

---

## *Введение в функции*

---

Функция - это фрагмент программного кода, который выполняет определенную задачу и может быть вызван из других частей программы для выполнения этой задачи. Функции используются для организации кода, упрощения его чтения, повторного использования и модульности.

*Важные концепции, связанные с функциями:*

1. Функция как блок кода: функция представляет собой блок кода, который имеет имя и выполняет некоторую задачу. Она определяется с использованием ключевого слова `def` в Python, за которым следует имя функции и список параметров, если они есть.


2. Вызов функции: после определения функции она может быть вызвана из других частей программы для выполнения задачи, которую она решает. Для вызова функции используется ее имя, после которого следует список аргументов в круглых скобках, если они требуются.

3. Функция как часть программы: Функции могут быть использованы повторно в разных частях программы, что делает код более модульным и легко поддерживаемым.

4. Функция и процедура: В некоторых языках программирования термины "функция" и "процедура" имеют различные значения. В Python различия между ними нет. Все функции могут выполняться как функции, возвращающие значение, так и процедуры, не возвращающие значение.

## Пример определения функции в Python:

python

 Copy code

```
def greet(name):  
    print("Hello, " + name + "!")
```

В этом примере функция `greet` принимает один параметр `name` и выводит приветствие на экран. Определение функции начинается с ключевого слова `def`, за которым следует имя функции, список параметров (в данном случае `name`) и двоеточие. Все инструкции, которые должны выполняться в теле функции, отступаются на один уровень вправо.

*Функции играют ключевую роль в программировании и используются по множеству причин.*

Рассмотрим основные из них:

1. **Модульность:** Функции позволяют разбить программу на небольшие логические блоки, каждый из которых выполняет определенную задачу. Это делает код более организованным и легким для понимания.

2. **Повторное использование кода:** Определив функцию один раз, вы можете вызывать ее в разных частях программы. Это позволяет избежать дублирования кода и делает его более эффективным и легким в обслуживании.

3. **Абстрагирование:** Функции позволяют абстрагировать детали реализации определенной операции. Вы можете сосредоточиться на том, что делает функция, а не на том, как она это делает. Это упрощает понимание и изменение кода.

4. Упрощение: Функции могут быть использованы для упрощения сложных операций или задач, разбивая их на более простые шаги. Это делает код более понятным и легким в написании.

5. Тестирование: Разделение программы на функции делает ее более тестируемой. Вы можете тестировать каждую функцию отдельно, что помогает выявить и исправить ошибки.

6. В целом, функции являются мощным инструментом, который помогает создавать чистый, модульный и эффективный код. Они позволяют разработчикам легко масштабировать и поддерживать программы любого размера и сложности.

*В Python функции определяются с использованием ключевого слова `def`, за которым следует имя функции и список параметров в круглых скобках. Далее идет двоеточие, а после него блок кода, который выполняется при вызове функции. Вот пример синтаксиса определения функции:*

```
python Copy code  
  
def my_function(parameter1, parameter2):  
    # Блок кода функции  
    # Может содержать одну или несколько операций  
    result = parameter1 + parameter2  
    return result
```

Рассмотрим основные компоненты синтаксиса определения функции:

- **def:** Ключевое слово `def` используется для определения функции.
- **Имя функции:** Имя функции должно быть допустимым идентификатором в Python. Оно следует за ключевым словом `def`.
- **Параметры функции:** Параметры функции определяются в скобках после имени функции. Параметры являются переменными, которые принимают значения, переданные в функцию при ее вызове.
- **Двоеточие:** После списка параметров ставится двоеточие, обозначающее начало блока кода функции.
- **Блок кода функции:** Этот блок содержит код, который будет выполняться при вызове функции. Он должен быть с отступом относительно ключевого слова `def`.
- **return:** Ключевое слово `return` используется для возврата значения из функции. Оно указывает, какое значение должно быть возвращено в место вызова функции.

Важно помнить, что блок кода функции должен иметь одинаковый отступ и быть корректно выровнен. Отступы в Python играют роль в определении блоков кода, поэтому неправильное выравнивание может привести к ошибкам.

