Python and Datascience Workshop

Author: Mohammad Akradi[1]

[1] Institute of Medical Science and Technology, Shahid Beheshti University, Tehran, Iran

Session 1

In this workshop we will learn basics of python, working with tabular data, data visualization, simple statistics and neuroimaging in python.

to install python, go to Python website and download the version compatible with your operating system. it is recommended to use python on Linux OS for this workshop as we will work with neuroimaging data.

---

1. Data structures in Python

- Integers and Floats

```
a = 2 #integer
pi = 3.14 #float
summation = a+pi
print(summation)
print(float(a))
print(int(pi))
```

- Strings

```
char = "a"
name = "Ali"
name_new = name + char + "r3"
print(name_new)
```

- Boolean and None

```
condition1, condition2 = True, False
none = None
print(int(condition1))
print(int(condition2))
```

- Lists

```
lst = [1, 2, "3", "character", 5, [6, 7, 8], 9, True, None]

# lists are mutable
lst[2] = 3
print(lst[2])
```

- Dictionaries

```
phonebook = {"Ali": 7475,
             "Fateme": 6218,
             "Reza": 1395}
print(phonebook["Ali"])
print(phonebook.keys())

# dictionaries are mutable
phonebook["Ali"] = 4235
print(phonebook)
```

- Tuples

```
tup1 = (1, "2", [3, 4])
print(tup1[2])

# tuples are immutable
tup1[1] = 2 # so this causes an error: 'tuple' object does not support
item assignment
```

there are lots examples and notes we should know about data structure, but for now, we will be content with that.

---

2. Conditional Code

somtimes we want to run one or more commands under certain circumstances. so, we use conditional code to this approach.
in conditional code, we use `if` with a following `condition`, which means that the specified commands will run if the `statement` arguman is `True`.

- Example #1:

```python
statement = 1
if statement == 1:
    print("the condition is true")
```

we can do something else if the statement is not True. to this approach we should use `else` like example below:

- Example #2:

```python
statement = -1
if statement >= 0:
    print("statement value is positive")
else:
    print("statement value is not positive")
```

we can search for other conditions with `elif`, if the first condition is not True.

- Example #3:

```python
statement = "string"
if isinstance(statement, str) == True:
    print("Statement type is string")
elif isinstance(statement, int) == True:
    print("Statement type is integer")
elif isinstance(statement, float) == True:
    print("Statement type is float")
else:
    print("Statement type is neither numeric nor string")
```

also, we can use logical `and`, `or` in our statement to check if two statements happen at once "(and)" or at least one of them "(or)". to this approach we can use both `and`/`&` for and statement, `or`/`|` for or statement.

- Example #4:

```python
exp1 = 5
exp2 = "apple"
if (statement1==5) and (statement2=="orange"):
    print("exp1 is 5 and exp2 is orange")
elif (statement1==5) & (statement2=="apple"):
    print("exp1 is 5 and exp2 is apple")

if (statement1==5) or (statement2=="orange"):
    print("At least one of the terms exp1 or exp2 is True")
```

```python
else:
    print("None of the expressions exp1 or exp2 are True")
```

---

3. Loops and Iterations

more often, specially in data analysis, we want to repeat one or more commands in our code, that's when we should use a loop to repeat our code for different inputs.
to this approach we use `for` and start iterating in data.

- Example #1:

```python
for i in range(10):
    print(i)
```

we can iterate in a prepared list or array:

- Example #2:

```python
lst = [1, 2, "Ali", 4, 5, [6, 7]]
for item in lst:
    print(lst)
```

---

4. Functions

some commands should be repeated in a code several times. but it doesn't look good to repeat a block of code, several time in a single code. so, we define a function which includes this block of code, and whenever we need to run these commands, we call the defined function. we already use lots of python built-in functions; e.g. `print`.

for defining a function we need a `name` for the function, `input` and `output` if it's needed. so, a function could have or not `input` and `output`.

- Example #1:

```python
def display():
    print("this a display function")
```

- Example #2:

```
def func_name(input1, input2):
    output = input1*input2
    print(output)
```

- Example #3:

```
def func_name():
    output = 2
    return output
```

- Example #4:

```
def func_name(input1, input2):
    output = input1*input2
    return output
```

---

5. Files

before getting through opening and processing files in python, let's have a look at the `input` function in python.

- Example #1:

```
dat1 = input("Enter the input value")
print(dat1)
```

this is the simplest way to get an input in python. for opening a file in python we use `open` function. this function takes to parameters, *filename* and *mode*. there are four different modes for opening a file:

> "r" - Read- Default value. opens a file for reading, error if the file does not exist
>
> "a" -Append- opens a file for appending, creates the file if it does not exist
>
> "w" -Write- opens a file for writing, creates the file if it does not exist
>
> "x" -Create- creates the specified file, returns an error if the file exist

> "r+" -Read and Write mode

first of all, we create a txt file and add some lines to that:

- Example #2:

```
f = open("file.txt", "w")
f.write("# this is a txt file created by python code \n")
f.write("# Date: 17 March 2022 \n \n")
f.write("Created by Mohammad")
f.close()
```

- Example #3:

```
f = open("file.txt")
for line in f:
    print(line)
f.close()
```

if you open a file in write mode, it will clear all context of file. so if you want to add some new lines to your txt file, you should open it in `append` mode.

- Example #4:

```
f = open("file.txt", "a")
f.write("\n This line is appended to the main txt file")
f.close()
```

it is better to use `with` method to open a file. so, any files opened will be closed automatically after one is done, so auto-cleanup

- Example #5:

```
with open("file.txt") as f:
    for line in f:
        print(line)
```

We can also split lines using file handling in Python. This splits the variable when space is encountered. You can also split using any characters as we wish. Here is the code:

- Example #6:

```python
with open("file.txt", "r") as f:
    data = f.readlines()
    for line in data:
        word = line.split()
        print (word)
```

Check if a file exists

if you want to check if a file exist in a directory, you can use `os.path.exists` command:

- Example #7:

```python
import os
os.path.exists("file.txt")
```

Removing a file

for removing files in your device, you can use `os.remove` command:

- Example #8:

```python
import os
os.remove("file.txt")
```

Creating a directory

- Example #9:

```python
import os
os.makedirs("test_dir", exist_ok=True)
```

copy and move files into another directory

- Example #10:

```python
import shutil
shutil.copy("file.txt", "test_dir/copy.txt") ## make a copy of file.txt in test_dir directory
shutil.move("file.txt", "test_dir/move_file.txt") ## moves file.txt into test_dir directory
```

remove a directory

- Example #11:

```
import os
os.rmdir("test_dir") ## Removes test_dir folder if it is empty

import shutil
shutil.rmtree("test_dir") ## Removes test_dir folder whether it is empty
or not
```

---

6. Error Handling

Sometimes we do not know if a command is wrong or not and it does not really matter to us. We may want to run a for loop that takes time and we do not want to stop in the event of an error. so we use `try` and `except` method in this case.

- Example #1:

```
a = "2"
b = 3
try:
    print(a+b)
except:
    print("an error occurred")
```

you can handle specific errors:

- Example #2:

```
while True:
    try:
        x = int(input("Please enter a number: "))
        if x == -1:
            break
    except ValueError:
        print("Oops!  That was no valid number.  Try again...")
```