Моя IT Школа

сертифицированные IT курсы

Функции. Модули. Пакеты

План занятия

Моя IT Школа
сертифицированные IT курсы

1. Модули и пакеты
2. Функции
3. Пустая функция (stub)
4. Передача аргументов функции
5. Область видимости
6. Анонимная функция

Модули и пакеты

Моя IT Школа

Под модулем в Python понимается файл с расширением .py. Модули предназначены для того, чтобы в них хранить часто используемые функции, классы, константы и т.п. Можно условно разделить модули и программы: программы предназначены для непосредственного запуска, а модули для импортирования их в другие программы.
Примеры модулей, которые мы использовали ранее уже: math,random

Пакет в Python – это каталог, включающий в себя другие каталоги и модули, но при этом дополнительно содержащий файл __init__.py. Пакеты используются для формирования пространства имен, что позволяет работать с модулями через указание уровня вложенности (через точку)

Пакет fincal содержит в себе модули для работы с простыми процентами (simper.py), сложными процентами (compper.py) и аннуитетами (annuity.py).

Пример пакета и его вызова его компонентов в код:

```
fincalc
|-- __init__.py
|-- simper.py
|-- compper.py
|-- annuity.py
```

```
import fincalc.simper
fv = fincalc.simper.fv(pv, i, n)
```

```
import fincalc.simper as sp
fv =sp.fv(pv, i, n)
```

```
from fincalc import simper
fv = simper.fv(pv, i, n)
```

Функция

Моя Школа

Функция – это структура, которую вы определяете. Вам нужно решить, будут ли в ней аргументы, или нет. Вы можете добавить как аргументы ключевых слов, так и готовые по умолчанию. **Функция** – это блок кода, который начинается с ключевого слова def, названия функции и двоеточия, пример:

```
def a_function():
    print("You just created a function!")
```

Функция

Моя Школа

Эта функция не делает ничего, кроме отображения текста. Чтобы вызвать функцию, вам нужно ввести название функции, за которой следует открывающаяся и закрывающаяся скобки:

```
def a_function():
    print("You just created a function!")


a_function()
```

Пустая функция (stub)

Моя Школа

Иногда, когда вы пишете какой-нибудь код, вам нужно просто ввести определения функции, которое не содержит в себе код.
Вот пример:

```
def empty_function():
    pass
```

А вот здесь кое-что новенькое: оператор pass. Это пустая операция, это означает, что когда оператор pass выполняется, не происходит ничего.

Задание №1

Создайте функцию, которая будет считать сумму чисел в массиве.

Решение

Моя IT Школа

```
def a():
    sum = 0
    for i in range(10):
        sum += i
    print(sum)


a()
```

Передача аргументов функции

Моя Школа

Теперь мы готовы узнать о том, как создать функцию, которая может получать доступ к аргументам, а также узнаем, как передать аргументы функции.
Создадим простую функцию, которая может суммировать два числа:

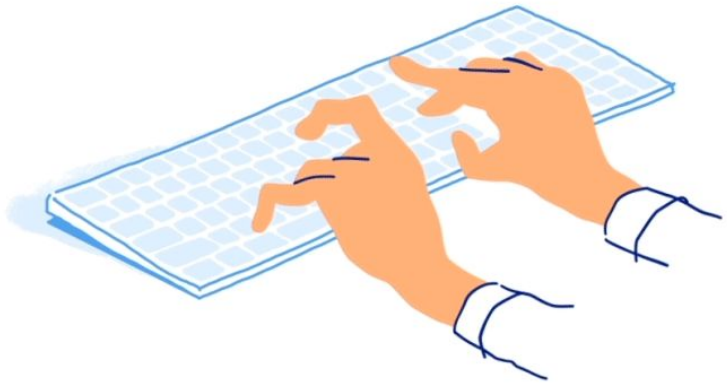
```
def add(a, b):
    return a + b


print(add(1, 2))  # 3
```

Передача аргументов функции

Последняя строка в блоке инструкций может начинаться с return, если нужно вернуть какое-то значение. Если инструкции return нет, тогда по умолчанию функция будет возвращать объект None.

В нашем примере мы указали выдать результат a + b. Как вы видите, мы можем вызвать функцию путем передачи двух значений. Если вы передали недостаточно, или слишком много аргументов для данной функции, вы получите ошибку.



Передача аргументов функции

Моя Школа

Вы также можете вызвать функцию, указав наименование аргументов:

```
def add(a, b):
    return a + b


print(add(a=2, b=3))  # 5

total = add(b=4, a=5)
print(total)  # 9
```

Стоит отметить, что **не важно, в каком порядке вы будете передавать аргументы функции** до тех пор, как они называются корректно. Во втором примере мы назначили результат функции переменной под названием total. Это стандартный путь вызова функции в случае, если вы хотите дальше использовать её результат.

Аргументы

Моя IT Школа

В функции можно использовать неограниченное количество параметров, но число аргументов должно точно соответствовать параметрам. Эти параметры представляют собой позиционные аргументы. Также Python предоставляет возможность определять значения по умолчанию, которые можно задавать с помощью аргументов-ключевых слов.

Параметр — это имя в списке параметров в первой строке определения функции. Он получает свое значение при вызове. **Аргумент** — это реальное значение или ссылка на него, переданное функции при вызове. В этой функции:

```
def add(a, b):
    return a + b
```

a и b — это параметры

```
add(1, 2)
```

1 и 2 — аргументы.

