Python programming for beginners

Stefan Zhauryd
Instructor

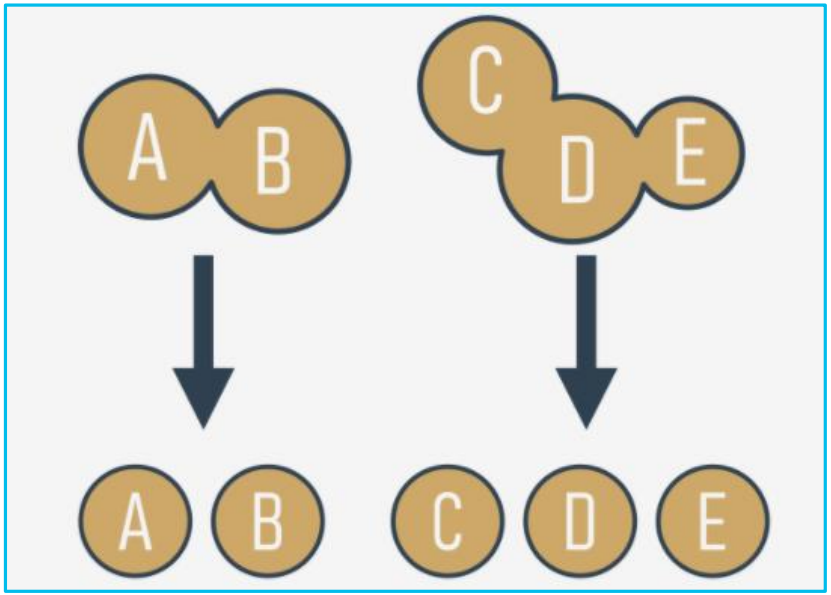
Module 4

Functions, Tuples, Dictionaries, and Data Processing

In this module, you will learn about:

- **code structuring and the concept of function;**
- **function invocation and returning a result from a function;**
- name scopes and variable shadowing;
- tuples and their purpose, constructing and using tuples;
- dictionaries and their purpose, constructing and using dictionaries.

Why do we need functions?

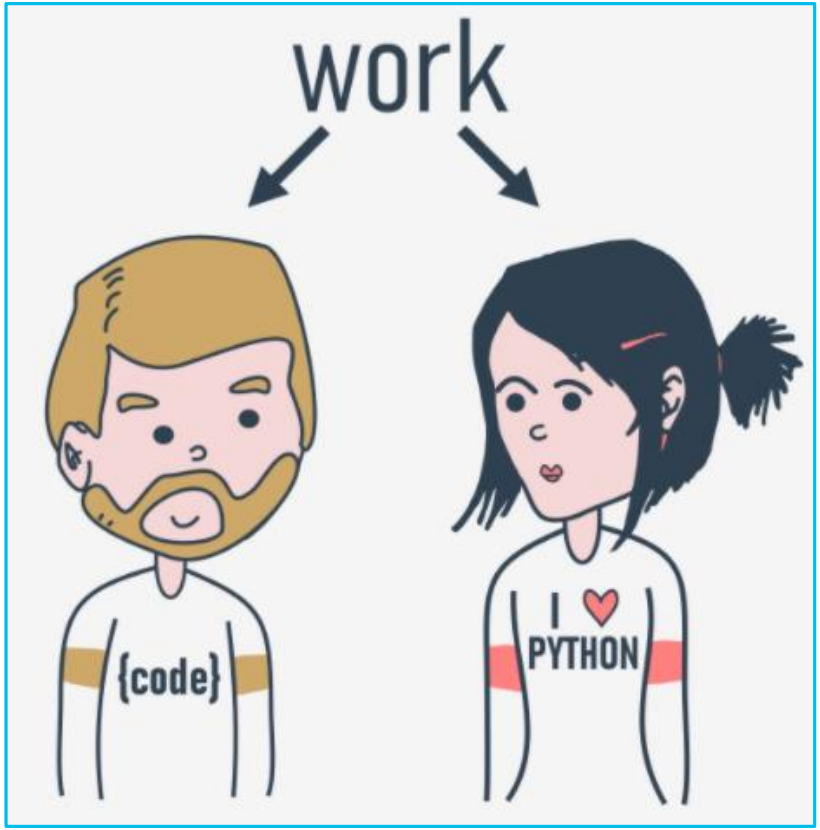


print() input() int() float()

len() bin() str() and so on

.strip(" ")