*Параметры функций в Python определяются в списке параметров в скобках после имени функции. Есть несколько типов параметров, которые могут использоваться для передачи аргументов в функцию:*

**Позиционные параметры:** Позиционные параметры - это параметры функции, которые передаются в определенном порядке, определяемом их позицией в списке параметров. При вызове функции значения аргументов связываются с параметрами функции в том же порядке, в котором они были переданы. Вот пример:

```
python Copy code  
  
def greet(name, age):  
    print(f"Привет, {name}! Тебе {age} лет.")  
  
# Вызов функции с позиционными аргументами  
greet("Анна", 25)
```

**Именованные параметры:** Именованные параметры позволяют явно указывать, какому параметру соответствует передаваемый аргумент, используя имя параметра. Это делает код более понятным и уменьшает вероятность ошибок из-за путаницы в порядке аргументов. Вот пример:

```
python Copy code  
  
def greet(name, age):  
    print(f"Привет, {name}! Тебе {age} лет.")  
  
# Вызов функции с именованными аргументами  
greet(name="Анна", age=25)
```

Значения параметров по умолчанию: В Python вы можете определить значения параметров по умолчанию, которые будут использоваться, если вызывающая сторона не передает аргумент для этого параметра. Это удобно, когда некоторые параметры функции часто используют одни и те же значения. Пример:

```
python Copy code

def greet(name, age=30):
    print(f"Привет, {name}! Тебе {age} лет.")

# Вызов функции без указания значения для параметра age
greet("Анна")
```

В этом примере, если значение не передано для параметра age, используется значение по умолчанию, равное 30. Однако, если значение передано, то оно заменит значение по умолчанию.

### *Написание простых функций:*

1. Функция для вычисления среднего значения списка чисел: Студенты должны написать функцию, которая принимает список чисел в качестве аргумента и возвращает среднее значение этого списка.

```
python Copy code

def calculate_average(numbers):
    total = sum(numbers)
    average = total / len(numbers)
    return average

# Пример использования функции
my_list = [1, 2, 3, 4, 5]
result = calculate_average(my_list)
print("Среднее значение списка:", result)
```

В этом примере функция `calculate_average` принимает список чисел в качестве аргумента `numbers`. Затем она вычисляет сумму всех чисел в списке с помощью функции `sum()` и находит среднее значение, разделив сумму на количество чисел в списке. Наконец, функция возвращает полученное среднее значение.

## 2. Функция для объединения двух строк с разделителем:

Студенты должны создать функцию, которая принимает две строки и разделитель как аргументы и возвращает строку, в которой обе строки объединены с указанным разделителем.

```
python Copy code  
  
def merge_strings(string1, string2, delimiter):  
    merged_string = string1 + delimiter + string2  
    return merged_string  
  
# Пример использования функции  
result = merge_strings("Hello", "World", " ")  
print("Объединенная строка:", result)
```

В этом примере функция `merge_strings` принимает две строки `string1` и `string2`, а также разделитель `delimiter`. Затем она объединяет обе строки с указанным разделителем и возвращает полученную строку.

## Практические задания

1. Напишите функцию `concat_names`, которая принимает на вход две строки - имя и фамилию, и возвращает их объединенными через пробел.
2. Напишите функцию `is_palindrome`, которая принимает строку и возвращает `True`, если строка является палиндромом, и `False` в противном случае.
3. Создайте функцию `compute_discount`, которая принимает на вход сумму покупки и процент скидки, а затем возвращает сумму с учетом скидки.
4. Напишите функцию `count_words`, которая принимает строку в качестве аргумента и возвращает количество слов в этой строке.
5. Напишите функцию `generate_random_list`, которая принимает длину списка и диапазон случайных чисел в качестве позиционных параметров и возвращает список указанной длины, заполненный случайными числами из указанного диапазона.
6. Напишите функцию `find_common_elements`, которая принимает два списка и возвращает список элементов, которые есть в обоих списках.
7. Напишите функцию `calculate_total_price`, которая принимает цену товара и процент налога в качестве именованных параметров со значениями по умолчанию и возвращает общую стоимость товара с учетом налога.
8. Напишите функцию `generate_password`, которая принимает длину пароля в качестве параметра со значением по умолчанию и возвращает случайно сгенерированный пароль указанной длины.