برنامه سازی پیشرفته
(توابع: معرفی و کاربرد)

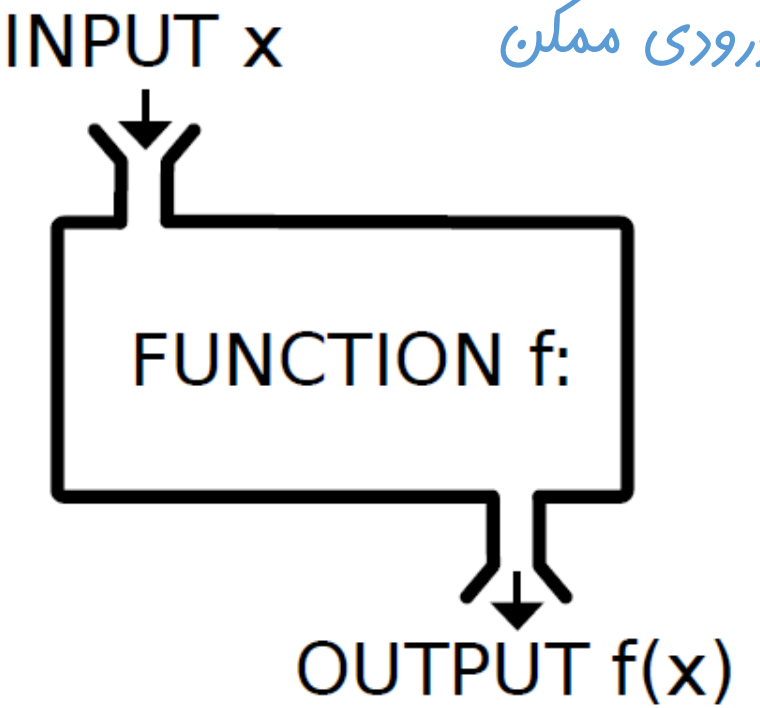
صادق اسکندری – دانشکده علوم ریاضی، گروه علوم کامپیوتر

eskandari@guilan.ac.ir

تابع: تعاریف

هر دستگاهی که یک ورودی را دریافت کرده و بر روی آن عملیاتی انجام داده و یک خروجی تولید کند.

دامنه: مجموعه مقادیر ورودی ممکن



شرط اساسی: برای ورودی های یکسان، خروجی های یکسان تولید کند.

