Продвинутые функции. Шпаргалка

Как вызвать функцию с позиционными аргументами:

```
def greet(name, greeting):
    return f"{greeting}, {name}!"


result = greet("Alice", "Hello")
print(result)

>>> Hello, Alice!
```

Как вызвать функцию с именованными аргументами:

```
def greet(name, greeting):
    return f"{greeting}, {name}!"


result = greet(greeting="Hi", name="Bob")
print(result)

>>> Hi, Bob!
```

Как создать параметр со значением по умолчанию:

```
def greet(name, greeting="Hello"):
    return f"{greeting}, {name}!"


result = greet("Alice")
print(result)

>>> Hello, Alice!
```

Как передать в функцию параметры неопределенного размера:

```
def example_function(a, b, *args):
    print(f"a: {a}, b: {b}, other args: {args}")


example_function(1, 2, 'three', 4, 5)
>>> a: 1, b: 2, other args: ('three', 4, 5)
```

```
def example_function(a, b, **kwargs):
    print(f"a: {a}, b: {b}, other kwargs: {kwargs}")

example_function(1, b=2, name='Alice', age=30)

>>> a: 1, b: 2, other kwargs: {'name': 'Alice', 'age': 30}
```

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


print_args_and_kwargs(1, 'two', 3.0, name='John', age=25)

>>> 1
    two
    3.0
    name: John
    age: 25
```

Как распаковать аргументы из коллекции при передаче в функцию:

```
def example_function(a, b, c):
    print(f"a: {a}, b: {b}, c: {c}")

# Создаем список
my_list = [1, 2, 3]

# Распаковываем список и передаем его значения в функцию
example_function(*my_list)

>>> a: 1, b: 2, c: 3
```

```
def example_function(name, age, city):
    print(f"name: {name}, age: {age}, city: {city}")

# Создаем словарь
my_dict = {'name': 'Alice', 'age': 30, 'city': 'Wonderland'}

# Распаковываем словарь и передаем его значения в функцию
example_function(**my_dict)

>>> name: Alice, age: 30, city: Wonderland
```

Как использовать лямбда-функции:

```
multiply = lambda x, y: x * y
result = multiply(3, 4)
print(result)

>>> 12
```

Как обработать исключения:

```
try:
    # Код, который может вызвать исключение
    result = 10 / 2
except ZeroDivisionError:
    # Код, который выполняется в случае исключения
    print("Ошибка: Вы пытаетесь сломать математику. Делить на 0 нельзя!")
else:
    # Код, который выполняется, если исключение не произошло
    print(f"Результат: {result}")
finally:
    # Код, который всегда выполняется независимо от того,
    # произошло исключение или нет
    print("Программа завершает работу")

>>> Результат: 5
    Программа завершает работу
```

Как самостоятельно вызвать исключение:

```
def divide(x, y):
    if y == 0:
        raise ZeroDivisionError("Cannot divide by zero")
    return x / y


try:
    result = divide(10, 0)
except ZeroDivisionError as e:
    print(f"Error: {e}")
```