**Part-2**

**Topics**

* Python strings and slicing.
* Python conditional statements.
* Python iterative statements.
* Python functions.
* Classes and objects in python.
* Python collections
* Lists
* Tuples
* Sets
* Dictionary

**Python strings & Slicing**

Strings in python are surrounded by either single quotation marks, or double quotation marks.



Output :

Hello

**String slicing:**

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

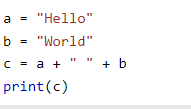


Output:

llo

**String concatenation:**

To concatenate, or combine, two strings you can use the + operator.



**Output**:

Hello World

Example 2:

str4 = "here is"

i=5

print(str4+" "+str(i)) #here we have to type cast because we cannot merge string with int using # + operator

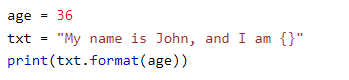
**Output:**

here is 5

**String format:**

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

**(Note: we cannot combine strings and numbers using ‘+’ operator)**



Output:

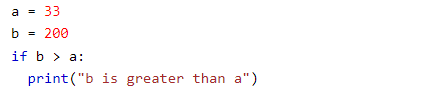
My name is John, and I am 36

## Python Conditions and If statements

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

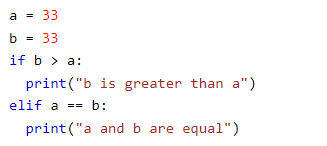
**If keyword:**



In this example we use two variables, a and b, which are used as part of the if statement to test whether b is greater than a. As a is 33, and b is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

**Elif:**

The elif keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

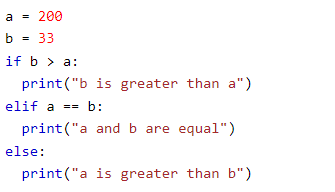


Output:

a and b are equal

**Else:**

The else keyword catches anything which isn't caught by the preceding conditions.



Output:

A is greater than b

**Python iterative statements**

**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 program:



**Output:**

b  
a  
n  
a  
n  
a

**The range() function:**

To loop through a set of code a specified number of times, we can use the range() function,

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Example program:



**Output:**

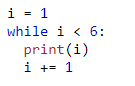
2

4

**Python while loop:**

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

Example program:



**Output:**

1

2

3

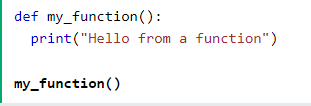
4

5

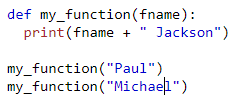
**Python Functions**

A function is a block of code which only runs when it is called (code reusability whenever needed).

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



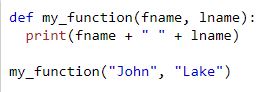
Information can be passed into functions as arguments.



Output:

Paul Jackson  
Michael Jackson

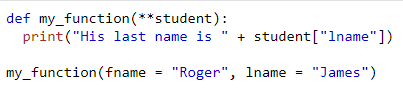
**Number of arguments:**



Output:

John Lake

**Arbitrary arguments(\*\*kwargs)**



Output:

His last name is James

**Class and Objects**

**Class:** A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made.

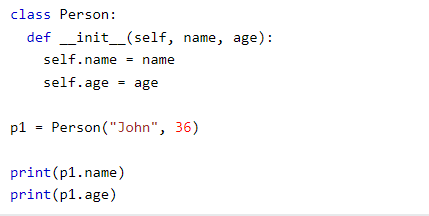
**Object:** An Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with actual values. Python is object-oriented programming language that stresses on objects.



In the above example code ‘MyClass’ is the class name and ‘p1’ is the object.

**\_init\_()**: All classes have a function called \_\_init\_\_(), which is always executed when the class is being initiated.

Use the \_\_init\_\_() function to assign values to object properties, or other operations that are necessary to do when the object is being created:



Output:

John

36

**Python Lists**

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

Lists are one of 4 built-in data types in Python used to store collections of data.

Lists allow duplicate values.



**Output:**

['John', 'Jake', 'Tom']

**Accessing List elements:**

**Range of indexes is [start, end-1, step]**

Example 1:



**Output:**

Cherry

Example 2:



**Output:**

**['cherry', 'orange', 'kiwi']**

**Python Tuples**

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

**Tuple is one of 4 built-in data types in Python used to store collections of data**

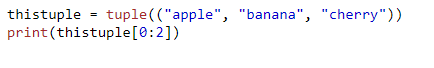
**Note:** A tuple is a collection which is ordered and **unchangeable**.



**Output:**

('apple', 10.25, 'cherry')

**Tuple slicing:**



**Output:**

**('apple', 'banana')**

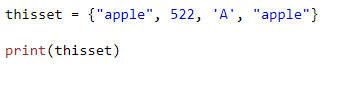
**Python Sets**

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

Set is one of 4 built-in data types in Python used to store collections of data.

**Note:** A set is a collection which is *unordered*, *unchangeable\**, and *unindexed*.

Example 1:



Output:

{'A', 522, 'apple'}

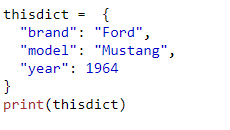
\*Prints every element one time, even if the element have duplicates.

**Python Dictionaries**

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

A dictionary is a collection which is ordered\*, changeable and do not allow duplicates.

Example:



**Output:**

{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}