13. Использование функций

13.1. Функции без аргументов (параметров)

Функции – это мини-программы в рамках большой программы.

Мы познакомились со встроенными функциями (например, print()), но можно писать и собственные функции.

Вид функции def без параметров:

```
def <имя функции> ():
        <тело функции>
```

где

<имя функции> дается по тем же правилам, что и имя переменной,

<тело функции> - инструкции, реализующие решаемую задачу с помощью этой функции.

Если функция описана, то *вызов функции* осуществляется инструкцией:

<имя функции> ()

Пример 1. Функция без параметров

```
# Программа func01
# Начало описания функции
def lin():
        for k in range(40):
                print('*',end='')
        print()
# Начало основной части программы
lin()
lin()
```

Будет напечатано:

```
****************************************

****************************************
```

Замечание. Функцию lin() можно было написать и короче:

```
def lin():
        print('*' *40)
```

13.2. Вид функции def с параметрами:

```
def <имя функции> (s):
        <тело функции>
```

где
s – список формальных параметров, перечень величин, от которых зависит результат выполнения функции. Формальные параметры используются в теле функции.

Вызов такой функции
имеет вид:

<имя функции> (p)

где
p – список фактических параметров,
перечень конкретных значений
параметров
для решения конкретной задачи,
записанных через запятую.

В качестве значений фактических параметров могут быть:

1. Константы;
2. Имена переменных;

При вызове функции следует выполнять требования:

1. Количество фактических и формальных параметров должно совпадать;
2. Тип фактического параметра должен совпадать с типом соответствующего формального параметра;
3. Соответствующие фактические и формальные параметры должны

Замечание.
Переменные величины, получающие значения в теле функции называются «локальными», а встречающиеся в основной программе – «глобальными».

Локальные переменные «живут» только вовремя выполнения функции и недоступны для основной программы и других функций.

Пример 2. Функция с параметром

```
# Программа func02
# Начало описания функции
def linp(m):
        print('*'*m)
# Начало основной части программы
linp(20)
linp(30)
```

Будет напечатано:

```
********************

******************************
```

Результат выполнения функции (linp) зависит от величины m, которая в нашем случае является единственным параметром функции.

Параметров может быть несколько и они разделяются запятой.

Пример 3.

```
# Программа func03. Вывод нужной строки из файла
# описание функции
def fprint(ftxt,n): # ftxt – имя файла, n – номер строки
        f=open(ftxt)
        for k in range(0,n): # читаем первые n строк
                s=f.readline()
        print(s)
        f.close()
# Основная часть программы
n=int(input('введите номер строки '))
fprint('f9.txt',n)
```

Пример 4. Одна функция использует другую

```
# Программа func04
def linp(m): # функция из примера 2
                print('*'*m)

def lin2p(m,s): # функция с двумя параметрами
                    # заставка к программе
        linp(m)
        print('Эту программу разработал ',s)
        linp(m)

lin2p(35, 'Вова') # основная программа
```

Результат выполнения программы:

****

Эту программу разработал Вова

****

13.3. Инструкция return и возвращаемое значение

Пусть в программе надо найти значение:

$$\frac{1+\sqrt{2}}{1+\sqrt{7}}+\frac{1+\sqrt{5}}{1+\sqrt{6}}+\frac{1+\sqrt{3}}{1+\sqrt{11}}.$$

Для удобства вычисления желательно иметь функцию:

$$y(a,b)=\frac{1+\sqrt{a}}{1+\sqrt{b}}.$$

В Python это можно оформить так:

```
# Программа func05
# функция вычисления дроби
def drob(a,b):
        x1=1+pow(a,0.5)
        x2=1+pow(b,0.5)
        x=x1/x2
        return x # возврат значения
# основная программа
y=drob(2,7)+drob(5,6)+drob(3,11)
print('%10.3e'%y)
```

или так:

```
# Программа func05
# функция вычисления дроби
def drob(a,b):
        x1=1+pow(a,0.5)
        x2=1+pow(b,0.5)
        return x1/x2 # возврат значения
# основная программа
y=drob(2,7)+drob(5,6)+drob(3,11)
print('%10.3e'y)
```

или так:

```
# Программа func05

# функция вычисления дроби
def drob(a,b):
    return (1+pow(a,0.5)) /(1+pow(b,0.5))

# основная программа
y=drob(2,7)+drob(5,6)+drob(3,11)
print('%10.3e'% y)
```

В любом варианте выполнение функции заканчивается инструкцией **return** .

В любом варианте выполнение функции заканчивается инструкцией **return** .

