	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Lab Work#1

NAME:
ROLL NO:
BATCH:

AIM: Applying Basic Python Programming Constructs- functions, conditional statements, and iteration statements for solving the given scenario in Problem Statement#1

OBJECTIVE: The objective of Lab Work#1 is:

1. Develop and understanding of programming solution for a given scenario.
2. Identify the suitable programming elements of Python Programming Language to apply for developing programming solution.

OUTCOMES: After completing the Lab Work#1 students will be able to:

1. Analyze Problem Statement#1 and choose which programming elements to apply.
2. Apply the Basic Python Programming Constructs, - functions, conditional statements, and iteration statements, in solving programming problem mentioned in Problem Statement#1

PRE-REQUISITE:

1. Syntax for Loops, Conditional Statement and Functions in Python Programming Language.

INPUT-OUTPUT:

1. The input will be RPM and output should be whether the watch unit is suitable to launch in market or must be discarded.

THEORY:

What is Python?

Python is a popular programming language. It was created by Guido van Rossum and released in 1991.
It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

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

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

Example

If statement:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

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).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Example

Print each fruit in a fruit list:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

The break Statement

With the break statement we can stop the loop before it has looped through all the items:

Example

Exit the loop when x is "banana":

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Python Functions

A function is a block of code which only runs when it is called.

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

A function can return data as a result.

Creating a Function

In Python a function is defined using the def keyword:

Example

```
def my_function():
  print("Hello from a function")
```

Calling a Function

To call a function, use the function name followed by parenthesis:

Example

```
def my_function():
  print("Hello from a function")
my_function()
```

Parameters

Information can be passed to functions as parameter.

Parameters are specified after the function name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

The following example has a function with one parameter (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

```
Example
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

Default Parameter Value

The following example shows how to use a default parameter value.
If we call the function without parameter, it uses the default value:

```
Example
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

SOURCE CODE:
//source code here

INPUT/OUTPUT:
//Execute the program for all possible combination of inputs and mention the output

ASSESSMENT QUESTIONS:
(To be written individually)

1. You enter the x-y coordinates of 3 points in 2-d space and find out if they form a valid triangle.

2. Compare the usage of Java and Python Programming Language

** Courtesy:
https://www.w3schools.com

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Lab Work#2

NAME:
ROLL NO:
BATCH:

AIM: Apply List Data Type in python to solve programming problems.

OBJECTIVE: The objective of this lab work is:

1. Students will be able to understand list data type and its operations.
2. Students will be to use list data type and its functions.
3. Students will be able to apply suitable list functions to solve given programming problem.

OUTCOMES: After completing the Lab Work#2 students will be able to:

1. Apply basics list data type and its functions to solve programming problems.

PRE-REQUISITE:

1. Conditional Statements in python, functions in python, loops in python.

INPUT-OUTPUT:

1. Input will be as specified in problem statement#2- profits of 12 months.
2. Output shall be as desired in problem statement#2- Identified pattern of profits/ losses.

THEORY:

Python Collections (Arrays)

There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable, and indexed. No duplicate members.

When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a data set could mean retention of meaning, and it could mean an increase in efficiency or security.

List

A list is a collection which is ordered and changeable. In Python lists are written with square brackets.

Example

Create a List:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

List Methods

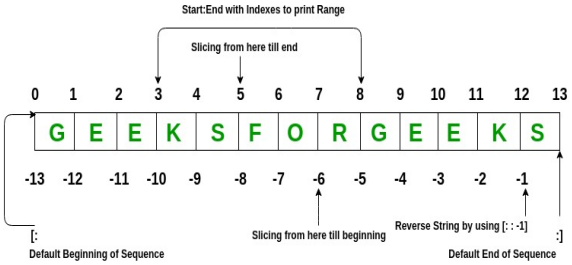
Python has a set of built-in methods that you can use on lists.

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list

count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Slicing of a List

In Python List, there are multiple ways to print the whole List with all the elements, but to print a specific range of elements from the list, we use Slice operation. Slice operation is performed on Lists with the use of colon(:). To print elements from beginning to a range use [:Index], to print elements from end use [:-Index], to print elements from specific Index till the end use [Index:], to print elements within a range, use [Start Index:End Index] and to print whole List with the use of slicing operation, use [:]. Further, to print whole List in reverse order, use [::-1].



SOURCE CODE:
Write source code with comments.

OUTPUT:
Write outputs with all input combinations.

ASSESSMENT QUESTIONS:

1. Write a function sumprimes(l) that takes as input a list of integers l and returns the sum of all the prime numbers in l. Here are some examples to show how your function should work.

```
>>>sumprimes([3,3,1,13])
 19
>>>sumprimes([2,4,6,9,11])
 13
>>>sumprimes([-3,1,6])
 0
```

**Courtesy:https://www.geeksforgeeks.org/

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Lab Work#3

NAME:
ROLL NO:
BATCH:

AIM: Apply Dictionary Data Type in Python to create solution for Problem Statement#3.

OBJECTIVE: The objective of this lab work is:

1. Students will be able to understand dictionary data type and its operations.
2. Students will be to use dictionary data type and its functions.
3. Students will be able to apply suitable dictionary functions to solve given programming problem.

OUTCOMES: After completing the Lab Work#3 students will be able to:

1. Apply dictionary data type to solve given programming problem.

PRE-REQUISITE:

1. Conditional Statements in python, functions in python, loops in python.
2. List data type

INPUT-OUTPUT:

1. Input will be as specified in problem statement#3.
2. Output shall be as desired in problem statement#3.

THEORY:

Dictionaries

Dictionaries are sometimes found in other languages as "associative memories" or "associative arrays". Unlike sequences, which are indexed by a range of numbers, dictionaries are indexed by *keys*, which can be any immutable type; strings and numbers can always be keys. Tuples can be used as keys if they contain only strings, numbers, or tuples; if a tuple contains any mutable object either directly or indirectly, it cannot be used as a key. You can't use lists as keys, since lists can be modified in place using index assignments, slice assignments, or methods like append()and extend().

It is best to think of a dictionary as a set of *key: value* pairs, with the requirement that the keys are unique (within one dictionary). A pair of braces creates an empty dictionary: {}. Placing a comma-separated list of key:value pairs within the braces adds initial key:value pairs to the dictionary; this is also the way dictionaries are written on output.

The main operations on a dictionary are storing a value with some key and extracting the value given the key. It is also possible to delete a key:value pair with del. If you store using a key that is already in use, the old value associated with that key is forgotten. It is an error to extract a value using a non-existent key.

Performing list(d) on a dictionary returns a list of all the keys used in the dictionary, in insertion order (if you want it sorted, just use sorted(d) instead). To check whether a single key is in the dictionary, use the inkeyword.

