Функции(!!!!!!)

Рассмотрим вычисление площади четырехугольнка по заданным длинам сторон и одной диагонали. Составим функцию вычисления площади одного треугольника. Далее сложим эти две площади.

In [1]:

```
import math
def Triangle(a, b, c):
    p = (a + b + c) / 2
    sq = math.sqrt(p * (p - a) * (p - b) * (p - c))
    return sq
```

In [4]:

```
a, b, c, d, L = map(float, input().split())
sqr = Triangle(a, b, L) + Triangle(c, d, L)
print("Площадь = ", sqr)
```

```
1 1 1 1 1
Площадь =  0.8660254037844386
```

Структура функции

После def пишется наименование функции и в круглях скобках список параметров. Результатом является переменная sq, которую нужно указать после оператора return. Обращаться к функция нужно также, как к обычным встроенным функциям.

В теле функции могут быть не только операторы присваивания, но и другие конструкции, например, оператор цикла

Напишем программу вычисления факториала без использования модуля math

In [5]:

```
def fact(n):
    fact = 1
    for i in range(1, n + 1):
        fact *= i
    return fact
```

In [7]:

```
n = int(input())
m = int(input())
c = fact(n) / fact(m) / fact(n - m)  # количество m-сочетаний из n элементов
print(c)
```

```
2
1
2.0
```

При образении к функциям задаются фактические параметры. При этом, в обзем случае, соотвествие с формальными параметрами по количеству и порядку следования. аименования фактических и формальных параметров могут быть, что чаще и происходит, равными.

Интересной особенностью функций является возможность задавать фактические параметры разных типов. Пусть функция sum служит для сложения двух чисел.

In [8]:

```
def sum(a, b):
    return a + b
```

In [9]:

```
# К ней можно обратиться:

# для нахождения суммы двух элеметов
print(sum(3, 5))
```

```
# для конкатенации строк, объединения списков, кортежей
print(sum("Вася", "!"))
print(sum([9, 8], [0, 1]))
print(sum(("qwe"), ("rty")))
```

```
8
Вася!
[9, 8, 0, 1]
qwerty
```

Можно создать "заглушку", что удобно делать на этапе отладки, но не только

In [10]:

```
def f1():
    pass
```

У функции может быть несоклько операторов return. Поесле выполнения любого из них функция заканчивает работу.

В операторе return иможно указывать в качестве результатов несколько переменных.

In [11]:

```
def add_subt(a, b):
    summ = a + b
    diff = a - b
    return (summ, diff)
```

In [14]:

```
x = float(input())
y = float(input())
s, d = add_subt(x, y)
print(s, d)
q = add_subt(9, 8)
print(q)
```

```
3
4
7.0 -1.0
(17, 1)
```

In [15]:

```
def mult(a, b):
    return(a * b)
print(mult(3, 5))
print(mult("Vasya", 5))
```

```
15
VasyaVasyaVasyaVasyaVasya
```

Функиця может не иметь оператора return. В этом случае её иногд аназывают процедурой. Образаться к такой функции чаще всего слудет отедльным оператором.

Можно в качестве параметров передовать нименование заполненных кортежей или списков. При этом при образении нужно перед ними поставить символ "*".

In [16]:

```
def add(a, b):
    return a + b
tp = (0, 1)
ls = [4, 5]
print(add(*tp))
print(add(*ls))
tp1 = (7, )
print(add(11, *tp1))
```

```
1
9
18
```

Но здесь определенные ограничения накладываются на рзамер кортежей и списков: их длина должно совпадать с количесвом параметров

Локальные и глобальные переменные

In [22]:

```
k = 7
def Sokolov():
    print(k)
    k1 = 100
    print(k1)
Sokolov()
```

```
7
100
```

У функции имеются локальные и глобальные величины. Здесь переменная k1 - локальная, к ней можно обратиться только внутри функции, вне ее локальная переменная несуществует. При обращении к ней вне функции возникает ошибка. Параметры функции, если они есть, также являются локальными

Глобальную переменную изменять внутри функции нельзя. Чтобы изменить значение глобальной переменной в функции, нужно воспользоваться словом global, после которого указать переменную, используемую в качестве глобальной.

In [30]:

```
def Masha():
    global z
    z = 100
    print(z)
Masha()
print(z + 50)
```

```
100
150
```

Функция может принимать не только конкретное количество входных параметров, но и переменное число их. Напишем функцию, которая находит произведение нескольких числе, заднных в кортеже. Перед такими параметрами нужнор поставить символ "*".

In [33]:

```
def multiplication(*x):
    p = 1
    for i in x:
        p *= i
    return p
print(multiplication(1, 2, 3, 4, 5))
print(multiplication(2, 3))
print(multiplication(111))
```

```
120
6
111
```

Именованые параметры

Параметры могут быть необязательнымы, если этому параметру присвоить начальное значение.

In [38]:

```
def stroka(string, length=15, end_st="!!!"):
    if len(string) > length:
        string = string[:length - len(end_st)] + end_st
    return string
print(stroka("Vova"))
print(stroka(length=7, string="Vladimir"))
print(stroka("Valdimir", end_st="?", length=3))
print(stroka("Vladimit", 5, "!"))
```

Vova
Vlad!!!
Va?
Vlad!