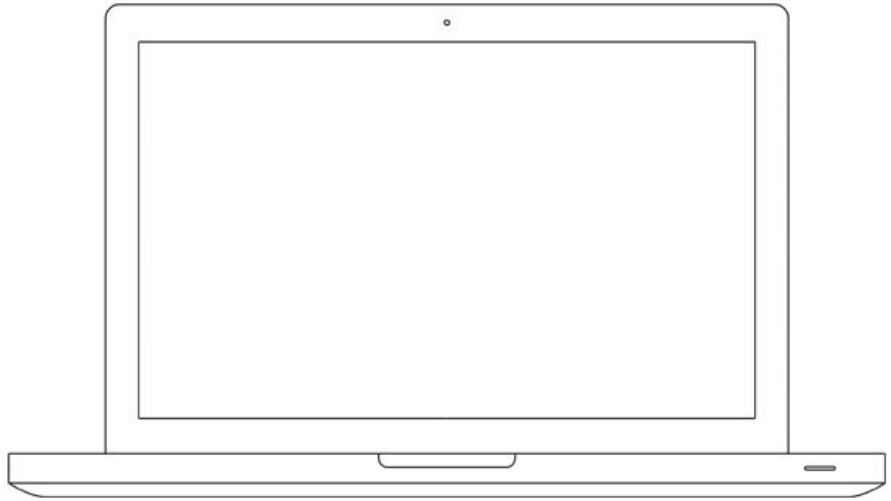
Моя IT Школа

сертифицированные IT курсы

Функции



План занятия

Моя IT Школа
сертифицированные IT курсы

1. Рекурсивные функции
2. Ключевые и позиционные аргументы
3. Присвоение функции переменной
4. Функция внутри функции в Python

Ключевые аргументы

Моя Школа

Функции также могут принимать ключевые аргументы. Более того, они могут принимать как регулярные, так и ключевые аргументы. Это значит, что вы можете указывать, какие ключевые слова будут ключевыми, и передать их функции.

Вы также можете вызвать данную функцию без спецификации ключевых слов. Эта функция также демонстрирует концепт аргументов, используемых по умолчанию.

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


print(keyword_function(b=4, a=5))  # 9
print(keyword_function())          # 3
```

Причина заключается в том, что **a** и **b** по умолчанию **имеют значение** 1 и 2 соответственно.

Ключевые аргументы

Моя Школа

Теперь попробуем создать функцию, которая имеет обычный аргумент, и несколько ключевых аргументов:

```
def mixed_function(a, b=2, c=3):
    return a + b + c



mixed_function(b=4, c=5)

print(mixed_function(1, b=4, c=5))  # 10

print(mixed_function(1))  # 6
```

Ключевые аргументы

Моя IT Школа

Выше мы описали три возможных случая. Проанализируем каждый из них.
В первом примере мы попробовали вызвать функцию, используя только ключевые аргументы. Это дало нам только ошибку. Traceback указывает на то, что наша функция принимает, по крайней мере, один аргумент, но в примере было указано два аргумента. Что же произошло? Дело в том, что первый аргумент необходим, потому что он ни на что не указывает, так что, когда мы вызываем функцию только с ключевыми аргументами, это вызывает ошибку.
Во втором примере мы вызвали смешанную функцию, с тремя значениями, два из которых имеют название. Это работает, и выдает нам ожидаемый результат: 1+4+5=10.
Третий пример показывает, что происходит, если мы вызываем функцию, указывая только на одно значение, которое не рассматривается как значение по умолчанию. Это работает, если мы берем 1, и суммируем её к двум значениям по умолчанию: 2 и 3, чтобы получить результат 6!

```
Traceback (most recent call last):
  File "C:\Users\dpris\OneDrive\Рабочий стол\Work_MyITSchool\6. Функции в программировании\6.py", line 5, in <module>
    mixed_function(b=4, c=5)
TypeError: mixed_function() missing 1 required positional argument: 'a'
```

```
10
6


Process finished with exit code 0
```

*args и **kwargs

Моя IT Школа

Вы также можете настроить функцию на прием любого количества аргументов, или ключевых аргументов, при помощи особого синтаксиса. Чтобы получить бесконечное количество аргументов, мы используем *args, а чтобы получить бесконечное количество ключевых аргументов, мы используем **kwargs. Сами слова “args” и “kwargs” не так важны. Это просто сокращение. Вы можете назвать их *lol и **omg, и они будут работать таким же образом. Главное здесь – это количество звездочек.
Давайте взглянем на следующий пример:

```
def many(*args, **kwargs):
    print(args)
    print(kwargs)


many(1, 2, 3, name="Mike", job="programmer")

# Результат:
# (1, 2, 3)
# {'name': 'Mike', 'job': 'programmer'}
```

*args и **kwargs

Моя IT Школа

Сначала мы создали нашу функцию, при помощи нового синтаксиса, после чего мы вызвали его при помощи трех обычных аргументов, и двух ключевых аргументов. Функция показывает нам два типа аргументов. Как мы видим,
параметр args превращается в кортеж, а kwargs – в словарь.

*args и **kwargs

Моя Школа

Вот несколько советов, которые помогут вам избежать распространённых проблем, возникающих при работе с функциями, и расширить свои знания:

- Используйте общепринятые конструкции *args и **kwargs для захвата позиционных и именованных аргументов.
- Конструкцию **kwargs нельзя располагать до *args. Если это сделать — будет выдано сообщение об ошибке.
- Остерегайтесь конфликтов между именованными параметрами и **kwargs, в случаях, когда значение планируется передать как **kwarg-аргумент, но имя ключа этого значения совпадает с именем именованного параметра.
- Оператор *можно использовать не только в объявлениях функций, но и при их вызове.

Рекурсивные функции

Моя Школа

Рекурсия — это не особенность Python. Это общепринятая и часто используемая техника в Computer Science, когда функция вызывает сама себя. Самый известный пример — вычисление факториала n! = n * n — 1 * n -2 * … 2 *1. Зная, что 0! = 1, факториал можно записать следующим образом:

```
def factorial(n):
    if n != 0:
        return n * factorial(n - 1)
    else:
        return 1


print(factorial(5))
```

Присвоение функции переменной

Моя Школа

С существующей функцией func синтаксис максимально простой:

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


variable = add(1, 2)

print(variable)
```

Присвоение функции переменной

Моя Школа

Переменным также можно присваивать встроенные функции. Таким образом позже есть возможность вызывать функцию другим именем. Такой подход называется непрямым вызовом функции.
Менять название переменной также разрешается:

```
def func(x): return x


a1 = func
a2 = a1

print(a2(10))
```

В этом примере a1, a2 и func имеют один и тот же id. Они ссылаются на один объект.

Присвоение функции переменной

Моя Школа

Переменным также можно присваивать встроенные функции. Таким образом позже есть возможность вызывать функцию другим именем. Такой подход называется непрямым вызовом функции.
Менять название переменной также разрешается:

```
def func(x): return x


a1 = func
a2 = a1

print(a2(10))
```

В этом примере a1, a2 и func имеют один и тот же id. Они ссылаются на один объект.

Присвоение функции переменной

Моя Школа

Практический пример — рефакторинг существующего кода. Например, есть функция sq, которая вычисляет квадрат значения:

```
def sq(x): return x * x
```

Позже ее можно переименовать, используя более осмысленное имя. Первый вариант — просто сменить имя. Проблема в том, что если в другом месте кода используется sq, то этот участок не будет работать. Лучше просто добавить следующее выражение:

```
square = sq
```

Функция внутри функции в Python

Моя Школа

Функции в Python мы можем создавать, вызывать и возвращать из других функций. Кстати, на этом основана идея замыкания (closures) в Python.

Давайте создадим функцию, умножающую 2 числа:

```
def mul(a):
    def helper(b):
        return a * b

    return helper
```

Функция внутри функции в Python

Моя Школа

В этой функции в Python реализованы два важных свойства:

- внутри функции mul() мы создаём ещё одну функцию helper();
- функция mul() возвращает нам функцию helper() в качестве результата работы.

Вызов этой функции в Python:

```
print(mul(3)(2))
```

Функция внутри функции в Python

Моя Школа

Особенность заключается в том, что мы можем создавать на базе функции mul() собственные кастомизированные функции.
Давайте создадим функцию в Python, умножающую на 3:

```
def mul(a):
    def helper(b):
        return a * b

    return helper


three_mul = mul(3)

print(three_mul(5))
```

В результате была построена функция three_mul(), умножающая на 3 любое переданное ей число.

