Python Basics

Python is a dynamically typed language, meaning we don’t need to define the type of the variable. The type is inferred at runtime based on the value of the variable.

For numbers, we can or cannot use the quotes.

A black screen with white text

Description automatically generated

Similar to string and integers, we can define other types of variables such as decimals, boolean as well

A black background with colorful text

Description automatically generated

Python is a strongly typed language meaning it doesn’t implicitly convert one type to another, if the operation is not valid. Whereas in weakly typed language (e.g. JavaScript) implicit type conversion occurs. So, we need to convert the variable into required type and then print it.

A screenshot of a computer

Description automatically generated

We can do a lot of operations with Strings. Some of the common operations are as follows

A screenshot of a computer program

Description automatically generated

Similar to strings, we can play around numbers as well. Some of the common functions related to numbers are as follows

A screenshot of a computer program

Description automatically generated

We can take user inputs and print it out with the help of the input () function

A black background with white text

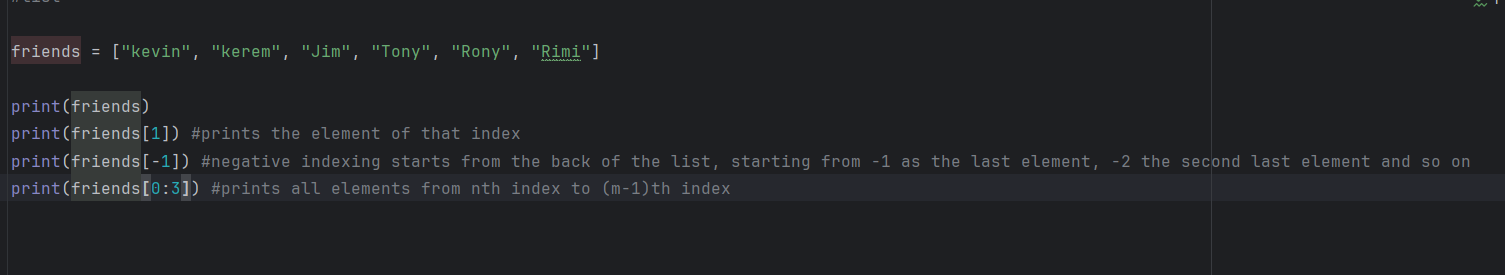
Description automatically generated

We can also write a Mad Libs game by taking user input like this –

A computer screen with text

Description automatically generated

**List**



In Python, inside a list we can store multiple variables of the same type, and also multiple variables of different types such as string, integer, float, etc. as well.

We can perform a bunch of operations with lists and some of them are listed as below -

A screen shot of a computer

Description automatically generated

**Tuples**

Tuples are data structures that are immutable. Once a tuple or a list of tuples are defined, the value cannot be changed. It can be accessed via indexing and index starts from 0, but none of the elements of a tuple can ever be changed.

A black background with numbers and symbols

Description automatically generated

**Functions**

A computer screen shot of colorful text

Description automatically generated

We can pass any type of values like string, integer, float, boolean, etc. inside the function and perform any desirable actions. Also functions work with indentation, so the lines of codes that we want to execute inside of the function have to be intended accordingly.

**Conditional statements – if, elif, else**

We can perform various operations using the conditional statements if, else if and else

A screenshot of a computer program

Description automatically generated

Using these conditional statements, we can build a little fun calculator

A screen shot of a computer program

Description automatically generated

**Dictionaries**

A dictionary in python is a data structure which stores (key: value) pairs. One of the characteristics is that every key inside a dictionary has to be unique, there cannot be any duplicates.

A screenshot of a computer program

Description automatically generated

We can keep the key in forms of strings, numbers, etc. Just we have to remember that the key has to be unique.

**While loop**

A screen shot of a computer program

Description automatically generated

All the statements written inside the indentation, i.e. within the while loop will be executed until the condition specified within the while loop is satisfied

We can combine such conditional statements and loops to create some fun game like a guessing game

A screen shot of a computer program

Description automatically generated

**For loop**

we can display a number of ways inside the for loop to perform the operations. Some of these are as below -

A screen shot of a computer program

Description automatically generated

Using functions and for loop we can perform multiple operations ranging from lower to higher complexity. One fine example is exponential function –

A computer screen shot of a program code

Description automatically generated

**2D lists** and accessing the elements through the nested loops

A screenshot of a computer program

Description automatically generated

Using all the for loops and the conditional statements, we can make something like a translator. In this example, all the vowels are converted to ‘g’ or ‘G’, depending on whether it’s an upper case or a lower-case vowel and the rest of the consonants are kept intact.

A screen shot of a computer program

Description automatically generated

**Comments**

Comments are lines of codes that will not be rendered by the programming language. Please refer to the above screenshots for the lines with # to identify comments. For a single line comment, we can use #, whereas for multiple lines of code we can use triple quotes ‘’’ <lines of comments in between>‘’’

A black background with white text

Description automatically generated

**Try Except**

There can be many unseen situations where the code can be broken due to faulty lines of codes, invalid user input and so on. To handle these situations, we can use Try...Except exception handling.