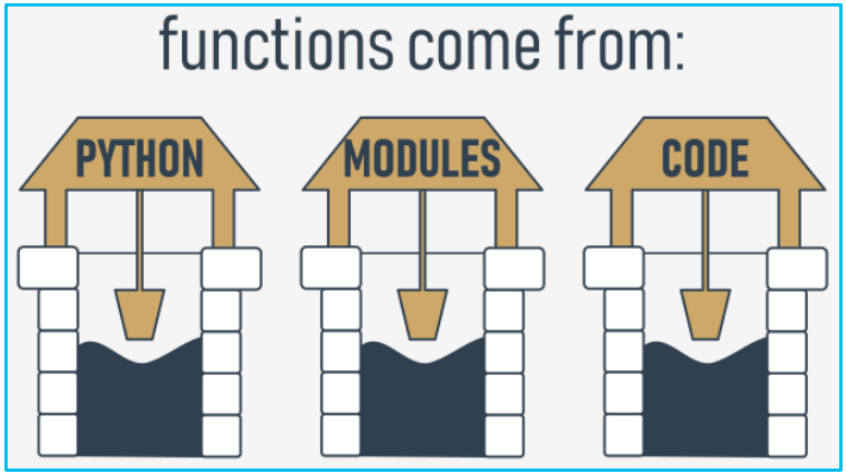
Decomposition



Where do the functions come from?

- from Python itself
- from Python's preinstalled modules
- directly from your code

there is one other possibility, but it's connected with classes, so we'll omit it for now.



Your first function

```
1  print("Enter a value: ")
2  a = int(input())
3
4  print("Enter a value: ")
5  b = int(input())
6
7  print("Enter a value: ")
8  c = int(input())
```

Console >_

```
Enter a value:
23
Enter a value:
32
Enter a value:
22
```

Your first function **def**

```
def function_name():
    function_body
```

```
1  def message():
2      print("Enter a value: ")
3      a = input()
4      print(a)
5
6  print("We start here.")
7  message()
8  print("We end here.")
```

Console >_

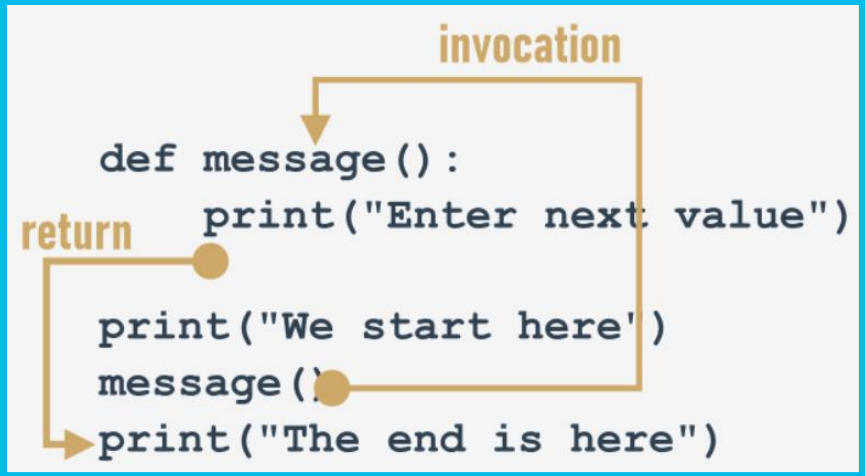
```
We start here.
Enter a value:
44444
44444
We end here.
```

```
print("We start here.")
message()
print("We end here.")


def message():
    print("Enter a value: ")
```

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

How functions work



You mustn't invoke a function which is not known at the moment of invocation.

Remember - Python reads your code from top to bottom. It's not going to look ahead in order to find a function you forgot to put in the right place ("right" means "before invocation".)

```
def message():
    print("Enter a value: ")

message = 1
```

```
print("We start here.")


def message():
    print("Enter a value: ")

message()

print("We end here.")
```

```
def message():
    print("Enter a value: ")

message()
a = int(input())
message()
b = int(input())
message()
c = int(input())
```

Key takeaways

```
def your_function(optional parameters):
    # the body of the function
```


1. A function is a block of code that performs a specific task when the function is called. You can use functions to make your code reusable, better organized, and more readable. Functions can have parameters and return values.

