برنامه سازی پیشرفته
(توابع: مفاهیم پیشرفته تر و بازگشتی)

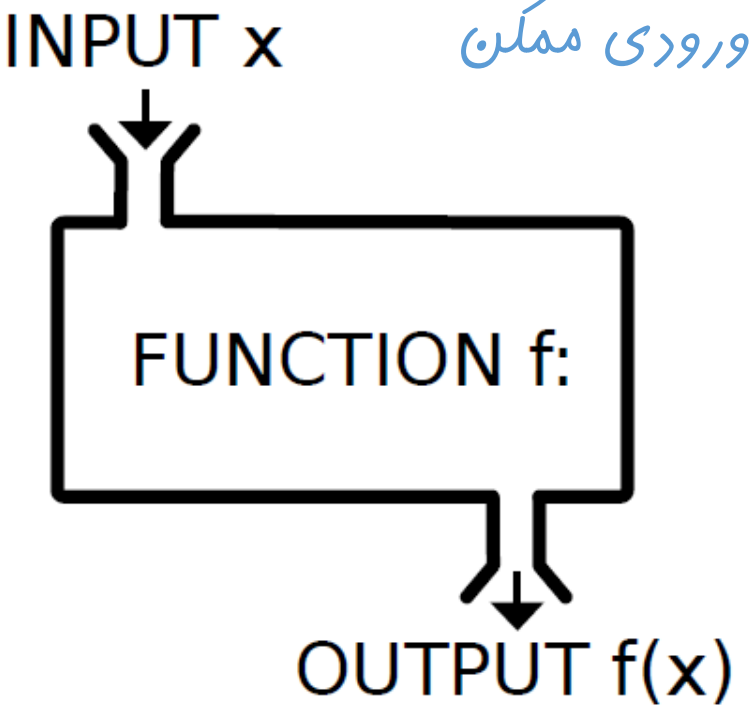
صادق اسکندری – دانشکده علوم ریاضی، گروه علوم کامپیوتر

eskandari@guilan.ac.ir

یادآوری ....

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



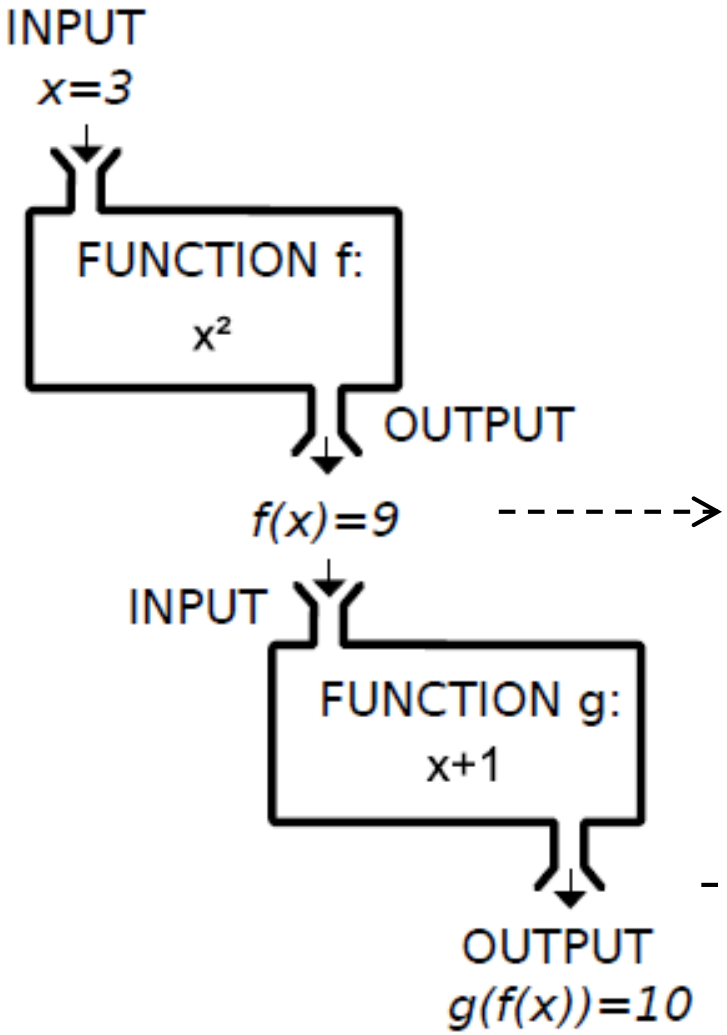
شرط اساسی: برای ورودی های یکسان، خروجی های یکسان تولید کند.