$$\forall x_1, x_2 : if\ (x_1 == x_2)\ then\ f(x_1) == f(x_2)$$

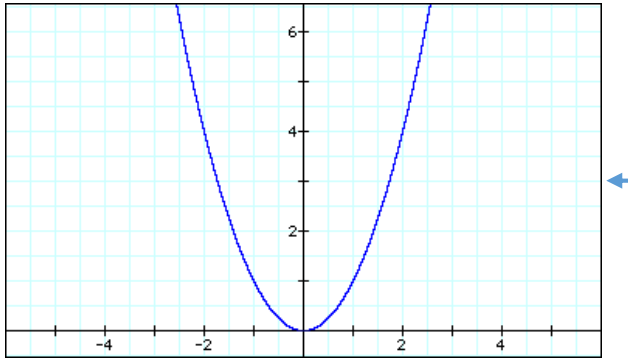
برد: مجموعه مقادیر خروجی ممکن

روش های مرسوم نمایش یک تابع

زوج مرتب ⟵ $\{(1,1),(2,4),(3,9),(4,16),\dots\}$

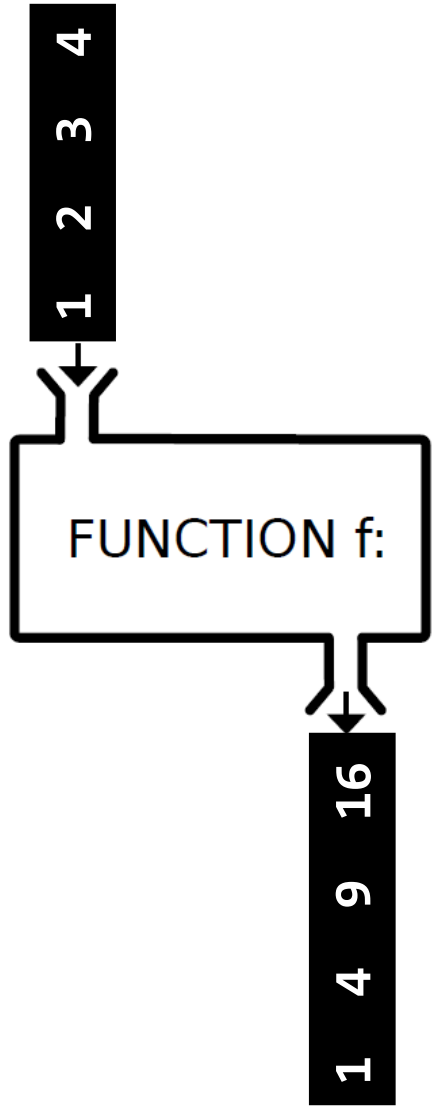
نمایش جبری ⟵ $f(x) = x^2$

نمودار ⟵

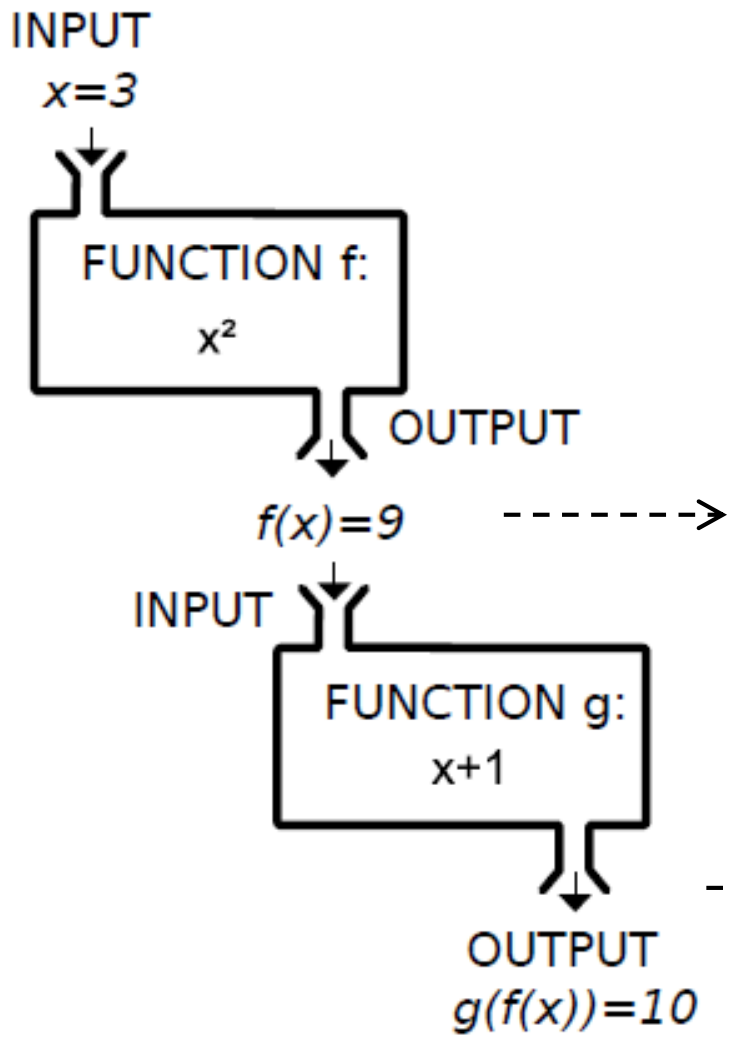


ماژول ⟵

```
def f(x):
    return x * x
```



ترکیب توابع: ورودی یک تابع، خروجی تابعی دیگر باشد.