Here is a small example using a dictionary:

```
>>>tel={'jack':4098,'sape':4139}
>>>tel['guido']=4127
>>>tel
{'jack': 4098, 'sape': 4139, 'guido': 4127}
>>>tel['jack']
4098
>>>delete tel['sape']
>>>tel['irv']=4127
>>>tel
```

```
{'jack': 4098, 'guido': 4127, 'irv': 4127}
>>>list(tel)
['jack', 'guido', 'irv']
>>>sorted(tel)
['guido', 'irv', 'jack']
>>>'guido'in tel
True
>>>'jack'notin tel
False
```

The dict() constructor builds dictionaries directly from sequences of key-value pairs:

```
>>>dict([('sape',4139),('guido',4127),('jack',4098)])
{'sape': 4139, 'guido': 4127, 'jack': 4098}
```

In addition, dict comprehensions can be used to create dictionaries from arbitrary key and value expressions:

```
>>>{x:x**2forxin(2,4,6)}
{2: 4, 4: 16, 6: 36}
```

When the keys are simple strings, it is sometimes easier to specify pairs using keyword arguments:

```
>>>dict(sape=4139,guido=4127,jack=4098)
{'sape': 4139, 'guido': 4127, 'jack': 4098}
```

Looping Techniques

When looping through dictionaries, the key and corresponding value can be retrieved at the same time using the items() method.

```
>>>knights={'gallahad':'the pure','robin':'the brave'}
>>>fork,vinknights.items():
... print(k,v)
...
gallahad the pure
robin the brave
```

When looping through a sequence, the position index and corresponding value can be retrieved at the same time using the enumerate() function.

```
>>>
```

```
>>>fori,vinenumerate(['tic','tac','toe']):
... print(i,v)
...
0 tic
1 tac
2 toe
```

To loop over two or more sequences at the same time, the entries can be paired with the zip() function.

```
>>>
```

```
>>>questions=['name','quest','favoritecolor']
>>>answers=['lancelot','the holy grail','blue']
>>>forq,ainzip(questions,answers):
... print('What is your {0}? It is {1}.'.format(q,a))
...
What is your name? It is lancelot.
What is your quest? It is the holy grail.
```

What is your favoritecolor? It is blue.

To loop over a sequence in reverse, first specify the sequence in a forward direction and then call the reversed() function.

```
>>>
>>>for i in reversed(range(1,10,2)):
... print(i)
...
9
7
5
3
1
```

To loop over a sequence in sorted order, use the sorted() function which returns a new sorted list while leaving the source unaltered.

```
>>>basket=['apple','orange','apple','pear','orange','banana']
>>>for f in sorted(set(basket)):
... print(f)
...
apple
banana
orange
pear
```

It is sometimes tempting to change a list while you are looping over it; however, it is often simpler and safer to create a new list instead.

```
>>>import math
>>>raw_data=[56.2,float('NaN'),51.7,55.3,52.5,float('NaN'),47.8]
>>>filtered_data=[]
>>>for value in raw_data:
... if not math.isnan(value):
... filtered_data.append(value)
...
>>>filtered_data
[56.2, 51.7, 55.3, 52.5, 47.8]
```

SOURCE CODE:
// Write the source code with comments here

OUTPUT:
// Mention all kinds of outputs obtained

ASSESSMENT QUESTIONS:

1. Create a dictionary to store score of each team in a cricket tournament, match-wise

Courtesy: https://www.geeksforgeeks.org

Lab Work#4

NAME:
ROLL NO:
BATCH:

AIM: Apply Dictionary Data Type in Python to create solution for Problem Statement#4.

OBJECTIVE: The objective of this lab work is:

1. Students will be able to understand dictionary data type and its operations.
2. Students will be to use dictionary data type and its functions.
3. Students will be able to apply suitable dictionary functions to solve given programming problem.

OUTCOMES: After completing the Lab Work#4 students will be able to:

1. Apply basics list data type and its functions to solve programming problems.

PRE-REQUISITE:

1. Conditional Statements in python, functions in python, loops in python.
2. List and tuple data type

INPUT-OUTPUT:

1. Input will be as specified in problem statement#4.
2. Output shall be as desired in problem statement#4.

THEORY:

A Dictionary in Python works similar to the Dictionary in the real world. Keys of a Dictionary must be unique and of immutable data type such as Strings, Integers and tuples, but the key-values can be repeated and be of any type.
Nested Dictionary: Nesting Dictionary means putting a dictionary inside another dictionary. Nesting is of great use as the kind of information we can model in programs is expanded greatly.

```
nested_dict = { 'dict1': {'key_A': 'value_A'},
                'dict2': {'key_B': 'value_B'}}
```

262
A nested dict is a dictionary within a dictionary. A very simple thing.

```
>>> d = {}
>>> d['dict1'] = {}
>>> d['dict1']['innerkey'] = 'value'
>>> d
{'dict1': {'innerkey': 'value'}}
```

You can also use a defaultdict from the collectionspackage to facilitate creating nested dictionaries.

```
>>> import collections
>>> d = collections.defaultdict(dict)
>>> d['dict1']['innerkey'] = 'value'
>>>d  # currently a defaultdict type
defaultdict(<type 'dict'>, {'dict1': {'innerkey': 'value'}})
>>>dict(d)  # but is exactly like a normal dictionary.
{'dict1': {'innerkey': 'value'}}
```

SOURCE CODE:

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

// Write the source code with comments here

OUTPUT:
// Mention all kinds of outputs obtained

ASSESSMENT QUESTIONS:

1. Up to how many levels can you nest a dictionary?

2. Write disadvantages of using python dictionary.

Courtesy: https://www.geeksforgeeks.org

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Lab Work#5

NAME:
ROLL NO:
BATCH:

AIM: Apply Dictionary Data Type in Python to create solution for Problem Statement#5 with lists and Tuples

OBJECTIVE: The objective of this lab work is:

1. Students will be able to use list and set data types.
2. Students will be able to apply dictionary.
3. Students will be to use dictionary data type and its functions.
4. Students will be able to choose appropriate data structure to come up with solution of given problem statement.

OUTCOMES: After completing the Lab Work#5 students will be able to:

1. Use dictionary data type to come up with programming solution for given data in the programming statement.
2. Apply basics list data type and its functions to solve programming problems.

PRE-REQUISITE:

1. List and tuple data type
2. Dictionary data type

INPUT-OUTPUT:

1. Input will be as specified in problem statement#5.
2. Output shall be as desired in problem statement#5.

THEORY:

A Dictionary in Python works similar to the Dictionary in the real world. Keys of a Dictionary must be unique and of immutable data type such as Strings, Integers and tuples, but the key-values can be repeated and be of any type.

