**Technical Blog on Python**

**1. What is python?**

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

Example : print("Hello, World!")

1. **Variables in Python ?**

Variables are the example of identifiers. An Identifier is used to identify the literals used in the program. The rules to name an identifier are given below.

* The first character of the variable must be an alphabet or underscore ( \_ ).
* All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore, or digit (0-9).
* Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, \*).
* Identifier name must not be similar to any keyword defined in the language.
* Identifier names are case sensitive; for example, my name, and MyName is not the same.
* Examples of valid identifiers: a123, \_n, n\_9, etc.
* Examples of invalid identifiers: 1a, n%4, n 9, etc.

1. **Python Data Types ?**

Variables can store data of different types, and different types can do different things.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Integers | int | 1,2 |
| Floating point | float | 2.3,3.3 |
| Strings | str | “hello” |
| Lists | list | [10,”hi”, 22.4] |
| Dictionaries | dict | {“mom” ,”son”} |
| Tuples | tup | (10,”hiii”) |
| Sets | set | {“a”,”b”} |
| Booleans | bool | True or False |

**String:** Strings in python are surrounded by either single quotation marks, or double quotation marks.'hello' is the same as "hello".

1. Index :[]
2. Slicing : [2: ]
3. Upper(): string\_name .upper()
4. Lower (): string\_name .Lower()
5. Split(): string\_name.split()

**Lists:** Lists are used to store multiple items in a single variable.

1. Append() :list.append()
2. Remove():list.remove()
3. Pop():list.pop()
4. Sorting():list.sort()
5. Reverse():list.reverse()

**Dictionaries:**Dictionaries are used to store data values in key:value pairs.

1. Dict.values()
2. Dict.items()
3. Dict.remove()
4. Mydict = dict.copy()

**Tuples:** Tuples are used to store multiple items in a single variable.

1. Update tuples: Once a tuple is created, you cannot change its values. Tuples are **unchangeable**, or **immutable** as it also is called.
2. Indexing is possible for both end in positive or negative.

**Sets:** Sets are used to store multiple items in a single variable.

Example: my\_set = {"apple", "banana", "cherry"}

1. My\_set: Set.add()
2. My\_set: Set.remove()
3. Join two sets using union() : set3 = set1.union(set2)

**Booleans :**

Booleans represent one of two values: True or False.

**Python statements**

**Python if-else :**

Python supports the usual logical conditions from mathematics:

* Equals: a == b
* Not Equals: a != b
* Less than: a < b
* Less than or equal to: a <= b
* Greater than: a > b
* Greater than or equal to: a >= b
* If , elif , else

Example: a = 200  
 b = 33  
 if b > a:  
 print("b is greater than a")  
 elif a == b:  
 print("a and b are equal")  
 else:  
 print("a is greater than b")

**Python while loops:**

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

Example: i = 1  
 while i < 6:  
 print(i)  
 i += 1

**Python 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).

Example: fruits = ["apple", "banana", "cherry"]  
 for x in fruits:  
 print(x)

**Object Oriented Programming**

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

**Class:** A class is a collection of objects. A class contains the blueprints or the prototype from which the objects are being created. It is a logical entity that contains some attributes and methods.

Some points on Python class:

1. Classes are created by keyword class.
2. Attributes are the variables that belong to a class.
3. Attributes are always public and can be accessed using the dot (.)

Example: class MyClass:  
 x = 5

**Object**: Objects can also contain methods. Methods in objects are functions that belong to the object.

Example: p1.my\_func()

**Inheritance:**Inheritance allows us to define a class that inherits all the methods and properties from another class.

**Parent class** is the class being inherited from, also called base class.

**Child class** is the class that inherits from another class, also called derived class.

**Polymorphism:**

Polymorphism simply means having many forms.

**Functions**

**Function:** 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.

Example : def my\_function():  
 print("Hello!!")

**Lambda function:**

1. A lambda function is a small anonymous function.
2. A lambda function can take any number of arguments, but can only have one expression.

Syntax : lambda *arguments* : *expression*

*Example:* x = lambda a : a + 10  
 print(x(5))

**Python Iterator:**

1. An iterator is an object that contains a countable number of values.
2. An iterator is an object that can be iterated upon, meaning that you can traverse through all the values.
3. Technically, in Python, an iterator is an object which implements the iterator protocol, which consist of the methods \_\_iter\_\_() and \_\_next\_\_().

Example: my\_tuple = ("apple", "banana", "cherry")  
 my\_it = iter(my\_tuple)  
 print(next(my\_it))  
 print(next(my\_it))  
 print(next(my\_it))

**Generator Functio**n: A generator is a special type of function which does not return a single value, instead, it returns an iterator object with a sequence of values. In a generator function, a yield statement is used rather than a return statement. The following is a simple generator function.

Syntax: def generator():

**Regular Expression:**

1. A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.
2. RegEx can be used to check if a string contains the specified search pattern.

