

Семинар 6

Введение в программирование на Python

Папулин С.Ю.
papulin_hse@mail.ru

2015

1. Словари
2. Множества
3. Работа с файлами

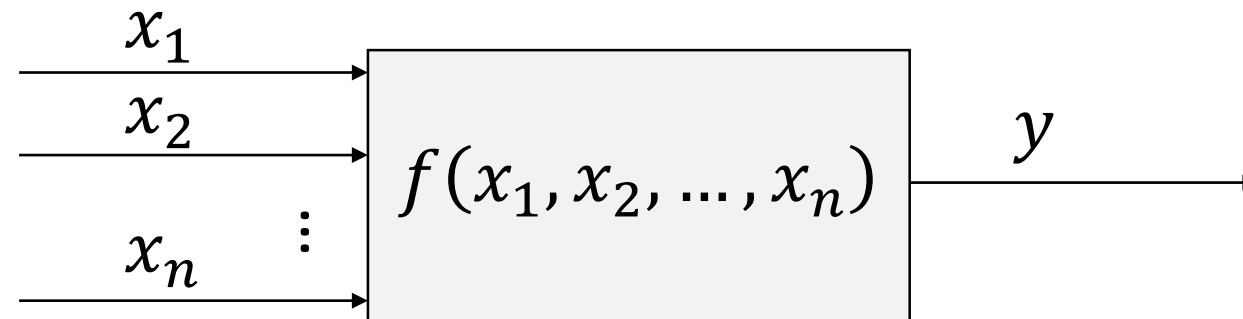
1. Функции
2. Рекурсия
3. Matplotlib

Функции

(см. семинар 4)

Функция

Функция в Python – организованный блок программы, предназначенный для выполнения действий (указанных в теле функции) с возможностью повторного использования. Функция может принимать аргументы и возвращать результат выполнения.



Определение функции

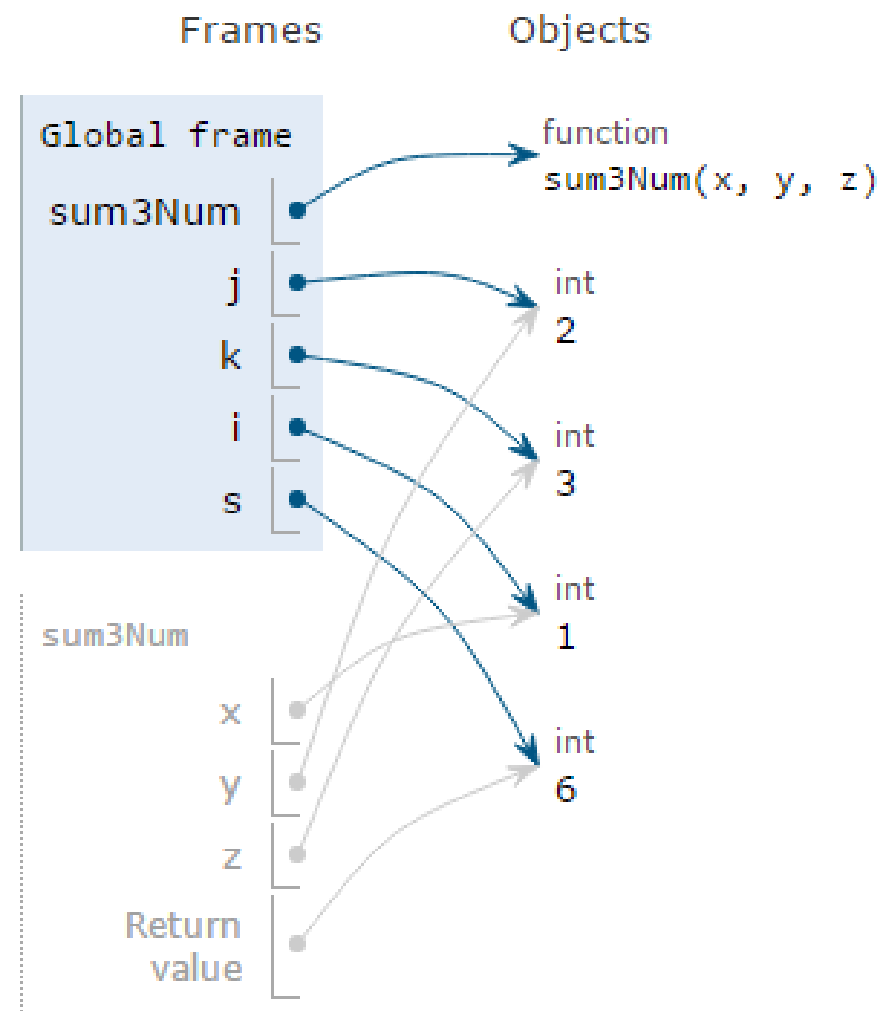
def имя_функции(аргументы):
 действия
 return выражение

```
def sum3Num(x, y, z):  
    return x + y + z
```

```
i = 1; j = 2; k = 3  
s = sum3Num(i, j, k)
```

```
print("Функция sum3Num: 1 + 2 + 3 = %i" % s)
```

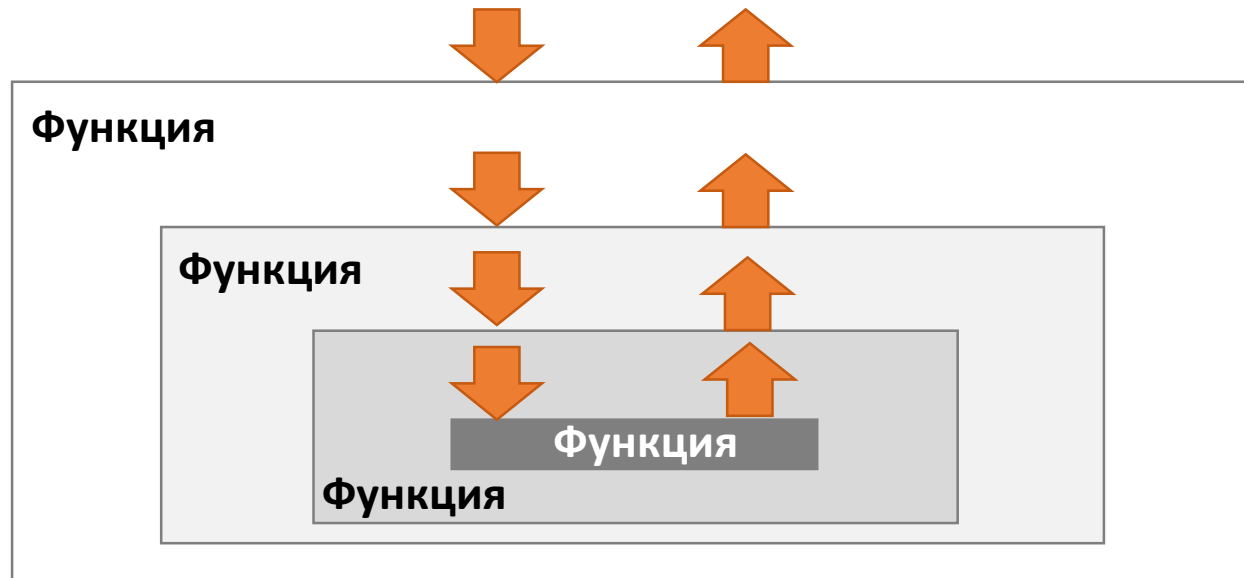
Функция sum3Num: 1 + 2 + 3 = 6



Рекурсия

Рекурсивная функция

Рекурсия в Python – обращение функции к самой себе для решения меньшей задачи



Рекурсия. Факториал

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$$

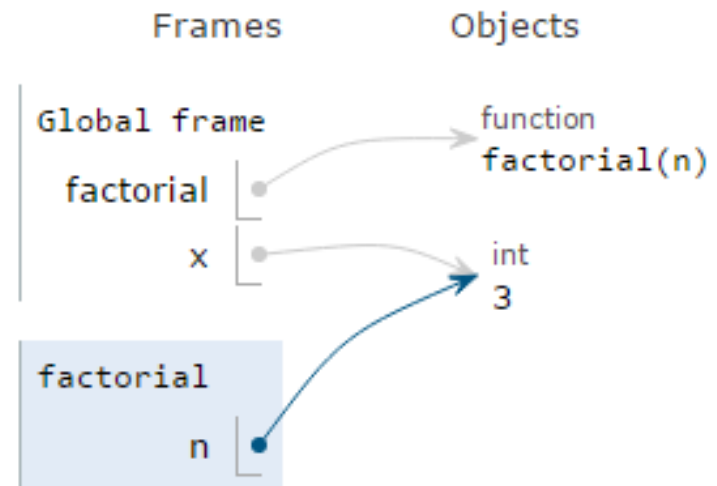
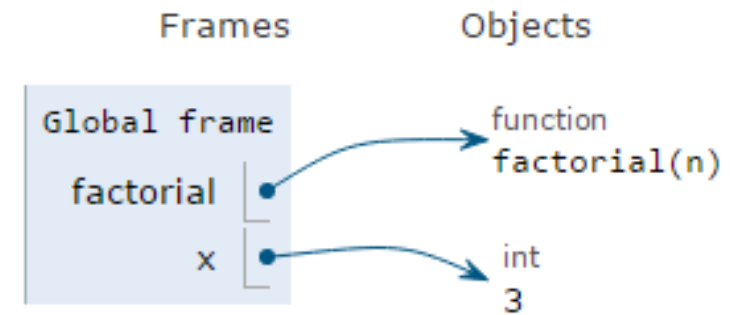
```
def factorial(n):  
    if n == 1:  
        return 1  
    else:  
        return n * factorial(n-1)  
  
x = 3  
y = factorial(x)  
print("Факториал %i! = %i" % (x, y))
```

Факториал $3! = 6$

Рекурсия. Факториал

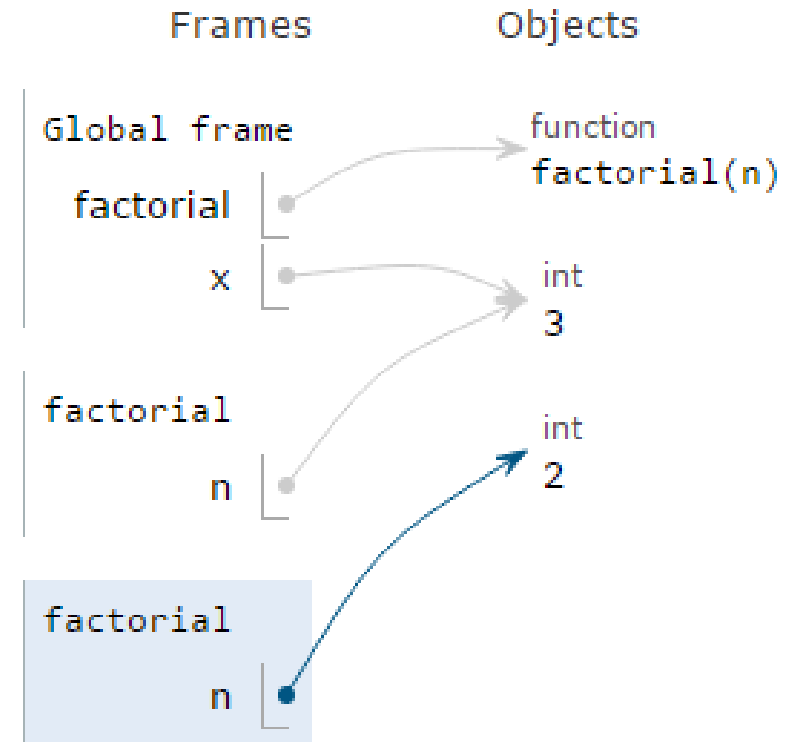
```
1 def factorial(n):
2     if n == 1:
3         return 1
4     else:
5         return n * factorial(n-1)
6
7 x = 3
8 y = factorial(x)
9 print("Факториал %i! = %i" % (x, y))
```

```
1 def factorial(n):
2     if n == 1:
3         return 1
4     else:
5         return n * factorial(n-1)
6
7 x = 3
8 y = factorial(x)
9 print("Факториал %i! = %i" % (x, y))
```