In the below example, we handled 2 different types of exceptions – Division by Zero and Value Error exception. These exceptions help to continue the normal flow of code, without breaking it. We can also catch the error (using ‘as’ keyword and some variable) and print the error in the console.

A screenshot of a computer program

Description automatically generated

**File Operations**

There are several file operations – opening a file, closing a file, reading from a file, writing to a file and a bunch of other operations.

Reading a file -

Below are some of the ways by which we can read a file and its contents. It also specifies the different mode of operations that we can perform on the file contents.

A screenshot of a computer program

Description automatically generated

Appending a file –

When we want to append something, we usually add the new contents to the end of the file. Also, we can’t change something already existing in the file, we can only add something to the end of the file. (note: “a” – open the file in append mode and not read mode)

A screenshot of a computer program

Description automatically generated

Writing to a file –

If we perform the write operation to an existing file, the previous content is completely overwritten by the new content provided. On the other hand, we can also create a new file with file name and then write some contents to the file.

A screenshot of a computer program

Description automatically generated

**Modules & pip**

Modules are basically a bunch of files that we can import in our current file to access and make use of the functions or methods that are written in that file. In this website <https://docs.python.org/3/py-modindex.html> , we can find all useful python modules and it is also described how to import and how to make use of the module.

There are basically 2 types of modules – built-in modules and external modules. For built-in modules, we can get access automatically, and we don’t need to import anything explicitly. On the other hand, we can find the external modules listed in external libraries –> Python –> Lib folder (these come up along with the installation of the python).

Apart from these external modules also, we can use other external 3rd party modules that other developers have scripted. For that, we need to find them as they are not pre-installed with python. One such module is *python-docx*. So, to install that we can make use of the command ‘pip install python-docx’ .(note: pip is a very powerful command. It’s basically referred to as a package manager that allows us to install, uninstall, update, manage external python modules).

For example,

A computer screen shot of a black screen

Description automatically generated

We can also uninstall the installed module –

A screen shot of a computer program

Description automatically generated

**Classes & Objects**

So, in the real world not everything can be represented by strings or number or boolean. So, we have to make use of something called classes and objects to represent different things. For example, a phone or a person can’t be described with strings or numbers or boolean, so we need user-defined datatypes and by using classes we can make use of these data types.

Below is the example of a student class and the main class that makes use of the student class and its attributes by creating its objects.

A computer screen shot of a program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Quiz game**

The objective is to try out a fun Quiz game in order to model a real-world entity. So, in this below example, we write 2 classes – 1 containing the main quiz game and another class (Question) to initialise the question prompts and answers to it.

Later on, we define a list ‘questions’ and passed the ‘question\_prompts’ list and the right answer to the function ‘\_\_init\_\_’ of the class ‘Questions’. So, here’s the workflow –

As soon as the method ‘run\_test’ is called, it enters the method and loops through each question and gets back with the correct answer, then compares that with the answer given by the user and then update the score.

So, when method ‘run\_test’ is called, it calls the list ‘questions’, which in turn passes the list of question prompts and answers to the method ‘\_\_init\_\_’ of the class ‘Question’ and then gets back the right answer to the right question. Then, the control flow returns to ‘run\_test’ where the validation is carried whether or not the user has given the right answer.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

**Class Functions**

So, we can define as many functions inside a class as we want and then later on call those functions from another class to structure and enhance readability. One such example is given below –

In this example, there are 2 functions that are defined in the class ‘Students’ and both of these functions are used by another class by importing the class. So once we import a class, we can make use of the functions defined in that class in several ways like to pass value of the parameters and initialise them, to validate something, etc.

A screenshot of a computer

Description automatically generated

A computer screen shot of text

Description automatically generated

**Inheritance**

Inheritance is an important of the concept of OOP, where one class can inherit the properties of another class. The class which inherits the properties is called child class or derived class or sub class. The class from which the properties are inherited is called Parent class or Super class.

In this below there are basically three class – let’s say the main class (app.py), the parent class (Chef.py) and the child class (ChineseChef.py).

So, when the ChineseChef inherits the Chef, it can make use of all of its methods, can override them if its required and also can define its additional methods as well.

So, when we create an object of the ChineseChef class in the ‘app.py’ class and call a method that’s initially defined in the Chef class, we can do that because the ChineseChef has already inherited all the methods of its parent class. Inheritance enhances the reusability of the code by a great extent.

The main class ‘app.py’

A computer screen shot of a program

Description automatically generated

The parent class ‘Chef.py’

A screen shot of a computer program

Description automatically generated

The child class ‘ChineseChef.py’

A computer screen shot of a computer program

Description automatically generated

**Python Interpreter**

The Python interpreter is basically a little environment that we can use to execute different python commands. So, its like a little sandbox environment where we can test out and try out different python commands or different Python functions in a very safe and neutral environment.

So, in windows systems, we have to add python file to our system’s path variable and then only system should be able to recognise the python interpreter.

We can perform a bunch of stuffs and can execute a lot of basic commands, can write any conditions statements or execute loops and can define functions also. But its definitely not recommended to script huge python codes here in this CLI, simply because an IDE provides a more structure, developer-friendly environments with many additional tools and features that we can make use of.

A black and white image of numbers and symbols

Description automatically generated

Functions defined in the terminal

A close up of text

Description automatically generated