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

برد: مجموعه مقادیر خروجی ممکن

یادآوری ....

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



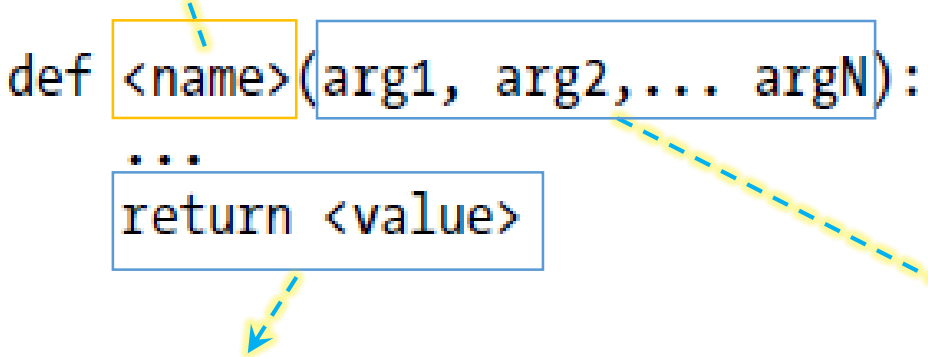
باید خروجی تابع f، عضوی از دامنه تابع g باشد.

$g(f(x))$

یادآوری ....

ساختار کلی توابع در زبان پایتون

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



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.

نکاتی در خصوص 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 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 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 توابع:

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 توابع:

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 توابع:

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 😐

فراخوانی توابع و آرگومانها

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 😐

آرگومانها با مقادیر پیش فرض

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)
```

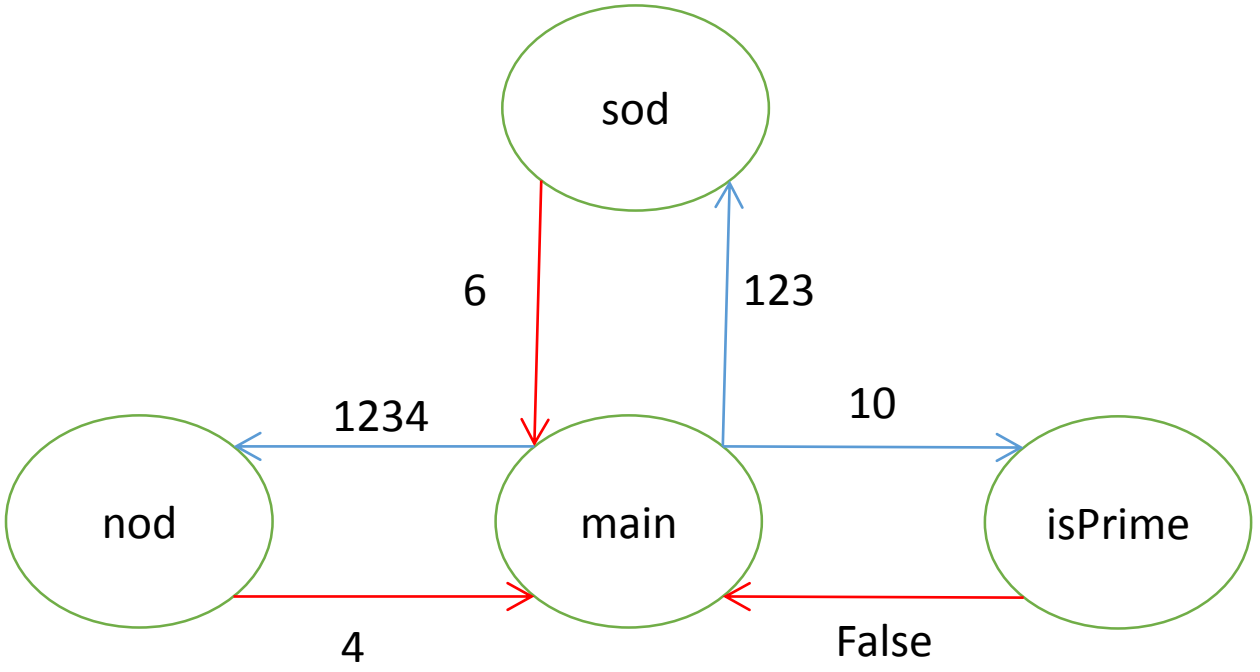
ارسال دو آرگومان اول الزامی و دو آرگومان دیگر اختیاری است.

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

بازگشتی

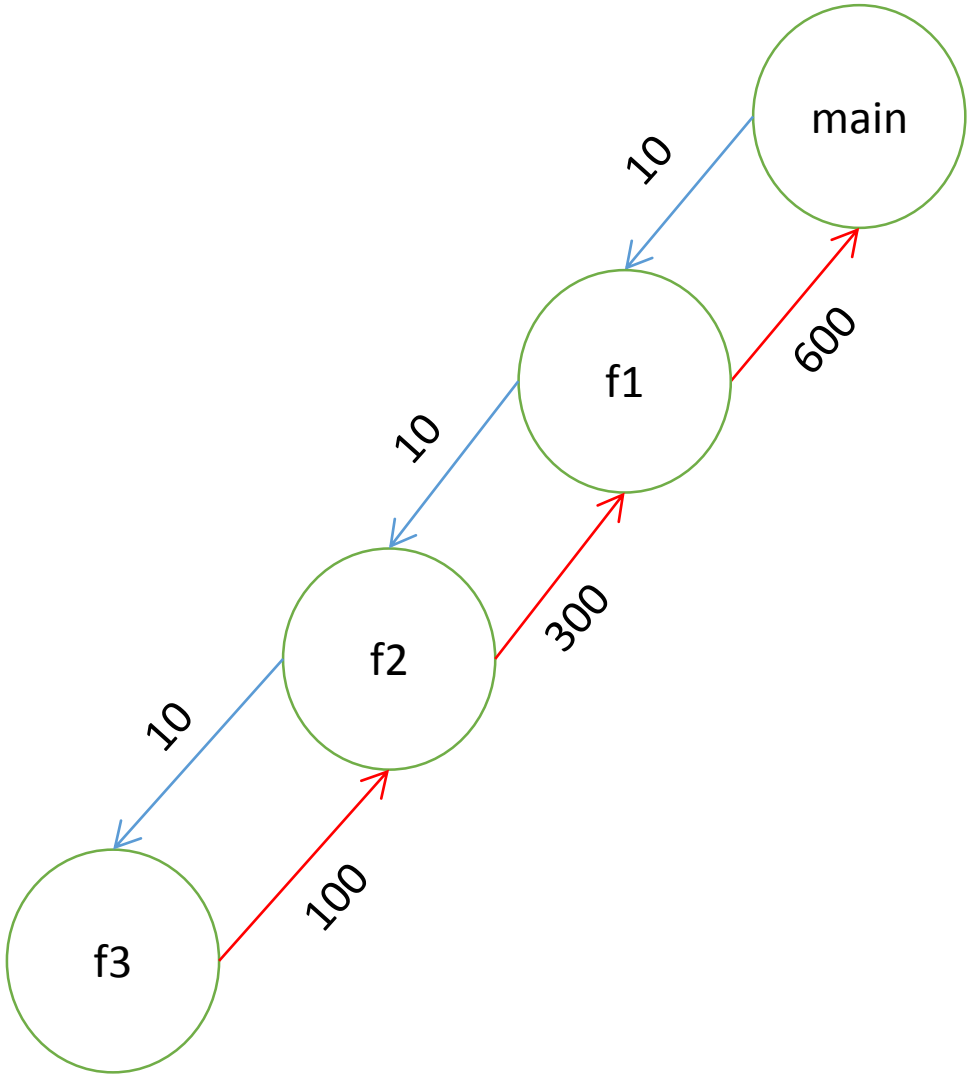
بازگشتی: درخت فراخوانی توابع

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

def sod(n):
    sum = 0
    while(n != 0):
        sum += n%10
        n //= 10
    return sum

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

print(isPrime(sod(123) + nod(1234) ))
```



