackbox
set = {1,2,3,4}

print(set)
print(set)
print(set)
print(set)
print(set)
print(set)
print(set)
print(set)
print(set)
print(set)
```




```
Idler (2)/test/tes
{1, 2, 3, 4}
{1, 2, 3, 4}
{1, 2, 3, 4}
{1, 2, 3, 4}
{1, 2, 3, 4}
{1, 2, 3, 4}
{1, 2, 3, 4}
{1, 2, 3, 4}
{1, 2, 3, 4}
{1, 2, 3, 4}
PS C:\Users\hamid
```

در ست ها درسته که وقتی خروجی میگیریم به ترتیب نشان داده میشوند ولی اگر مقداری رو به اون ها اضافه کنیم معلوم نیست که داخل کدام ایندکس قرار میگیرد یعنی هر بار که مثلا از آن ها خروجی میگیریم شاید هر دفعه در جاهای مختلفی قرار بگیرد.

هم ست ها و هم توپل ها قابل تغییر نیستند :

```
test.py > ...
Click here to ask Blackbox to help you code faster
1 set = {1,2,3,4}
2
3 set[0] = 'a'
4
5 print(Set)
```




```
Traceback (most recent call last):
  File "c:\Users\hamid\Desktop\New folder (2)\test\test.py", line 3, in <module>
    set[0] = 'a'
    ~~~~~^
TypeError: 'set' object does not support item assignment
```

```
Click here to ask blackbox
tuple = (1,2,3,4)

tuple[0] = 'a'

print(tuple)
```




```
Traceback (most recent call last):
  File "c:\Users\hamid\Desktop\New folder (2)\test\test.py", line 3, in <module>
    tuple[0] = 'a'
    ~~~~~^
TypeError: 'tuple' object does not support item assignment
```

در دیکشنری ها هم فقط کی ها یونیک هستند ولی مقادیر اختصاص داده شده به کی ها هرچی میتوانند باشند و اگر چندین کی یکسان داشته باشید فقط آخری را قبول میکند (چون کد ها از بالا به پایین ترجمه میشوند):

```
my_dic = {
    'name' : "erfan",
    'name' : "mamad"
}

print(my_dic)
```



```
y
{'name': 'mamad'}
PS C:\Users\hamid\Desktop\
```

فیلتر کردن :

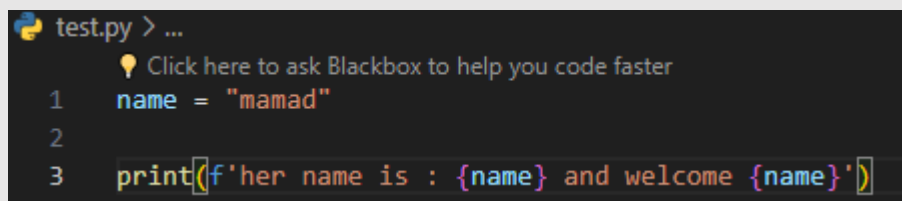
زمانی فرا میرسه که ما میخاییم متغیر هارو در کنار رشته ها نشان دهیم مثلا در سی پلاس پلاس مینوشتیم :

```
>> string name = "mamad";  
>> cout << "her name is : " << name;
```

در پایتون این چنین نیست ما باید از تابع فیلتر برای این کار استفاده کنیم :

1- قبل از رشته یک حرف f بگذارید (با این کار تابع فیلتر را صدا میزنید , خود تابع را در آموزش پیشرفته پایتون ببینید)

2- متغیر را در رشته در کروشه قرار دهید : {variable}




```
test.py > ...  
Click here to ask Blackbox to help you code faster  
1 name = "mamad"  
2  
3 print(f'her name is : {name} and welcome {name}')
```

متد (method) چیست : توابعی هستند که در کلاس های دیتاتایپ ها مثل لیست ها رشته ها توپل ها و .. وجود دارند

متد های رشته ها :