1. import re(built-in package called re)

## **RegEx Functions**

The re module offers a set of functions that allows us to search a string for a match:

|  |  |
| --- | --- |
| **Function** | **Description** |
| Findall | Returns a list containing all matches |
| search | Returns a [Match object](https://inc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?new=1&ui=en%2DUS&rs=en%2DUS&wdorigin=OFFICECOM-WEB.MAIN.NEW-INSTANT&wdenableroaming=1&mscc=1&wdodb=1&hid=D7E829A0-C0EA-0000-F48C-B93CA3D3BD9A&wopisrc=https%3A%2F%2Fzehntechtechnologies-my.sharepoint.com%2Fpersonal%2Froopali_verma-int_zehntech_com%2F_vti_bin%2Fwopi.ashx%2Ffiles%2F86994da0627f469c89b2eb4e771c25e4&wdhostclicktime=1647327618867&jsapi=1&jsapiver=v1&newsession=1&corrid=d7fbc366-fdc6-4450-9d24-63c9b47203bd&usid=d7fbc366-fdc6-4450-9d24-63c9b47203bd&sftc=1&mtf=1&sfp=1&wdredirectionreason=Unified_SingleFlush&rct=Medium&ctp=LeastProtected#matchobject) if there is a match anywhere in the string |
| split | Returns a list where the string has been split at each match |
| sub | Replaces one or many matches with a string |

**Python File Handling**

## **File Handling:**

1. The key function for working with files in Python is the open() function.
2. The open() function takes two parameters; filename, and mode.
3. There are four different methods (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 exists

1. In addition you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

Syntax: f = open("demofile.txt")

**Read Files :** The open() function returns a file object, which has a read() method for reading the content of the file:

Example: f = open("demofile.txt", "r")  
 print(f.read())

**Write Files:** To write to an existing file, you must add a parameter to the open() function:

Example : f = open("demofile2.txt", "a")  
 f.write("Now the file has more content!")  
 f.close()  
  
 #open and read the file after the appending:  
 f = open("demofile2.txt", "r")  
 print(f.read())

**Delete Files:** To delete a file, you must import the OS module, and run its os.remove() function:

Example : import os  
 os.remove("demofile.txt")

1. Delete folder : To delete an entire folder, use the os.rmdir() method.

**Error and Exception Handling**

**Try and exception :** The code which can cause an exception to occur is put in the try block and the handling of the exception is then implemented in the except block of code. The syntax follows:

**try:**

You do your operations here...

...

**except** :

If there is Exception, then execute this block.

**Finally :** The finally: block of code will always be run regardless if there was an exception in the try code block. The syntax is:

**try:**

Code block here

...

Due to any exception, this code may be skipped!

**finally:**

This code block would always be executed.

# **Web Scraping**

1. Always be respectful and try to get premission to scrape, do not bombard a website with scraping requests, otherwise your IP address may be blocked!
2. Be aware that websites change often, meaning your code could go from working to totally broken from one day to the next.
3. Pretty much every web scraping project of interest is a unique and custom job, so try your best to generalize the skills learned here.

## l Basic components of a WebSite

1. HTML:
2. CSS

## **Web Scraping with Python:** y**ou can use pip install, for example:**

1. pip install requests
2. pip install lxml
3. pip install bs4

**Working with pdf’s and speradsheet:**

# **Working with CSV Files:**

CSV stands for “Comma Separated Values.” It is the simplest form of storing data in tabular form as plain text. It is important to know to work with CSV because we mostly rely on CSV data in our day-to-day lives as data scientists.

**Reading CSV files :** import CSV

data **=** open('example.csv')

data

## **Writing to CSV Files:**

We can also write csv files, either new ones or add on to existing ones.

file **=** open('to\_save\_file.csv','w',newline**=**'')

**Exist file :**

f **=** open('to\_save\_file.csv','a',newline**=**'')

**Emails with python**

**Sending Email :**

Create an SMTP object for a server. Here are the main Server Domain Name for the top email services. If you don't see your email server here, you may need to do a quick Google Search to see if there SMTP server domain name is available:

**import** smtplib

**Received Email :**

Now that we understand how to send emails progammatically with Python, let's explore how we can read and search recieved emails. To do we will use the built-in [imaplib library](https://inc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?new=1&ui=en%2DUS&rs=en%2DUS&wdorigin=OFFICECOM-WEB.MAIN.NEW-INSTANT&wdenableroaming=1&mscc=1&wdodb=1&hid=D7E829A0-C0EA-0000-F48C-B93CA3D3BD9A&wopisrc=https%3A%2F%2Fzehntechtechnologies-my.sharepoint.com%2Fpersonal%2Froopali_verma-int_zehntech_com%2F_vti_bin%2Fwopi.ashx%2Ffiles%2F86994da0627f469c89b2eb4e771c25e4&wdhostclicktime=1647327618867&jsapi=1&jsapiver=v1&newsession=1&corrid=d7fbc366-fdc6-4450-9d24-63c9b47203bd&usid=d7fbc366-fdc6-4450-9d24-63c9b47203bd&sftc=1&mtf=1&sfp=1&wdredirectionreason=Unified_SingleFlush&rct=Medium&ctp=LeastProtected#imap4-example). We will also use the built in [email](https://docs.python.org/3/library/email.examples.html) library for parsing through the recieved emails.

**import** imaplib

**graphical user interfaces(GUI)**

Python has a huge number of GUI frameworks (or toolkits) available for it, from [TkInter](https://wiki.python.org/moin/TkInter) (traditionally bundled with Python, using Tk) to a number of other cross-platform solutions, as well as bindings to platform-specific (also known as "native") technologies.

**Using Interact:**

The interact function (ipywidgets.interact) automatically creates user interface (UI) controls for exploring code and data interactively. It is the easiest way to get started using IPython's widgets.

1. Fixing arguments using fixed
2. Widget abbreviations
3. Using function annotations with interact