Функции

Функции позволяют разделить большой код на структурированные блоки. Если мы несколько раз повторяем одну и ту же операцию, стоит задуматься - возможно, ее стоит вынести в функцию.

Функция (по аналогии с математикой) принимает в себя любое количество разных значений - **аргументов**, и возвращает одно значение - **результат функции**.

Синтаксис функций таков:

```
def имя функции(параметр функции, параметр функции)
    действия
    return возвращаемое значение
```

С определенной версии языка (3.0) можно так же явно указывать типы принимаемых и возвращаемых значений функции, например:

```
def имя_функции(параметр функции: int, параметр функции: list) -> bool
    действия
    return возвращаемое значение # тип этого значения - bool
```

> Примечание: функция **всегда** должна быть объявлена до того, как она будет использована. Если в задании требуется найти ошибки в коде, одна из ошибок наверняка будет связана с этим.

Примеры различных простых функций

Функция, складывающая 2 целых числа:

- Принимает 2 числа: a и b
- Возвращает их сумму

```
def add(a, b)
    return a + b

print(add(2, 2))
print(add(-100, 50))
```

Вывод:

```
4
-50
```

Функция, проверяющая число на четность:

- Принимает число n
- Возвращает `True` или `False`

```
def is_even(n):
    return n % 2 == 0

print(is_even(42))
print(is_even(57))
```

Вывод:

```
True
False
```

Функция, печатающая все положительные четные числа в списке:
Функция может не возвращать значения.

- Принимает список $numbers$ из N элементов
- Печатает все положительные четные числа в нем в строчку
- Ничего не возвращает

```
def print_all_even_positive_numbers(numbers):
    for num in numbers:
        if num > 0 and num % 2 == 0:
            print(num, end=" ")

print_all_even_positive_numbers([-2, -5, 4, 7, 10, 35, -4, -8, 0, 42, 98])
```

Вывод:

```
4 10 42 98
```

Функция, возводящая число в квадрат и инвертирующая его:
Функция может вызывать и другие функции.

- Принимает число N
- Возвращает $\frac{1}{N^2}$

```
def inverse(n):
    return 1 / n

def square(n):
    return n ** 2

def inverse_square(n):
    return inverse(square(n))

print(inverse_square(2.5))
print(inverse_square(7.0))
print(inverse_square(-0.1))
```

Вывод:

```
0.16
0.02040816326530612
99.99999999999999
```

Функция, рекурсивно вызывающая саму себя:
Пример: функция, вычисляющая числа Фибонначи.

- Принимает число N, равное номеру числа Фибонначи
- Возвращает это число

```
def fibonacci(n):
    if n in (1, 2):
        return 1
    return fibonacci(n - 1) + fibonacci(n - 2)

for i in range(30):
    print(fibonacci(i), end=" ")
```

Вывод:

```
1 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393
```