باید خروجی تابع f، عضوی از دامنه تابع g باشد.

$g(f(x))$

در زبانهای برنامه نویسی از توابع به دو منظور استفاده می شود:

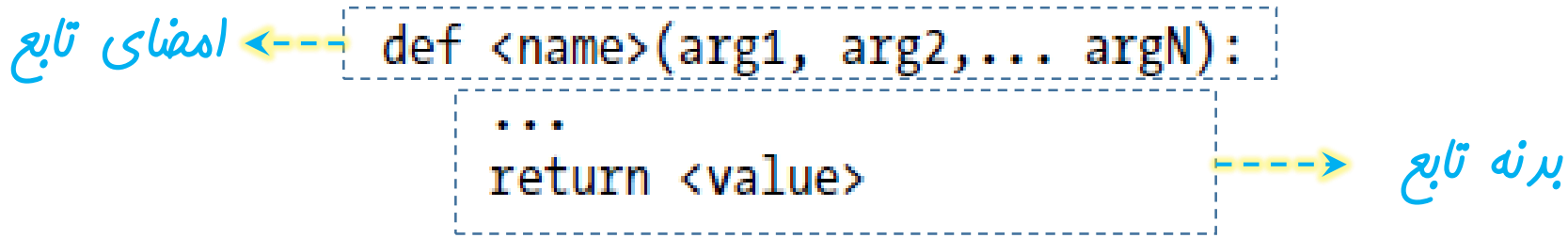
۱- ماژولار سازی برنامه و افزایش ساخت یافتگی

به جای حل یک مسئله پیچیده، آن را به چند مسئله ساده تبدیل کرده و آنها را به شکل مجزا حل می کنیم (برای هر مسئله، یک تابع ایجاد می کنیم). در نهایت، مسئله اصلی از طریق ترکیب توابع ساده حل می شود.

۲- کاهش افزونگی کد در برنامه

برای یک مسئله، یک تابع نوشته می شود و در هر جا و به هر تعداد که نیاز باشد، فراخوانی می شود.

ساختار کلی توابع در زبان پایتون



ساختار کلی توابع در زبان پایتون

نام تابع: باید از قوانین نامگذاری شناسه ها تبعیت کند. بهتر است متناسب با کارکرد تابع انتخاب شود.

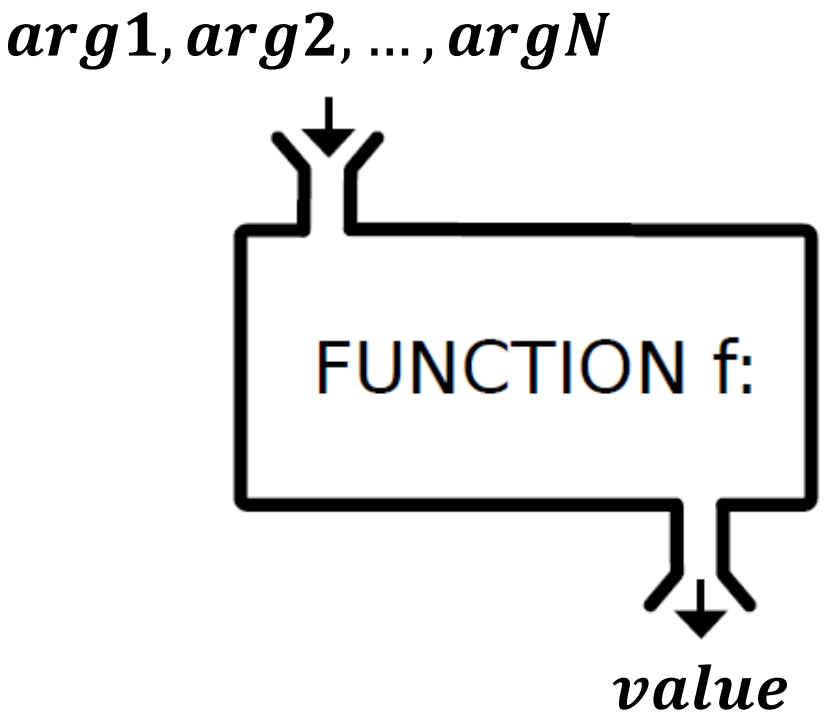
```
def <name>(arg1, arg2,... argN):
    ...
    return <value>
```

