## Basics

Basic syntax from the python programming language

### Showing Output To User

the print function is used to display or print output

print("Content that you wanna print on screen")

Copy

### Taking Input From User

the input function is used to take input from the user

var1 = input("Enter your name: ")

Copy

### Empty List

This method allows you to create an empty list

my\_list = []

Copy

### Empty Dictionary

By putting two curly braces, you can create a blank dictionary

my\_dict = {}

Copy

### Range Function

range function returns a sequence of numbers, eg, numbers starting from 0 to n-1 for range(0, n)

range(int\_value)

Copy

## Comments

Comments are used to make the code more understandable for programmers, and they are not executed by compiler or interpreter.

### Single line comment

#This is a single line comment

Copy

### Multi-line comment

'''This is a

multi-line

comment'''

Copy

## Escape Sequence

An escape sequence is a sequence of characters; it doesn't represent itself when used inside string literal or character.

### Newline

Newline Character

\n

Copy

### Backslash

It adds a backslash

\\

Copy

### Single Quote

It adds a single quotation mark

\'

Copy

### Tab

It gives a tab space

\t

Copy

### Backspace

It adds a backspace

\b

Copy

### Octal value

It represents the value of an octal number

\ooo

Copy

### Hex value

It represents the value of a hex number

\xhh

Copy

### Carriage Return

Carriage return or \r is a unique feature of Python. \r will just work as you have shifted your cursor to the beginning of the string or line.

\r

Copy

## Strings

Python string is a sequence of characters, and each character can be individually accessed. Using its index.

### String

You can create Strings by enclosing text in both forms of quotes - single quotes or double-quotes.

variable\_name = "String Data"

Copy

### Slicing

Slicing refers to obtaining a sub-string from the given string.

var\_name[n : m]

Copy

### String Methods isalnum() method

Returns True if all characters in the string are alphanumeric

string\_variable.isalnum()

Copy

### isalpha() method

Returns True if all characters in the string are alphabet

string\_variable.isalpha()

Copy

### isdecimal() method

Returns True if all characters in the string are decimals

string\_variable.isdecimal()

Copy

### isdigit() method

Returns True if all characters in the string are digits

string\_variable.isdigit()

Copy

### islower() method

Returns True if all characters in the string are lower case

string\_variable.islower()

Copy

### isspace() method

Returns True if all characters in the string are whitespaces

string\_variable.isspace()

Copy

### isupper() method

Returns True if all characters in the string are upper case

string\_variable.isupper()

Copy

### lower() method

Converts a string into lower case

string\_variable.lower()

Copy

### upper() method

Converts a string into upper case

string\_variable.upper()

Copy

### strip() method

It removes leading and trailing spaces in the string

string\_variable.string([chars])

Copy

## List

A List in Python represents a list of comma-separated values of any data type between square brackets.

### List

var\_name = [element1, element2, and so on]

Copy

### List Methods index method

Returns the index of the first element with the specified value

list.index(element)

Copy

### append method

Adds an element at the end of the list

list.append(element)

Copy

### extend method

Add the elements of a list (or any iterable) to the end of the current list

list.extend(iterable)

Copy

### insert method

Adds an element at the specified position

list.insert(position, element)

Copy

### pop method

Removes the element at the specified position and returns it

list.pop(position)

Copy

### remove method

The remove( ) method removes the first occurrence of a given item from the list

list.remove(element)

Copy

### clear method

Removes all the elements from the list

list.clear()

Copy

### count method

Returns the number of elements with the specified value

list.count(value)

Copy

### reverse method

Reverse the order of the list

list.reverse()

Copy

### sort method

Sorts the list

list.sort(reverse=True|False)

Copy

## Tuples

Tuples are represented as a list of comma-separated values of any data type within parentheses.

### Tuple Creation

variable\_name = (element1, element2, ...)

Copy

### Tuple Methods count method

It returns the number of times a specified value occurs in a tuple

tuple.count(value)

Copy

### index method

It searches the tuple for a specified value and returns the position.

tuple.index(value)

Copy

## Sets

A set is a collection of multiple values which is both unordered and unindexed. It is written in curly brackets.

### Set Creation: Way 1

var\_name = {element1, element2, ...}

Copy

### Set Creation: Way 2

var\_name = set([element1, element2, ...])

Copy

### Set Methods: add() method

Adds an element to a set

set.add(element)

Copy

### clear() method

Remove all elements from a set

set.clear()

Copy

### discard() method

Removes the specified item from the set

set.discard(value)

Copy

### intersection() method

Returns intersection of two or more sets

set.intersection(set1, set2 ... etc)

Copy

### issubset() method

Checks if a Set is Subset of Another Set

set.issubset(set)

Copy

### pop() method

Removes an element from the set

set.pop()

Copy

### remove() method

Removes the specified element from the Set

set.remove(item)

Copy

### union() method

Returns the union of Sets

set.union(set1, set2...)

Copy

## Dictionaries

The dictionary is an unordered set of comma-separated key: value pairs, within {}, with the requirement that within a dictionary, no two keys can be the same.