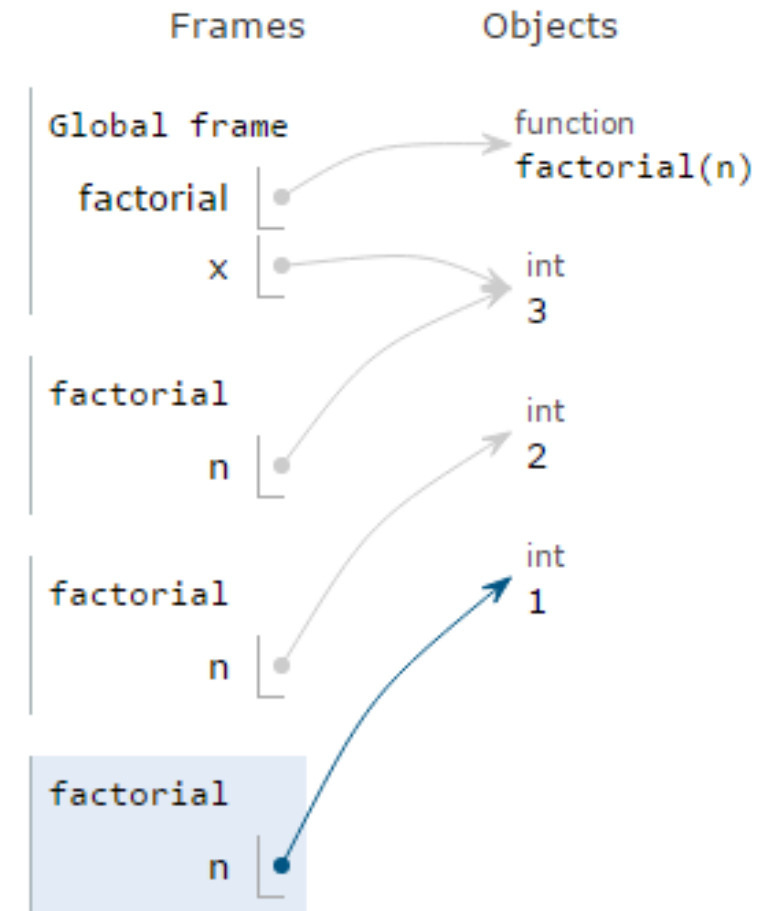
Рекурсия. Факториал

```
→ 1 def factorial(n):  
  2     if n == 1:  
  3         return 1  
  4     else:  
→ 5         return n * factorial(n-1)  
  6  
  7 x = 3  
  8 y = factorial(x)  
  9 print("Факториал %i! = %i" % (x, y))
```



Рекурсия. Факториал

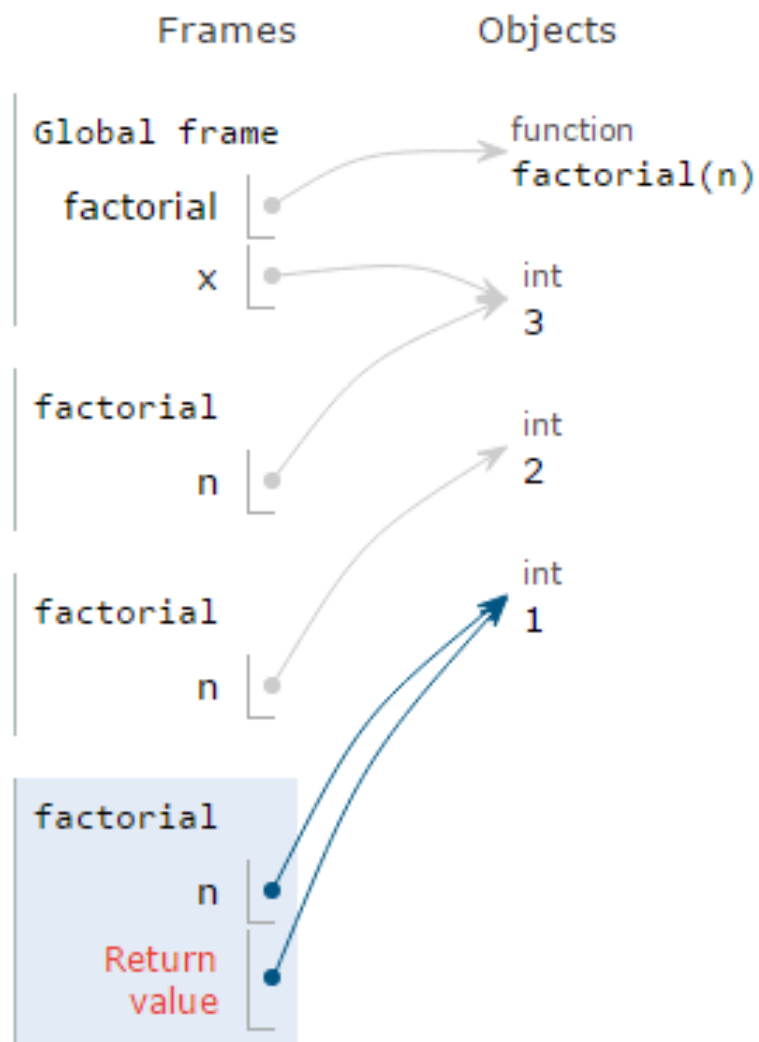
```
→ 1 def factorial(n):  
  2     if n == 1:  
  3         return 1  
  4     else:  
→ 5         return n * factorial(n-1)  
  6  
  7 x = 3  
  8 y = factorial(x)  
  9 print("Факториал %i! = %i" % (x, y))
```



Рекурсия. Факториал

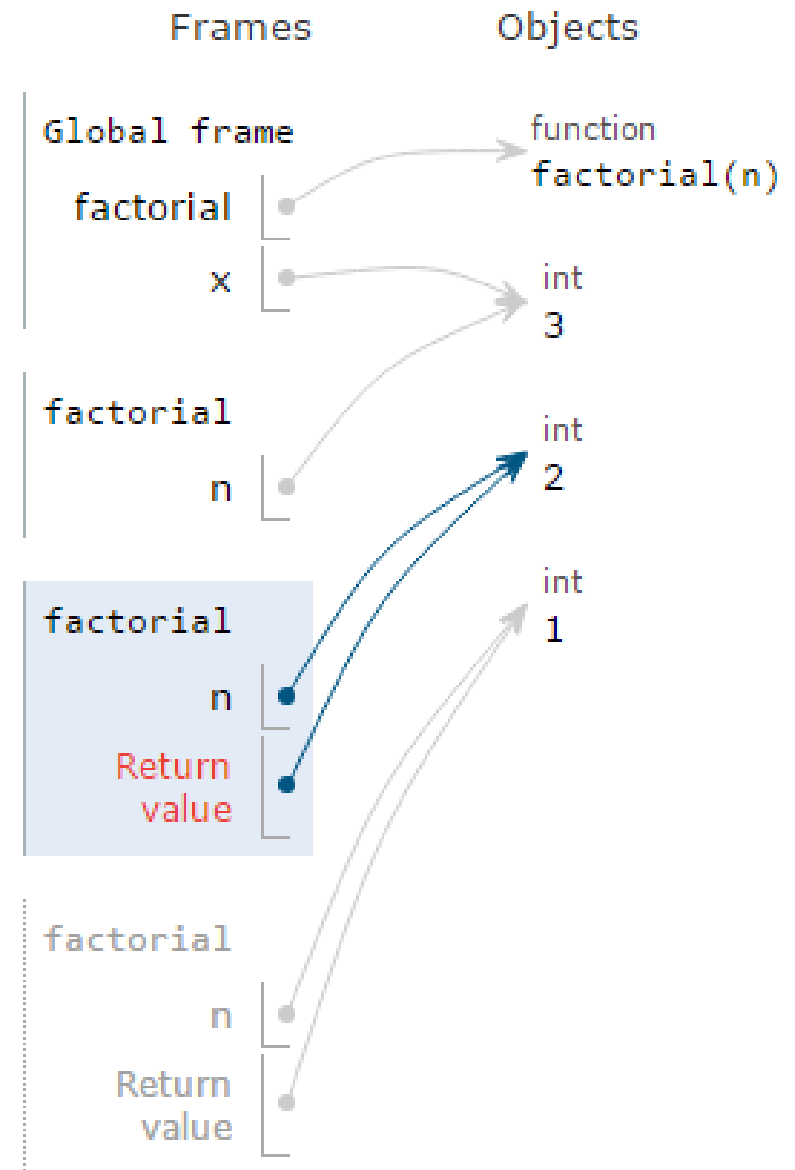
```
→ 1 def factorial(n):  
→ 2     if n == 1:  
3         return 1  
4     else:  
5         return n * factorial(n-1)  
6  
7 x = 3  
8 y = factorial(x)  
9 print("Факториал %i! = %i" % (x, y))
```

```
1 def factorial(n):  
2     if n == 1:  
→ 3         return 1  
→ 4     else:  
5         return n * factorial(n-1)  
6  
7 x = 3  
8 y = factorial(x)  
9 print("Факториал %i! = %i" % (x, y))
```



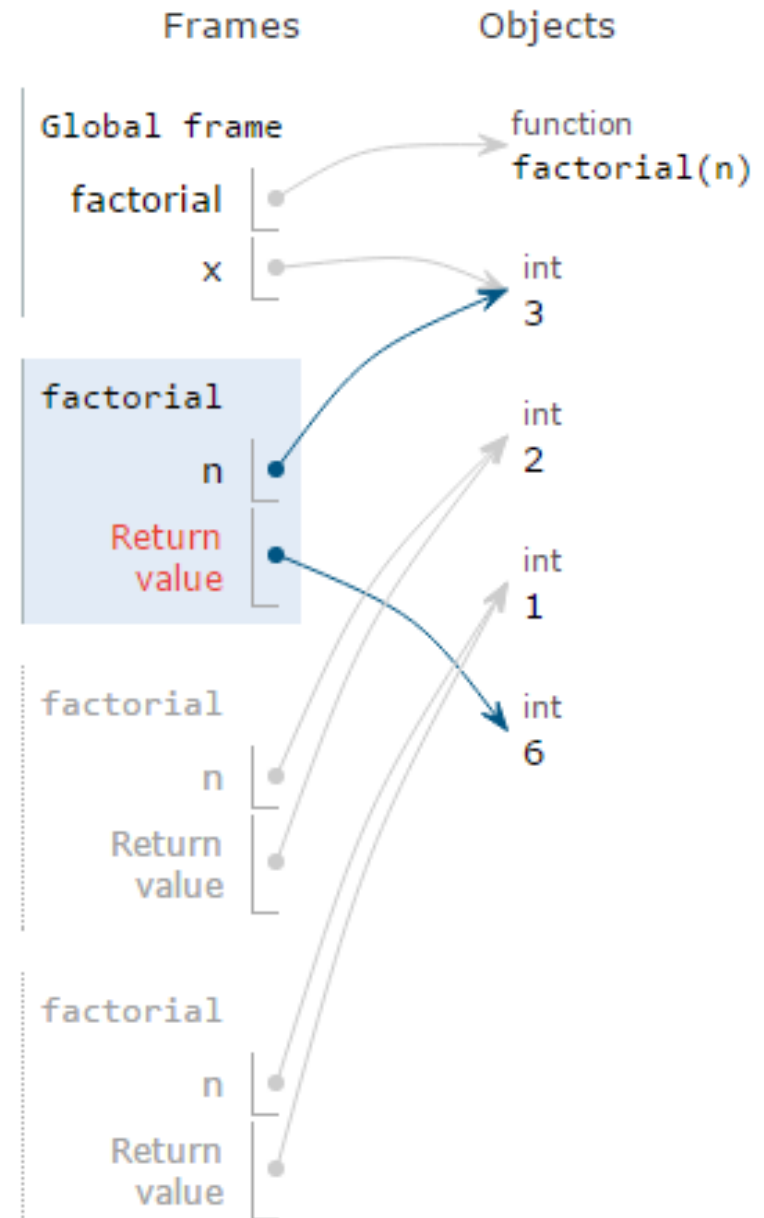
Рекурсия. Факториал

```
1 def factorial(n):  
2     if n == 1:  
3         return 1  
4     else:  
5         return n * factorial(n-1)  
6  
7 x = 3  
8 y = factorial(x)  
9 print("Факториал %i! = %i" % (x, y))
```