2. There are at least four basic types of functions in Python:

- **built-in** functions which are an integral part of Python (such as the print() function). You can see a complete list of Python built-in functions at https://docs.python.org/3/library/functions.html
- the ones that come from **pre-installed** modules
- **user-defined** functions which are written by users for users - you can write your own functions and use them freely in your code,
- the **lambda** functions (you'll learn about them in next time)

3. You can define your own function using the **def** keyword and the following syntax:

The `input()` function is an example of a:

a) user-defined function
b) built-in function

```
hi()

def hi():
    print("hi!")
```

```
def hi():
    print("hi")

hi(5)
```

Examples

```
def hello(name):    # defining a function
    print("Hello,", name)    # body of the function


name = input("Enter your name: ")

hello(name)    # calling the function
```

```
def message():    # defining a function
    print("Hello")    # body of the function

message()    # calling the function
```

```
def function(parameter):
    ###
```

Parameterized functions

id(name)

- **parameters exist only inside functions** in which they have been defined, and the only place where the parameter can be defined is a space between a pair of parentheses in the def statement;
- assigning a value to the parameter is done at the time of the function's invocation, by specifying the corresponding argument.
- **parameters live inside functions** (this is their natural environment)
- **arguments exist outside functions**, and are carriers of values passed to corresponding parameters.

```
def message(number):
    print("Enter a number:", number)
```

```
1 def message(number):
2     print("Enter a number:", number)
3
4 message()
```

```
Traceback (most recent call last):
  File "main.py", line 4, in <module>
    message()
TypeError: message() missing 1 required positional argument: 'number'
```

Parametrized functions: continued

```
def message(number):
    print("Enter a number:", number)

message(1)
```

```
Console >_

Enter a number: 1
```

```
1 def message(number):
2     print("Enter a number:", number)
3
4 number = 1234
5 message(1)
6 print(number)
```

```
Console >_

Enter a number: 1
1234
```

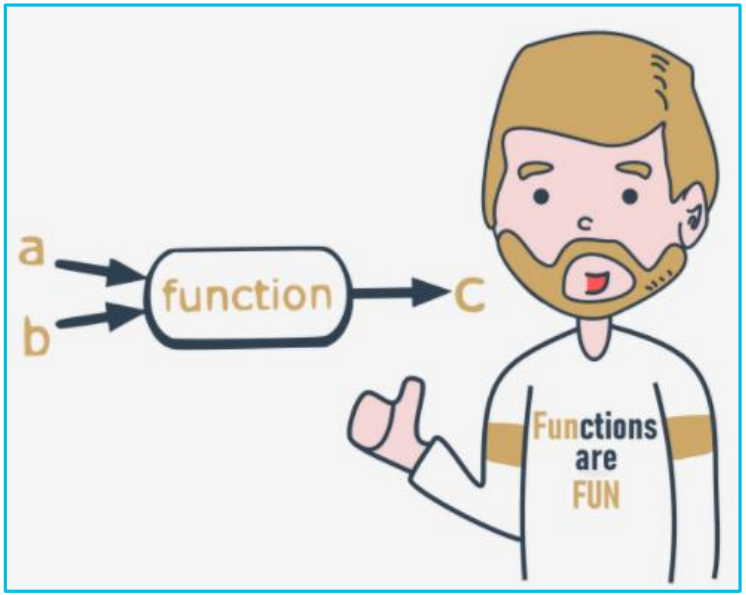
Console >_

```
Enter NANANA number 333
Enter telephone number 11
Enter price number 5
Enter number number number
```

```python
def message(what, number):
    print("Enter", what, "number", number)

# invoke the function
a = "NANANA"
message(a, "333")

message("telephone", 11)
message("price", 5)
message("number", "number")
```



```python
def my_function(a, b, c):
    print(a, b, c)

my_function(1, 2, 3)

def introduction(first_name, last_name):
    print("Hello, my name is", first_name, last_name)

introduction("Luke", "Skywalker")
introduction("Jesse", "Quick")
introduction("Clark", "Kent")

def introduction(first_name, last_name):
    print("Hello, my name is", first_name, last_name)

introduction("Skywalker", "Luke")
introduction("Quick", "Jesse")
introduction("Kent", "Clark")
```

Console >_

```
1 2 3
Hello, my name is Luke Skywalker
Hello, my name is Jesse Quick
Hello, my name is Clark Kent
Hello, my name is Skywalker Luke
Hello, my name is Quick Jesse
Hello, my name is Kent Clark
```

```python
def introduction(first_name, last_name):
    print("Hello, my name is", first_name, last_name)

introduction(first_name = "James", last_name = "Bond")
introduction(last_name = "Skywalker", first_name = "Luke")
```

Console >_

```
Hello, my name is James Bond
Hello, my name is Luke Skywalker
```

Keyword argument passing

```python
def introduction(first_name, last_name):
    print("Hello, my name is", first_name, last_name)

introduction(surname="Skywalker", first_name="Luke")
```

