Python IX:
Распаковка, *args и **kwargs

Оператор распаковки * позволяет распаковывать итерабельные объекты, такие как списки или кортежи, при передаче аргументов в функцию. Также он может использоваться для объединения двух списков в один.

```
# Распаковка списка при передаче аргументов в функцию
def my_func(a, b, c):
      print(a, b, c)

my_list = [1, 2, 3]
my_func(*my_list) # выводит: 1 2 3

# Объединение двух списков
list1 = [1, 2, 3]
list2 = [4, 5, 6]
my_list = [*list1, *list2] # [1, 2, 3, 4, 5, 6]
```

Оператор ** используется для распаковки словарей в именованные аргументы функции. Например, если у вас есть словарь с именованными параметрами, вы можете передать его значения в функцию с помощью оператора **:

```
def my_function(x, y, z):
   print(x, y, z)

my_dict = {'x': 1, 'y': 2, 'z': 3}
my_function(**my_dict) # Распаковка словаря в именованные
аргументы   функции
```

Операторы *args и **kwargs позволяют передавать переменное количество аргументов в функцию.

*args позволяет передать произвольное количество позиционных аргументов в виде кортежа. В функции они могут быть обработаны как обычные кортежи:

```
def my_func(*args):
      for arg in args:
      print(arg)

my_func(1, 2, 3) # выводит: 1 2 3
```

**kwargs позволяет передавать произвольное количество именованных аргументов в виде словаря. В функции они могут быть обработаны как обычные словари.

```
def my_func(**kwargs):
    for key, value in kwargs.items():
        print(key, value)
my_func(a=1, b=2, c=3) # выводит: a 1, b 2, c 3
```

Кроме того, операторы *args и **kwargs могут быть использованы вместе при определении функций, чтобы позволить передавать как позиционные, так и именованные аргументы.

```
def my_func(*args, **kwargs):
    for arg in args:
        print(arg)
    for key, value in kwargs.items():
        print(key, value)

my_func(1, 2, 3, a=4, b=5) # выводит: 1 2 3, a 4, b 5
```