Рекурсия. Факториал

```
1 def factorial(n):  
2     if n == 1:  
3         return 1  
4     else:  
5         return n * factorial(n-1)  
6  
7 x = 3  
8 y = factorial(x)  
9 print("Факториал %i! = %i" % (x, y))
```

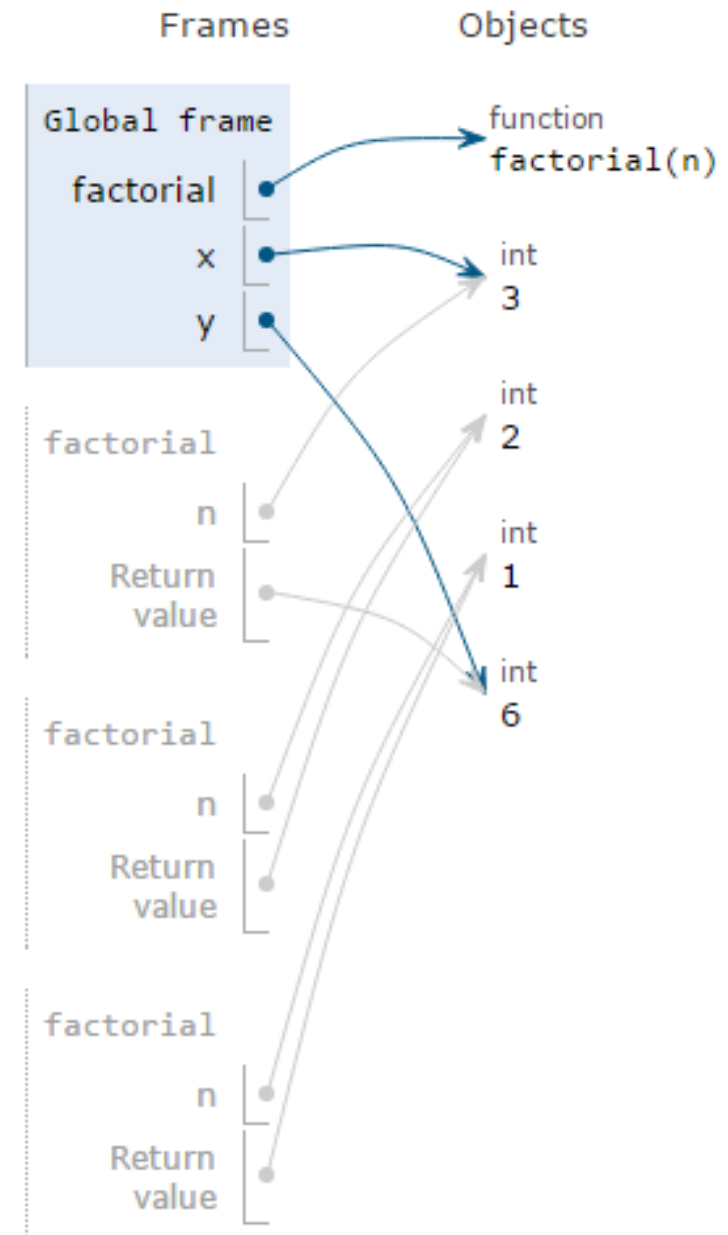


Рекурсия. Факториал

```
1 def factorial(n):  
2     if n == 1:  
3         return 1  
4     else:  
→ 5         return n * factorial(n-1)  
6  
7 x = 3  
8 y = factorial(x)  
→ 9 print("Факториал %i! = %i" % (x, y))
```



Факториал 3! = 6



Рекурсивная функция. Произведение суммами

$$n \cdot m = m + \underbrace{\left(m + \left(m + \left(\dots (m) \right) \right) \right)}_n$$

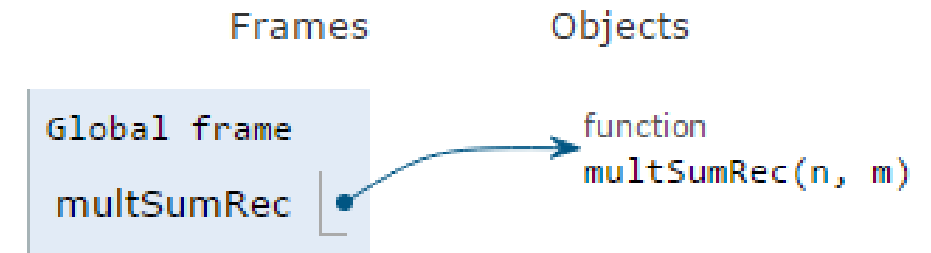
ПРИМЕР: $2 \cdot 3 = 3 + (3)$

$$4 \cdot 3 = 3 + \left(3 + \left(3 + (3) \right) \right)$$

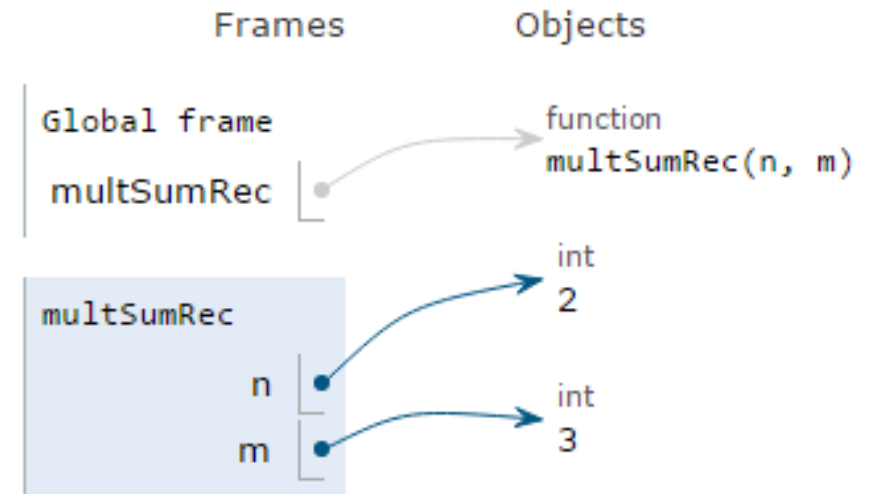
```
def multSumRec(n, m):  
    if (n==1):  
        return m  
    else:  
        return multSumRec(n-1, m) + m  
  
nm = multSumRec(2, 3)  
print(nm)
```

Рекурсивная функция. Произведение суммами

```
→ 1 def multSumRec(n, m):  
  2     if(n==1):  
  3         return m  
  4     else:  
  5         return multSumRec(n-1, m) + m  
  6  
→ 7 nm = multSumRec(2, 3)  
  8 print(nm)
```

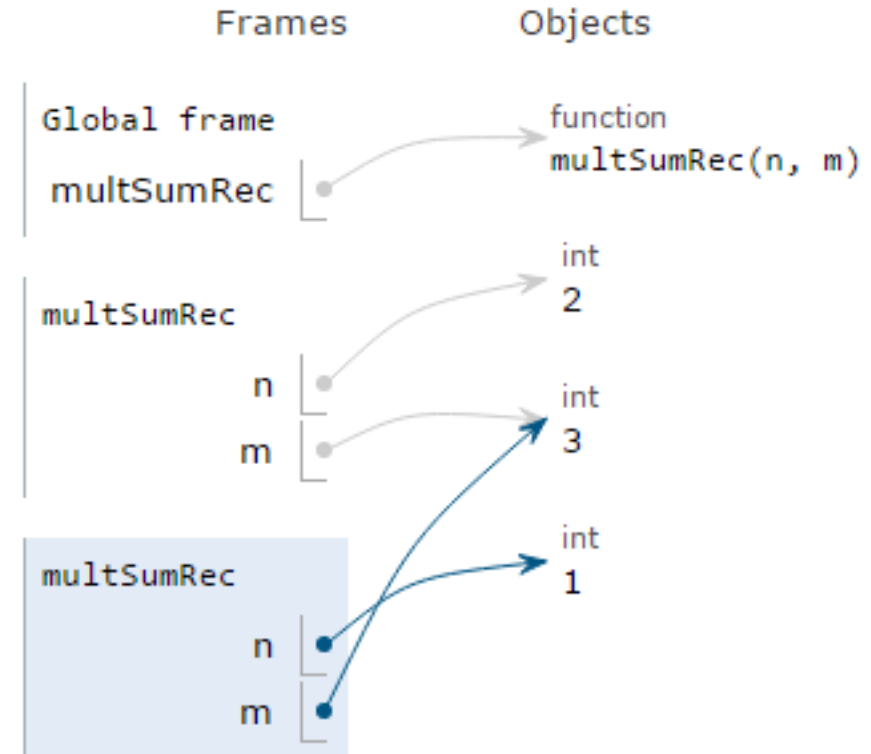


```
→ 1 def multSumRec(n, m):  
  2     if(n==1):  
  3         return m  
  4     else:  
  5         return multSumRec(n-1, m) + m  
  6  
→ 7 nm = multSumRec(2, 3)  
  8 print(nm)
```



Рекурсивная функция. Произведение суммами

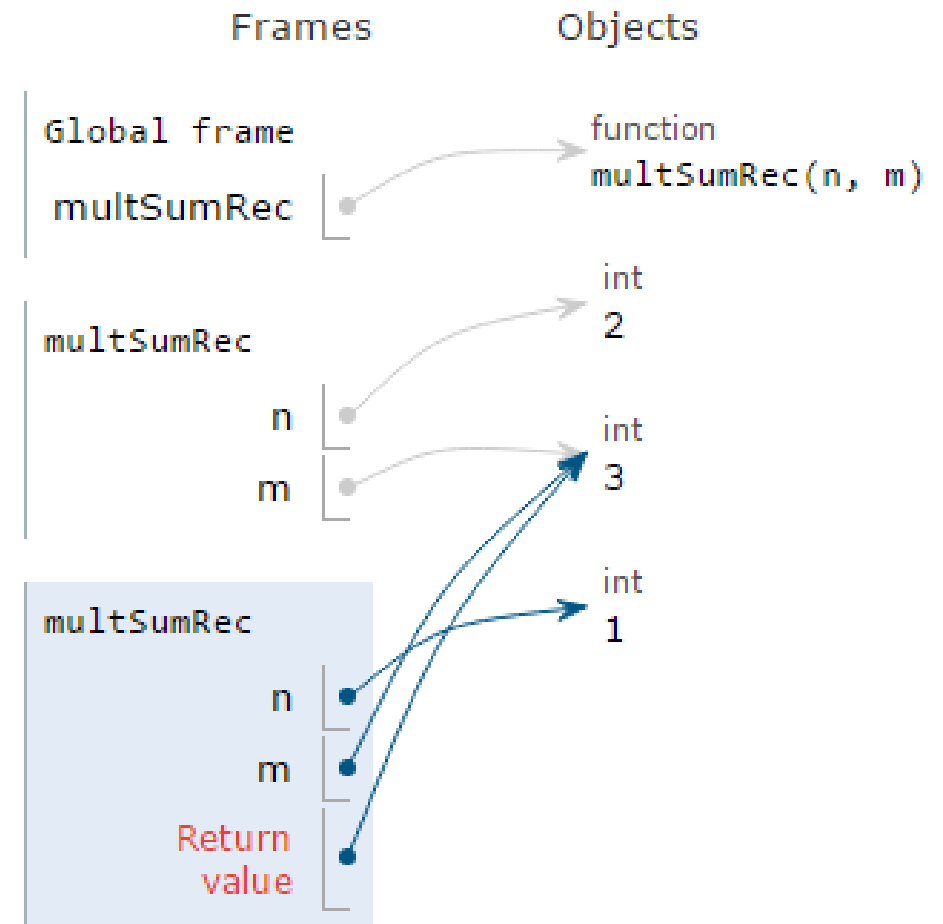
```
→ 1 def multSumRec(n, m):  
  2     if(n==1):  
  3         return m  
  4     else:  
→ 5         return multSumRec(n-1, m) + m  
  6  
  7 nm = multSumRec(2, 3)  
  8 print(nm)
```



