Function

مزایای استفاده از توابع

- اجتناب از نوشتن دستورات مکرر
- بسته بندی مجموعه از دستورات
- تقسیم یک برنامه بزرگ به چند قسمت
- مخفی سازی جزئیات پیچیده عملکرد تابع
- تمرکز روی کاری که یک تابع انجام می دهد جای جزئیات آن

فراخوانی تابع

- نام تابع به همراه پرانتز باز و بسته
- آرگومان، مقداری که باید برای تابع ارسال شود.

```
>>> type('2')
Output: str
```

- به نتیجه ای که هر تابع بر می گرداند، مقدار بازگشتی گفته می شود.
- مقدار بازگشتی یک تابع را می توان به یک متغیر نسبت دهیم.

```
>>> s = type('2')
>>> print(s)
Output: str
```

مقدار بازگشتی – Return Value

- هر تابع می تونه مقدار بازگشتی داشته باشد.
- به این مثال توجه کنید:

```
>>> V = print(“Hello”)
Output: Hello
>>> print(V)
Output: None
```

- تابعی که چیزی برای برگرداندن نداشته باشد None را برمی گرداند.
- توابع می تواند مقدار و یا مقدارهایی برگردانند.
- مثال:

```
>>> name = input(“enter your name :”)
Output: enter your name : Ali
>>> name
Output: Ali
```

تبدیل نوع داده

- توابع پیش ساخته ای که قادر به تغییر نوع مقدار ورودی هستن (البته در صورت امکان!)

```
print(int('2'))
print(int(5.98))
print(int('Hello'))
print(float('3.1415'))
print(float(6))
print(str(35))
```

Output:

```
2
5
error
3.1415
6.0
'35'
```

تبدیل نوع موقت

- به مثال توجه کنید:

```
>>> min = 59
>>> min/60.0
Output: 0.983333333
```

توابع ریاضی

- ماژول math شامل تعداد بسیار زیادی توابع ریاضی
- ماژول (module) فایلیست تشکیل شده از مجموعه ای از توابع مرتبط و مجتمع
- برای اضافه شدن به فضای کار از دستور import استفاده میشود.

```
>>> import math
```

- برای فراخوانی یک تابع می بایست نام ماژول و نام تابع را هم آورده شوند (البته با یک نقطه جداکننده)

```
>>> print(math.log10(14))
```

ماژول ها - Module

- ماژول یک فایل است شامل مجموعه ای از توابع، متغیرها و ثابت ها
- دسترسی با کمک نمادگذاری نقطه ای انجام می شود.
- برای دسترسی به کل و یا بخشی از ماژول می توانیم از دستور زیر استفاده کنیم:

```
>>> from module import name1, name2, ...
```

- مثال: وارد کردن دو تابع exp و cos

```
>>> from math import exp, cos
```

- همچنین برای دسترسی به تمام محتویات ماژول از عملگر * استفاده می شود.

```
>>> from math import *
```

تعریف یک تابع جدید

- تابع، دنباله ای از دستورات با نام مشخص که عملیات خاصی انجام می دهند.
- الگوی تعریف تابع:

```
def functionName(list of parameters):
    statements
```

- اولین مثال: تعریف یک تابع خالی بدون هیچ پارامتری!

```
def do_nothing():
    pass
```

- کلمه کلیدی pass در پایتون یعنی هیچ کاری نکن!

