IntroductionToPython

May 9, 2019

1 Python Documentation and Tutorials

You can access the documentation and tutrials at https://www.python.org/ Use Python 3.x Documentation at https://docs.python.org/3/ If you are rusty in Python programming, go through this tutorial at https://docs.python.org/3/tutorial/index.html Here is an excellent resource to learn Python step by step; https://www.techbeamers.com/python-tutorial-step-by-step/

2 Simple Programs

Here is a sample program for implementing an Interest Calculator. By looking at it, you can feel how easy it is to write and understand the code in Python.

```
In [3]: # Good code is self-documenting. (this is a single line comment)

        print('Interest Calculator:')

        amount = float(input('Principal amount ?'))
        roi = float(input('Rate of Interest ?'))
        yrs = int(input('Duration (no. of years) ?'))

        total = (amount * pow(1 + (roi/100), yrs))
        interest = total - amount
        print('\nInterest = %0.2f' %interest)
```

```
Interest Calculator:
Principal amount ?1000
Rate of Interest ?2
Duration (no. of years) ?3

Interest = 61.21
```

3 Functions, Comments, and DocStrings

```
In [6]: #functions are defined with def follwed by the function name
        def theFunction():
```

```
    '''
This function demonstrate the use of docstring in Python.
The strings beginning with triple quotes are still regular strings except the fact that
It means they are executable statements. if such a string is placed immediately after a
You can access them using the following special variable.
    '''
    print("Python docstrings are not comments.")

print("\nJust printing the docstring value...")
print(theFunction.__doc__) #this is the special variable
```

```
Just printing the docstring value...

This function demonstrate the use of docstring in Python.
The strings beginning with triple quotes are still regular strings except the fact that they cou
It means they are executable statements. if such a string is placed immediately after a function
You can access them using the following special variable.
```

```
In [7]: #a simple example where a function typeOfNum() has nested functions to decide on a numbe

        def typeOfNum(num): # Function header
            # Function body
            if num % 2 == 0:
                def message():
                    print("You entered an even number.")
            else:
                def message():
                    print("You entered an odd number.")
            message()
        # End of function

        typeOfNum(2)  # call the function
        typeOfNum(3)  # call the function again
```

```
You entered an even number.
You entered an odd number.
```

4 Data Types

If you noticed in the previous code block, we did not use data types. This is because, * Python is a dynamically typed language which means the types correlate with values, not with variables. Hence, the polymorphism runs unrestricted. * That's one of the primary differences between Python and other statically typed languages such as C++ or Java. * In Python, you don't have to mention the specific data types while coding. * However, if you do, then the code limits to the types anticipated at the time of coding. * Such code won't allow other compatible types that may require in the future. * Python doesn't support any form of function overloading.

*****You can learn more about Data types using 02.01-Understanding-Data-Types.ipynb

```
In [8]: # Demonstrate type binding
        def product(x, y) : return x * y
        print(product(4, 9)) # function returns 36
        print(product('Python!', 2))  # function returns
                                      # Python!Python!
        print(product('Python 2 or 3?', '3')) # TypeError occurs
```

```
36
Python!Python!
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-8-17e3f5213033> in <module>()
          3 print(product('Python!', 2))  # function returns
          4                               # Python!Python!
    ----> 5 print(product('Python 2 or 3?', '3')) # TypeError occurs


        <ipython-input-8-17e3f5213033> in product(x, y)
    ----> 1 def product(x, y) : return x * y
          2 print(product(4, 9)) # function returns 36
          3 print(product('Python!', 2))  # function returns
          4                               # Python!Python!
          5 print(product('Python 2 or 3?', '3')) # TypeError occurs


        TypeError: can't multiply sequence by non-int of type 'str'
```

5 Python List

A list in Python, also known as a sequence, is an ordered collection of objects. It can hold values of any data types such as numbers, letters, strings, and even the nested lists as well.

Every element rests at some position (i.e., index) in the list. The index can be used later to locate a particular item. The first index begins at zero, next is one, and so forth.

Unlike strings, lists are mutable (or changeable) in Python. It implies that you can replace, add or remove elements.

```
In [12]: theList = ['Python', 'C', 'C++', 'Java', 'CSharp'] # create a list
```

```
         theList[4] = 'Angular' # item replacement
         print(theList)

         theList[1:4] = ['Ruby', 'TypeScript', 'JavaScript']
         print(theList)

         print('\n Iterating through the list')
         #iterate through list with index
         for index, element in enumerate(theList):
          print(index, element)
```

```
['Python', 'C', 'C++', 'Java', 'Angular']
['Python', 'Ruby', 'TypeScript', 'JavaScript', 'Angular']

 Iterating through the list
0 Python
1 Ruby
2 TypeScript
3 JavaScript
4 Angular
```

```
In [26]: # If you would like to fetch the index for first matching item
         theList = ['a','e','i','o','u']

         def matchall(theList, value, pos=0):
             loc = pos - 1
             try:
                 loc = theList.index(value, loc+1)
                 yield loc
             except ValueError:
                 pass

         value = 'i'
         for loc in matchall(theList, value):
             print("match at", loc+1, "position.")
```

```
match at 3 position.
```

6 Sorting a List

Python list implements the sort() method for ordering (in both ascending and descending order) its elements in place.

```
In [28]: theList = ['a','e','i','o','u']
         theList.sort()
         print(theList)
```

```
        theList = ['gs', 'ge', 'gt', 'es', 'ee', 'et', 'ts', 'te', 'tt'] # reassignment of theL
        theList.sort()
        print(theList)
```

```
['a', 'e', 'i', 'o', 'u']
['ee', 'es', 'et', 'ge', 'gs', 'gt', 'te', 'ts', 'tt']
```

7 Object Orient Programming

- Python supports object-oriented programming (OOP)
- A class is an arrangement of variables and functions into a single logical entity. It works as a template for creating objects. Every object can use class variables and functions as its members.