Рекурсивная функция. Произведение суммами

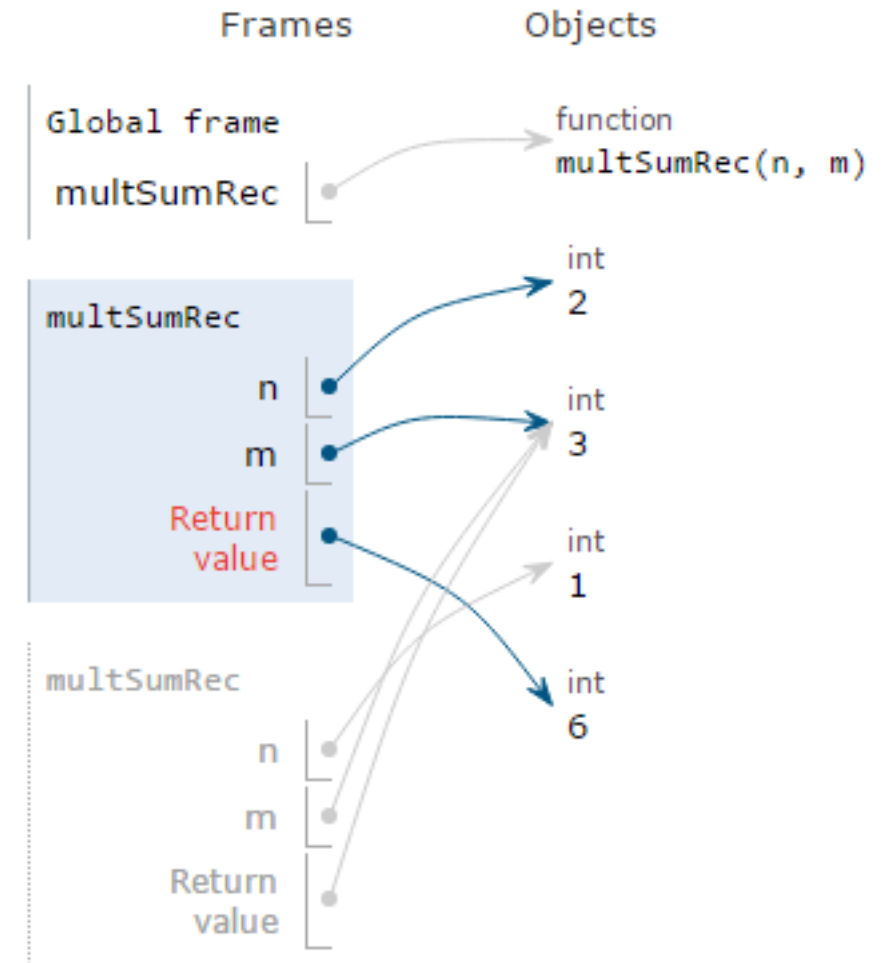
```
→ 1 def multSumRec(n, m):  
→ 2     if(n==1):  
3         return m  
4     else:  
5         return multSumRec(n-1, m) + m  
6  
7 nm = multSumRec(2, 3)  
8 print(nm)
```

```
1 def multSumRec(n, m):  
2     if(n==1):  
→ 3         return m  
→ 4     else:  
5         return multSumRec(n-1, m) + m  
6  
7 nm = multSumRec(2, 3)  
8 print(nm)
```



Рекурсивная функция. Произведение суммами

```
1 def multSumRec(n, m):  
2     if(n==1):  
3         return m  
4     else:  
5         return multSumRec(n-1, m) + m  
6  
7 nm = multSumRec(2, 3)  
8 print(nm)
```

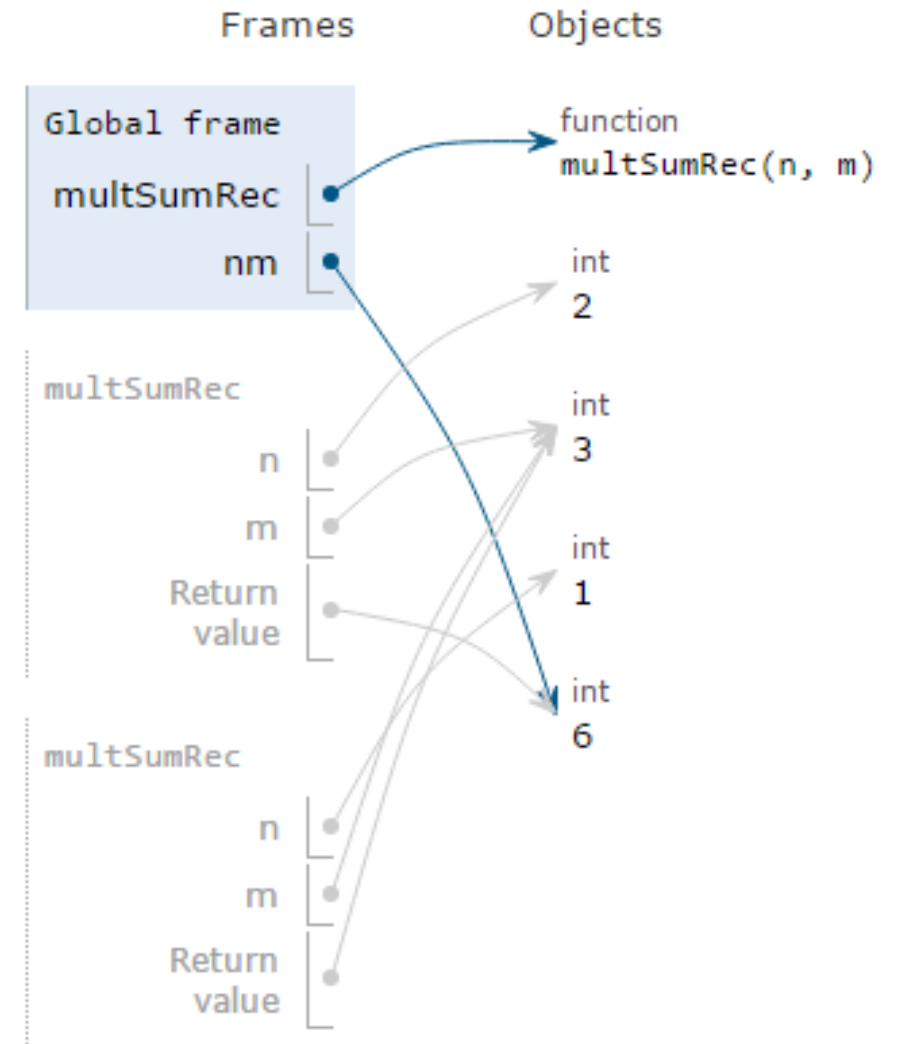


Рекурсивная функция. Произведение суммами

```
1 def multSumRec(n, m):  
2     if(n==1):  
3         return m  
4     else:  
→ 5         return multSumRec(n-1, m) + m  
6  
7 nm = multSumRec(2, 3)  
→ 8 print(nm)
```



6



Рекурсивная функция. Произведение суммами

1 Рекурсия

```
def multSumRec (n, m) :  
    if (n==1) :  
        return m  
    else:  
        return multSumRec (n-1, m) + m
```

2 Цикл WHILE

```
def multSumIter (n, m) :  
    s = 0  
    while (n > 0) :  
        s = s + m  
        n = n - 1  
    return s
```

Рекурсивная функция. Сумма первых N чисел

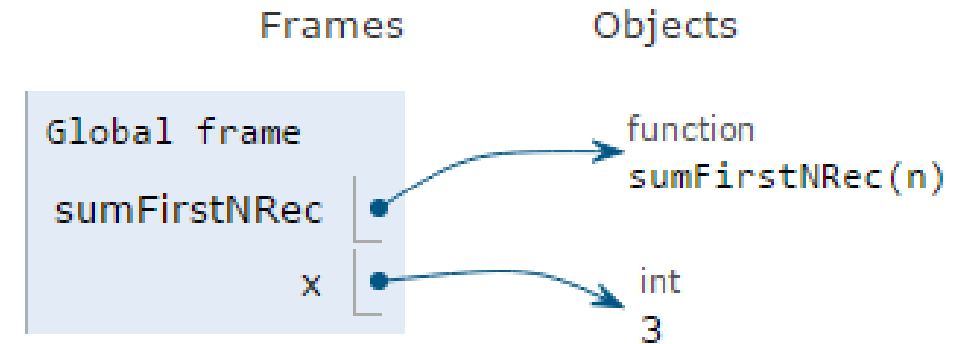
$$1 + 2 + \dots + n = n + \left(n - 1 + \left(n - 1 - 1 + \left(\dots + (1) \right) \right) \right)$$

ПРИМЕР: $1 + 2 + 3 = 3 + (2 + (1))$

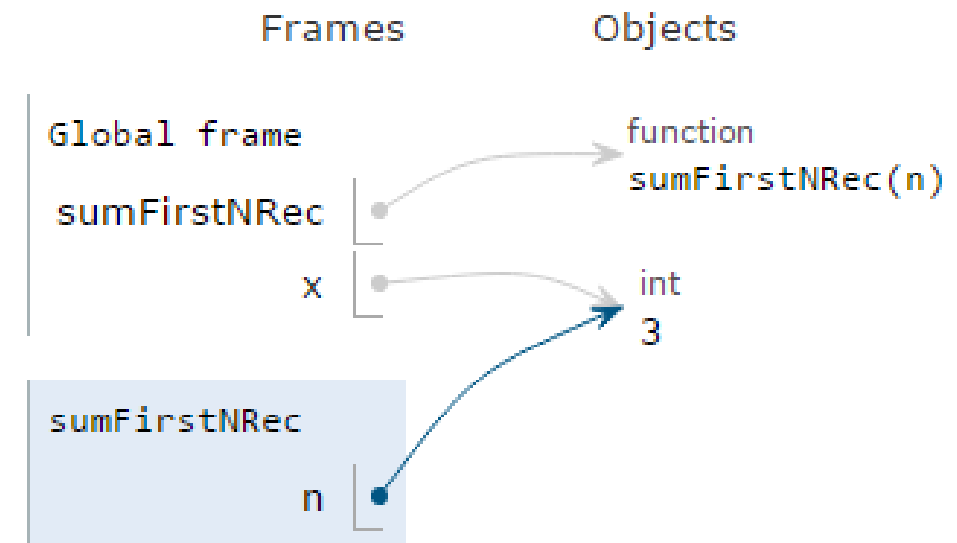
```
def sumFirstNRec(n):  
    if (n==1):  
        return 1  
    return sumFirstNRec(n-1)+n  
  
x = 3  
y = sumFirstNRec(x)  
print(y)
```