آرگومانها یا پارامترهای تابع: ورودی های ممکن تابع را مشخص می کنند. تابع می تواند فاقد آرگومان باشد. هر کدی، در زمان استفاده از تابع باید آرگومانهای تابع را مقداردهی کند.

مقدار بازگشتی تابع: return یک ساختار پرش می باشد. هر زمان درون تابع، دستور return اجرا شود، اجرای تابع خاتمه یافته و نتایج آن برگردانده می شود. یک تابع می تواند فاقد مقدار بازگشتی باشد. علاوه براین، در پایتون یک تابع می تواند بیش از یک مقدار را برگرداند.

ساختار کلی توابع در زبان پایتون

```
def <name>(arg1, arg2,... argN):
    ...
    return <value>
```



تابع در پایتون

مثال: برنامه ای بنویسید که سه مقدار را از ورودی دریافت کرده و تعداد ارقام هر یک را چاپ کند.

راهکار بدون استفاده از تابع

```
1  a = int(input('enter first number: '))
2  b = int(input('enter second number: '))
3  c = int(input('enter third number: '))
4
5  nod = 1
6  while a >= 10:
7      a = a // 10
8      nod = nod + 1
9  print(nod)
10 nod = 1
11 while b >= 10:
12     b = b // 10
13     nod = nod + 1
14 print(nod)
15 nod = 1
16 while c >= 10:
17     c = c // 10
18     nod = nod + 1
19 print(nod)
```

محاسبه تعداد ارقام a

محاسبه تعداد ارقام b

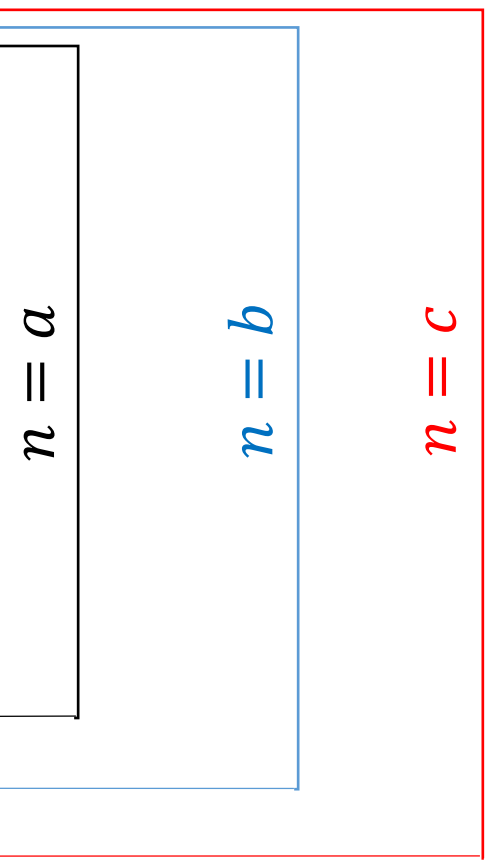
محاسبه تعداد ارقام c

تابع در پایتون

مثال: برنامه ای بنویسید که سه مقدار را از ورودی دریافت کرده و تعداد ارقام هر یک را چاپ کند.