```
test.py > ...  
Click here to ask Blackbox to help you code faster  
1 name = "mamad"  
2  
3 print(name.capitalize())  
4 print(name.casefold())  
5 print(name.count("a"))  
6 print(name.index('d'))  
7 print(name.isalpha())  
8 print(name.isdigit())  
9 print(name.isdecimal())  
10 print(name.isupper())  
11 print(name.islower())  
12 print(name.replace(name[0] , "k"))  
13 print(name.replace("d" , "k"))  
14
```



```
PROBLEMS OUTPUT  
Mamad  
mamad  
2  
4  
True  
False  
False  
False  
True  
kakad  
mamak
```

Capitalize : حرف اول رو بزرگ میکنه

Casefold : تمام رشته رو کوچیک میکنه

Count : مقداری را میگیره و تعداد بار تکرار شده را برمیگرداند

Index : مقداری را میگیرد و شماره ایندکس آنرا در آن برمیگرداند

Isalpha : اگر حاوی فقط حروف الفبا باشد ترو و اگر حتی فقط یک چیز غیر از حروف داشته باشد فالس برمیگرداند

Isdigit : اگر حاوی فقط اعداد باشد ترو و اگر حتی فقط یک چیز غیر از عدد داشته باشد فالس برمیگرداند

Isdecimal : اگر حاوی فقط اعداد اعشاری باشد ترو و اگر اعداد اعشاری نداشته باشد فالس برمیگرداند (اگر عدد غیر اعشاری هم بزارید مثلا 22 باز هم ترو برمیگرداند و فقط زمانی فالس میده که ترکیبی از عدد و حروف داشته باشید)

Isupper : اگر حروف اولش بزرگ باشد ترو برمیگرداند در غیر این صورت فالس

Islower : اگر حروف اولش کوچک باشد ترو برمیگرداند و در غیر این صورت فالس

Replace : جایگزین میکند ایندکس و یا خود المان را با المان داده شده

متد های لیست :

Append : دو یا چند لیست را میگیرد و باهم جمع میکند (حواستون باشه که این متد نمیاد توی یه متغیر جدید بریزه یا برگردونه که بخوایم پرینتش کنیم فقط میاد همون اولی رو دایمی تغییر میده) :

```
Click here to ask Blackbox to help you code faster
mylist = [1, 2, 3, 4, 5, 'a', True, [1,2]]
mylist2 = [False, ["B", False], [1,2]]
mylist.append(mylist2)
print(mylist)
```

→

```
y"
[1, 2, 3, 4, 5, 'a', True, [1, 2], [False, ['B', False], [1, 2]]]
```

اگر بیاییم داخل یک متغیر بریزیم یا پرینت کنیم حاصل رو به ما None برمیگردونه چون به قول سی پلاس پلاس کار ها این یک تابع Void هست یعنی چیزی برنمیگردونه که بخوایم ذخیره در متغیر کنیم یا چاپ کنیم . اگر این کارو کنیم همچین اتفاقی میوفته :

```
test.py > ...
Click here to ask Blackbox to help you code faster
1 mylist = [1, 2, 3, 4, 5, 'a', True, [1,2]]
2 mylist2 = [False, ["B", False], [1,2]]
3
4 sum = mylist.append(mylist2)
5
6 print(sum)
```

→

```
y
None
```



```
test.py > ...
Click here to ask Blackbox to help you code faster
1 mylist = [1, 2, 3, 4, 5, 'a', True, [1,2]]
2 mylist2 = [False, ["B", False], [1,2]]
3
4 print(mylist.append(mylist2))
5
```

→