Nested Dictionary: Nesting Dictionary means putting a dictionary inside another dictionary. Nesting is of great use as the kind of information we can model in programs is expanded greatly.

```
nested_dict = { 'dict1': {'key_A': 'value_A'},
                'dict2': {'key_B': 'value_B'}}
```

262

A nested dict is a dictionary within a dictionary. A very simple thing.

```
>>> d = {}
>>> d['dict1'] = {}
>>> d['dict1']['innerkey'] = 'value'
>>> d
{'dict1': {'innerkey': 'value'}}
```

You can also use a defaultdict from the collectionspackage to facilitate creating nested dictionaries.

```
>>> import collections
>>> d = collections.defaultdict(dict)
>>> d['dict1']['innerkey'] = 'value'
>>>d  # currently a defaultdict type
defaultdict(<type 'dict'>, {'dict1': {'innerkey': 'value'}})
>>>dict(d)  # but is exactly like a normal dictionary.
{'dict1': {'innerkey': 'value'}}
```

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

SOURCE CODE:
// Write the source code with comments here

OUTPUT:
// Mention all kinds of outputs obtained

ASSESSMENT QUESTIONS:

1. Compare lists and tuples.

Courtesy: https://www.geeksforgeeks.org

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Lab Work#6

NAME:
ROLL NO:
BATCH:

AIM: Application of dictionary, list, and tuple data type

OBJECTIVE: The objective of this lab work is:

1. Students will be able to use list and set data types.
2. Students will be able to apply dictionary.
3. Students will be able to apply tuple data type.

OUTCOMES: After completing the Lab Work#6 students will be able to:

1. Use list, tuple and dictionary data type to come up for solution of given problem statement.

PRE-REQUISITE:

1. List and tuple data type
2. Dictionary data type

INPUT-OUTPUT:

1. Input will be as specified in problem statement#6.
2. Output shall be as desired in problem statement#6.

THEORY:

Tuples in Python

A Tuple is a collection of Python objects separated by commas. In someways a tuple is similar to a list in terms of indexing, nested objects and repetition but a tuple is immutable unlike lists which are mutable.

Create a Tuple:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Access Tuple Items

You can access tuple items by referring to the index number, inside square brackets:

Example

Print the second item in the tuple:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```

Negative Indexing

Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second last item etc.

Example

Print the last item of the tuple:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[-1])
```

Range of Indexes

You can specify a range of indexes by specifying where to start and where to end the range.

When specifying a range, the return value will be a new tuple with the specified items.

Example

Return the third, fourth, and fifth item:

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
```

```
print(thistuple[2:5])
```

Note: The search will start at index 2 (included) and end at index 5 (not included).
Remember that the first item has index 0.

Range of Negative Indexes

Specify negative indexes if you want to start the search from the end of the tuple:

Example

This example returns the items from index -4 (included) to index -1 (excluded)

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[-4:-1])
```

Change Tuple Values

Once a tuple is created, you cannot change its values. Tuples are **unchangeable**, or **immutable** as it also is called.
But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

Example

Convert the tuple into a list to be able to change it:

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

Run example »

Loop Through a Tuple

You can loop through the tuple items by using a for loop.

Example

Iterate through the items and print the values:

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)
```

Run example »

You will learn more about for loops in our Python For Loops Chapter.

Check if Item Exists

To determine if a specified item is present in a tuple use the in keyword:

Example

Check if "apple" is present in the tuple:

```
thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
  print("Yes, 'apple' is in the fruits tuple")
```

Run example »

Tuple Length

To determine how many items a tuple has, use the len() method:

Example

Print the number of items in the tuple:

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

Run example »

Add Items

Once a tuple is created, you cannot add items to it. Tuples are **unchangeable**.

Example

You cannot add items to a tuple:

```
thistuple = ("apple", "banana", "cherry")
thistuple[3] = "orange" # This will raise an error
print(thistuple)
```

Run example »

Create Tuple With One Item

To create a tuple with only one item, you have add a comma after the item, unless Python will not recognize the variable as a tuple.

Example

One item tuple, remember the commma:

```
thistuple = ("apple",)
print(type(thistuple))

#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
```

Remove Items

Note: You cannot remove items in a tuple.

Tuples are **unchangeable**, so you cannot remove items from it, but you can delete the tuple completely:

Example

The del keyword can delete the tuple completely:

```
thistuple = ("apple", "banana", "cherry")
del thistuple
print(thistuple) #this will raise an error because the tuple no longer exists
```

Join Two Tuples

To join two or more tuples you can use the + operator:

Example

Join two tuples:

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

The tuple() Constructor

It is also possible to use the tuple() constructor to make a tuple.

Example

Using the tuple() method to make a tuple:

```
thistuple = tuple(("apple", "banana", "cherry")) # note the double round-brackets
print(thistuple)
```

Tuple Methods

Python has two built-in methods that you can use on tuples.

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

SOURCE CODE:
// Write the source code with comments here

OUTPUT:
// Mention all kinds of outputs obtained

ASSESSMENT QUESTIONS:

1. Categorize operations of tuple and set data type.

Courtesy: https://www.geeksforgeeks.org
https://www.w3schools.com

Lab Work#7

NAME:
ROLL NO:
BATCH:

AIM: Apply Python Object Oriented Programming concepts to give programming solution to problem statement#7.

OBJECTIVE: The objective of Lab Work#7 is:

1. Develop and understanding of programming solution for a given scenario.
2. Identify the suitable Object-Oriented programming elements of Python Programming Language to apply for developing programming solution.

OUTCOMES: After completing the Lab Work#7 students will be able to:

1. Analyze Problem Statement#1 and choose which programming elements to apply.
2. Create object-oriented programs in python.

PRE-REQUISITE:

1. Syntax for Loops, Conditional Statement and Functions in Python Programming Language.

INPUT-OUTPUT:

1. Simulate the programming solution as asked in problem statement#7

THEORY:

Python has been an object-oriented language since it existed. Because of this, creating and using classes and objects are downright easy. This chapter helps you become an expert in using Python's object-oriented programming support.

Overview of OOP Terminology

- Class − A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- Class variable − A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- Data member − A class variable or instance variable that holds data associated with a class and its objects.

Creating Classes

The class statement creates a new class definition. The name of the class immediately follows the keyword class followed by a colon as follows −

```
class ClassName:
   'Optional class documentation string'
class_suite
```

- The class has a documentation string, which can be accessed via ClassName.__doc__.
- The class_suite consists of all the component statements defining class members, data attributes and functions.

Creating Instance Objects

To create instances of a class, you call the class using class name and pass in whatever arguments its __init__ method accepts.

```
"This would create first object of Employee class"
emp1 = Employee("Zara", 2000)
"This would create second object of Employee class"
emp2 = Employee("Manni", 5000)
```

Accessing Attributes

You access the object's attributes using the dot operator with object. Class variable would be accessed using class name as follows −

```
emp1.displayEmployee()
emp2.displayEmployee()
print "Total Employee %d" % Employee.empCount
```

Destroying Objects (Garbage Collection)

Python deletes unneeded objects (built-in types or class instances) automatically to free the memory space. The process by which Python periodically reclaims blocks of memory that no longer are in use is termed Garbage Collection.

Python's garbage collector runs during program execution and is triggered when an object's reference count reaches zero. An object's reference count changes as the number of aliases that point to it changes.