Рекурсивная функция. Сумма первых N чисел

```
1 def sumFirstNRec(n):  
2     if (n==1):  
3         return 1  
4     return sumFirstNRec(n-1)+n  
→ 5 x = 3  
→ 6 y = sumFirstNRec(x)  
7 print(y)
```

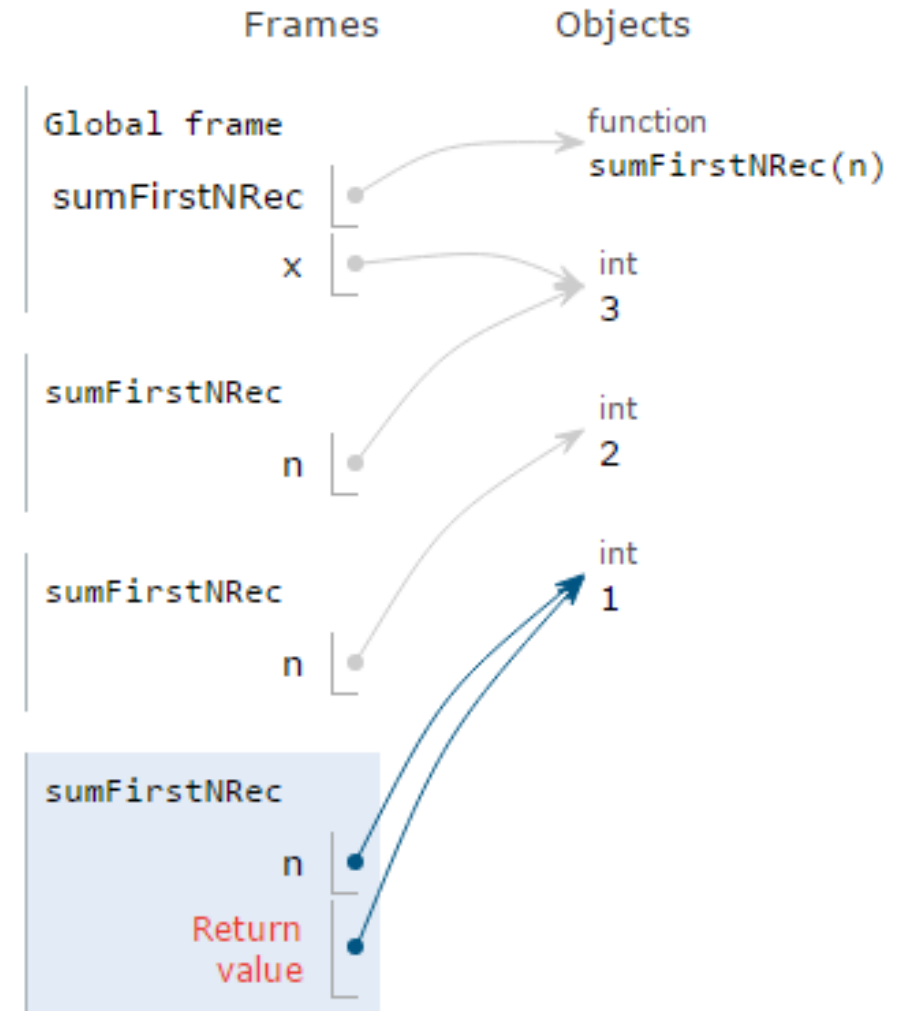


```
→ 1 def sumFirstNRec(n):  
2     if (n==1):  
3         return 1  
4     return sumFirstNRec(n-1)+n  
5 x = 3  
→ 6 y = sumFirstNRec(x)  
7 print(y)
```



Рекурсивная функция. Сумма первых N чисел

```
1 def sumFirstNRec(n):  
2     if (n==1):  
3         return 1  
4     return sumFirstNRec(n-1)+n  
5 x = 3  
6 y = sumFirstNRec(x)  
7 print(y)
```

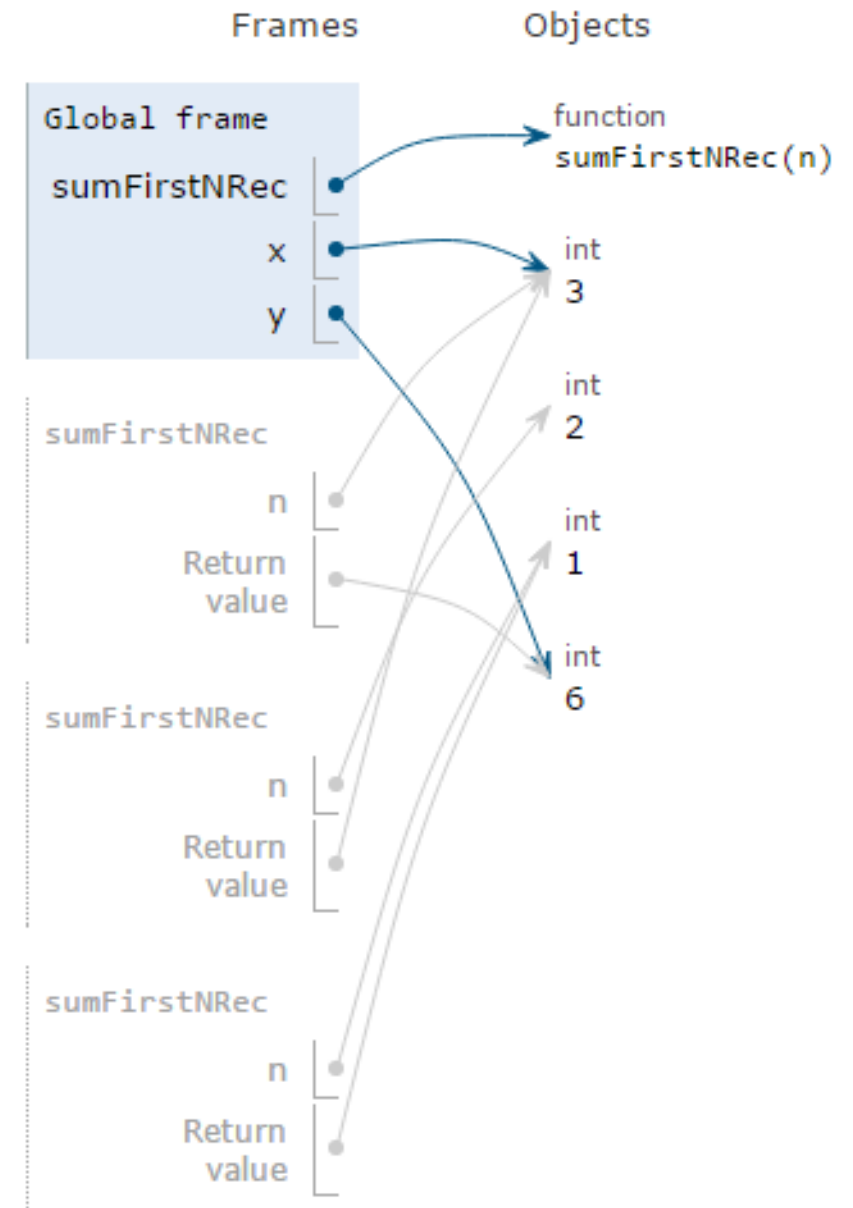


Рекурсивная функция. Сумма первых N чисел

```
1 def sumFirstNRec(n):  
2     if (n==1):  
3         return 1  
→ 4     return sumFirstNRec(n-1)+n  
5 x = 3  
6 y = sumFirstNRec(x)  
→ 7 print(y)
```



6



Рекурсивная функция. Сумма первых N чисел

1 Рекурсия

```
def sumFirstNRec(n):  
    if (n==1):  
        return 1  
    return sumFirstNRec(n-1) + n
```

2 Цикл FOR

```
def sumFirstNIter(n):  
    s = 0  
    for i in range(1, n+1):  
        s = s + i  
    return s
```

Рекурсивная функция. Фибоначчи

$$F_n = F_{n-1} + F_{n-2} \qquad F_0 = 0$$

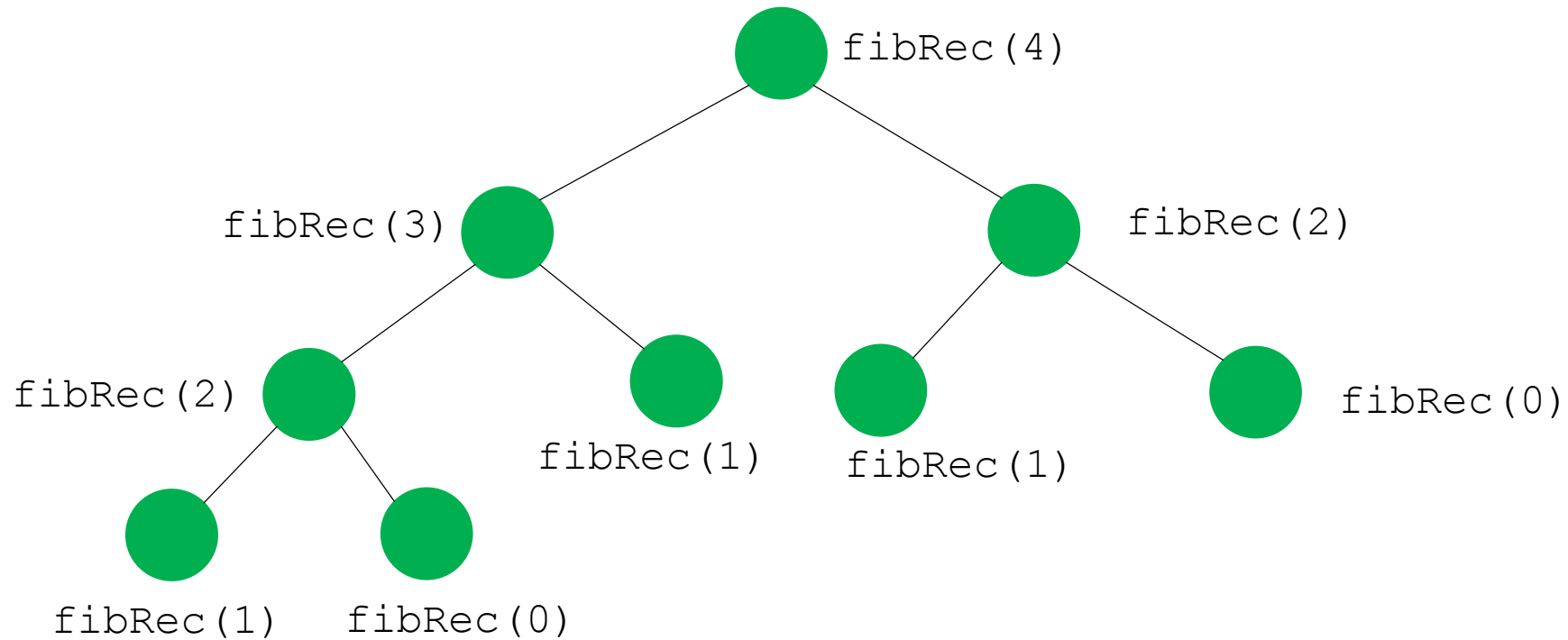
$$F_1 = 1$$

n: 0 1 2 3 4 5 6 7 8 9 10 ...

