تابع

توابع در پایتون به صورت زیر با کلمه کلیدی `def` تعریف می‌شوند و محدوده مربوط به تابع باید یک `tab` جلوتر از تعریف تابع و بقیه‌ی کد باشد. خروجی تابع را هم با کلمه‌ی کلیدی `return` برمی‌گردانند.

Copy Python

```
>>> def add(x, y):
...     print("x is {} and y is {}".format(x, y))
...     return x + y
...
>>> add(5,6)
x is 5 and y is 6
11
```

می‌توان با استفاده از پَک کردن چند متغیر به صورت تاپل، چند متغیر را با هم خروجی داد.

```
>>> def two(x,y):
...     return x+y , x-y
...
>>> a , b = two(5,10)
>>> a
15
>>> b
-5
```

آرگومان‌های کلمه‌ای (*kwargs*)

می توان آرگومان‌های یک تابع را بر اساس اسم آنها مقداردهی کرد و در این حالت مهم نیست که به چه ترتیبی باشند.

```
>>> def add(x, y):
...     print("x is {} and y is {}".format(x, y))
...     return x + y
...
>>> add(y=15,x=4)
x is 4 and y is 15
19
```

می‌توان به آرگومان‌ها مقدار پیش‌فرض داد. در این صورت لازم نیست تا حتما موقع فراخوانی تابع مقداردهی شوند.

```
>>> def add(x=1,y=2):
...     return x+y
...
>>> add()
3
>>> add(5)
7
```

```
>>> add(y=3)
4
```

- **نکته**: آرگومان‌های بدون مقدار پیش‌فرض، نمی‌توانند بعد از آرگومان‌های دارای مقدار پیش‌فرض بیایند.

بسته‌بندی/باز کردن آرگومان‌ها (*packing/unpacking*)

می‌توان آرگومان‌های پوزیشنی(*positional arguments*) را به صورت تاپل و آرگومان‌های کلمه‌ای(*keyword arguments*) را به صورت دیکشنری بسته‌بندی کرد. با این روش می‌توان به هر تعدادی آرگومان از کاربر گرفت. هم‌زمان با وجود اینها می‌توان آرگومان‌های معمولی را نیز از کاربر دریافت کرد.

```
>>> varargs(1,2,5,6)
(1, 2, 5, 6)
>>> a = [1,4,3,12]
>>> varargs(*a)
(1, 4, 3, 12)
>>> varargs(*a , 9)
(1, 4, 3, 12, 9)
```

همان طور که می‌بینید می‌توان یک لیست یا دیکشنری از قبل تعریف شده را به آرگومان‌های یک تابع، باز (*unpack*) کرد.

```
>>> def keyword_args(**kwargs):
...     return kwargs
...
>>> keyword_args(big="foot", loch="ness")
{'big': 'foot', 'loch': 'ness'}
>>> d = {"one":1 , "two":2 , "three":3}
>>> keyword_args(**d , four=4)
{'one': 1, 'two': 2, 'three': 3, 'four': 4}
```

دیکشنری‌ها را به وسیله‌ی `**` و لیست/تاپل‌ها را به وسیله‌ی `*` می‌توان باز یا بسته‌بندی کرد. همچنین می‌توان همزمان از بسته‌بندی آرگومان‌های کلمه‌ای و پوزیشنی استفاده کرد.

```
>>> def all_the_args(*args, **kwargs):
...     print(args)
...     print(kwargs)
...
>>> all_the_args(1, 2, a=3, b=4)
(1, 2)
{'a': 3, 'b': 4}
>>> args = (1, 2, 3, 4)
>>> kwargs = {"a": 3, "b": 4}
>>> all_the_args(*args)
(1, 2, 3, 4)
{}
>>> all_the_args(**kwargs)
()
```

```
{'a': 3, 'b': 4}
>>> all_the_args(*args, **kwargs)
(1, 2, 3, 4)
{'a': 3, 'b': 4}
```

مثال: تابعی برای عوض کردن مقدار دو متغیر:

```
>>> def swap(x, y):
...     return y, x
...
>>> x = 10
>>> y = 20
>>> x , y = swap(x,y)
>>> x
20
>>> y
10
```

محدوده (*Scope*)

متغیرهای داخل تابع در محدوده‌ی تابع هستند. به این معنی که از متغیرهای خارج از تابع مستقل هستند و حتی می‌توانند اسمی مشابه با آنها داشته باشند. ولی می‌توان با کلمه `global` از متغیرهای خارج تابع درون آن استفاده کرد.

```
>>> x = 5
>>> def set_x(num):
...     # Local var x is not the same as global variable x
...     x = num
...     print(x)
...
>>> set_x(43)
43
>>> x
5
```

در کد بالا `x` داخل تابع و `x` خارج تابع دو متغیر جدا هستند و نمی‌توان با عوض کردن یکی دیگری را تغییر داد.

```
>>> x = 5
>>> def set_global_x(num):
...     global x
...     print(x)   # => 5
...     x = num    # global var x is now set to 6
...     print(x)   # => 6
...
>>> set_global_x(6)
5
6
>>> x
6
```

در کد بالا با استفاده از کلمه‌ی `global` مشخص شده که `x` تعریف شده در بیرون تابع در داخل تابع نیز استفاده می‌شود و به این وسیله تغییر داده شده است.