Bootcamp 134 | Python

Course 03 | Basic Python

Amir Hossein Chegouniyan

Head of the Technical Team at Dariche Tejarat

Lecturer of Python – Django at Maktab Sharif

Linkedin: Amirhossein-chegounian

Content

Function | Define

```
def  <function_name>(arg1, arg2, …):
        <function_body>        # use a tab (4 space) for function body
        return …….             # every function can return every value that you want or return None
```

- Arg1, arg2, …. is optional. You can define a function without every argument.

Function | Parameters

- You can send some parameters to your function. For example:

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

- You can don’t send every parameter to your function. For example:

```
def print_hello():
    print(“Hello World”)
```

Function | Parameters | Optional argument

- You can define a function with optional parameter. For example:

```
def hello(name="student"):
        print(f"Hello {name}")
```

- In this case, if you send a name to function:

```
hello("Amir")    # print => Hello Amir
```

- If don't send every data:

```
hello()          # print => Hello student
```

Function | Return statement

- If you need to output of function, can use from **return statement.**
- Return statement help you to store output of a function to a variable and use from it.
- Return statement is optional. If you don’t use from return, function return None by default.
- Exmaple:

```
def sum(a, b):
    return a + b
a_plus_b = sum(10, 30)
print(a_plus_b) # print => 40
```

Function | Keyword arguments

- *args

```
def test(*args):
    for x in args:
        print(x)
```

- **kwargs

```
def test(**kwargs):
    for index, value in kwargs.items():
        print(f“Value of {index} of input is {value}”)
```

Module | Intro

- Modules in python are just python files with a .py extention
- The name of module is the same the file name
- A python module can have a set of functions, classes, vars

Module | Import

- Default import (import a part of module in your working file):

  from <module_name> import <requested_part>

- Import all part (all functions, classes and vars of module in your working file):

  *from <module_name> import **

- Import a part of module in your working file with new name:

  from <module_name> import <requested_part> as <new_name>

Module | Load path

- PYTHONPATH
  - *nano ~/.bashrc*
  - *export PYTHONPATH = <address>*
  - *echo $PYTHONPATH*
- sys.path.append(address)
  - Execute if before running the impor in python file

Module | Exploring Built-in modules

- Use from help function:

  help('modules')

  help(math)

Module | Writing Packages

- from foo import bar
- import foo.bar

Pip and Pypi

- Is a package manager for python packages
- Package is contains all the files you need for a module
- Check if pip is installed:

  pytthon –m pip –version
- Create new env:

  Python3 –m venv <name_of_env> # for linux

  virtualenv <name_of_env> # for window
- Active env:

  source <address_of_env>/bin/activate # for linux

  <address_of_env>\Scripts\activate # for window

Pip and Pypi | Install library

- For install apps, see pypi site, search your app and install it.
- Install apps:

  python3 -m pip install "SomeProject“

  python3 -m pip install "SomeProject==1.4“

  python3 -m pip install "SomeProject>=1,<2“

  python3 -m pip install "SomeProject~=1.4.2“

- Deactive env:

  deactivate

Any question?

Next course