راهکار با استفاده از تابع

```
1  def num_of_digits(n):
2      nod = 1
3      while n >= 10:
4          n = n // 10
5          nod = nod + 1
6      return nod
7
8  a = int(input('enter first number: '))
9  b = int(input('enter second number: '))
10 c = int(input('enter third number: '))
11
12 nod1 = num_of_digits(a)
13 nod2 = num_of_digits(b)
14 nod3 = num_of_digits(c)
15
16 print(nod1)
17 print(nod2)
18 print(nod3)
```



مثال: برنامه ای بنویسید که مجموع تعداد ارقام کلیه اعداد مابین ۲۳۵۰ و ۱۲۸۴۰ را محاسبه و چاپ کند.

```python
def num_of_digits(n):
    nod = 1
    while n >= 10:
        n = n // 10
        nod = nod + 1
    return nod

sum = 0
for i in range(2350, 12841):
    sum += num_of_digits(i)

print(sum)
```

```
44805
```

مثال: برنامه ای بنویسید که مجموع تعداد ارقام کلیه اعداد اول مابین ۲۳۵۰ و ۱۲۸۴۰ را محاسبه و چاپ کند.

```python
import math
def is_prime(n):
    for i in range(2, int(math.sqrt(n))+1):
        if n%i == 0:
            return False
    return True

def num_of_digits(n):
    nod = 1
    while n >= 10:
        n = n // 10
        nod = nod + 1
    return nod

sum = 0
for i in range(2350, 12841):
    if(is_prime(i)):
        sum += num_of_digits(i)
print(sum)

```

```
5029
```

تابع در پایتون

مثال: برنامه ای بنویسید که عدد صحیح n را از ورودی دریافت کرده و اول بودن مجموع ارقام عبارت زیر را محاسبه کند:

$$1 + 2 + 4 + 9 + \cdots + n^2$$

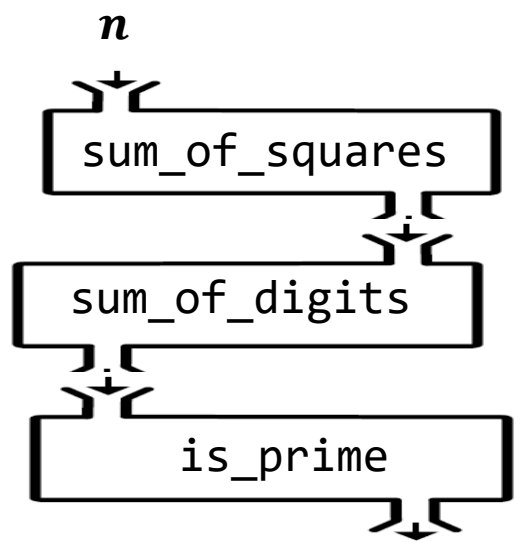
```python
import math
# a function to calculate 1+4+9+ ... + n**2
def sum_of_squares(n):
    sum = 0
    for i in range(1,n+1):
        sum += i**2
    return sum

# a function to calculate sum of digits
def sum_of_digits(n):
    sum = 0
    while(n != 0):
        sum += n%10
        n //= 10
    return sum

def is_prime(n):
    for i in range(2, int(math.sqrt(n))+1):
        if n%i == 0:
            return False
    return True

n = int(input('enter an integer: '))
print(sum_of_squares(n))
print(sum_of_digits(sum_of_squares(n)))
print(is_prime(sum_of_digits(sum_of_squares(n))))
```

```
enter an integer: 123
627874
34
False
```



نکاتی در خصوص کلمه کلید def

def در واقع یک دستورالعمل است که مجموعه ای از کدها را به یک نام تخصیص می دهد. بنابراین، def می تواند در هر جایی که یک دستورالعمل معمولی قابل بیان است، ظاهر شود (درون یک ساختار کنترلی، درون یک تابع و ...).

