Базовые функции. Шпаргалка

Как объявить функцию:

```
def function_name():
    some_code
```

Как определить функцию с параметрами:

```
def hello_someone(name, age):
    print(f"Привет, {name}! Тебе {age} лет!")
```

Как вернуть значение из функции:

```
def multiplication(a, b):
    result = a * b
    return result
```

Как распечатать результат работы функции в консоль:

```
multiplication_result = multiplication(3, 4)
print(multiplication_result)
```

Как использовать return для обработки ошибок:

```
def safe_division(a, b):
    if b == 0:
        return "Ошибка: деление на ноль!"
    else:
        return a / b
```

Как использовать return в циклах:

```
def serach_in_list(list_for_search, target):
    for element in list_for_search:
        if element == target:
            return True
    return False


my_list = [1, 2, 3, 4, 5]
goal = 3

result = serach_in_list(my_list, goal)
print(result)

>>> True
```

Как использовать return в обработке данных из словаря:

```
def process_person_info(person_info):
    if 'name' in person_info and 'age' in person_info and 'city' in person_info:
        # Извлечение данных из словаря
        name = person_info['name']
        age = person_info['age']
        city = person_info['city']

        # Обработка информации
        result = f"{name} - {age} лет, проживает в городе {city}."
        return result
    else:
        return "Неполная информация о человеке."


person_data = {'name': 'Иван', 'age': 25, 'city': 'Москва'}
processed_info = process_person_info(person_data)
print(processed_info)

>>> Иван - 25 лет, проживает в городе Москва.
```

Локальная область видимости:

```
def foo():
    local_variable = 10
    print(local_variable)

foo()

>>> 10
```

Правила оформления функций в коде

- После объявления функций ставится два переноса строки:

```
def new_sum(*nums):
    sum = 0
    for n in nums:
    sum += n
    return sum
# раз пустая строка
# два пустая строка
new_sum(5, 7, 8, 11)    # Здесь пишем любой код
```

- Аргументы разделяются запятой с пробелом:

```
def any(arg1, arg2, arg3):
    pass
```

- Если есть значения по умолчанию, знаки равно в них НЕ окружаются пробелами:

```
def precheck(prices, tip=10):
```

- Если имя аргумента конфликтует с зарезервированным словом, в конце ставится нижнее подчеркивание:

```
def any_func(class_, try_, finally_)
```

- Функции документируются с помощью `"""`:

```
def remove_from_string(string, *symbols_to_remove):
  """Удаляет символы, перечисленные после первого аргумента"""

    for symbol in symbols_to_remove:
        string = string.replace(symbol,"")
    return string

print(remove_from_string.__doc__)

# Вернет "Удаляет символы, перечисленные после первого аргумента"
```

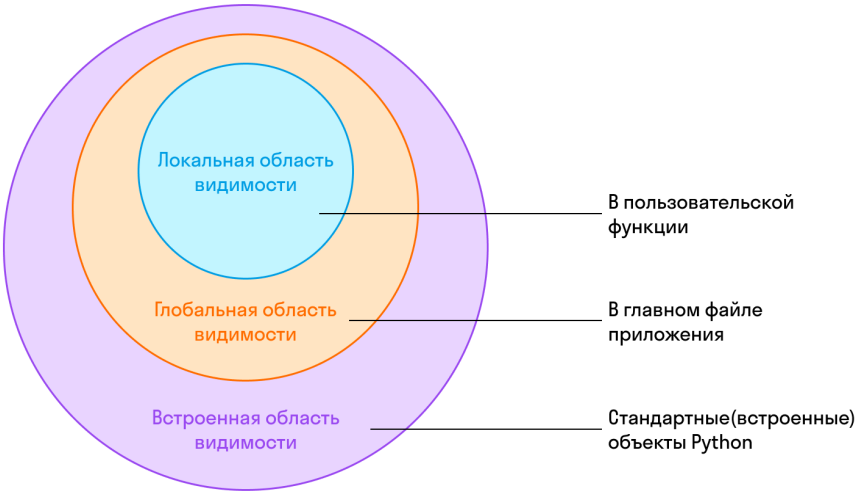
Глобальная область видимости:

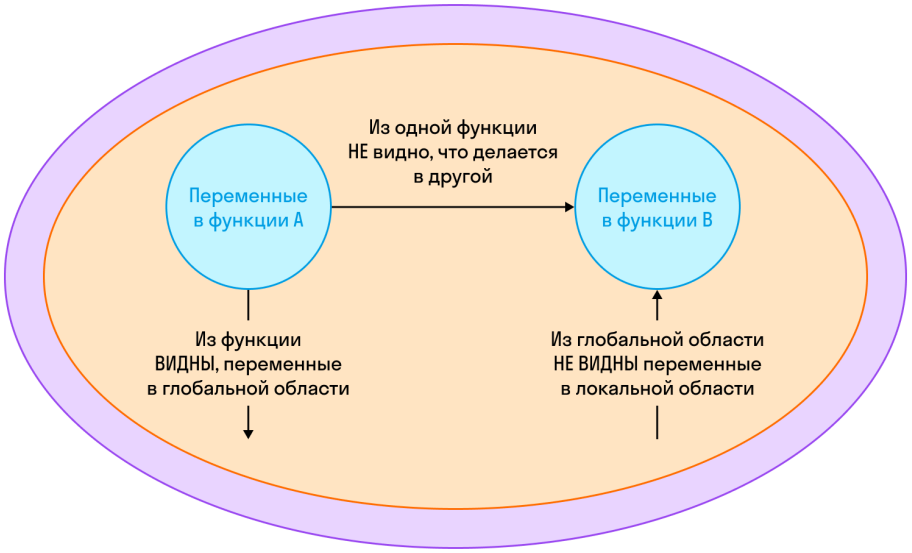
```
global_variable = 30

def foo():
    print(global_variable)

foo()
print(global_variable)

>>> 30
    30
```





Заглушка pass:

```
def some_func():
    pass  # Временная заглушка, нужно добавить логику позже
```