### Dictionary

<dictionary-name> = {<key>: value, <key>: value ...}

Copy

### Adding Element to a dictionary

By this method, one can add new elements to the dictionary

<dictionary>[<key>] = <value>

Copy

### Updating Element in a dictionary

If the specified key already exists, then its value will get updated

<dictionary>[<key>] = <value>

Copy

### Deleting Element from a dictionary

del let to delete specified key: value pair from the dictionary

del <dictionary>[<key>]

Copy

### Dictionary Functions & Methods len() method

It returns the length of the dictionary, i.e., the count of elements (key: value pairs) in the dictionary

len(dictionary)

Copy

### clear() method

Removes all the elements from the dictionary

dictionary.clear()

Copy

### get() method

Returns the value of the specified key

dictionary.get(keyname)

Copy

### items() method

Returns a list containing a tuple for each key-value pair

dictionary.items()

Copy

### keys() method

Returns a list containing the dictionary's keys

dictionary.keys()

Copy

### values() method

Returns a list of all the values in the dictionary

dictionary.values()

Copy

### update() method

Updates the dictionary with the specified key-value pairs

dictionary.update(iterable)

Copy

## Conditional Statements

The if statements are the conditional statements in Python, and these implement selection constructs (decision constructs).

### if Statement

if(conditional expression):

statements

Copy

### if-else Statement

if(conditional expression):

statementselse:

statements

Copy

### if-elif Statement

if (conditional expression) :

statementselif (conditional expression) :

statementselse :

statements

Copy

### Nested if-else Statement

if (conditional expression):if (conditional expression):

statementselse:

statementselse:

statements

Copy

## Iterative Statements

An iteration statement, or loop, repeatedly executes a statement, known as the loop body, until the controlling expression is false (0).

### For Loop

The for loop of Python is designed to process the items of any sequence, such as a list or a string, one by one.

for <variable> in <sequence>:

statements\_to\_repeat

Copy

### While Loop

A while loop is a conditional loop that will repeat the instructions within itself as long as a conditional remains true.

while <logical-expression> :

loop-body

Copy

### Break Statement

The break statement enables a program to skip over a part of the code. A break statement terminates the very loop it lies within.

for <var> in <sequence> :

statement1if <condition> :break

statement2

statement\_after\_loop

Copy

### Continue Statement

The continue statement skips the rest of the loop statements and causes the next iteration to occur.

for <var> in <sequence> :

statement1if <condition> :continue

statement2

statement3

statement4

Copy

## Functions

A function is a block of code that performs a specific task. You can pass parameters into a function. It helps us to make our code more organized and manageable.

### Function Definition

def my\_function(parameters):# Statements

Copy

## File Handling

File handling refers to reading or writing data from files. Python provides some functions that allow us to manipulate data in the files.

### open() function

var\_name = open("file name", "opening mode")

Copy

### close() function

var\_name.close()

Copy

### Read () function

The read functions contains different methods, read(),readline() and readlines()

read() #return one big string

Copy

It returns a list of lines

read-lines

Copy

It returns one line at a time

readline

Copy

### Write () function

This function writes a sequence of strings to the file.

write () #Used to write a fixed sequence of characters to a file

Copy

It is used to write a list of strings

writelines()

Copy

### Append () function

The append function is used to append to the file instead of overwriting it. To append to an existing file, simply open the file in append mode (a):

file = open("Hello.txt", "a")

Copy

## Exception Handling

An exception is an unusual condition that results in an interruption in the flow of the program.

### try and except

A basic try-catch block in python. When the try block throws an error, the control goes to the except block.

try:[Statement body block]raise Exception()except Exception as e:[Error processing block]

Copy

## OOPS

It is a programming approach that primarily focuses on using objects and classes. The objects can be any real-world entities.

### class

The syntax for writing a class in python

class class\_name:#Statements

Copy

### class with a constructor

The syntax for writing a class with the constructor in python

class CodeWithHarry:

# Default constructordef \_\_init\_\_(self):

self.name = "CodeWithHarry"

# A method for printing data membersdef print\_me(self):print(self.name)

Copy

### object

Instantiating an object

<object-name> = <class-name>(<arguments>)

Copy

### filter function

The filter function allows you to process an iterable and extract those items that satisfy a given condition

filter(function, iterable)

Copy

### issubclass function

Used to find whether a class is a subclass of a given class (classinfo) or not

issubclass(class, classinfo)

Copy

## Iterators and Generators

Here are some of the advanced topics of the Python programming language like iterators and generators

### Iterator

Used to create an iterator over an iterable

iter\_list = iter(['Harry', 'Aakash', 'Rohan']) print(next(iter\_list)) print(next(iter\_list)) print(next(iter\_list))

Copy

### Generator

Used to generate values on the fly

# A simple generator functiondef my\_gen():

n = 1print('This is printed first')# Generator function contains yield statementsyield n

n += 1print('This is printed second')yield n

n += 1print('This is printed at last')yield n

Copy

## Decorators

Decorators are used to modifying the behavior of function or class. They are usually called before the definition of a function you want to decorate.

### property Decorator (getter)

@propertydef name(self):return self.\_\_name

Copy

### setter Decorator

It is used to set the property 'name'

@name.setterdef name(self, value):

self.\_\_name=value

Copy

### Deletor Decorator

It is used to delete the property 'name'

@name.deleter #property-name.deleter decoratordef name(self, value):print('Deleting..')del self.\_\_name