An object's reference count increases when it is assigned a new name or placed in a container (list, tuple, or dictionary). The object's reference count decreases when it's deleted with del, its reference is reassigned, or its reference goes out of scope. When an object's reference count reaches zero, Python collects it automatically.

```
a = 40      # Create object <40>
b = a       # Increase ref. count  of<40>
c = [b]     # Increase ref. count  of <40>

del a       # Decrease ref. count  of<40>
b = 100     # Decrease ref. count  of<40>
c[0] = -1   # Decrease ref. count  of <40>
```

SOURCE CODE:

// Write the source code with comments here

OUTPUT:

// Mention all kinds of outputs obtained

ASSESSMENT QUESTIONS:

(To be written individually)

1. Give example of a class for Employee with its constructor.

** Courtesy:
https://www.tutorialspoint.com/

Lab Work#8

NAME:
ROLL NO:
BATCH:

AIM: Apply concept of Polymorphism to give programming solution to problem statement#7

OBJECTIVE: The objective of Lab Work#8 is:

1. Develop and understanding of programming solution for a given scenario.
2. Identify the suitable Object-Oriented programming elements of Python Programming Language to apply for developing programming solution.
3. Apply the concept of polymorphism- method overloading and method overriding.

OUTCOMES: After completing the Lab Work#8 students will be able to:

1. Analyze Problem Statement#7 and choose which programming elements to apply.
2. Create object-oriented programs in python.
3. Apply method overloading and method overriding.
4. Apply inheritance.

PRE-REQUISITE:

1. Syntax for Loops, Conditional Statement and Functions in Python Programming Language.
2. Syntax of class, object, inheritance in python

INPUT-OUTPUT:

1. Simulate the programming solution as asked in problem statement#9

THEORY:

Inheritance

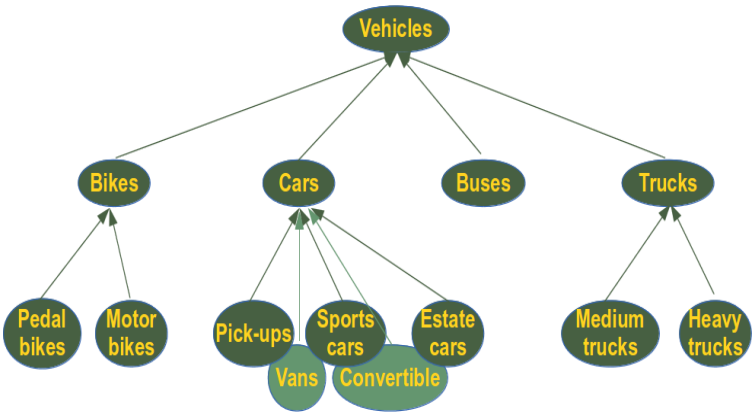
Every object-oriented programming language would not be worthy to look at or use, if it weren't to support inheritance. Inheritance was invented in 1969 for Simula. Python not only supports inheritance but multiple inheritance as well. Generally speaking, inheritance is the mechanism of deriving new classes from existing ones. By doing this we get a hierarchy of classes. In most class-based object-oriented languages, an object created through inheritance (a "child object") acquires all, - though there are exceptions in some programming languages, - of the properties and behaviors of the parent object.
Inheritance allows programmers to create classes that are built upon existing classes, and this makes it possible that a class created through inheritance inherits the attributes and methods of the parent class. This means that inheritance supports code reusability. The methods or generally speaking the software inherited by a subclass is considered to be reused in the subclass. The relationships of objects or classes through inheritance give rise to a directed graph.

The class from which a class inherits is called the parent or superclass. A class which inherits from a superclass is called a subclass, also called heir class or child class. Super classes are sometimes called ancestors as well. There exists a hierarchy relationship between classes. It's similar to relationships or categorizations that we know from real life. Think about vehicles, for example. Bikes, cars, buses and trucks are vehicles. Pick-ups, vans, sports cars, convertibles and estate cars are all cars and by being cars they are vehicles as well. We could implement a vehicle class in Python, which might have methods like accelerate and brake. Cars, Buses and Trucks and Bikes can be implemented as subclasses which will inherit these methods from vehicle.

Syntax of Inheritance in Python
The syntax for a subclass definition looks like this:
class DerivedClassName(BaseClassName):

pass

Classification of vehicles



Instead of the pass statement, there will be methods and attributes like in all other classes. The name BaseClassName must be defined in a scope containing the derived class definition.
Several ways to call a method (method overloading)
In Python you can define a method in such a way that there are multiple ways to call it.

- Given a single method or function, we can specify the number of parameters ourself.
- Depending on the function definition, it can be called with zero, one, two or more parameters.
- This is known as method overloading. Not all programming languages support method overloading, but Python does.

Method overloading example

We create a class with one method sayHello(). The first parameter of this method is set to None, this gives us the option to call it with or without a parameter.
An object is created based on the class, and we call its method using zero and one parameter.

```
#!/usr/bin/env python

class Human:

    def sayHello(self, name=None):

        if name is not None:
print('Hello ' + name)
        else:
```

```
print('Hello ')


# Create instance
obj = Human()

# Call the method
obj.sayHello()

# Call the method with a parameter
obj.sayHello('Guido')
```

Output:

```
Hello
Hello Guido
```

To clarify method overloading, we can now call the method sayHello() in two ways:

```
obj.sayHello()
obj.sayHello('Guido')
```

Overriding

Let us get back to our new PhysicianRobot class. Imagine now that an instance of a PhysicianRobot should say hi in a different way. In this case, we have to redefine the method say_hi inside of the subclass PhysicianRobot:

```
class Robot:

    def __init__(self, name):
        self.name = name

    def say_hi(self):
print("Hi, I am " + self.name)

class PhysicianRobot(Robot):
    def say_hi(self):
print("Everything will be okay! ")
print(self.name + " takes care of you!")
y = PhysicianRobot("James")
y.say_hi()
Everything will be okay!
James takes care of you!
```

What we have done in the previous example is called overriding. A method of a parent class gets overridden by simply defining in the child class a method with the same name.

If a method is overridden in a class, the original method can still be accessed, but we have to do it by calling the method directly with the class name, i.e.Robot.say_hi(y). We demonstrate this in the following code:

```
y = PhysicianRobot("Doc James")
y.say_hi()
print("... and now the 'traditional' robot way of saying hi :-)")
Robot.say_hi(y)
```

Everything will be okay!
Doc James takes care of you!
... and now the 'traditional' robot way of saying hi :-)
Hi, I am Doc James

SOURCE CODE:
// Write the source code with comments here

OUTPUT:
// Mention all kinds of outputs obtained

ASSESSMENT QUESTIONS:
(To be written individually)

1. Write advantages of inheritance.

2. Mention types of Inheritance.

** Courtesy:
https://www.python-course.eu

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Lab Work#9

NAME:
ROLL NO:
BATCH:

AIM: Apply Python File Handling to create solution for Problem Statement#8.

OBJECTIVE: The objective of this lab work is:

1. Students will be able to understand File Handling Operations in Python
2. Students will be to use Python File handling to solve problems.
3. Students will be able to apply suitable Python File handling functions to solve given programming problem.