Область видимость и глобальные переменные

Моя IT Школа

Концепт области (scope) в Пайтон такой же, как и в большей части языков программирования. Область видимости указывает нам, когда и где переменная может быть использована. Если мы определяем переменные внутри функции, эти переменные могут быть использованы только внутри этой функции. Когда функция заканчиваются, их можно больше не использовать, так как они находятся вне области видимости.

Область видимость и глобальные переменные

Моя Школа

Давайте взглянем на пример:

```
def function_a():
    a = 1
    b = 2
    return a + b


def function_b():
    c = 3
    return a + c


print(function_a())
print(function_b())
```

Область видимость и глобальные переменные

Моя Школа

Если вы запустите этот код, вы получите ошибку:

```
NameError: name 'a' is not defined

Process finished with exit code 1
```

Это вызвано тем, что переменная определена только внутри первой функции, но не во второй. Вы можете обойти этот момент, указав в Пайтоне, что переменная a – глобальная (global).

Область видимость и глобальные переменные

Моя Школа

Давайте посмотрим на то, как это работает:

```
def function_a():
    global a
    a = 1
    b = 2
    return a + b


def function_b():
    c = 3
    return a + c


print(function_a())
print(function_b())
```

Область видимость и глобальные переменные

Моя Школа

Этот код работает, так как мы указали Пайтону сделать a – глобальной переменной, а это значит, что она работает где-либо в программе. Из этого вытекает, что это настолько же хорошая идея, насколько и плохая. Причина, по которой эта идея – плохая в том, что нам становится трудно сказать, когда и где переменная была определена. Другая проблема заключается в следующем: когда мы определяем «a» как глобальную в одном месте, мы можем случайно переопределить её значение в другом, что может вызвать логическую ошибку, которую не просто исправить.

Функцию можно записать в одну строку, если блок инструкций представляет собой простое выражение:

```
def sum(a, b): return a + b



print(sum(1, 5))
```

Анонимная функция: лямбда

Моя Школа

Лямбда-функция — это короткая однострочная функция, которой даже не нужно имя давать. Такие выражения содержат лишь одну инструкцию, поэтому, например, if, for и while использовать нельзя.
Их также можно присваивать переменным:

```
product = lambda x, y: x * y

print(product(2, 3))
```

Анонимная функция: лямбда

Моя Школа

В отличие от функций, здесь не используется ключевое слово return. Результат работы и так возвращается.

С помощью type() можно проверить тип:

```
type(product)
```

```
<class 'function'>
```

Анонимная функция: лямбда

Моя Школа

На практике эти функции редко используются. Это всего лишь элегантный способ записи, когда она содержит одну инструкцию.

```
power = lambda x=1, y=2: x ** y
square = power
print(square(5))
```

Задание №2

Написать функцию is_year_leap, принимающую 1 аргумент — год, и возвращающую True, если год високосный, и False иначе.

Решение

Моя Школа

```python
def is_year_leap(year):
    return year % 4 == 0 and year % 100 != 0 or year % 400 == 0


print(is_year_leap(int(input('Введите год: '))))
```

Задание №3

Написать функцию square, принимающую 1 аргумент — сторону квадрата, и возвращающую 3 значения (с помощью кортежа): периметр квадрата, площадь квадрата и диагональ квадрата. Сторону квадрата вводить с клавиатуры.

Решение

Моя Школа

```python
import math


def square(a):
    p = 4 * a
    s = a ** 2
    d = math.sqrt(2) * a

    return p, s, d


print(square(int(input('Введите сторону квадрата: '))))
```

Задание №4

Написать функцию season, принимающую 1 аргумент — номер месяца (от 1 до 12), и возвращающую время года, которому этот месяц принадлежит (зима, весна, лето или осень). Номер месяца вводить с клавиатуры.

Решение

Моя IT Школа

```python
def season(num):
    if num == 12 or 1 <= num <= 2:
        print("Зима")

    elif 3 <= num <= 5:
        print("Весна")

    elif 6 <= num <= 8:
        print("Лето")

    elif 9 <= num <= 11:
        print("Осень")

    else:
        print("Неверно введён номер месяца!")


n = int(input("Введите номер месяца (1-12): "))

season(n)
```

Задание №5

Написать функцию is_prime, принимающую 1 аргумент — число от 0 до 1000, и возвращающую True, если оно простое, и False - иначе.

Решение

Моя Школа

```python
def is_prime(n):
    d = 2
    while n % d != 0:
        d += 1
    return d == n


n = int(input('Введите число: '))
print(is_prime(n))
```

Задание №6

Функция, вычисляющая среднее арифметическое элементов массива. Массив должен состоять из случайных чисел, длинной 10 элементов.

Решение

Моя Школа

```python
import random

N = 10

arr = [0] * N

for i in range(N):
    arr[i] = random.randint(1, 100)


def average(arr):
    s = 0
    for i in range(N):
        s += arr[i]
    return s / N


print(arr)
print(average(arr))
```

Домашнее задание

Простейший калькулятор с введёнными двумя числами вещественного типа.

Ввод с клавиатуры: операции + - * / и два числа. Операции являются функциями.

Обработать ошибку: “Деление на ноль”

Ноль использовать в качестве завершения программы (сделать как отдельную операцию).