В нашем примере инструкция **return** в функции последняя, но может быть и не последней.

В функции может быть несколько инструкций **return** , но после выполнения любой из них работа функции заканчивается.

В нашей функции drob(a,b) переменные являются позиционными.

В нашей функции drob(a,b) переменные являются позиционными.
Порядок параметров важен.
drob(2,7)
возвращает
0.6621991892442395 ,
а
drob(7,2)
возвращает
1.5101196380824 46 .

```
# Программа func06
def sign(a): # в этой функции несколько return
        if a < 0:
                return -1
        elif a==0:
                return 0
        else:
                return 1

# основная программа
a= float(input('Введите a '))
b=float(input('Введите b '))
print('результат: ', sign(a)+sign(b))
```

```
# Программа func07
def sum (a): # функция вычисления суммы
                # элементов списка
        s=0
        for i in range(len(a)):
                s=s+a[i]
        return s

# основная программа
b=[1,23,45,67,54]

print('Сумма равна', sum (b))
```

или

```
def sum (a): # функция вычисления суммы
                # элементов списка
        s=0
        for e in a:
                s=s+e
        return s

# основная программа
b=[1,23,45,67,54]

print('Сумма равна', sum (b))
```

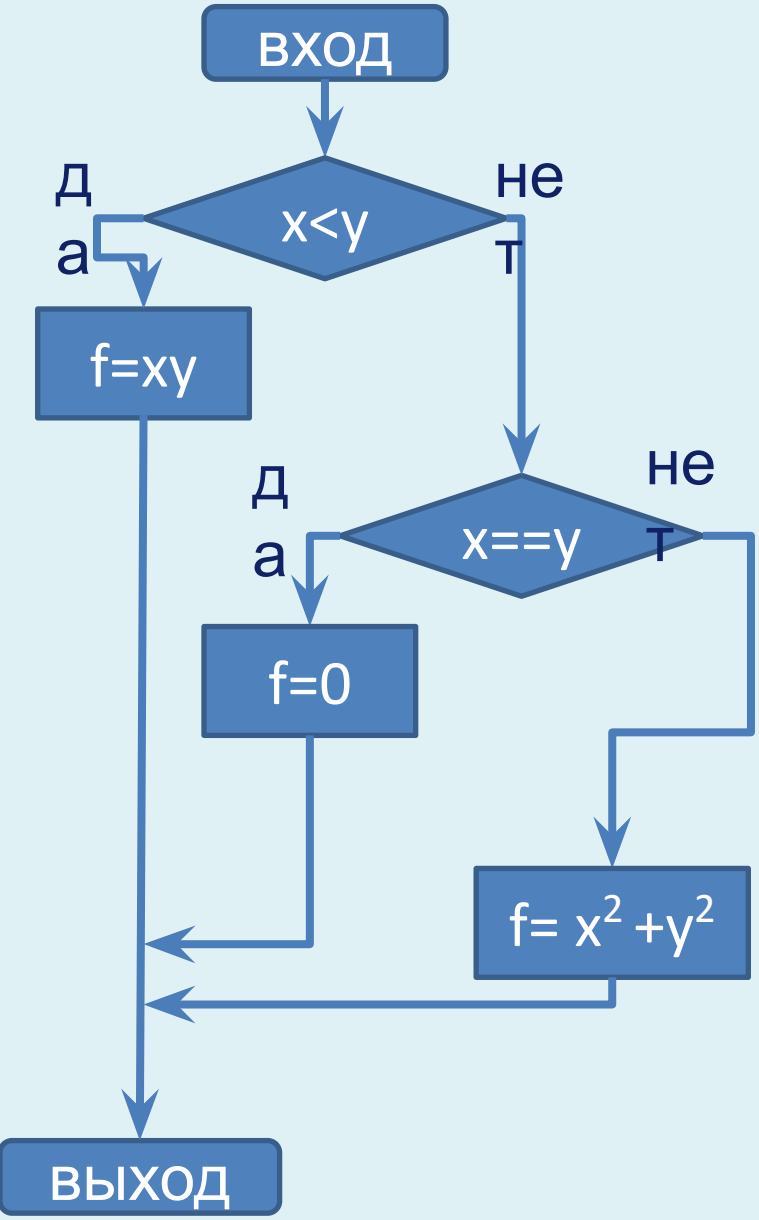
Задача. Написать программу вычисления величины

