Programming Mental Health Data in Python

Course Content

- Python Basics
- Python Data Structures
- Looping and Function in Python
- Importing Datasets
- Data Wrangling
- Exploratory Data Analysis
- Data Visualization Using Python
- Univariate Analysis
- Model evaluation and Refinement

3.0 Python Looping and Functions

Python has two primitive loop commands:

- while loops
- for loops

While Loops

With the while loop we can execute a set of statements as long as a condition is true.

```
import random

MIN = 1
MAX = 6

roll_again = 'y'

while roll_again == 'y':
    print('Rolling the dices...')
    print('The values are....')
    dice1 = random.randint(MIN, MAX)
    print(dice1)
    dice2 = random.randint(MIN, MAX)
    print(dice2)
    roll_again = input('Roll the dices again? (y / n): ')
```

Break and Continue statement with While Loop

With the break statement we can stop the loop even if the while condition is true:

With the continue statement we can stop the current iteration, and continue with the next:

```
[18]: # Break statement
i = 0
while i < 6:
  i += 1
  if i == 3:
    break
  print(i)

1
2

[17]: # Continue statement
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)

1
2
4
5
6

[ ]:
```

For Loops

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

```
9  Done

[28]: for i in range(0, 10):
    print(i, ' ', end=' ')
print()
print('Done')

0  1  2  3  4  5  6  7  8  9
Done
```

Loop through a list

```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

Loop through a Dictionary

```python
[31]: my_dict = {'name': 'Alice', 'age': 30, 'city': 'New York'}

for key in my_dict:
    print(key, ':', my_dict[key])
```

```
name : Alice
age : 30
city : New York
```

```python
[32]: # Using items() method for key-value pairs
for key, value in my_dict.items():
    print(key, ':', value)
```

```
name : Alice
age : 30
city : New York
```

Functions

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

Using def keyword to define a function

```python
def my_function ():
  print("Hello from a function")
```

call a function : myfunction()

Parameters and arguments

From a function's perspective:

A parameter is the variable listed inside the parentheses in the function definition.

An argument is the value that is sent to the function when it is called.

```
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

Passing arbitrary arguments

```
def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Emil", "Tobias", "Linus")
```

Other functions

A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

```
x = lambda a : a + 10
print(x(5))
```

Recursive function

Python also accepts function recursion, which means a defined function can call itself.

An example of recursion vs iteration

```
[33]: def factorial(n):
          # Base case: factorial of 0 or 1 is 1
          if n == 0 or n == 1:
              return 1
          # Recursive case: factorial of n is n times factorial of (n-1)
          else:
              return n * factorial(n - 1)

      # Example usage
      number = 5
      print("Factorial of", number, "is", factorial(number))

      Factorial of 5 is 120

[34]: number = 5
      result = 1

      # Calculate factorial using a for loop
      for i in range(1, number + 1):
          result *= i

      # Print the factorial
      print("Factorial of", number, "is", result)
```

Module

Consider a module to be the same as a code library. A file containing a set of functions you want to include in your application.

```
Mygreetingmodule.py

def your_age(dob):
  pass

def greeting(name):
  print("Hello, " + name)

person1 = {
  "name": "John",
  "age": 36,
  "country": "Norway"
}
```

import mygreetingmodule

mygreetingmodule.greeting("james")

from mygreetingmodule import person1

print(person1[“age”])