Local and Global

- در برنامه نویسی دو نوع متغیر مورد استفاده قرار می گیرد؛ سراسری و محلی
- متغیرهای سراسری در همه جای برنامه قابل استفاده هستند.
- مثال:

```
a = ‘I’
b = ‘am’
c = ‘student’
def print_abc():
    print(a,b,c)
>>> print_abc()
```

Output: I am student

Local and Global

- متغیرهای محلی فقط درون تابعی که تعریف شده اند قابل استفاده هستند.
- مثال:

```
def print_d():
    d = ‘hello’
    print(d)
>>> print_d()
Output: hello
>>>print(d)
error
```

ورودی تابع

- هر تابع می تواند آرگومان هایی بصورت ورودی دریافت کند.
- مثال:

```
def print_n(name):
    print(name)
```

- فراخوانی تابع به دو روش انجام میشود؛ positional و keyword
- positional

```
>>> print_n(‘hello')
Output: hello
```

- keyword

```
>>> print_n(name=‘hello')
Output: hello
```

روند اجرا

- ترتیب اجرای برنامه
- تعریف تابع به هیچ وجه روند اجرای برنامه را تغییر نمی دهد.
- فراخوانی تابع شبیه یک راه فرعی در روند اجرای برنامه است.

پارامترها و آرگومان ها

- تفاوت اساسی بین پارامتر و آرگومان
- آرگومان، مقادیری که برای اجرا به تابع پاس داده می شوند.
- پارامتر، متغیری محلی ست که مقادیری که به تابع پاس داده می شوند را دریافت می کند..

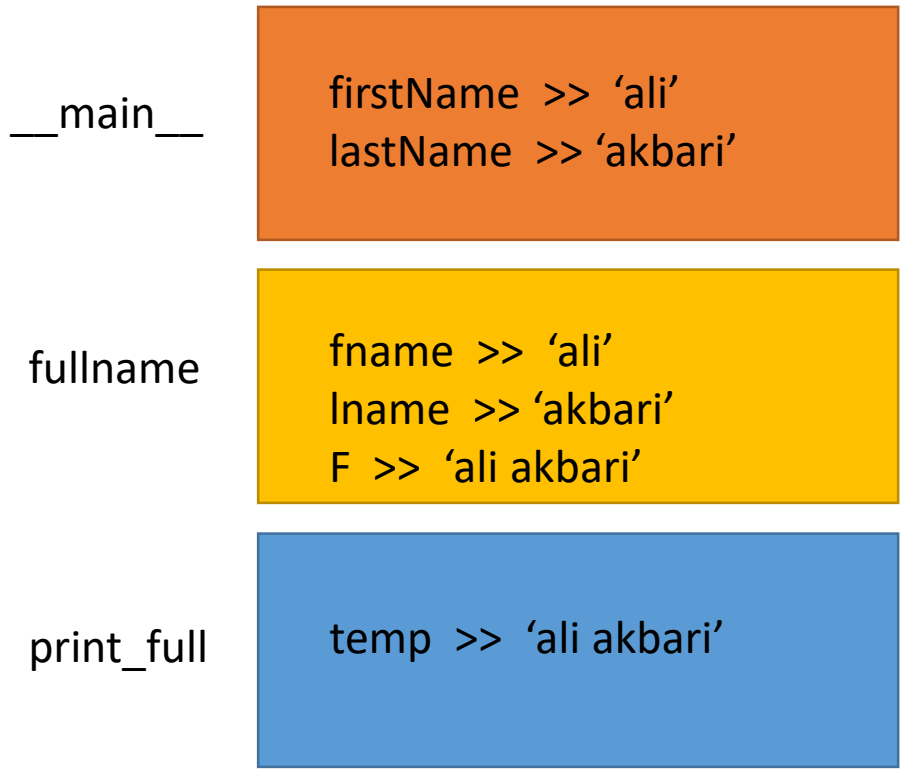
نمودار پشته – Stack Diagram

- جهت ثبت متغیرها و تعیین تابعی که به آن تعلق دارند.
- مثال:

```
def fullname(fname,lname):
    f = fname + lname
    print_full(f)

def print_full(temp):
    print(temp)

>>> firstName = ‘ali’
>>> lastName = ‘akbari’
>>> fullname(firstName,lastName)
Output: ali akbari
```



خروجی

- مقدار بازگشتی یک تابع با دستور return برگردانده می شود.

```
def times_two(thing):
    return thing*thing
>>> times_two(7)
Output: 49
```