$$w = f(a, x) + f(a,4) + f(2,1) \quad ,$$

где

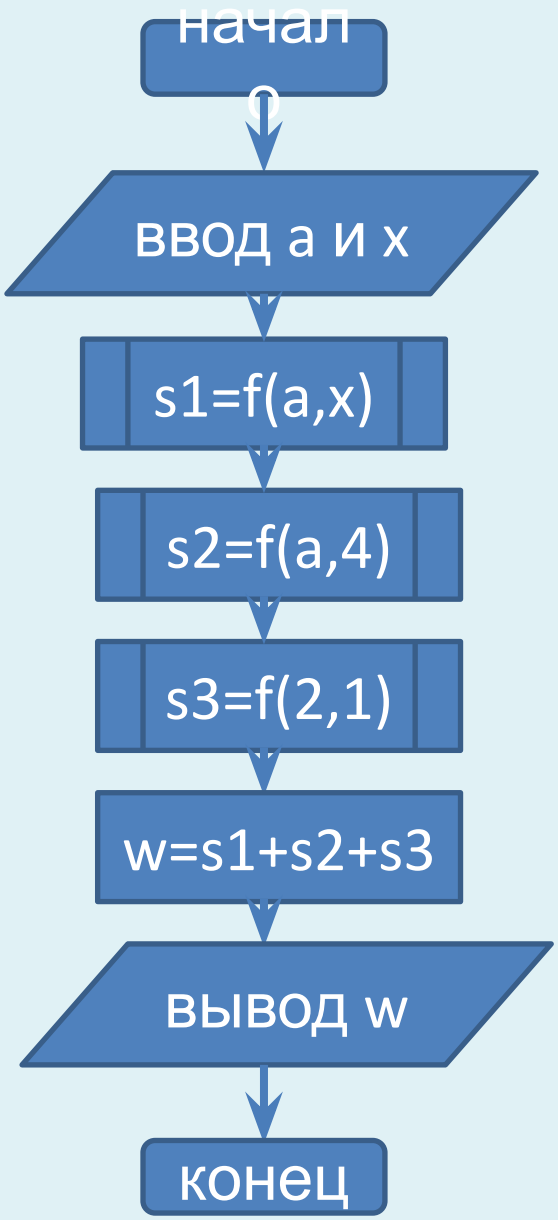
$$f(x, y) = \begin{cases} xy, & \text{если } x < y; \\ 0, & \text{если } x = y; \\ x^2 + y^2, & \text{если } x > y. \end{cases}$$

Вычисление f(x,y) оформить в виде вспомогательной функции.




```
# описание функции
f(x,y)
def f(x,y):
        if x<y :
                f=x*y
        elif x==y:
                f=0
        else:
                f=x*x+y*y


        return f
```




```
# описание основной части
a=float(input('Введите a '))
x=float(input('Введите x '))
s1=f(a,x)
s2=f(a,4)

s3=f(2,1)

w=s1+s2+s3

print(w)

input()
```

13.4. Значение None

Значение None – это «значение, не имеющее значения».

Оно используется, например, в качестве возвращаемого значения в функции print().

Введем в интерактивной оболочке:

```
>>> a=print('Привет!' )
Привет!
>>> a==None
True
>>>
```

Мы видим, что переменная a имеет значение None

Python незаметно для пользователя добавляет инструкцию

return None

в конец любой функции,

в которой инструкция **return** отсутствует

или

используется без указания возвращаемого значения (т.е. только слово **return**)

13.5. Именованные параметры

В Python можно использовать именованные аргументы, именованным аргументам можно присваивать значения невзирая на порядок.

С именованными аргументами мы встречались в функции

```
print(…, end=…, sep=…)
```

Пример с именованными параметрами

```
# функция вычисления дроби
def drob(a,b):
    return (1+pow(a,0.5)) /(1+pow(b,0.5))

# основная программа
print(drob(2,7))
print(drob(b=7,a=2))
```

Результат:

```
0.6621991892442395
0.6621991892442395
```

13.6. Параметры по умолчанию

При описании функции параметрам можно, хотя и не обязательно, присваивать значения по умолчанию.

```
# функция поздравления
def denr(name='Вова',god=7):
        print('С днем рождения!',name)
        print('Вам сегодня исполняется', god)

# основная программа
denr()
denr(god=12)
denr(name='Катя')
denr('Ира',70)
denr(70,'Ира')
```

13.7. Локальная и глобальная область видимости

О переменных, получающих значения в теле функции, говорят, что они существуют в *локальной область видимости* этой функции.

О переменных, получающих значения вне функции, говорят, что они существуют в *глобальной область видимости*.

