Функции с произвольным количеством параметров

Функции в Python могут принимать произвольное количество параметров. Это достигается с помощью специальных символов в определении функции.

В Python есть два основных способа передачи произвольного количества параметров:

- *args: этот параметр позволяет передавать произвольное количество позиционных аргументов в функцию. Внутри функции они обрабатываются как кортеж.
- **kwargs: этот параметр позволяет передавать произвольное количество именованных аргументов в функцию.

1. *args (позиционные аргументы):

Параметр *args в определении функции позволяет передавать произвольное количество позиционных аргументов в функцию. Эти аргументы собираются в кортеж.

Пример использования *args:

```
def sum_all(*args):
    total = 0
    for num in args:
        total += num
    return total

print(sum_all(1, 2, 3, 4, 5))  # Вывод: 15
```

Здесь функция sum_all может принимать любое количество аргументов и вычисляет их сумму.

Еще примеры:

```
def concatenate(*args):
    result = ""
    for word in args:
        result += word
    return result

print(concatenate("Hello", " ", "world", "!"))  # Вывод: Hello world!
```

В этом примере функция concatenate принимает произвольное количество аргументов и объединяет их в одну строку.

```
def multiply_numbers(*args):
    result = 1
    for num in args:
        result *= num
    return result

print(multiply_numbers(2, 3, 4))  # Вывод: 24
```

В этом примере функция multiply_numbers принимает произвольное количество числовых аргументов и возвращает их произведение.

2. **kwargs (именованные аргументы):

Параметр **kwargs в определении функции позволяет передавать произвольное количество именованных аргументов в функцию. Эти аргументы собираются в словарь.

Пример использования **kwargs:

```
def print_info(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")

print_info(name="John", age=30, city="New York")
# Вывод:
# name: John
# age: 30
# city: New York
```

Здесь функция print_info принимает произвольное количество именованных аргументов и выводит их на экран.

Еще примеры:

```
def print_contact_info(**kwargs):
    print("Contact information:")
    for key, value in kwargs.items():
        print(f"{key}: {value}")

print_contact_info(name="Alice", phone="123-456-7890", email="alice@example.com")
# Вывод:
# Contact information:
# name: Alice
# phone: 123-456-7890
# email: alice@example.com
```

Здесь функция print_contact_info принимает произвольное количество именованных аргументов, представляющих контактную информацию, и выводит ее на экран.

args и kwargs вместе:

```
def print_info(*args, **kwargs):
    print("Positional arguments:")
    for arg in args:
        print(arg)
    print("Keyword arguments:")
    for key, value in kwargs.items():
        print(f"{key}: {value}")

print_info("Hello", "World", name="Alice", age=30)
# Вывод:
# Positional arguments:
# Hello
# World
# Keyword arguments:
# name: Alice
# age: 30
```

Этот пример позволяет передавать и позиционные, и именованные аргументы в одну функцию, что обеспечивает гибкость в использовании.

```
def black_hole_mixed(var_1, var_2=3, *args, **kwargs):
    print("var_1:", var_1)
    print("var_2:", var_2)
    for arg in args:
        print("*args - ", arg)
    for key, value in kwargs.items():
        print(key, value)


black_hole_mixed( var_1: 1.2,  var_2: "hello",  *args: "world",
                 name="Nick", planet="Earth", galaxy="Milky Way", age=13800000000)

# var_1: 1.2
# var_2: hello
# *args -  world
# name Nick
# planet Earth
# galaxy Milky Way
# age 13800000000
```

Распаковка параметров функции

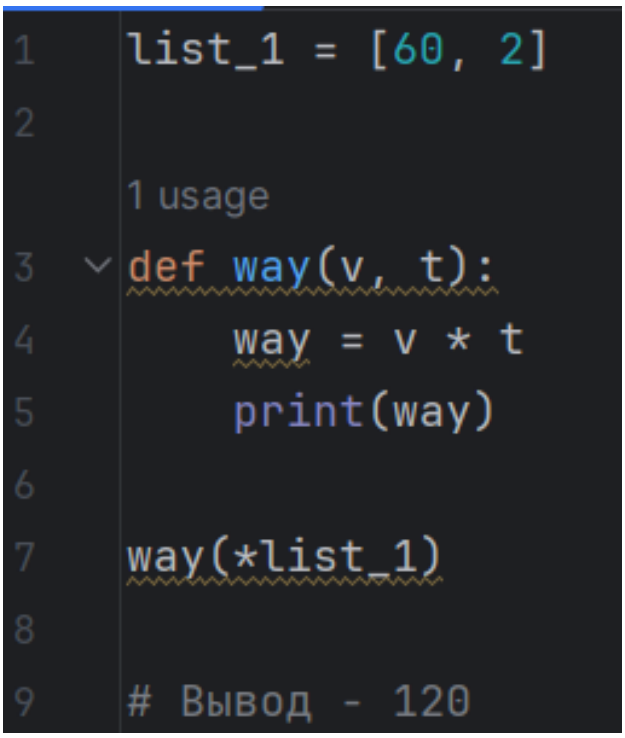
Распаковка параметров функции в Python позволяет передавать аргументы функции не только явно, но и через итерируемые объекты, такие как списки или кортежи. Это делает вызов функции более гибким и удобным.