```
y
None
```

متد append چیز جدیدی درست نمیکنه فقط همون اولیه رو تغییر میده پس اگر بیاییم خود شو بریزیم توی یه متغیر دیگه یا چاپش کنیم چیزی به ما نشون نمیده

Extends : این تابع هر 2 یا چند لیست را باهم مخلوط میکند و مثل بالایی روی لیست اولیه تاثیر میگذارد و فرق آن با بالایی این است که در **append** وقتی لیست هارا قاطی میکنه آیتم های لیست هارو داخل لیست آخری به صورت لیست وارد میکنه ولی این یکی خود آیتم هارا وارد میکنه . میدونم چیزی نفهمیدید پس این عکسای پایین رو بخونید :

The image shows two code snippets comparing the `extend` and `append` methods. In the first snippet, `list1.extend(list2)` is used, and the output is a flat list `[1, 2, 3, 4, 5, 6, 7, 8]`. In the second snippet, `list1.append(list2)` is used, and the output is a list containing the first list and the second list as an element: `[1, 2, 3, 4, [5, 6, 7, 8]]`. Red arrows point from the method calls in the code to the corresponding parts of the output lists.

```
test.py > ...  
Click here to ask Blackbox to help you code faster  
1 list1 = [1,2,3,4]  
2 list2 = [5,6,7,8]  
3  
4 list1.extend(list2)  
5 print(list1)  
y  
[1, 2, 3, 4, 5, 6, 7, 8]
```

```
test.py > ...  
Click here to ask Blackbox to help you code faster  
1 list1 = [1,2,3,4]  
2 list2 = [5,6,7,8]  
3  
4 list1.append(list2)  
5 print(list1)  
y  
[1, 2, 3, 4, [5, 6, 7, 8]]
```

همانطور که میبینید در **append** آیتم های اضافه شده خود در یک لیست قرار گرفته و به عنوان یک لیست اضافه شده اند ولی در **extend** خود آیتم ها به لیست اضافه شده


Insert : یک ایتm میگیرد همراه با یک شماره ایندکس و آن ایتm را به آن ایندکس میچسباند این تابع هم مثل تابع بالایی چیزی برنمیگرداند و روی لیست اولیه تاثیر میزاره. حواستون باشه که اون ایندکسی که جایگزینش کردید رو پاک نمیکنه , حولش میده میره جلو!

The image shows a code snippet using the `insert` method. It creates a list `list1` with elements `[1, 2, 3, 4]` and inserts the string `'x'` at index `0`. The output is `['x', 1, 2, 3, 4]`. A red arrow points from the index `0` in the code to the first element in the output list.

```
test.py > ...  
Click here to ask Blackbox to help you code faster  
1 list1 = [1,2,3,4]  
2 list1.insert(0, 'x')  
3 print(list1)  
['x', 1, 2, 3, 4]
```

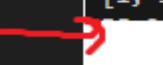
Clear : لیست را پاک سازی میکند :

```
test.py > ...
Click here to ask Blackbox to help you
1 list1 = [1,2,3,4]
2 list1.clear()
3 print(list1)
```



Sort : میاد و لیست رو اول از لحاظ الفبایی و بعد از لحاظ ریاضی ترتیب بندی میکنه : در نسخه های قدیمی پایتون میشد اعداد و حروف رو کنار هم گذاشت ولی الان نه لیست تون باید فقط از یک نوع باشه :


```
test.py > ...
Click here to ask Blackbox to help you
1 list1 = [6,3,7,1,34,222]
2 list1.sort()
3 print(list1)
```



اسلایسینگ لیست ها و کلن هر چیزی که قابل پیشمایش باشه :

یادتون باشه هر وقت ما چیزی رو اسلایس میکنیم عدد مقصد شامل نمیشود یعنی مثلا [2:5] از 2 به ما میده تا ایتمی که ایندکس 4 دارد یعنی مقصد شامل نمیشود