بازگشتی: درخت فراخوانی توابع

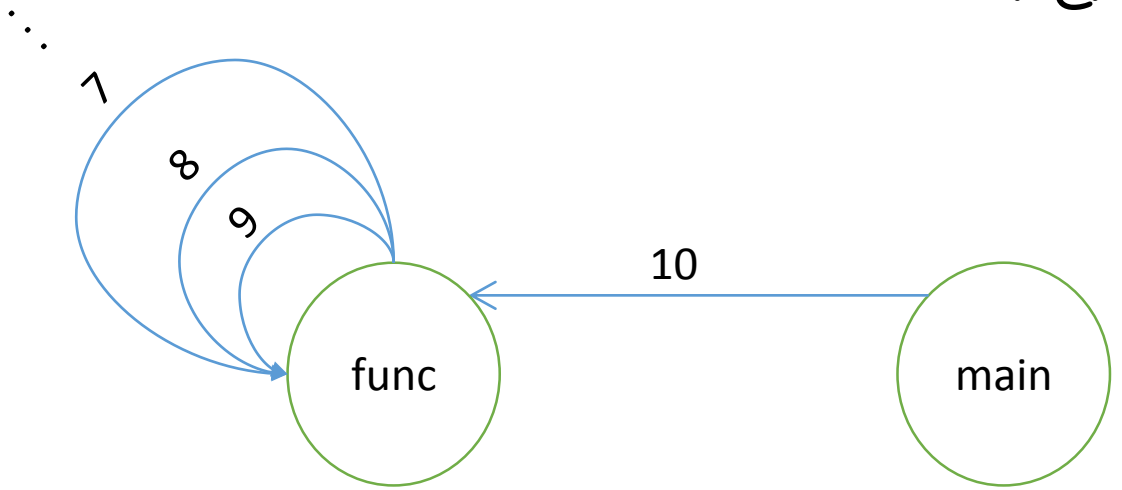
```python
def f1(n):
    a = 2*f2(n)
    return a

def f2(n):
    x = 3*f3(n)
    return x

def f3(n):
    return n**2

f1(10)
```