OUTCOMES: After completing the Lab Work#9 students will be able to:

1. Apply python file handling to find programming solutions for given situation.

PRE-REQUISITE:

1. Conditional Statements in python, functions in python, loops in python.
2. List data type and its functions

INPUT-OUTPUT:

1. Input will be a coded file.
2. Output will be a decoded file.

THEORY:

Python provides basic functions and methods necessary to manipulate files by default. You can do most of the file manipulation using a **file** object.

File Handling

The key function for working with files in Python is the open() function.
The open() function takes two parameters; *filename*, and *mode*.
There are four different methods (modes) for opening a file:
"r" - Read - Default value. Opens a file for reading, error if the file does not exist
"a" - Append - Opens a file for appending, creates the file if it does not exist
"w" - Write - Opens a file for writing, creates the file if it does not exist
"x" - Create - Creates the specified file, returns an error if the file exists
In addition you can specify if the file should be handled as binary or text mode
"t" - Text - Default value. Text mode
"b" - Binary - Binary mode (e.g. images)

Syntax

To open a file for reading it is enough to specify the name of the file:
f = open("demofile.txt")
The code above is the same as:
f = open("demofile.txt", "rt")
Because "r" for read, and "t" for text are the default values, you do not need to specify them.

Open a File on the Server

Assume we have the following file, located in the same folder as Python:
demofile.txt
Hello! Welcome to demofile.txt
This file is for testing purposes.

Good Luck!

To open the file, use the built-in open() function.

The open() function returns a file object, which has a read() method for reading the content of the file:

Example

```
f = open("demofile.txt", "r")
print(f.read())
```

Read Only Parts of the File

By default the read() method returns the whole text, but you can also specify how many characters you want to return:

Example

Return the 5 first characters of the file:

```
f = open("demofile.txt", "r")
print(f.read(5))
```

Write to an Existing File

To write to an existing file, you must add a parameter to the open() function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

Example

Open the file "demofile.txt" and append content to the file:

```
f = open("demofile.txt", "a")
f.write("Now the file has one more line!")
```

Example

Open the file "demofile.txt" and overwrite the content:

```
f = open("demofile.txt", "w")
f.write("Woops! I have deleted the content!")
```

Note: the "w" method will overwrite the entire file.

Create a New File

To create a new file in Python, use the open() method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exist

"a" - Append - will create a file if the specified file does not exist

"w" - Write - will create a file if the specified file does not exist

Example

Create a file called "myfile.txt":

```
f = open("myfile.txt", "x")
```

Result: a new empty file is created!

Example

Create a new file if it does not exist:

```
f = open("myfile.txt", "w")
```

Delete a File

To delete a file, you must import the OS module, and run its os.remove() function:

Example

Remove the file "demofile.txt":

```
import os
os.remove("demofile.txt")
```

Check if File exist:

To avoid getting an error, you might want to check if the file exists before you try to delete it:

Example

Check if file exists, *then* delete it:

```
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```

Delete Folder

To delete an entire folder, use the os.rmdir() method:

Example

Remove the folder "myfolder":

```
import os
os.rmdir("myfolder")
```

SOURCE CODE:

// Write the source code with comments here

OUTPUT:

// Mention all kinds of outputs obtained

ASSESSMENT QUESTIONS:

1. Write python function to count number of words in a file.

2. Write python program to create a file F3 by copying one line each alternately from file1 and file 2.

Lab Work#10

NAME:
ROLL NO:
BATCH:

AIM: Study of NumPy Library Functions in Python

OBJECTIVE: The objective of this lab work is:

1. Students will be able to list functions of NumPy.
2. Students will be able to configure NumPy.
3. Students will be able to apply functions of NumPy.

OUTCOMES: After completing the Lab Work#10 students will be able to:

1. Configure NumPy in their system.
2. Understand use of NumPy functions.
3. Apply NumPy Functions

PRE-REQUISITE:

1. Basic Syntax of Python

THEORY:

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

What is NumPy?

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for Numerical Python.

Why Use NumPy?

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

Data Science: is a branch of computer science where we study how to store, use and analyze data for deriving information from it.

Why is NumPy Faster Than Lists?

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.

This behavior is called locality of reference in computer science.

This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.

- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

Standard Python distribution doesn't come bundled with NumPy module. A lightweight alternative is to install NumPy using popular Python package installer, **pip**.

pip install numpy

The best way to enable NumPy is to use an installable binary package specific to your operating system. These binaries contain full SciPy stack (inclusive of NumPy, SciPy, matplotlib, IPython, SymPy and nose packages along with core Python).

Windows

Anaconda (from https://www.continuum.io) is a free Python distribution for SciPy stack. It is also available for Linux and Mac.

Canopy (https://www.enthought.com/products/canopy/) is available as free as well as commercial distribution with full SciPy stack for Windows, Linux and Mac.

Python (x,y): It is a free Python distribution with SciPy stack and Spyder IDE for Windows OS. (Downloadable from https://www.python-xy.github.io/)

Linux

Package managers of respective Linux distributions are used to install one or more packages in SciPy stack.

For Ubuntu

sudo apt-get install python-numpy

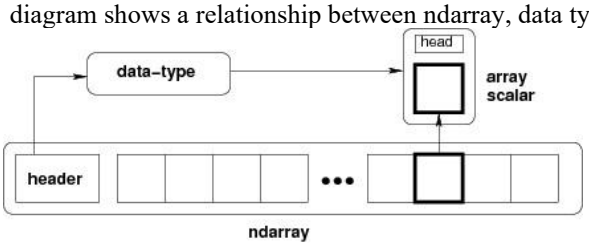
python-scipy python-matplotlibipythonipythonnotebook python-pandas

python-sympy python-nose

The most important object defined in NumPy is an N-dimensional array type called **ndarray**. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.

Every item in anndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called **dtype**).

Any item extracted from ndarray object (by slicing) is represented by a Python object of one of array scalar types. The following diagram shows a relationship between ndarray, data type object (dtype) and array scalar type −



Take a look at the following examples to understand better.

Example 1

```
importnumpyasnp
a =np.array([1,2,3])
print a
```

The output is as follows −

[1, 2, 3]

Example 2

```
# more than one dimensions
importnumpyasnp
a =np.array([[1,2],[3,4]])
print a
```

The output is as follows −

[[1, 2]
 [3, 4]]

Example 3

```
# minimum dimensions
importnumpyasnp
a =np.array([1,2,3,4,5],ndmin=2)
print a
```

The output is as follows −

```
[[1, 2, 3, 4, 5]]
```

Example 4

```
# dtype parameter
importnumpyasnp
a =np.array([1,2,3],dtype= complex)
print a
```

The output is as follows −