Задание №1

Написать функцию, которая определяет количество разрядов введенного целого числа.

Решение

Моя Школа

```python
def digits(n):
    i = 0
    while n > 0:
        n = n // 10
        i += 1
    return i


num = abs(int(input('Введите число: ')))
print('Количество разрядов:', digits(num))
```

Задание №2

В зависимости от выбора пользователя вычислить площадь круга, прямоугольника или треугольника. Для вычисления площади каждой фигуры должна быть написана отдельная функция.

Решение

Моя Школа

```python
import math


def circle(r):
    return math.pi * r ** 2


def rectangle(a, b):
    return a * b


def triangle(a, b, c):
    p = (a + b + c) / 2
    return math.sqrt(p * (p - a) * (p - b) * (p - c))
```

```python
choice = input("Круг(к), прямоугольник(п) или треугольник(т): ")
if choice == 'к':
    rad = float(input("Радиус: "))
    print("Площадь круга: %.2f" % circle(rad))
elif choice == 'п':
    l = float(input("Длина: "))
    w = float(input("Ширина: "))
    print("Площадь прямоугольника: %.2f" % rectangle(l, w))
elif choice == 'т':
    AB = float(input("Первая сторона: "))
    BC = float(input("Вторая сторона: "))
    CA = float(input("Третья сторона: "))
    print("Площадь треугольника: %.2f" % triangle(AB, BC, CA))
```

Задание №3

Написать функцию, которая заполняет массив длинной 10 элементов, случайными числами в диапазоне, указанном пользователем с клавиатуры. Функция должна принимать два аргумента – начало диапазона и его конец, при этом ничего не возвращать.

Решение

Моя Школа

```
import random

N = 10
a = [0] * N


def func(mn, mx):
    for i in range(N):
        a[i] = random.randint(mn, mx)


mn = int(input('Начало диапазона: '))
mx = int(input('Конец диапазона: '))
func(mn, mx)
print(a)
```

Задание №4

Написать функцию и сделать так, чтобы число секунд отображалось в виде дни:часы:минуты:секунды.

Решение

Моя Школа

```python
def convert(seconds):
    days = seconds // (24 * 3600)
    seconds %= 24 * 3600
    hours = seconds // 3600
    seconds %= 3600
    minutes = seconds // 60
    seconds %= 60
    print(f'{days}:{hours}:{minutes}:{seconds}')


convert(1234565)
```

Задание №5

Написать функцию, которая считает сколько гласных и согласных в строке. Строку вводить с клавиатуры.

Решение

Моя Школа

```
def func(a):
    h = 0
    d = 0
    for i in a:
        if i.isalpha():
            if i in "aeoiu":
                h += 1
            else:
                d += 1
    print("Гласные", h)
    print("Согласные", d)


func(input('Введите строку: '))
```

Задание №6

Функцию которая при заданном целом числе n посчитает n + nn + nnn.

Решение

Моя Школа

```python
def solve(n):
    n1 = n
    n2 = int(str(n) * 2)
    n3 = int(str(n) * 3)
    print(n1 + n2 + n3)


solve(5)
```

Задание №6

Вычислить значения нижеприведенной функции в диапазоне значений x от -10 до 10 включительно с шагом, равным 1.

y = x^2 при -5 <= x <= 5;

y = 2*|x|-1 при x < -5;

y = 2x при x > 5.

Вычисление значения функции оформить в виде программной функции, которая принимает значение x, а возвращает полученное значение функции (y).

Решение

Моя Школа

```
def func(x):
    if -5 <= x <= 5:
        return x * x
    elif x < -5:
        return 2 * abs(x) - 1
    else:
        return 2 * x


for i in range(-10, 11):
    print(func(i), end=' ')
print()
```

Домашнее задание

Если в функцию передаётся кортеж, то посчитать длину всех его слов.
Если список, то посчитать кол-во букв и чисел в нём.
Число – кол-во нечётных цифр.
Строка – кол-во букв.
Сделать проверку со всеми этими случаями.

Домашнее задание

Если в функцию передаётся кортеж, то посчитать длину всех его слов.
Если список, то посчитать кол-во букв и чисел в нём.
Число – кол-во нечётных цифр.
Строка – кол-во букв.
Сделать проверку со всеми этими случаями.