F_n: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

```
def fibRec(x):  
    if (x == 0):  
        return 0  
    elif (x == 1):  
        return 1  
    else:  
        return fibRec(x-1) + fibRec(x-2)
```

Рекурсивная функция. Фибоначчи



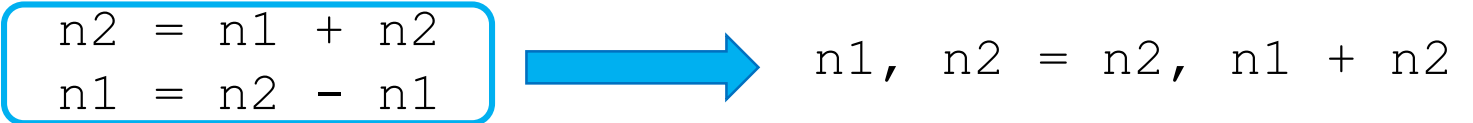
Рекурсивная функция. Фибоначчи

1 Рекурсия

```
def fibRec(x):  
    if (x == 0):  
        return 0  
    elif (x == 1):  
        return 1  
    else:  
        return fibRec(x-1) + fibRec(x-2)
```

2 Цикл FOR

```
def fibIter(x):  
    if (x == 0):  
        return 0  
    elif (x == 1):  
        return 1  
    else:  
        n1 = 0; n2 = 1  
        for n in range(2, x+1):  
            n2 = n1 + n2  
            n1 = n2 - n1  
        return n2
```



$n1, n2 = n2, n1 + n2$

Время выполнения рекурсии

```
import timeit as tmr
```

Main.py

```
tmrRec = tmr.Timer('fib.fibRec(30)', setup='import moduleFib as fib')
```

```
timeRec = tmrRec.timeit(10)
```

```
tmrIter = tmr.Timer('fib.fibIter(30)', setup='import moduleFib as fib')
```

```
timeIter = tmrIter.timeit(10)
```

```
print("Время вычисления Фибоначчи для x = 30 (10 раз)")
```

```
print("Рекурсия - " + str(timeRec))
```

```
print("Цикл - " + str(timeIter))
```

Время вычисления Фибоначчи для x = 30 (10 раз)

Рекурсия - 12.774560673552303

Цикл - 8.526778030493176e-05

Время вычисления Фибоначчи для x = 35 (10 раз)

Рекурсия - 154.36022873871582

Цикл - 9.79000440679556e-05

moduleA.py

```
def fibRec(x): ...
```

```
def fibIter(x): ...
```


Matplotlib

Время вычисления Фибоначчи

Определить время вычисления чисел Фибоначчи от 0 до n рекурсивным и итеративным способами, результат отобразить в виде графика.

```
import mfib
import mfib_plot as plt

n = int(input("Введите число n: "))

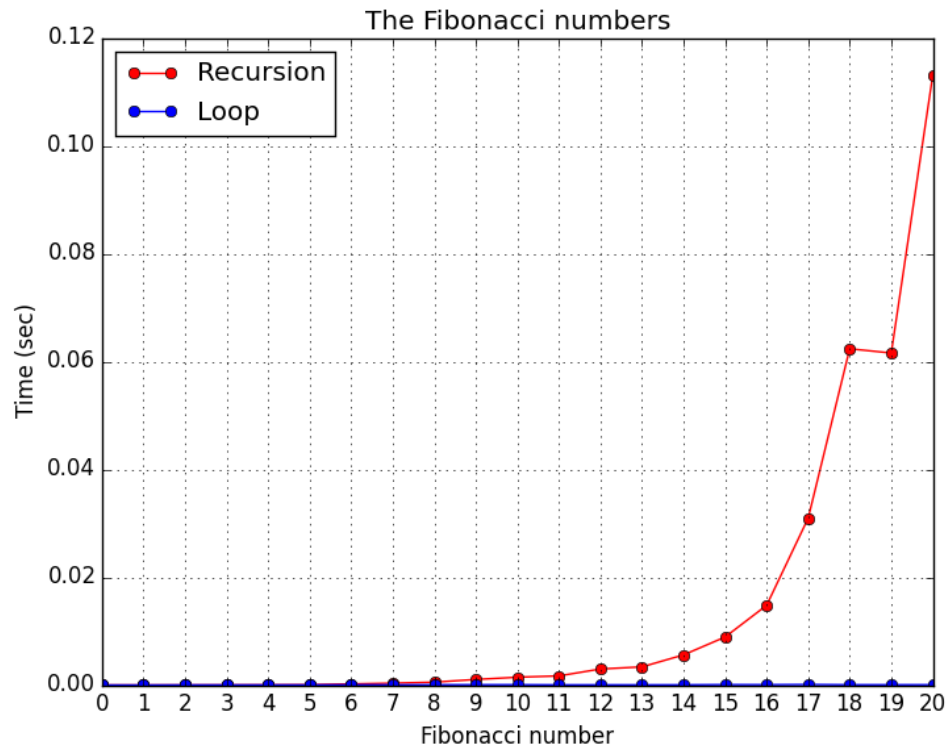
#Получить список времени вычисления чисел Фибоначчи [от 0 до n]
#lTimeRec - рекурсия; lTimeIter - цикл
lTimeRec, lTimeIter = mfib.getFibTime(n)

#Преобразовать значения в логарифмы
lTimeRecLog = mfib.getLogList(lTimeRec)
lTimeIterLog = mfib.getLogList(lTimeIter)

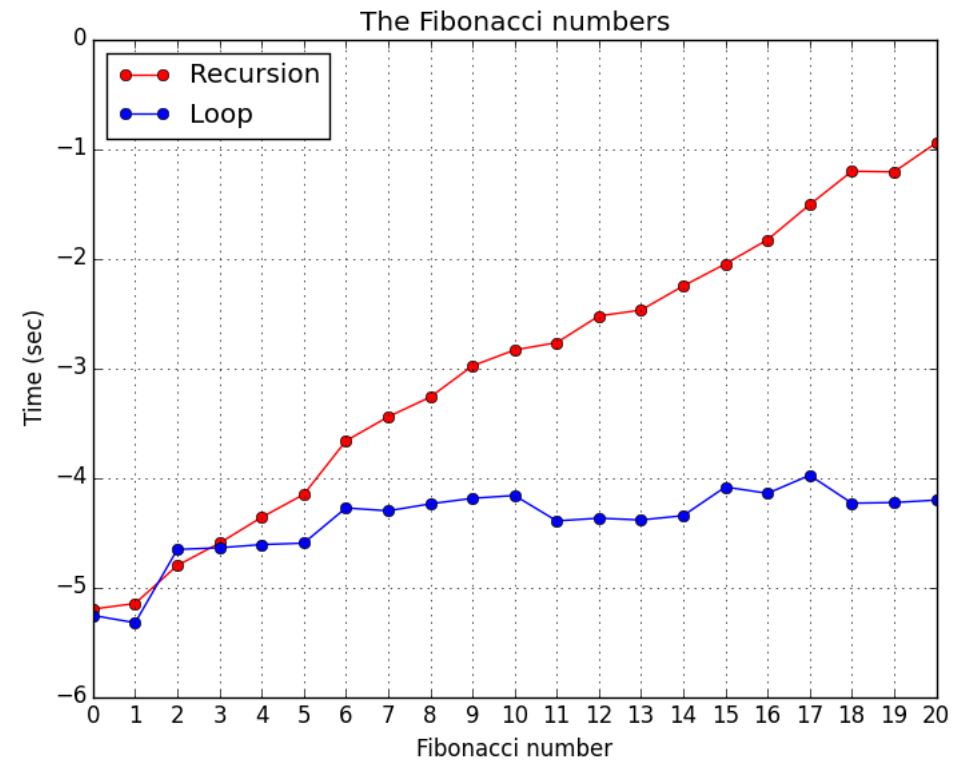
#Отобразить график 2 в 1
plt.showPlotFibTwoInOne(lTimeRec, lTimeIter,
                        lTimeRecLog, lTimeIterLog, vert=True)
```