Рассмотрим несколько способов использования распаковки параметров:

Распаковка позиционных аргументов:

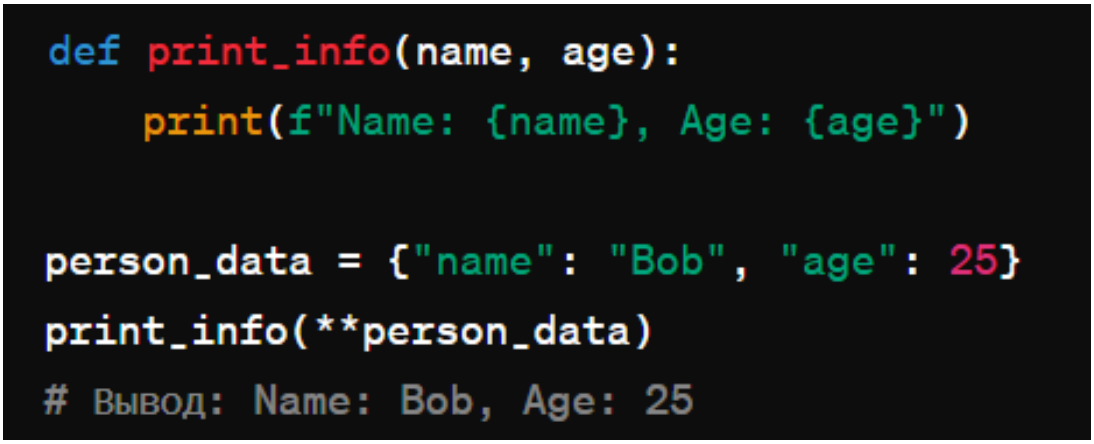
```
def print_info(name, age):
    print(f"Name: {name}, Age: {age}")

person_info = ["Alice", 30]
print_info(*person_info)
# Вывод: Name: Alice, Age: 30
```

Здесь список person_info содержит два элемента: имя и возраст. При вызове функции print_info(*person_info) каждый элемент списка распаковывается и передается в функцию как отдельный позиционный аргумент.



Итак, первый элемент списка был присвоен скорости (v), а второй – времени (t).

Распаковка именованных аргументов:



В этом примере словарь person_data содержит ключи "name" и "age" с соответствующими значениями. При вызове функции print_info(**person_data) каждая пара ключ-значение из словаря распаковывается и передается в функцию как именованный аргумент.

Совместное использование с другими аргументами:

```
def print_info(name, age, city):
    print(f"Name: {name}, Age: {age}, City: {city}")

person_info = ["Alice", 30]
additional_info = {"city": "New York"}

print_info(*person_info, **additional_info)
# Вывод: Name: Alice, Age: 30, City: New York
```

В этом примере мы используем и позиционные, и именованные аргументы. Сначала распаковываем позиционные аргументы из списка person_info, а затем распаковываем именованный аргумент из словаря additional_info. В результате получаем полный набор аргументов для функции.

Практические задания

1. Задачи на args (позиционные аргументы):

a) Напишите функцию sum_numbers, которая принимает произвольное количество целых чисел в качестве аргументов и возвращает их сумму.

b) Создайте функцию max_value, которая принимает произвольное количество чисел и возвращает наибольшее из них.

c) Реализуйте функцию merge_lists, которая принимает произвольное количество списков и возвращает один список, содержащий все элементы из всех переданных списков.

2. Задачи на kwargs (именованные аргументы):

a) Напишите функцию print_info, которая принимает именованные аргументы (имя, возраст, город) и выводит информацию о человеке.

b) Реализуйте функцию create_dict, которая принимает произвольное количество именованных аргументов и возвращает словарь, содержащий переданные аргументы.

3. Задачи на args и kwargs вместе:

Создайте функцию print_info_extended, которая принимает имя человека и произвольное количество именованных аргументов (любые дополнительные данные) и выводит информацию о человеке, включая переданные дополнительные данные.

4. Напишите функцию sum_and_max_of_numbers, которая принимает произвольное количество чисел в качестве аргументов и возвращает их сумму и наибольшее из них.

5. Напишите функцию average, которая принимает произвольное количество чисел в качестве аргументов и возвращает их среднее значение.

6. Напишите функцию merge_lists, которая принимает произвольное количество списков в качестве аргументов и возвращает объединенный список.

7. Напишите функцию contains_element, которая принимает число для поиска и произвольное количество элементов в качестве аргументов и возвращает True, если это число есть в списке переданных элементов, и False в противном случае.
8. Напишите функцию merge_lists, которая принимает произвольное количество списков в качестве аргументов и возвращает один список, содержащий все элементы из переданных списков.
9. Напишите функцию remove_duplicates, которая принимает произвольное количество списков и возвращает список, содержащий уникальные элементы из всех переданных списков.
10. Напишите функцию remove_element, которая принимает произвольное количество списков и элемент, который нужно удалить из них. Функция должна вернуть список, в котором удалены все указанные элементы.