# Урок 08.02. Функции с одним параметром

Вспомним, что функциями мы пользуемся постоянно, и первое что приходит в голову, это print и input. Так как же они работают. Чтобы немного лучше их понимать рассмотрим задачу «Почта».

Необходимо написать программу вывода информации о пользователе (фамилия, имя, улица, дом) три раза без использования цикла и с использованием функции.

Вот эта программа:

def user\_info():

 print('Фамилия: Иванов')

 print('Имя: Артем')

 print('Улица: Пушкина')

 print('Дом: 13')

user\_info()

user\_info()

user\_info()

А теперь представим, что теперь нам нужно вывести информацию о трех членах семьи: Артеме, Василии, Алене. Адрес и фамилия у них всех совпадает, а вот имя у каждого своё. Можем перед каждым вызовом функции написать это имя, а из функции удалить такую строчку, но тогда сначала будет идти имя, а потом фамилия, что не правильно.

Посмотрим на нашу программу: вообще на месте имени у нас должно быть всё время разное имя, т.е. должно меняться, таким свойством обладают переменные, так давайте поменяем в программе имя «Артем» на переменную name. И теперь нам пришло время узнать для чего нужны скобки. Внутри функции наша переменная должна быть всё время разная, т.е. наша функция должна вызваться с разным значением переменной name. И для того, чтобы функции получила эту переменную и нужны круглые скобки. Именно в них и пишется имя переменной, которую мы передаем в функцию. А при вызове функции также нужно эту переменную заполнять. И такая программа будет выглядеть вот так:

def user\_info(name):

 print('Фамилия: Иванов')

 print('Имя:', name)

 print('Улица: Пушкина')

 print('Дом: 13')

 print()

user\_info('Артём')

user\_info('Василий')

user\_info('Алёна')

Такую переменную чаще всего называют параметр или аргумент.

Если при вызове функции убрать аргумент, то программа выдаст ошибку «TypeError: user\_info() missing 1 required positional argument: 'name'», которая говорит, что при вызове функции не хватает одного обязательного позиционного аргумента «name».

## Функции с несколькими параметрами

В прошлом уроке мы увидели как можно передавать параметр (или аргумент) в написанную нами функцию. Но что делать если таких параметров несколько? Как с ними работать. И снова будем учиться на примерах.

Возьмем задачу, которую мы уже решали, «Семья». Нам нужно вывести данные о членах семьи, но теперь они все живут в разных домах, но на одной улице.

Возьмем наш ранее написанный код:

def user\_info(name):

 print('Фамилия: Иванов')

 print('Имя:', name)

 print('Улица: Пушкина')

 print('Дом: 13')

 print()

user\_info('Артём')

user\_info('Василий')

user\_info('Алёна')

и будем его модифицировать. Для начала начнем действовать как в предыдущий раз. Нам нужно менять номер дома, значит в функции это изменяемая часть, а значит это лучше сделать через переменную, соответственно произведем замену:

def user\_info(name):

 print('Фамилия: Иванов')

 print('Имя:', name)

 print('Улица: Пушкина')

 print('Дом:', house)

 print()

user\_info('Артём')

user\_info('Василий')

user\_info('Алёна')

А теперь самое интересное и простое. Номер нашего дома, как и имя, должно передаваться в функцию как аргумент, и делается это просто – нужно через запятую указать этот аргумент, а при вызове функции также через запятую передать значение этого аргумента.

def user\_info(name, house):

 print('Фамилия: Иванов')

 print('Имя:', name)

 print('Улица: Пушкина')

 print('Дом:', house)

 print()

user\_info('Артём', 7)

user\_info('Василий', 11)

user\_info('Алёна', 19)

Таким образом, мы с вами теперь передаем два аргумента и функция принимает два аргумента. Стоит обратить внимание на порядок, а именно в каком порядке функция принимает аргументы, в таком порядке значения аргументов и должны быть перечислены при вызове функции. Такие аргументы называются позиционными, т.е. значение аргументов зависит от позиции на кокой он стоит.

Для закрепления рассмотрим ещё одну старую задачу «GPS-навигатор 2.0».

Нам поручили усовершенствовать GPS-навигатор, добавив в него новую фишку. Теперь пользователь может не только смотреть расстояние от себя до объекта, но и задавать в навигаторе две произвольные точки, после чего на экран ему выводится расстояние между ними. Для этого пользователь вводит четыре действительных числа x1, y1, x2, y2 — это как раз координаты этих двух точек.

Напишите программу, где у пользователя спрашивается, чего он хочет — найти расстояние от себя до точки или найти расстояние между двумя произвольными точками, после чего запрашиваются необходимые координаты точек и выводится ответ на экран.

Примечание. Координаты каждой точки задается двумя числами x и y. Расстояния между двумя точками (x1, y1) и (x2, y2) рассчитывается следующим образом: $d=\sqrt{\left(x2-x1\right)^{2}+\left(y2-y1\right)^{2}}$.