بازگشتی: تعریف

فراخوانی یک تابع به وسیله خودش

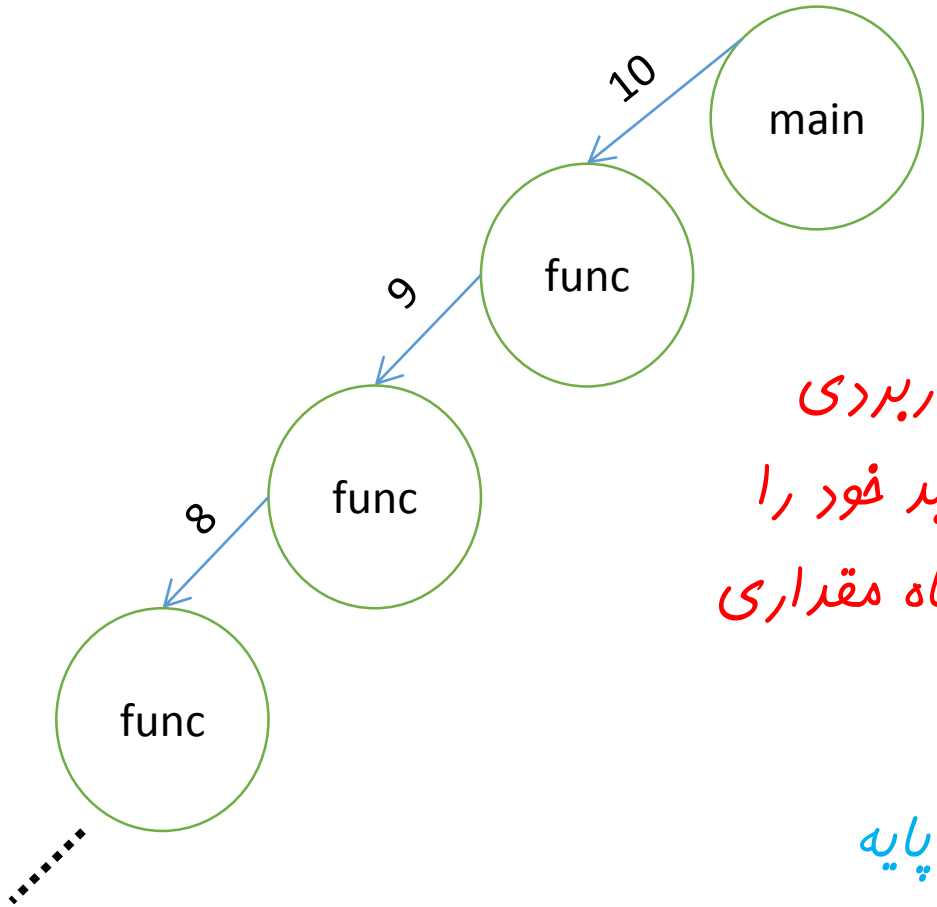
```
1  def func(n):
2      a = func(n-1)
3      return a
4
5  func(10)
```



بازگشتی: تعریف

فراخوانی یک تابع به وسیله خودش

```
1  def func(n):
2      a = func(n-1)
3      return a
4
5  func(10)
```



بازگشتی به این شکل، هیچ کاربردی ندارد زیرا تابع می تواند تا ابد خود را فراخوانی کند و در نتیجه، هیچگاه مقداری برگردانده نمی شود ☹

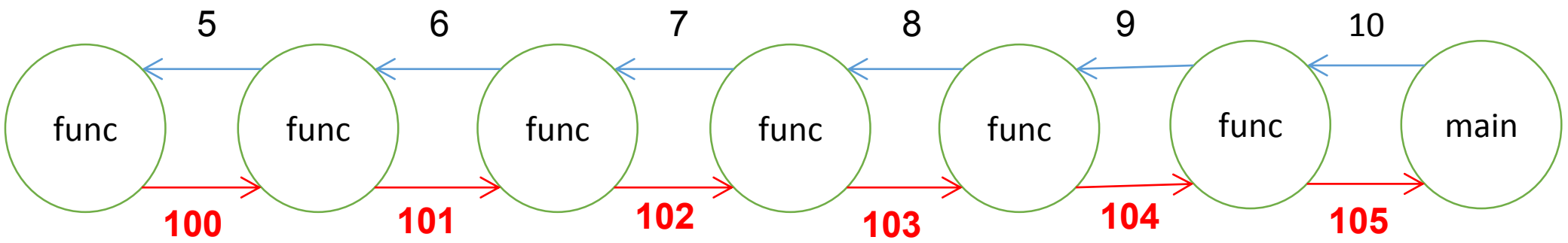
سوال: راهکار چیست؟ مقدار پایه

بازگشتی: مقدار پایه

```
1  def func(n):
2      if n == 5:
3          return 100
4      a = 1+func(n-1)
5      return a
6
7  func(10)
```

بررسی مقدار پایه:

مقدار پایه: ۵



بازگشتی

تمرین: درخت های فراخوانی هر یک از توابع زیر را رسم نموده و خروجی نهایی (آنچه که در صفحه نمایش نشان داده می شود) را برای هر یک مشخص کنید.

```python
def f(n):
    return 1+n**2

def func(n):
    if n == 5:
        return f(n)
    a = f(n) + func(n-1)
    return a

func(10)
```

```python
def func(n):
    print(n)
    if n == 5:
        return 100
    a = 1 + func(n-1)
    print(n)
    return a

func(10)
```