Время вычисления Фибоначчи

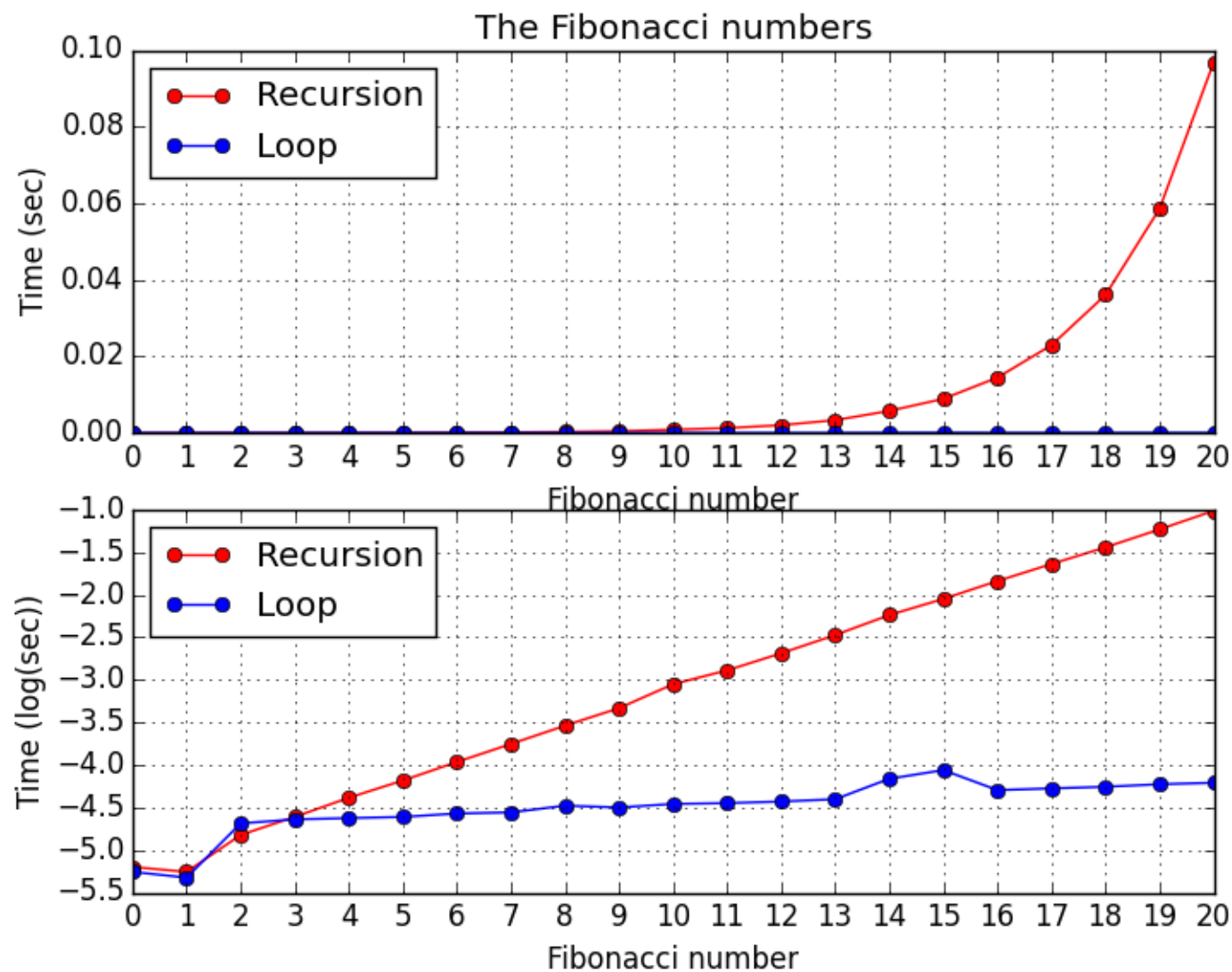
Координаты число–время



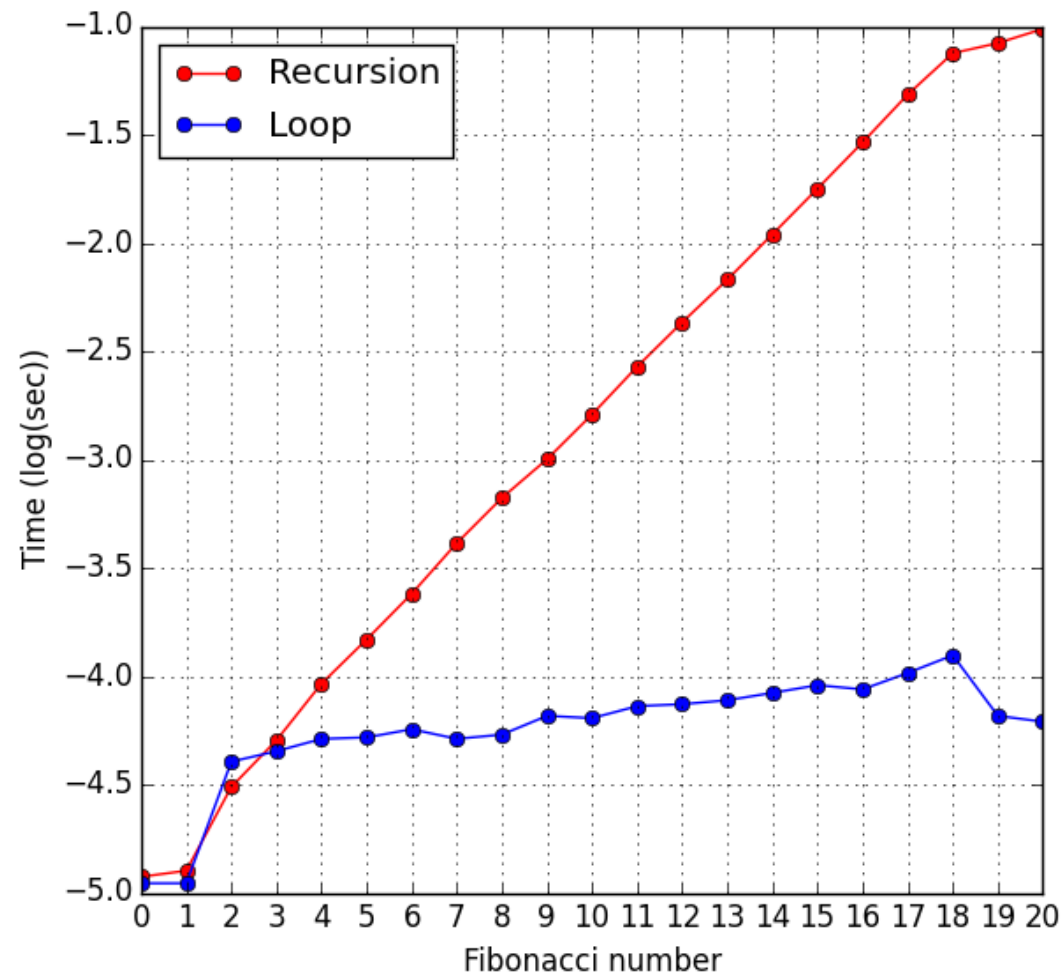
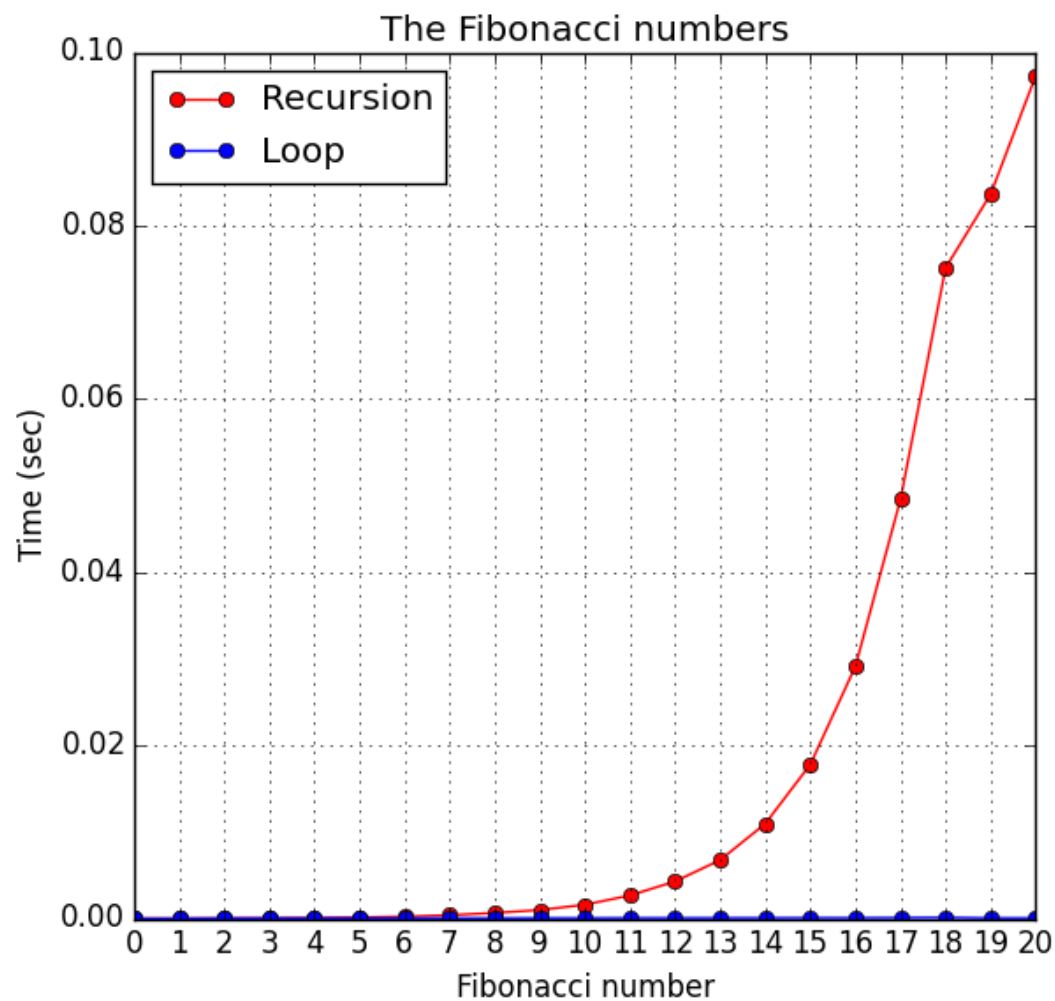
Координаты число–логарифм времени



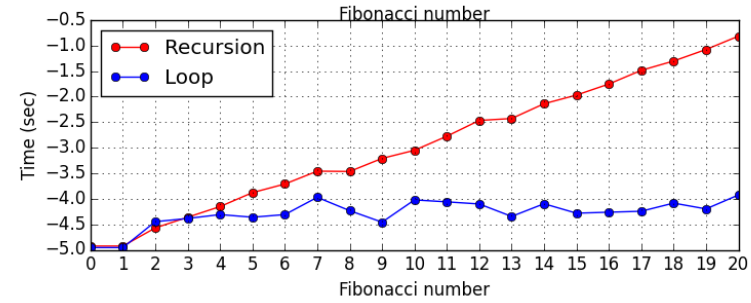
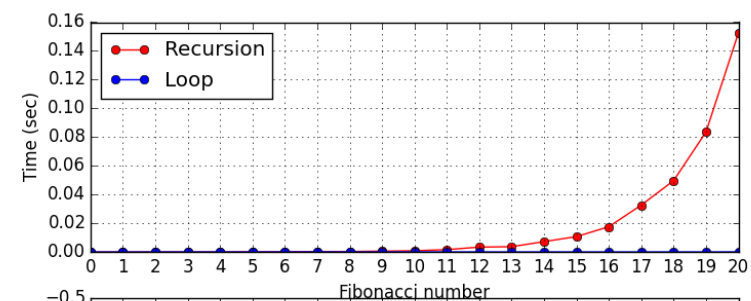
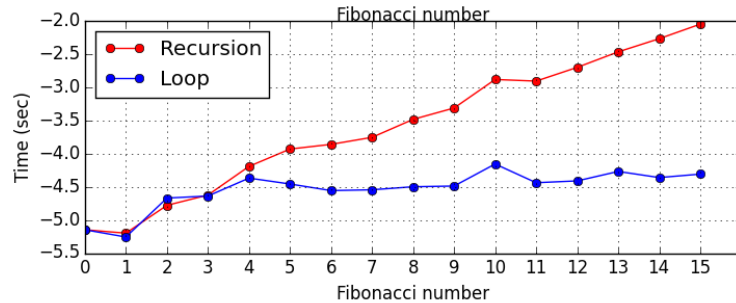
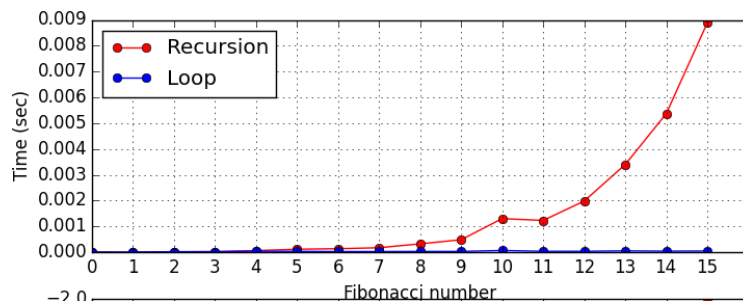
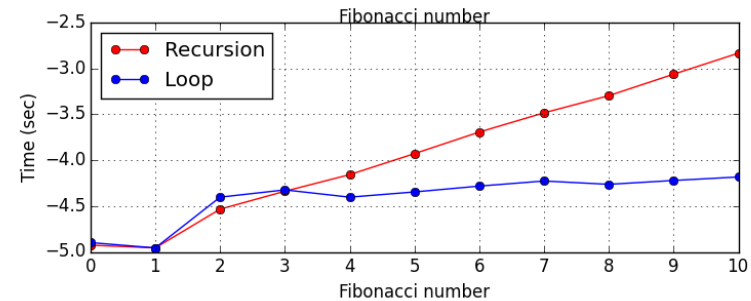
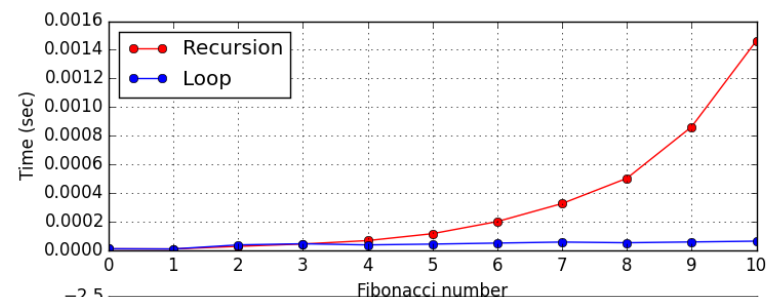
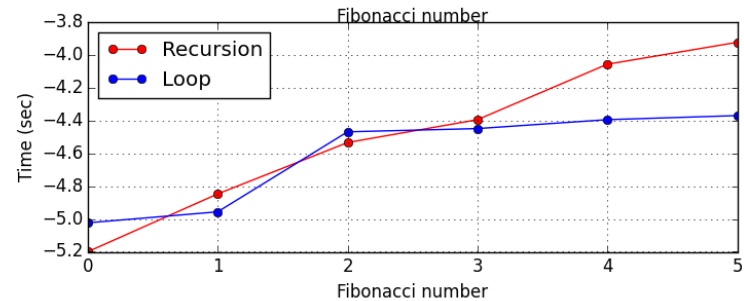
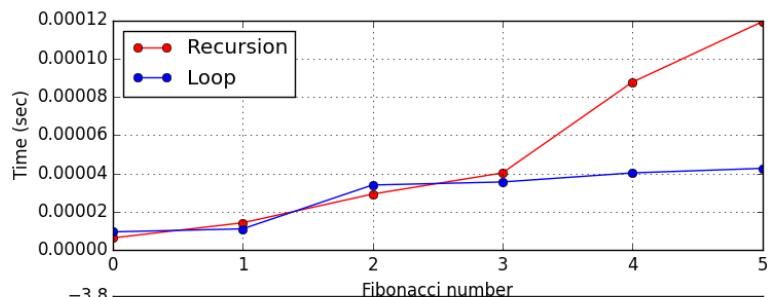
Время вычисления Фибоначчи. Графики 2 в 1