```
[ 1.+0.j,  2.+0.j,  3.+0.j]
```

NumPy supports a much greater variety of numerical types than Python does. The following table shows different scalar data types defined in NumPy.

Sr.No.	Data Types & Description
1	**bool_** Boolean (True or False) stored as a byte
2	**int_** Default integer type (same as C long; normally either int64 or int32)
3	**intc** Identical to C int (normally int32 or int64)
4	**intp** Integer used for indexing (same as C ssize_t; normally either int32 or int64)
5	**int8** Byte (-128 to 127)
6	**int16** Integer (-32768 to 32767)
7	**int32** Integer (-2147483648 to 2147483647)
8	**int64** Integer (-9223372036854775808 to 9223372036854775807)
9	**uint8** Unsigned integer (0 to 255)
10	**uint16** Unsigned integer (0 to 65535)

11	**uint32** Unsigned integer (0 to 4294967295)
12	**uint64** Unsigned integer (0 to 18446744073709551615)
13	**float_** Shorthand for float64
14	**float16** Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
15	**float32** Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
16	**float64** Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
17	**complex_** Shorthand for complex128
18	**complex64** Complex number, represented by two 32-bit floats (real and imaginary components)
19	**complex128** Complex number, represented by two 64-bit floats (real and imaginary components)

Example 1

```
# using array-scalar type
importnumpyasnp
dt =np.dtype(np.int32)
printdt
```

The output is as follows −

```
int32
```

Example 2

```
#int8, int16, int32, int64 can be replaced by equivalent string 'i1', 'i2','i4', etc.
importnumpyasnp

dt =np.dtype('i4')
printdt
```

The output is as follows −

```
int32
```

Example 3

```
# using endian notation
importnumpyasnp
dt =np.dtype('>i4')
printdt
```

The output is as follows −

```
>i4
```

The following examples show the use of structured data type. Here, the field name and the corresponding scalar data type is to be declared.

Example 4

```
# first create structured data type
importnumpyasnp
dt =np.dtype([('age',np.int8)])
printdt
```

The output is as follows −

```
[('age', 'i1')]
```

Example 5

```
# now apply it to ndarray object
importnumpyasnp

dt =np.dtype([('age',np.int8)])
a =np.array([(10,),(20,),(30,)],dtype= dt)
print a
```

The output is as follows −

```
[(10,) (20,) (30,)]
```

Example 6

```
# file name can be used to access content of age column
importnumpyasnp

dt =np.dtype([('age',np.int8)])
a =np.array([(10,),(20,),(30,)],dtype= dt)
print a['age']
```

The output is as follows −

```
[10 20 30]
```

Example 7

The following examples define a structured data type called **student** with a string field 'name', an **integer field** 'age' and a **float field** 'marks'. This dtype is applied to ndarray object.

```
importnumpyasnp
student =np.dtype([('name','S20'),('age','i1'),('marks','f4')])
printstudent
```

The output is as follows −

```
[('name', 'S20'), ('age', 'i1'), ('marks', '<f4')])
```

Example 8

```
importnumpyasnp

student =np.dtype([('name','S20'),('age','i1'),('marks','f4')])
a =np.array([('abc',21,50),('xyz',18,75)],dtype= student)
print a
```

The output is as follows −

```
[('abc', 21, 50.0), ('xyz', 18, 75.0)]
```

ndarray.shape

This array attribute returns a tuple consisting of array dimensions. It can also be used to resize the array.

Example 1

```
importnumpyasnp
a =np.array([[1,2,3],[4,5,6]])
printa.shape
```

The output is as follows −

(2, 3)

Example 2

```
# this resizes the ndarray
importnumpyasnp

a =np.array([[1,2,3],[4,5,6]])
a.shape=(3,2)
print a
```

The output is as follows −

```
[[1, 2]
 [3, 4]
 [5, 6]]
```

Example 3

NumPy also provides a reshape function to resize an array.

```
importnumpyasnp
a =np.array([[1,2,3],[4,5,6]])
b =a.reshape(3,2)
printb
```

The output is as follows −

```
[[1, 2]
 [3, 4]
 [5, 6]]
```

ndarray.ndim

This array attribute returns the number of array dimensions.

Example 1

```
# an array of evenly spaced numbers
importnumpyasnp
a =np.arange(24)
print a
```

The output is as follows −

```
[0 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 17 18 19 20 21 22 23]
```

Example 2

```
# this is one dimensional array
importnumpyasnp
a =np.arange(24)
a.ndim

# now reshape it
b =a.reshape(2,4,3)
printb
# b is having three dimensions
```

The output is as follows −

```
[[[ 0,  1,  2]
  [ 3,  4,  5]
  [ 6,  7,  8]
  [ 9, 10, 11]]
  [[12, 13, 14]
   [15, 16, 17]
   [18, 19, 20]
   [21, 22, 23]]]
```

numpy.itemsize

This array attribute returns the length of each element of array in bytes.

Example 1

```
# dtype of array is int8 (1 byte)
importnumpyasnp
x =np.array([1,2,3,4,5],dtype= np.int8)
printx.itemsize
```

The output is as follows −

1

Example 2

```
# dtype of array is now float32 (4 bytes)
importnumpyasnp
x =np.array([1,2,3,4,5],dtype= np.float32)
printx.itemsize
```

The output is as follows −

4

SOURCE CODE& OUTPUT:

// Execute the examples given above with different parameters and write their outputs

ASSESSMENT QUESTIONS:

1. Write advantages of using NumPy Library

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Lab Work#11

NAME:
ROLL NO:
BATCH:

AIM: Study of MatplotLib Library Functions in Python

OBJECTIVE: The objective of this lab work is:

1. Students will be able to list functions of MatplotLib.
2. Students will be able to configure MatplotLib.
3. Students will be able to apply functions of MatplotLib.

OUTCOMES: After completing the Lab Work#11 students will be able to:

1. Configure MatplotLib in their system.
2. Understand use of MatplotLibfunctions.
3. Apply MatplotLib Functions

PRE-REQUISITE:

1. Basic Syntax of Python

THEORY:

Installing an official release

Matplotlib and its dependencies are available as wheel packages for macOS, Windows and Linux distributions:

```
python -m pip install -U pip
python -m pip install -U matplotlib
```

If this command results in Matplotlib being compiled from source and there's trouble with the compilation, you can add `--prefer-binary` to select the newest version of Matplotlib for which there is a precompiled wheel for your OS and Python.

Note

The following backends work out of the box: Agg, ps, pdf, svg

Python is typically shipped with tk bindings which are used by TkAgg.

For support of other GUI frameworks, LaTeX rendering, saving animations and a larger selection of file formats, you need to install additional dependencies.

Although not required, we suggest also installing IPython for interactive use. To easily install a complete Scientific Python stack, see Scientific Python Distributions below.

Test data

The wheels (*.whl) on the PyPI download page do not contain test data or example code.

If you want to try the many demos that come in the Matplotlib source distribution, download the *.tar.gz file and look in the examples subdirectory.

To run the test suite:

- extract the lib/matplotlib/tests or lib/mpl_toolkits/tests directories from the source distribution.
- install test dependencies: pytest, MiKTeX, GhostScript, ffmpeg, avconv, ImageMagick, and Inkscape.
- run python -mpytest.

Third-party distributions of Matplotlib
Scientific Python Distributions

Anaconda and ActiveState are excellent choices that "just work" out of the box for Windows, macOS and common Linux platforms. WinPython is an option for Windows users. All of these distributions include Matplotlib and *lots* of other useful (data) science tools.

Linux: using your package manager

If you are on Linux, you might prefer to use your package manager. Matplotlib is packaged for almost every major Linux distribution.

- Debian / Ubuntu: sudo apt-get install python3-matplotlib
- Fedora: sudo dnf install python3-matplotlib
- Red Hat: sudo yum install python3-matplotlib
- Arch: sudo pacman -S python-matplotlib

Installing from source

If you are interested in contributing to Matplotlib development, running the latest source code, or just like to build everything yourself, it is not difficult to build Matplotlib from source. Grab the latest *tar.gz* release file from the PyPI files page, or if you want to develop Matplotlib or just need the latest bugfixed version, grab the latest git version, and see Install from source.

Matplotlib can be installed from the source directory with a simple