```
test.py > ...
Click here to ask Blackbox to help you code faster
1 list = ['a' , 'b' , 'c' , 'd']
2
3 print(list[0:2])
```



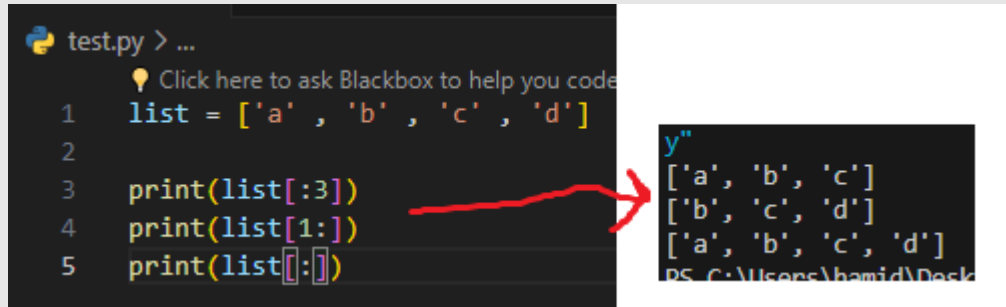
همانطور که میبینید آخرین ایندکس یعنی ایندکس شماره 2 که حرف c را حمل میکند چاپ نشده

حال اگر هر کدام از طرف هارو خالی بزاریم معنی خواصی دارد :

اگر شروع را خالی بزاریم یعنی از اولش شروع کن

اگر پایان رو خالی بزاریم یعنی تا تهش برو (آخرین رو هم میاره برخلاف مسئله قبل)

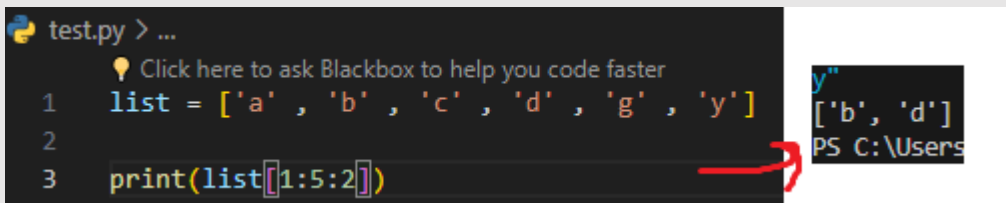
اگر هر دو طرف رو خالی بزاریم میاد و هرچی داره رو برمیگردونه حتی آخرین آیتم



```
test.py > ...
Click here to ask Blackbox to help you code faster
1 list = ['a', 'b', 'c', 'd']
2
3 print(list[:3])
4 print(list[1:])
5 print(list[:])
```

```
y"
['a', 'b', 'c']
['b', 'c', 'd']
['a', 'b', 'c', 'd']
PS C:\Users\hamid\Desktop
```

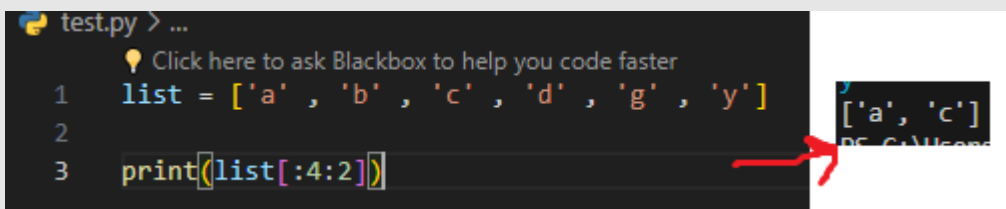
اگر یک دونقطه دیگر هم بزاریم بعدش میاد و step هارو مشخص میکنه مثلا : [2:5:2]
این یعنی از خود 2 شروع کن و تا نخود 5 برو بجلو و 2 تا 2 تا برو



```
test.py > ...
Click here to ask Blackbox to help you code faster
1 list = ['a', 'b', 'c', 'd', 'g', 'y']
2
3 print(list[1:5:2])
```

```
y"
['b', 'd']
PS C:\Users\hamid\Desktop
```

حال ترکیبی از این ها :



```
test.py > ...
Click here to ask Blackbox to help you code faster
1 list = ['a', 'b', 'c', 'd', 'g', 'y']
2
3 print(list[:4:2])
```

```
y"
['a', 'c']
PS C:\Users\hamid\Desktop
```

میگه از بیخ شروع کن و تا نخود 4 برو جلو و 2 تا 2 تا بخونش