Console >_

```
Traceback (most recent call last):
  File "main.py", line 4, in <module>
    introduction(surname="Skywalker", first_name="Luke")
TypeError: introduction() got an unexpected keyword argument 'surname'
```

Mixing positional and keyword passing

b = 2, 4, a = 1

```
1  def adding(a, b, c):
2      print(a, "+", b, "+", c, "=", a + b + c)
3
4  # Call the adding function here.
5  adding(1, 2, 3)
6
7  adding(c = 1, a = 2, b = 3)
8
9  adding(4, 3, c = 2)
10
11 adding(3, c = 1, b = 2)
12
13 adding(3, a = 1, b = 2)
```

Console >_

```
1 + 2 + 3 = 6
2 + 3 + 1 = 6
4 + 3 + 2 = 9
3 + 2 + 1 = 6
Traceback (most recent call last):
  File "main.py", line 13, in <module>
    adding(3, a = 1, b = 2)
TypeError: adding() got multiple values for argument 'a'
```

Parame function details

```
1  def introduction(first_name, last_name="Smith"):
2      print("Hello, my name is", first_name, last_name)
3
4  # Call the function here.
5  introduction("James", "Doe")
6
7  introduction("Henry")
8  introduction(first_name="William")
9
10 def introduction1(first_name="John", last_name="Smith"):
11     print("Hello, my name is", first_name, last_name)
12
13 introduction1()
14 introduction1(last_name="Hopkins")
```

Console >_

```
Hello, my name is James Doe
Hello, my name is Henry Smith
Hello, my name is William Smith
Hello, my name is John Smith
Hello, my name is John Hopkins
```

```
def hi(name):
    print("Hi,", name)

hi("Greg")
```

```
def hi_all(name_1, name_2):
    print("Hi,", name_2)
    print("Hi,", name_1)

hi_all("Sebastian", "Konrad")
```

Key takeaways

```
def address(street, city, postal_code):
    print("Your address is:", street, "St.,", city, postal_code)

s = input("Street: ")
p_c = input("Postal Code: ")
c = input("City: ")

address(s, c, p_c)
```

```
Ex. 1
def subtra(a, b):
    print(a - b)

subtra(5, 2)
subtra(2, 5)


Ex. 2
def subtra(a, b):
    print(a - b)

subtra(a=5, b=2)
subtra(b=2, a=5)

Ex. 3
def subtra(a, b):
    print(a - b)

subtra(5, b=2)
subtra(5, 2)
```

```
def name(first_name, last_name="Smith"):
    print(first_name, last_name)

name("Andy")
name("Betty", "Johnson")
```

Examples

```
def intro(a, b="Bond"):
    print("My name is", b + ".", a + ".")

intro("Susan")
```

```
def intro(a="James Bond", b="Bond"):
    print("My name is", b + ".", a + ".")

intro(b="Sean Connery")
```

```
def intro(a="James Bond", b="Bond"):
    print("My name is", b + ".", a + ".")

intro()
```

```
def add_numbers(a, b=2, c):
    print(a + b + c)

add_numbers(a=1, c=3)
```

Home work 5.0

```
###############
#Author:
#Date:
#Task: Своим языком
###############

# Для того-то.
def plus(a=1, b=1):
        lsvlkmvkd
        print(a, "+", b, "=", a+b)

def minus(a=1, b=1):
```


Implement the calculator using your own functions.

while, exit by put "exit"

Set of operations:

+, -, /, *, %, //, **, odd/even, type(), max/min, avg

use: def, float(), input() and all that you want.

Example:

```
def plus(a, b):
    print("Plus: ", a, "+", b, "=",  a + b)
```

ЗАДАНИЯ

1) Прорешать всю классную работу
2) Выполнить все домашние задания

Почитать:

1) Byte of Python
стр. 64-75 - там даже больше чем мы сегодня прошли, до всего дойдем по порядку не переживайте

Крайний срок сдачи DD/MM в 21:00 (можно раньше, но не позже)

ЗАДАНИЯ

Название файлов, которые вы отправляете мне в telegram:

Vasia_Pupkin_class_work_L5_P0.py

+все задания ОДНИМ ФАЙЛОМ - Vasia_Pupkin_L5_P0.py

Формат сообщения которое вы присылаете мне

(после полного выполнения домашнего задания, только один раз) в Telegram:

Добрый день/вечер. Я Вася Пупкин, и это мои домашние задания к лекции 5 часть 0 про функции.

И отправляете файлы

Крайний срок сдачи DD/MM в 21:00 (можно раньше, но не позже)

https://docs.github.com/articles/using-pull-requests

Create your possibilities. Bye bye.