Bootcamp 134 | Python

Course 04 | Basic Python

Amir Hossein Chegouniyan

Head of the Technical Team at Dariche Tejarat

Lecturer of Python – Django at Maktab Sharif

Linkedin: Amirhossein-chegounian

Content

Exceptions

- Why exceptions accure?
- Why we handle exceptions?
- Why use python assert statement? (Debugging, Documentation, Testing, Security)

Exceptions | Basic Structure

- We can use from try, except:

```
try:
        <Handled Codes>
except:
        <Exception Codes>
```

Exceptions | Advance Structure

- We can use from try, except:

```
try:
        <Handled Codes>
except:
        print(“Wrong”)
else:
        print(“Correct)
finally:
        print(“Both”)
```

Exceptions | Types

- NameError
- KeyError
- IndexError
- TypeError
- ZeroDivisionError
- …

Exceptions | Raised Error

- You can use from “raise” to raise an exception (in of a condition)
  - *raise Exception(“<message>”)*   *# only print a message*
  - *raise TypeError(“<message>”)*   *# define what kind of error to raise*
- For example:

```
if type(x) != int:
        raise TypeError(“Your input is not integer”)
```

Exceptions | Assert

- You can use from "assert" to raise an exception (in of a condition)

  assert <condition>, <message>   # only print a message

- For example:

  assert type(x) != int, "Your input is not integer"

- Execute when condition return <u>False</u>
-

Files | Intro

- By default, application data is not saved.
- We need to store some data.
- This is where we use files.

Files | Open

f = open(<address>, <mode>)

- Mode ([r|a|w|x][t|b]):
  - r: Read | Opens a file for reading | Return error if does not exist (is default)
  - a: Append | Opens a file for appending | Create if does not exist
  - w: Write | Opens a file for writing | Return error if does not exist
  - x: Create | Create a file specified file | Return error if file exist
  - t: Text mode (is default)
  - b: Binary mode
- For example: rt, rb, at, ab, wt, wb, xt, xb

Files | Read

- *Open file with r mode*
- *f.read()   # read all file*
- *f.readline(limit=<number>) # read a line from file (limit by bit)*
- *f.readlines(limit=<number>) # read all file, return a list containg each line in the file (limit by line)*

Files | Write

- *Open file with a or w mode*
- *f.write(<content>) # write content on file*
- *f.close()   # close file to store data in file*
- *f.open(<address>)*
- *f.read()*

Files | Delete

- *import os*
- *os.remove(<address>) # return error if does not exist*
- *Check file is exist:*

```
 if os.path.exists(<address>):
     os.remove(<address>)
```

Files | With statement

```
With open(<address>) as <new_name>:
        print(<content>, file=new_name)
```

- *With statement will automatically close the file*

Data Encoding | String

- *my_string.encode(encoding="ascii",errors="replace")*
- *my_string.encode(encoding="utf-8",errors="replace")*

Data Encoding | Number

- *oct(my_decimal_number)*
- *hex(my_decimal_number)*
- *bin(my_decimal_number)*
- *int(my_number)*

Any question?

Next course

- None type
- Dictionaries
- Tuple
- List
- Comprehension
- Set