Переменные, существующие в локальной область видимости, называются *локальными переменными*.

Переменные, существующие в глобальной область видимости, называются *глобальными переменными*.

Переменная не может быть локальной и глобальной одновременно.

Локальная область видимости создается каждый раз, когда вызывается функция.

Любая переменная, которой присваивается значение в этой функции, существует в данной локальной видимости.

При возврате из функции локальная область видимости уничтожается, и эти переменные теряются.

Когда в следующий раз будет вызвана эта функция, локальные переменные не будут помнить предыдущие значения.

Пример 1. Локальные переменные не видны в глобальной области видимости.

```
def spam():
        n=20123 # локальная область видимости

spam()  # глобальная область видимости
print(n) # глобальная область видимости
```

Выполнение этого кода приведет к ошибке:

NameError: name 'n' is not defined

В основном коде переменная **n** не определена.

Пример 2. В локальных областях видимости не видны переменные из других локальных областей видимости.

```
def spam():
        n=55
        b()
        print(n)
def b():
        n=1
spam()
```

В этом примере две разные переменные **n**.

В результате выполнения этого кода будет выведено:

55

Пример 3. Глобальные переменные могут читаться из локальной области видимости

```
def spam():
        print(n)
n=11

spam()
```

В этом примере в теле функции spam переменная **n** не определяется и функция использует глобальную переменную **n**.

В результате выполнения этого кода будет выведено:

```
11
```

Пример 4. Разные переменные могут иметь одно и то же имя, если они в разных областях видимости.

```
def spam():
        n='локальная в spam'
        print(n)
def b():
        n='локальная в b'
        print(n)
        spam()
        print(n)
n='глобальная'
b()
print(n)
```

В этом примере три разных переменных **n**.

В результате выполнения этого кода будет выведено:

локальная в b

локальная в spam

локальная в b

глобальная

13.8. Инструкция global

Если возникает потребность изменить в коде функции глобальную переменную, используют инструкцию global.

Пример 1. Использование инструкция global

```
def spam():
        global n # n глобальная переменная
        n='spam'
# основная часть кода
n='global'
spam()
print(n)
```

В результате выполнения этого кода будет выведено:

```
spam
```

Правила определения типа переменной (локальная или глобальная)

1. Если переменная используется в глобальной области видимости (т.е. вне функции), то она является глобальной.
2. Если переменная была объявлена в функции с использованием инструкции global, то она является глобальной.
3. Если переменной нигде в функции не присваивается значение, то она является глобальной.
4. Если переменная используется в операции присваивания в функции, то она является

Правила определения типа переменной (локальная или глобальная)

1. Если переменная используется в глобальной области видимости (т.е. вне функции), то она является глобальной.
2. Если переменная была объявлена в функции с использованием инструкции global, то она является глобальной.
3. Если переменной нигде в функции не присваивается значение, то она является глобальной.
4. Если переменная используется в операции присваивания в функции, то она является локальной.

Пример 2. Локальные и глобальные переменные

```
def spam():
        global n
        n='spam' # n глобальная переменная
def b():
        n= 'b' # n локальная переменная
def h():
        print(n) # n глобальная переменная
# основная часть кода
n=55 # n глобальная переменная
spam()
print(n)
```

В результате выполнения этого кода будет выведено:

spam

Пример 3. Нельзя использовать в функции локальную переменную до присвоения ей значения

```
def spam():
        print(n) # ошибка!!!
        n= 'локальная spam' # n локальная переменная
# основная часть кода
n= 55 # n глобальная переменная
spam()
print(n)
```

В результате выполнения этого кода получим:

```
Traceback (most recent call last):
  File "……", line 6, in <module>
    spam()
  File "……", line 2, in spam
    print(n) # ошибка!!!
UnboundLocalError: local variable 'n' referenced before
    assignment (локальная переменная 'n', на которую
    ссылаются перед присвоением)
```

Лабораторная работа № 5
«Вспомогательные функции»

Задание:

Написать программу вычисления величины z, которая вычисляется по формуле (согласно своего варианта).

Вычисление функции y(u,t), через которую описывается величина z, оформить в виде вспомогательной функции.