نکاتی در خصوص کلمه کلید def

کدهای زیر در پایتون معتبر هستند (تعریف تابع درون یک ساختار کنترلی):

```
a = int(input('enter an integer: '))
if a%2 == 0:
    def f(n):
        for i in range(0,n+1,2):
            print(i)
else:
    def f(n):
        for i in range(1,n+1,2):
            print(i)

b = int(input('enter an other integer: '))
print(f(b))
```

```
enter an integer: 4
enter an other integer: 18
0
2
4
6
8
10
12
14
16
18
```

```
enter an integer: 3
enter an other integer: 18
1
3
5
7
9
11
13
15
17
```

در زبانهای غیر مفسری مانند C++ تعریف تابع به شکل فوق امکان پذیر نیست 😐

نکاتی در خصوص کلمه کلید def

کدهای زیر در پایتون معتبر هستند (تعریف تابع درون یک تابع دیگر):

```
1  def f_out():
2      a = int(input('enter an integer: '))
3      b = int(input('enter an other integer: '))
4      def f_in(x,y):
5          if x>y:
6              return x
7          else:
8              return y
9
10     return f_in(a,n)
11
12 f_out()
13
```

تابع درونی: این تابع خارج از تابع **f_out** قابل دسترس نیست.

در زبانهای غیر مفسری مانند C++ تعریف تابع به شکل فوق امکان پذیر نیست 😐

نکاتی در خصوص نام تابع

در پایتون، نام تابع مانند یک متغیر یا شیء عمل می کند. بنابراین، می توان یک تابع را در متغیر دیگری قرار داد و یا آن را به عنوان پارامتر، ارسال کرد.

نکاتی در خصوص نام تابع

کدهای زیر در پایتون معتبر هستند (تخصیص یک تابع به یک شیء دیگر):

```
1  def num_of_digits(n):
2      nod = 1
3      while n >= 10:
4          nod += 1
5          n //= 10
6      return nod
7  digits_num = num_of_digits
8  print(digits_num(12345))
```

در زبانهای غیر مفسری مانند C++ تعریف تابع به شکل فوق امکان پذیر نیست 😐

نکاتی در خصوص نام تابع

کدهای مقابل در پایتون معتبر هستند (ارسال تابع به عنوان پارامتر):

```python
def fun(f,n):
    return f(n)

def num_of_digits(n):
    nod = 1
    while n >= 10:
        nod += 1
        n //= 10
    return nod

def isPrime(n):
    for i in range(2,n):
        if n%i == 0:
            return False
    return True

print(fun(isPrime,13))
print(fun(num_of_digits,13))
```

```
True
2
```

در زبانهای غیر مفسری مانند C++ تعریف تابع به شکل فوق امکان پذیر نیست 😐

در زمان فراخوانی تابع، می توان به شکل های مختلف پارامترها را برای تابع ارسال نمود.

- ارسال بر اساس ترتیب آرگومانها
- ارسال بر اساس نام آرگومانها

```
1  def func(a,b,c):
2      print(a,b,c)
3
4  func(1,3,2)
5  func(a=1, b=3, c=2)
6  func(b=3, c=2, a=1)
7  func(1, c=2, b=3)
```

بر اساس ترتیب آرگومانها

بر اساس نام آرگومانها

به شکل ترکیبی

```
1 3 2
1 3 2
1 3 2
1 3 2
```

بهتر است از روش ارسال بر اساس نام آرگومانها استفاده کنیم 😐

در زمان تعریف توابع، برای برخی از آرگومانها می توان مقادیر پیش فرض در نظر گرفت. بدین ترتیب، در صورتی که در زمان فراخوانی تابع، مقداری برای آرگومان مورد نظر ارسال نشود، از مقدار پیش فرض استفاده خواهد شد.

```
1  def func(a,b,c=3,d=4):
2      print(a,b,c,d)
3
4  func(a=1, b=2)
5  func(a=1, b=2, c=7)
6  #func(a=1)
```

ارسال دو آرگومان اول الزامی و دو آرگومان دیگر اختیاری است.

```
1 2 3 4
1 2 7 4
```