**PYTHON NOTES**

| | | | |
|---|---|---|---|
| **Notebook:** | First Notebook | | |
| **Created:** | 06-04-2020 00:01 | **Updated:** | 06-04-2020 15:22 |
| **Author:** | CodeBot | | |

# PYTHON NOTES

## BASIC DATA TYPES

1. Int : (7,8) : Any integer type Value is int.
2. float :(3.3,4.5) any value with decimal.
3. String:(name) array of characters.

## . STRINGS :-

1. Declaring : Variable = "string"
2. Printing : Print("text")/Print(var)
3. Length = len("text")/len(var)
4. To change Case : Var.upper() {all char upper} , Var.lower() {all char lower} , Var.captalize() {first char upper}
5. Find position of char in string : Variable.find(char to find)
6. To split list : Variable.split('char' on which to split)

7. To split as first position occurred : Variable.partition("char")

## LISTS :-

1. Declaring : List = [int,float,string,other list]
2. Put values at particular index : List[pos] = input
3. Put input at end : List.append(input)
4. Count no. of repeating items : List.count(var)
5. To extend the list with other list : List.extend([])
6. Insert input at particular index : List.insert(index,var)
7. To pop item : List.pop(index)

## Dictionaries :-

1. from collection import ordered dict{}
2. Declaring : Dict = orderedDict{"key":value,"key1":value1,....}
3. Printing value : print(Dict[key]) // value will be printed.
4. To get keys : Dict.keys()
5. To get values : Dict.values()
6. to get both : Dict.items()

## Sets :-

- Declaring : myset = set() //{there is no repetitive addition of elements }
- To push Elements  : myset.add(element)

## Bool :-

- Comparison of 2 things : {if 2 things are same: true} , {if 2 things are not same: false}

## String {slicing & formatting} :-

Eg. string="Name" :

1. indexing : string[0] = 'N' , string[2] = 'm' , string[-1] = 'e'
2. Reversing a string : string[::-1]
3. Sub - Section Slicing : string[start index : ending index] will give cut of string from start to end index.
4. Step Slicing : string[start index : end index : step size to skip] will give string between 2 defined points with skip of chars accordingly to step size.
5. String formatting : used to put var in string { "hello var is added after:{}".format(var) }

# FILE - HANDLING

## Letters used :

1. 'w' : it is used to create new file.
2. 'a' : it is used to append a file.
3. 'r' : it is used to read/open a file.

- ## Basic operation and syntax :

1. Var = open("filename.txt",'w,a,r') {letter acc. to use}
2. Var.write(input) {to add anything in file}
3. Var.read("filename.txt") {to read file}

# Basic statements

- ## Comparison Operators :

1. OR operator : a<=b or b>=c // we can write a<=c<=b.
2. AND operator : a<=b and b>=c // both must be true.
3. NOT operator : not a>b //to condition any no.

- ## Decision Making :

```
1. If condition:

        statement
2. elif condtion:
            statement
3. else :
        statement
```

- ## Iterations :

1. For loop : //for var in Datatype: // for var in range(0,range){for ranged execution}.
2. While loop : //to use loop until a condition id fulfilled. //declare var , while condition: , executing statement , inc/dec in var//

# Functions on Data Types

- ## Zip function :

1. Collects 2 lists or more than 2.
2. zip(list,list,list,.....n)

- ## Min,Max functions :

1. min(list) -> gives min value of list.
2. max(list) -> gives max value of list.
3. sort(list) -> sorts the list.

# Functions

- ## Declaring :

def function_name(argument):

body
return value

## Calling :

```
__init__ == "__main__"
function_name(argument)
```

## *args :

1. use : allows to take infinite number of arguments in function.
2. storing : it stores arguments as list.
3. declaring : function_name(*args)

## **kwargs :

1. Use : same use as args.
2. storing : stores arguments in dictionary.
3. declaring : function_name(**kwargs)

## Map Function :

- Use : applying function on full list.
- declaring : map(function_name , list)

## L.E.G.B. Rule :

1. L -> local var , E -> Enclosed var , G -> global var , B -> builtin var.
2. Use : gives priority order of choosing var done by interpreter.

3. L > E > G > B

# Object Oriented Programming

## . Declaration :

class name_class:
def __init__(self,arguments)
self.argument = argument(//it is given by user in argument)
# it is giving value to instance of data type.

## . Example :

class dog:
def __init__(self,name,breed)
self.name = name
self.breed = breed
calling : Dog.breed() will give breed provided by user.

## . Functions of class :

def __function_name__(self,other arguments)
function body
return value

- ## Inheritance :

1. Use : to draw properties of parent class in another class.
2. Syntax : class dog(animal) // where animal is parent class.
3. Advantages : use functions , vars of parent class in derived class.

- ## Polymorphism :

1. use : we can use functions of same name in different classes.
2. use : we can use variables of same name in different functions and classes.

# Exception - Handling

- ## Use :

It avoids error and executes a command provided by programmer whenever program counters with error on run-time.

- ## Syntax :

```
try :
        execution command
    except error e:
        print(e) //shows error//optional
        execution command
    finally :                    //optional
          execution command
```

# Python - Decorators

## . Syntax :

```
@function_1()
 def function_2():
      function body
      return value
```

## . Use :

1. function_1 will be executed with return of function_2 as argument for it.

---------------------------------------------------------------------------------------------------------------------------------------------