**Some terms which you need to know while working with classes in Python.

1. The "class" keyword
2. The instance attributes
3. The class attributes
4. The "self" keyword
5. The "__init_" method

- With the class keyword, we can create a Python class
- The "**init**()" is a unique, it is the constructor, and it is called automatically
- Instance attributes are object-specific attributes defined as parameters to the **init** method.
- Unlike the instance attributes which are visible at object-level, the class attributes remain the same for all objects. (More like a static variable in Java class)
- Python provides the "self" keyword to represent the instance of a class. It works as a handle for accessing the class members such as attributes from the class methods. (More like 'this' in Java or C++)

**Here is an example

```
In [29]: class BookStore: # create a Python class
             noOfBooks = 0 # class attribute

             def __init__(self, title, author): #instance attributes are parameters to construct
                 self.title = title
                 self.author = author
                 BookStore.noOfBooks += 1

             def bookInfo(self):
                 print("Book title:", self.title)
                 print("Book author:", self.author,"\n")

         # Create a virtual book store
```

```
    b1 = BookStore("Great Expectations", "Charles Dickens") # individual object creation
    b2 = BookStore("War and Peace", "Leo Tolstoy")
    b3 = BookStore("Middlemarch", "George Eliot")

    # call member functions for each object
    b1.bookInfo()
    b2.bookInfo()
    b3.bookInfo()

    print("BookStore.noOfBooks:", BookStore.noOfBooks)
```

```
Book title: Great Expectations
Book author: Charles Dickens

Book title: War and Peace
Book author: Leo Tolstoy

Book title: Middlemarch
Book author: George Eliot

BookStore.noOfBooks: 3
```

8 File Handling

In Python, file processing takes place in the following order.

- Open a file which returns a file handle.
- Use the handle to perform read or write action.
- Close the file handle.

**Open a file file object = open(file_name [, access_mode][, buffering])

1. access_mode - It's an integer representing the file open mode e.g. read, write, append, etc. ** It opens a file in read-only mode while the file offset stays at the root. ** It allows write-level access to a file. If the file already exists, then it'll get overwritten. It'll create a new file if the same doesn't exist. ** It opens the file in append mode. The offset goes to the end of the file. If the file doesn't exist, then it gets created. ** It opens the file in both (read + write) modes while the file offset is again at the root level. ** It opens a file in both (read + write) modes. Same behavior as for write-only mode. ** It opens a file in (append + read) modes. Same behavior as for append mode.

2. buffering - The default value is 0 which means buffering won't happen. If the value is 1, then line buffering will take place while accessing the file.

3. file_name - It's a string representing the name of the file you want to access.

Python File Encoding In Python 3.x, there is a clear difference between strings (text) and a byte (8-bits). It states that the char 'a' doesn't represent the ASCII value 97 until you specify it like that. So, while you want to use a file in text mode, then better you mention the correct encoding type.

**Example f = open('app.log', mode = 'r', encoding = 'utf-8')

- Encoding for Windows and for Linux/Unix is standard by default for text

```
In [33]: try: # use a try to avoid exceptions
            f = open("Berkeley.csv", mode = 'r', encoding = 'utf-8')
            # do file operations.

         finally:
            f.close()
```

***Another way to close a file is by using the WITH clause. It ensures that the file gets closed when the block inside the WITH clause executes. The beauty of this method is that it doesn't require to call the close() method explicitly.

```
In [36]: with open('app.log', 'w', encoding = 'utf-8') as f: # write to a file name app.log
            #first line
            f.write('my first file\n')
            #second line
            f.write('This file\n')
            #third line
            f.write('contains three lines\n')

         with open('app.log', 'r', encoding = 'utf-8') as f:
            content = f.readlines() # read the content

         for line in content:
            print(line) # output the content read from file
```

```
my first file

This file

contains three lines
```

```
In [37]: with open('app.log', 'w', encoding = 'utf-8') as f:
            #first line
            f.write('It is my first file\n')
            #second line
            f.write('This file\n')
            #third line
            f.write('contains three lines\n')

         #Open a file
         f = open('app.log', 'r+')
         data = f.read(19);
         print('Read String is : ', data)
```

```
    #Check current position
    position = f.tell();
    print('Current file position : ', position)

    #Reposition pointer at the beginning once again
    position = f.seek(0, 0);
    data = f.read(19);
    print('Again read String is : ', data)

    #Close the opened file
    f.close()
```

```
Read String is :  It is my first file
Current file position :  19
Again read String is :  It is my first file
```

Write a program to read Berkeley.csv file and output only the admitted female frequencey of all departments.