Возьмем ранее написанную программу:

import math

x = float(input('Введите координату x: '))

y = float(input('Введите координату y: '))

distance = math.sqrt(x \*\* 2 + y \*\* 2)

print('Расстояние:', distance)

И теперь модифицируем её: вычисление расстояния от нос до точки уберем в отдельную функцию, и добавим выбор для пользователя.

import math

def my\_distance(x, y):

 distance = math.sqrt(x \*\* 2 + y \*\* 2)

 print('Расстояние:', distance)

choice = input('1 - расстояние от меня до точки;\n'

 '2 - расстояние от точки до точки\n'

 'Выберете действие: ')

if choice == '1':

 x1 = float(input('Введите координату x: '))

 y1 = float(input('Введите координату y: '))

 my\_distance(x1, y1)

Конечно, от начальной программы наша уже очень сильно отличается. Но продолжим.

Теперь нам нужно написать новые возможности, для этого создадим новую функцию вычисления расстояния между двумя точками, добавим запрос на ввод координат двух точек и вызов функции:

import math

def my\_distance(x, y):

 distance = math.sqrt(x \*\* 2 + y \*\* 2)

 print('Расстояние:', distance)

def distance2(x1, y1, x2, y2):

 distance = math.sqrt((x2 - x1) \*\* 2 + (y2 - y1) \*\* 2)

 print('Расстояние между точками:', distance)

choice = input('1 - расстояние от меня до точки;\n'

 '2 - расстояние от точки до точки\n'

 'Выберете действие: ')

if choice == '1':

 x1 = float(input('Введите координату x: '))

 y1 = float(input('Введите координату y: '))

 my\_distance(x1, y1)

elif choice == '2':

 x1 = float(input('Введите координату x1: '))

 y1 = float(input('Введите координату y1: '))

 x2 = float(input('Введите координату x2: '))

 y2 = float(input('Введите координату y2: '))

 distance2(x1, y1, x2, y2)

Ну вот и всё. Наша программа готова.

## Вложенный вызов функций

Мы продолжаем углублять наши знания в программировании. И наши программы уже напоминают настоящие программы, где есть основная программа откуда уже вызываются различные функции. И давайте рассмотрим следующую задачу «Игра».

Мы пишем свою игру, где человеку нужно будет делать выбор, от которого будет зависеть дальнейшая судьба его персонажа. Для начала реализуем простые действия «сделать что-то хорошее» и «сделать что-то плохое». При этом если «сделали что-то плохое», то программа завершается, а если «сделали что-то хорошее», то программа продолжает спрашивать. Эти условия пока все. Давайте сделаем эту программу.

Начнем рассуждать с того, что если будет большая игра, то и функций у нас будет много, однако всё будет запускаться из главного меню с выбором, значит и наша программа должна начинаться с того, что выводится главное меню с выбором вариантов ответа и вызовом соответствующих функций.

def main\_menu():

 pass

main\_menu()

Вот такая простая заготовка. Напомню, что команда pass ничего не делает и является «заглушкой», т.е. команда по синтаксису языка программирования должна быть, но мы пока не знаем, что писать и поэтому пишем команду, которая ничего не делает.

Продолжим и добавим вывод вариантов выбора условный оператор для них.

def main\_menu():

 print('1 - сделать что-то хорошее')

 print('2 - сделать что-то плохое')

 choice = int(input('Выберете что сделать: '))

 if choice == 1:

 pass

 elif choice == 2:

 pass

 else:

 print('Ошибка выбора')

main\_menu()

Сейчас в нашей программе всё просто, нужно выводить текст после каждого выбора, а это одна строчка. Конечно, мы можем написать её в условном операторе, но мы хотим в дальнейшем усложнять нашу программу, и каждый выбор может быть более объемней, поэтому заранее вынесем действия на каждый выбор в отдельную функцию.

def main\_menu():

 print('1 - сделать что-то хорошее')

 print('2 - сделать что-то плохое')

 choice = int(input('Выберете что сделать: '))

 if choice == 1:

 good()

 elif choice == 2:

 bad()

 else:

 print('Ошибка выбора')

def good():

 print('Всё хорошо!')

def bad():

 print('Всё плохо!')

main\_menu()

Запустим программу, введем 1. И получим, что всё хорошо, а наша программа завершилась. Кажется, что надо сделать бесконечный цикл… Но есть вариант немного попроще. На самом деле после вывода сообщения о том, что всё хорошо, нужно просто ещё раз вызвать главное меню. Также нужно вызвать главное меню и при вводе ошибочного варианта выбора.

def main\_menu():

 print('1 - сделать что-то хорошее')

 print('2 - сделать что-то плохое')

 choice = int(input('Выберете что сделать: '))

 if choice == 1:

 good()

 elif choice == 2:

 bad()

 else:

 print('Ошибка выбора')

 main\_menu()

def good():

 print('Всё хорошо!')

 main\_menu()

def bad():

 print('Всё плохо!')

main\_menu()

И теперь наша программа работает правильно. Однако тут мы познакомились с очень важной конструкцией, но об этом в дальнейшем.