Варианты для величины z

Вариант	z
1	$y(a,b)+y(a^2,b^2)+y(a^2-1,a)+y(3,0.1)$
2	$y(a,x)+y(a-1,x+1)+y(a^2+x,1)$
3	$y(a,bxt)+y(ax+bt,0)+y(2,xt)$
4	$y(b,a^2+1)+y(b,a-b)+y(b^2-1,a^2+b^2)$
5	$y(a,x)+y(b,t)+y(a^2+b^2+x^2+t^2,5)$
6	$y(a,b)+y(c^2,1)+y(1,3)$
7	$y(xt,ab)+y(xt,2)+y(ab,3)$
8	$y(2,2xt)+y(7,3ab)+y(ab,xt)$
9	$y(a,5)+y(1,3)+y(x,a)$

Варианты для величины z

Вариант	z
10	$y(a^2,b^2)+y(a^2-1,a)+y(a,b)++y(2,0.5)$
11	$y(a^2+x,2)+y(a,x)+y(a-2,x+2)$
12	$y(ax+bt,0)+y(3,xt)+y(a,bxt)$
13	$y(b,a-b)+y(b^2-4,a^2+b^2)+y(b,a^2+1)$
14	$y(a^2+b^2+x^2,3)+y(a,x)+y(b,t)$
15	$y(c^2,2)+y(3,2)+y(a,2b)$
16	$y(xt,4)+y(ab,5)+y(xt,ab)$
17	$y(8,ab)+y(ab,xt)+y(3,3xt)$
18	$y(x,a)+y(a,8)+y(4,1)$

Варианты задания функции y(u,t)

вариант	функция	вариант	функция	вариант	функция
1	$y(u,t)=\begin{cases} u+2t, & u\ge 0 \\ u+t, & u\le -1 \\ u^2-2t+1, & -1<u<0 \end{cases}$	2	$y(u,t)=\begin{cases} u^2, & u<t-1 \\ t^2, & t-1\le u<t+1 \\ u+t^2, & u\ge t+1 \end{cases}$	3	$y(u,t)=\begin{cases} ut, & u<t \\ t^2, & u=t \\ u^2+t^2, & u>t \end{cases}$
4	$y(u,t)=\begin{cases} u+t, & u>1 \\ u-t, & 0\le u\le 1 \\ t-u, & u<0 \end{cases}$	5	$y(u,t)=\begin{cases} t-3u, & u<-1 \\ t^2+u, & u>1 \\ 0, & -1\le u\le 1 \end{cases}$	6	$y(u,t)=\begin{cases} 2u+t, & u\ge 2 \\ u+2t, & u\le -1 \\ u^2+2t-1, & -1<u<2 \end{cases}$
7	$y(u,t)=\begin{cases} u-t, & u<0 \\ t^2, & u=0 \\ u+t^2, & u>0 \end{cases}$	8	$y(u,t)=\begin{cases} u-t, & t<u+2 \\ u+t, & t>u+4 \\ 2ut, & u+2\le t\le u+4 \end{cases}$	9	$y(u,t)=\begin{cases} 3u+t, & u>5 \\ u-2t, & 2\le u\le 5 \\ 5t-u, & u<2 \end{cases}$

Варианты задания функции y(u,t)

вариант	функция	вариант	функция	вариант	функция
10	$y(u,t)=\begin{cases}2t+u, & u\ge 0\\ t+u, & u\le -1\\ u^2+1-2t, & -1<u<0\end{cases}$	13	$y(u,t)=\begin{cases}u^2, & t<u-1\\ t^2, & u-1\le t<u+1\\ u+t^2, & t\ge u+1\end{cases}$	16	$y(u,t)=\begin{cases}tu, & t<u\\ t^2, & t=u\\ t^2+u^2, & t>u\end{cases}$
11	$y(u,t)=\begin{cases}u-t, & t>1\\ u+t, & 0\le t\le 1\\ t+u, & t<0\end{cases}$	14	$y(u,t)=\begin{cases}3t-u, & u<-1\\ t+u^2, & u>1\\ 3, & -1\le u\le 1\end{cases}$	17	$y(u,t)=\begin{cases}2t+u, & u\ge 2\\ t+2u, & u\le -1\\ t^2+2u-1, & -1<u<2\end{cases}$
12	$y(u,t)=\begin{cases}t-u, & u<0\\ u^2, & u=0\\ t+u^2, & u>0\end{cases}$	15	$y(u,t)=\begin{cases}2u-t, & u<t+2\\ u+3t, & u>t+4\\ ut, & t+2\le u\le t+4\end{cases}$	18	$y(u,t)=\begin{cases}3t+u, & t>5\\ t-2u, & 2\le t\le 5\\ 5u-t, & t<2\end{cases}$

Отчет по лабораторной работе должен содержать:

- Задание сформулированное для конкретного (своего) варианта
- Тексты программ на языке Python
- Тестовые примеры
- Результат выполнения программ для тестовых примеров