Время вычисления Фибоначчи. Графики 2 в 1



Время вычисления Фибоначчи. Графики 2 в 1



Количество слов в тексте

Определить n наиболее часто встречающихся слов в тексте, упорядочить их по убыванию и построить bar график.

```
import mcountword as mcw
import mcountword_plot as plt
```

```
#Получить словарь вида <слово-количество> из файла "input.txt"
```

```
dWords = mcw.getWords("input.txt")
```

```
topNum = int(input("Введите число n: "))
```

```
#Получить списки:
```

```
#1) topWord: n наиболее часто встречающихся слов в тексте
```

```
#2) topCount: их количество в тексте
```

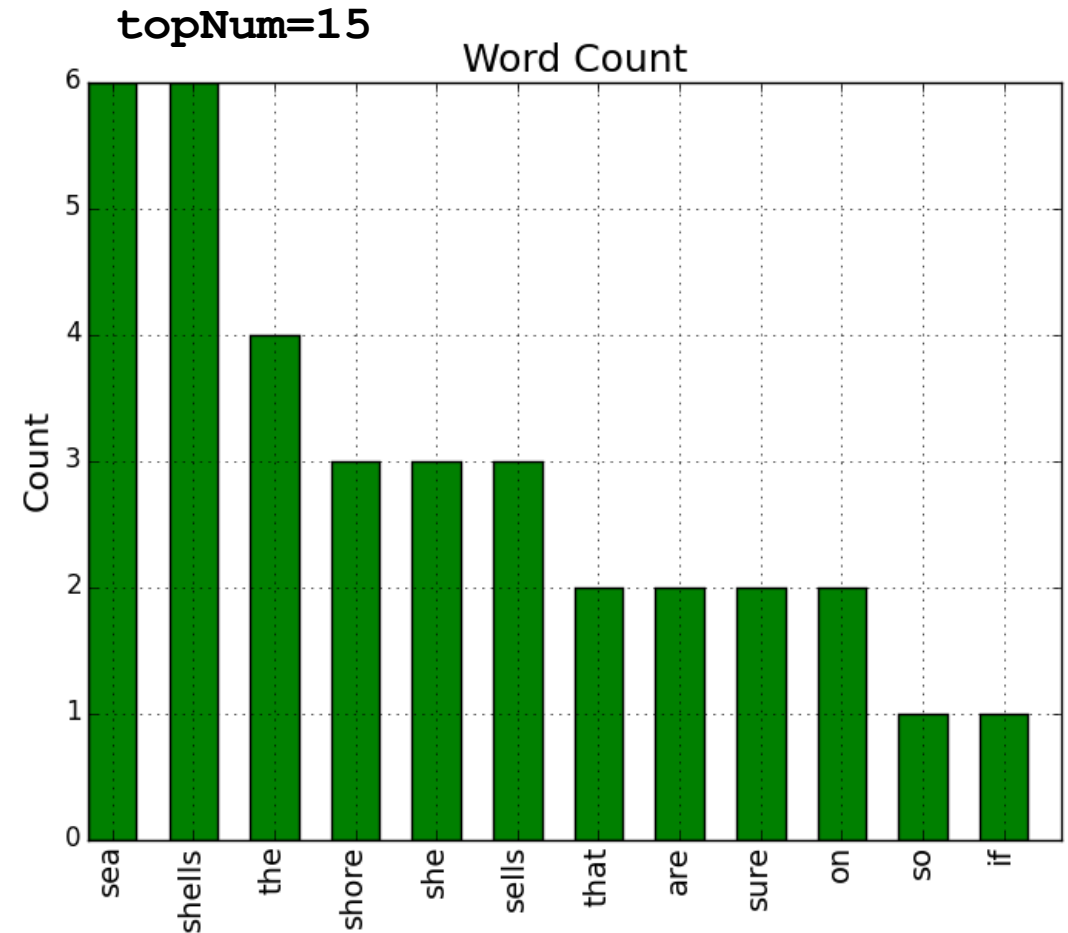
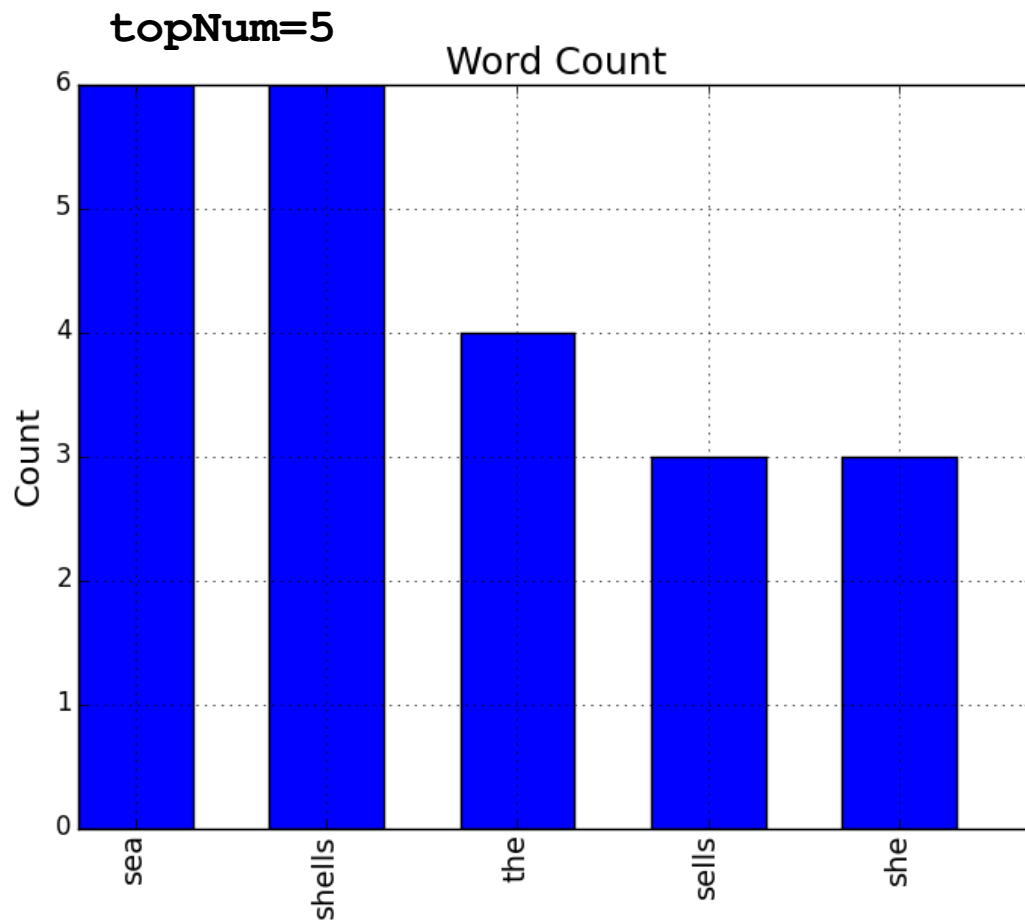
```
topCount, topWord = mcw.getNTop(topNum, dWords)
```

```
#Вывести график
```

```
plt.plotBarCWord(topCount, topWord)
```

Количество слов в тексте

She sells sea shells on the sea shore;
The shells that she sells are sea shells I'm sure.
So if she sells sea shells on the sea shore,
I'm sure that the shells are sea shore shells.



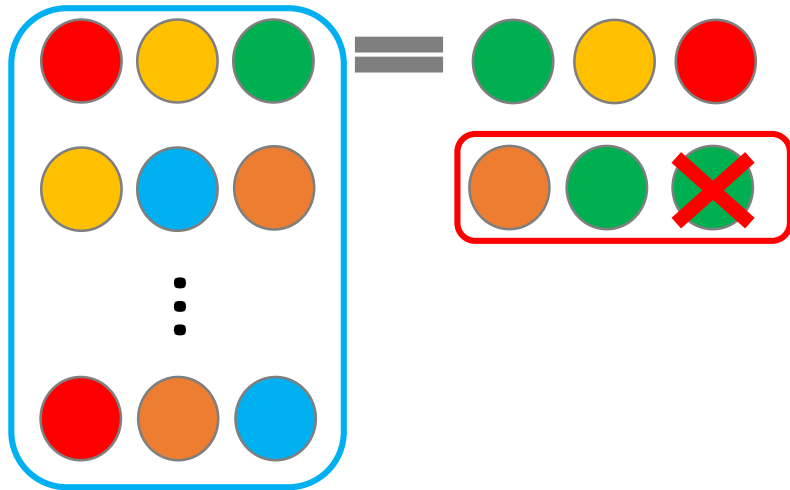
Комбинаторика

Комбинаторика

Элементы:



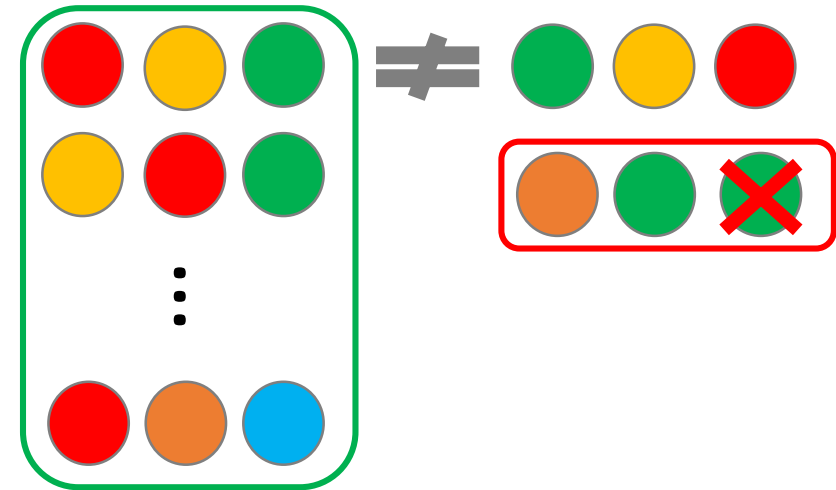
Сочетания из 5 элементов по 3



$$C_n^m = \frac{n!}{m! \cdot (n - m)!}$$

$$C_5^3 = \frac{5!}{3! \cdot (5 - 3)!}$$

Размещения из 5 элементов по 3



$$A_n^m = \frac{n!}{(n - m)!}$$

$$A_5^3 = \frac{5!}{(5 - 3)!}$$

```
import math  
x = 4  
f = math.factorial(x)  
print("Факториал %i! = %i" % (x, f))
```



Факториал 4! = 24

1. Recursive Functions

2. Measure execution time of small code snippets

3. Pyplot tutorial