?????

??????????????????

В этой книге описываются общие определения, методология и подходы в языках программирования.

- Основные определения

???????? ????????????

????????? ? ?????????

????????

> **Параметр** — это **переменная в объявлении функции**.

Это часть сигнатуры функции, "заглушка" или "формальный placeholder", которую вы указываете в скобках при создании функции.

- **Что это?** Переменная в объявлении функции.
- **Когда определяется?** Во время написания кода функции.
- **Аналог из жизни:** "Реквизит" в заявке на документ. В форме заявления написано: `ФИО: _____`, `Дата рождения: _____`. Это параметры — они определяют, какая информация *нужна*, но сами по себе не содержат значений.

Пример (Python):

```
# Здесь 'name' и 'age' — это ПАРАМЕТРЫ
def greet(name, age):
    print(f"Привет, {name}! Тебе {age} лет.")
```

????????

> **Аргумент** — это **конкретное значение, которое передается в функцию** при ее вызове.

Это реальное значение, которое вы подставляете вместо параметра, когда вызываете функцию.

- **Что это?** Конкретное значение, переданное в функцию.
- **Когда определяется?** В момент вызова функции.
- **Аналог из жизни:** Конкретные данные, которые вы вписываете в ту самую заявку: `ФИО: Иванов Иван`, `Дата рождения: 15.03.1990`. Это аргументы — фактические значения.

Пример (Python):

```
# Здесь 'Анна' и 25 — это АРГУМЕНТЫ
greet('Анна', 25)
```

??????? ??????? ????????

Критерий	Параметр (Parameter)	Аргумент (Argument)
Что представляет	Переменная в объявлении функции	Фактическое значение, переданное в функцию
Контекст	Объявление функции	Вызов функции
Также известен как	Формальный параметр (formal parameter)	Фактический параметр (actual parameter) или фактический аргумент
Существование	Существует только внутри функции	Может быть переменной или литералом из внешней области видимости

?????????????? ??????

???? ??????????

Когда вы передаете аргументы, они могут быть разных типов:

- **Позиционные аргументы (Positional arguments):** Передаются в том порядке, в котором объявлены параметры.

  ```
  greet('Анна', 25)  # 'Анна' идет для параметра `name`, 25 — для `age`
  ```

- **Именованные аргументы (Keyword arguments):** Передаются с указанием имени параметра. Порядок не важен.

  ```
  greet(age=25, name='Анна')  # Порядок изменен, но результат тот же
  ```

- **Аргумент по умолчанию (Default argument):** Параметр, которому в объявлении функции присвоено значение по умолчанию. Если соответствующий аргумент не передан, используется значение по умолчанию.

  ```
  # Параметр 'age' имеет значение по умолчанию 18
  def greet(name, age=18):
      print(f"Привет, {name}! Тебе {age} лет.")

  greet('Максим')  # Вызовет: "Привет, Максим! Тебе 18 лет."
  ```

```
greet('Мария', 30) # Вызовет: "Привет, Мария! Тебе 30 лет."
```

???????? ????? ????????

Одно и то же слово в разных контекстах может быть и параметром, и аргументом.

```
# 1. ОБЪЯВЛЕНИЕ ФУНКЦИИ
# x и y — это ПАРАМЕТРЫ
def multiply(x, y):
    return x * y

a = 5
b = 10

# 2. ВЫЗОВ ФУНКЦИИ
# a и b — это АРГУМЕНТЫ
result = multiply(a, b)

# 3. ВНУТРИ ФУНКЦИИ 'multiply' во время ее выполнения:
# Параметр x получает значение аргумента a (5)
# Параметр y получает значение аргумента b (10)
# Функция работает с числами 5 и 10.
```

??????????

Хотя эти термины часто используют как синонимы, и во многих случаях вас поймут, точное понимание разницы делает ваш код и общение с другими разработчиками более четкими.

Простая мнемоника:

> **П**араметр — **П**олучатель (в объявлении).
> **А**ргумент — **А**ктуальное значение (при вызове).

?????

Опция — это модификатор команды или функции, который *меняет ее поведение*. Часто она просто включает или выключает какой-то режим.

- **Аналогия:** Представьте, что вы заказываете кофе. Вы говорите: "С сахаром" или "Без кофеина". Слова "с" и "без" — это опции, которые меняют стандартное приготовление кофе.
- **В программировании/CLI:** Опции обычно начинаются с дефиса (`-`) или двойного дефиса (`--`).
  - Короткие опции: `-l`, `-h`, `-v`
  - Длинные опции: `--help`, `--version`, `--all`
- **Ключевая особенность:** Опция может не требовать дополнительного значения. Она просто "включена".

Примеры:

```
# Опция '-l' меняет вывод команды ls на подробный список
ls -l


# Опция '--help' выводит справку, а не выполняет основное действие
program --help
```