Lab #01 task:

Exercise: Dir and Help

Learn about the methods Python provides for strings. To see what methods Python provides for a datatype, use the dir and help commands:

```
>>> s = 'abc'
```

>>> dir(s)

['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__',
'__ge__', '__getattribute__', '__getitem__', '__getnewargs__',
'__getslice__', '__gt__', '__hash__', '__init__','__le__', '__len__',
'__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__','__repr__', '__rmod__', '__rmul__', '__setattr__', '__str__', 'capitalize', 'center', 'count', 'decode',
'encode', 'endswith',
'expandtabs', 'find', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower',
'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
'replace', 'rfind','rindex', 'rjust', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

>>> help(s.find)

Help on built-in function find:

find(...) method of builtins.str instance
S.find(sub[, start[, end]]) -> int

Return the lowest index in S where substring sub is found, such that sub is contained within S[start:end]. Optional arguments start and end are interpreted as in slice notation.

Return -1 on failure.

```
>> s.find('b')
```

1 Try out some of the string functions listed in dir (ignore those with underscores '_' around the method name).

Exercise Python input /output Basic operations

(i)Write a Python program to swap 4 variables values (input four values.

Sample input:
Before swapping
a=2,b=56,c=78,d=9

After Swapping
a=,9,b=78,c=56,d=2

(ii) Write a Python program to convert temperatures to and from celsius,
Fahrenheit.
Formula : c/5 = f-32/9
Expected Output :
Enter temp in Celsius: 60°C
Temperature in Fahrenheit is :140

Exercise: Lists

(i)Play with some of the list functions. You can find the methods you can call on an object via the dir and get information about them via the help command:

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__',   '__delslice__', '__doc__', '__eq__', '__ge__',
'__getattribute__',
'__getitem__', '__getslice__', '__gt__', '__hash__', '__iadd__', '__imul__',
'__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__',
'__rmul__', '__setattr__', '__setitem__', '__setslice__', '__str__',   'append', 'count', 'extend', 'index', 'insert', 'pop',
'remove', 'reverse',  'sort']

>>> help(list.reverse)
Help on built-in function reverse:


reverse(...)
   L.reverse() -- reverse *IN PLACE*

>>> lst = ['a','b','c']
>>> lst.reverse()
>>> ['c','b','a']
```

Note: Ignore functions with underscores "_" around the names; these are private helper methods. Press 'q' to back out of a help screen

(ii)Write a Python program to count the number of strings where the string length is 2 or more and the first and last character are same from a given list of strings.
Sample List : ['abc', 'xyz', 'aba', '1221']
Expected Result : 2.

Exercise: Dictionaries

(i)Use dir and help to learn about the functions you can call on dictionaries and implement it.

(ii)Write a Python script to concatenate following dictionaries to create a new one.

Sample Dictionary :
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

Exercise: List Comprehensions

(i)Write a list comprehension which, from a list, generates a lowercased version of each string that has length greater than five.

(ii)Write a Python program to print a specified list after removing the 0th, 4th and 5th elements
Sample List : ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow','Teapink']
Expected Output : ['Green', 'White', 'Black']

Exercise : Operators:

Play with some Operators in Python(assignment ,bitwise ,logical, arithmetic, identity, membership)

(i) What will be the output of the given program

Identity Operators in Python
x = 6
if (type(x) is int):
  print ("true")
else:
  print ("false")
	Output:

if (type(x) x = 7.2
if (type(x) is not int):
  print ("true")
else:
  print ("false")
	Output:

Membership operator:
list1=[1,2,3,4,5]
list2=[6,7,8,9]
for item in list1:
  if item in list2:
  print("overlapping")
else:
  print("not overlapping")
	Output:

Floor division and Exponent and Assign a//=3 a**=5 print(“floor divide=”,a) print(“exponent=”,a)	**Output:**

Bitwise Operaotors:
a = 60 /* 60 = 0011 1100 */
b = 13 /* 13 = 0000 1101 */
int c = 0
 	
c = a & b /* 12 = 0000 1100 */
print("Line 1", c)
 	
c = a | b /* 61 = 0011 1101 */
print("Line 2 ", c)
 	
c = a ^ b /* 49 = 0011 0001 */
print("Line 3 ", c)
 	
c = ~a /*-61 = 1100 0011 */
print("Line 4", c)
 	
c = a << 2 /* 240 = 1111 0000 */
printf("Line 5 ", c);
 	
c = a >> 2 /* 15 = 0000 1111 */
printf("Line 6 -", c);
	output

Exercise

Create a Python Program that perform following tasks for any problem of your choice:
Task 1: Introduction
Task 2: Terminal
Task 3: Python Interpreter
Task 4: Variables
Task 5: Text Editor
Task 6: Functions
Task 7: Lists and Tuples
Task 8: Conditional Statements
Task 9: The For Loop
Task 10: User Input and the While Loop

Resources:

1. https://github.com/rajuiit/Machine-Learning-Summer-2020/blob/master/week_2_basicpython.ipynb
2. https://github.com/rajuiit/Machine-Learning-Summer 2020/blob/master/week_2_use_of_build_in_function.ipynb