```
python -m pip install .
```

We provide a setup.cfg file which you can use to customize the build process. For example, which default backend to use, whether some of the optional libraries that Matplotlib ships with are installed, and so on. This file will be particularly useful to those packaging Matplotlib.

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

SOURCE CODE & OUTPUT:
// Execute the examples given at the links, write code and draw outputs obtained:

https://matplotlib.org/3.3.3/tutorials/index.html

1. Introductory – ALL
2. Intermediate- any 3
3. Advanced- any 1
4. 1 each from colors, text.

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Lab Work#12

NAME:
ROLL NO:
BATCH:

AIM: Study of use of Jupyter Notebook and Open CV

OBJECTIVE: The objective of this lab work is:

1. Students will be able to list use of Jupyter notebook.
2. Configure Jupyter notebook and execute simple functions
3. Students will be able to list uses of OpenCV

OUTCOMES: After completing the Lab Work#12 students will be able to:

1. Configure and use Jupyter Notebook.
2. Understand use of OpenCV and its applications.

PRE-REQUISITE:

1. Basic Syntax of Python

THEORY:

What is Jupyter Notebook?

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects. This article will walk you through how to use Jupyter Notebooks for data science projects and how to set it up on your local machine.
First, though: **what is a "notebook"?**

A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media. In other words: it's a single document where you can run code, display the output, and also add explanations, formulas, charts, and make your work more transparent, understandable, repeatable, and shareable.
Using Notebooks is now a major part of the data science workflow at companies across the globe. If your goal is to work with data, using a Notebook will speed up your workflow and make it easier to communicate and share your results.
Best of all, as part of the open source Project Jupyter, Jupyter Notebooks are completely free. You can download the software on its own, or as part of the Anaconda data science toolkit.

Although it is possible to use many different programming languages in Jupyter Notebooks, this article will focus on Python, as it is the most common use case. (Among R users, R Studio tends to be a more popular choice).

Installation

The easiest way for a beginner to get started with Jupyter Notebooks is by installing Anaconda.
Anaconda is the most widely used Python distribution for data science and comes pre-loaded with all the most popular libraries and tools.
Some of the biggest Python libraries included in Anaconda include NumPy, pandas, and Matplotlib, though the full 1000+ list is exhaustive.

Anaconda thus lets us hit the ground running with a fully stocked data science workshop without the hassle of managing countless installations or worrying about dependencies and OS-specific (read: Windows-specific) installation issues.

To get Anaconda, simply:

1. Download the latest version of Anaconda for Python 3.8.
2. Install Anaconda by following the instructions on the download page and/or in the executable.

If you are a more advanced user with Python already installed and prefer to manage your packages manually, you can just use pip:

```
pip3 install jupyter
```

OpenCV

OpenCV was started at Intel in 1999 by **Gary Bradsky**, and the first release came out in 2000. **Vadim Pisarevsky** joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge. Later, its active development continued under the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the project. OpenCV now supports a multitude of algorithms related to Computer Vision and Machine Learning and is expanding day by day.

OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development.

OpenCV-Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language.

OpenCV-Python

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

Python is a general purpose programming language started by **Guido van Rossum** that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability.

Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to

write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-Python makes use of **Numpy**, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

SOURCE CODE & OUTPUT:

// Configure Jupyter notebook and execute simple code on it
https://www.dataquest.io/blog/jupyter-notebook-tutorial/

// Installopencv at your system and try executing examples given at the link
https://docs.opencv.org/master/d0/d3d/tutorial_general_install.html

Appendix- A :Problem Statements

Problem Statement#1

FasterTrack is a company which manufactures analog wrist watches. There is a speed with which its internal gears need to move to keep watch times and speed up to date. FasterTrack recently launched its new model of sports watch- "Brodies". Usually, the life span of fasterTrack Watch is is more than 10 years, i.e. It should work for atleast 10 years. Unfortunately, a serious bug was detected while manufacturing of 'Brodies' model was about to complete,duetowhichthe**RPMincreasesordecreaseseveryyear**inthefollowing manner:

- multiply the number 323 to sum of the squares of the digits of theRPM,
- shiftthedigitsoftheRPMtotherightby1positoninacyclicway,andfinally
- extract the last two digits of the new number obtained and add to the result obtained in'a'.
- The result (c) is the newRPM.

This issue is that if the RPM increases by a certain limit, then the watch disk crashes, which is also dangerous to the person who is using it. **The maximum limit of the RPM of the watch is 8 times of the RPM**, i.e. if RPM of watch is 1000, then the maximum RPM can be 1000 * 8 = 8000.

This issue needs to be resolved before they release the 'Brodies' model of FasterTrack to the general public. You are given a contract to find out (based on the RPM) which watch will last for more than 10 years, and which will not.

Assume that the RPM is between 3524 and 8524 only, both inclusive.

Hint:

Task: In this program, the value of RPM is taken from the user using scanf() statement. You are required to write code that does the following:

1. Find out the sum of the squares of the digits of theRPM
2. Multiply the number obtained in (1) by323
3. Do a cyclic right shift of digits of the RPM i.e. if Number is 1234, after cyclic right shift, the number will be 4123. Thereafter, take the last two digits of the number obtained just now, and add it to the number obtained in point 2. Thus, obtaining a new RPMvalue.
4. Do these steps till the number of years are 10 or the watch has reached the maximum RPM

Example 1: Assume an **RPM of 5524**. So, the **maximum limit of RPM is 5524 * 8 = 44192**

RPM	Years	SSD	ASSD 323 *	B Cyclic Right Shift	A + last 2 digits of B	Result (NextRPM)
5524	1	$4^2 + 2^2 + 5^2 + 5^2$ = 16 + 4 + 25 + 25	70 * 323 = 22610	4552	22610 + 52	22662
22662	2	$2^2 + 6^2 + 6^2 + 2^2 + 2^2$ = 4 + 36 + 36 + 4 + 4	84 * 323 = 27132	22266	27132 + 66	27198
27198	3	$8^2 + 9^2 + 1^2 + 7^2 + 2^2$ = 64 + 81 + 1 + 49 + 4	199 * 323 = 64277	82719	64277 + 19	64296
Stop computing further as the maximum limit of RPM is 44192, and the RPM after 3 years is 64296						

****Courtesy: IITBombayX: CS101.1x Introduction to Computer Programming**

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Problem Statement#2

Steven Allan Spielberg is an American filmmaker. He is considered one of the founding pioneers of the New Hollywood era and one of the most popular directors and producers in film history.

With numerous films releasing everyday and millions of dollars going in and out of Steven's account he has lost track of his profits or losses.

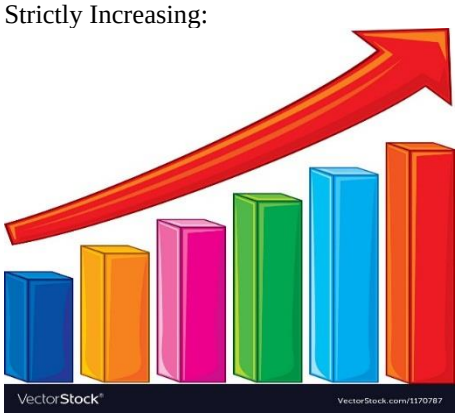
Recently
Ammanswami joined Steven's crew, Ammanswami an Indian Engineer perused his career in films due to strong inclination in this field. One fine day Steven and Ammanswami were at a script reading when Steven shared his poor money management and lack of analysis of his past finances.

Ammanswami had studied Python programming in his engineering curriculum and an idea striked him!!
He decided to use python programming to output the profit pa tterns of Speilberg's earning.

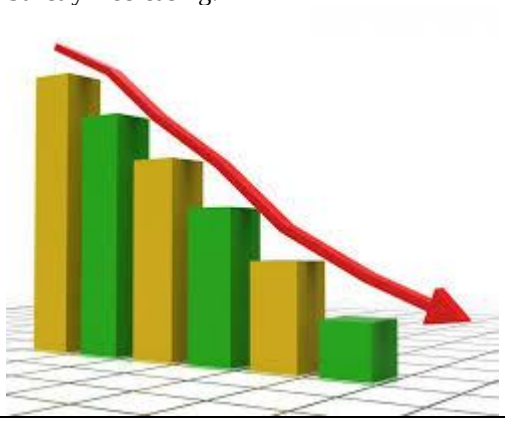
You are a good friend of Ammmanswami, you are now a CEO of a multinational firm. Help Amman solve this problem using python programming!!

Divide the profits into five patterns:

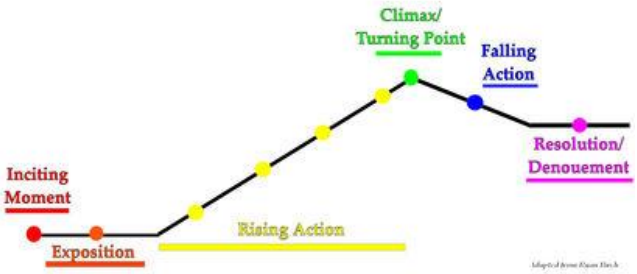
Strictly Increasing:



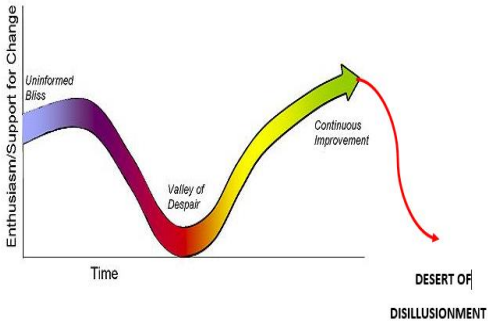
Strictly Decreasing:



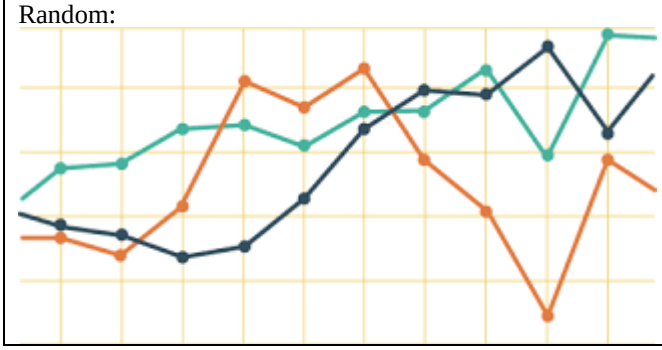
Mountain:



Valley:



Random:



You get free tickets to every Speilberg's movies' premiers!! YaY!

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Problem Statement#3

VeeruShahtrabuddhe, an old-fashioned schoolteacher recently started teaching class 9th students. According to him everyone should be good in Physics, Chemistry and Mathematics so he introduced a new method of calculating average and deciding the class topper.

He would collect the student-wise marks of Physics, Chemistry and Mathematics and will find their average. The student scoring highest will be declared topper and will get VeeruShahtrabuddhe's precious fountain pen.

But Veeru though brilliant in teaching but little weaker when it comes to numbers, he appoints Silencer- his favorite student to maintain the student wise marks and find topper.

You and your 2 other friends also study in this class and decide to create a python program for same.

Try outsmarting Veeru and his favorite Silencer!!

Problem Statement#4

South Africa is holding a one-day match series for four countries:
Total Matches: 5
Teams are:

1. India
2. South Africa
3. Pakistan
4. Bangladesh

These teams will compete and will have total 6 matches among them. The record of each match is scored in this way:

Match 1	India	Sachin	129
		Virat	100
	And similarly all other players of India (consider all players who got batting)		
	Pakistan	Irfan	34
		Shahid	56
		Imran	10
	And similarly all other players of India (consider all players who got batting)		

The match organizers are keeping the records digitally and they need to find following:

1. **Man of the Match**, Player who scored highest among two teams in amatch.
Output Example: The Man of the Match for Match 1 is Sachin from India with Score 126

2. **Man of the Tournament**, player who scored highest total in all matches in the tournament. Output Example: The Man of the Tournament is Virat from India with score378

Write a python code to simplify the above solution and present the output.

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Problem Statement#5

India is experiencing a climate shift and there is huge imbalance in the rainfall. Marathwada had been a heavily drought affected area since last 5 years. You have seen a havoc in the condition of farmers. The thought always stays at the back of your mind as to somehow contribute in handling the situation.
You recently came across the rainfall details of several cities for past few years and decided to work on them as per below details, so that later you can replicate the same for your region.

We have a list of annual rainfall recordings of cities. Each element in the list is of the form (c,r) where c is the city and r is the annual rainfall for a particular year. The list may have multiple entries for the same city, corresponding to rainfall recordings in different years.

Write a Python function rainaverage(l) that takes as input a list of rainfall recordings and computes the avarage rainfall for each city. The output should be a list of pairs (c,ar) where c is the city and ar is the average rainfall for this city among the recordings in the input list. Note that ar should be of type float. The output should be sorted in dictionary order with respect to the city name.

Here are some examples to show how rainaverage(l) should work.

```
>>>rainaverage([(1,2),(1,3),(2,3),(1,1),(3,8)])

[(1, 2.0), (2, 3.0), (3, 8.0)]


>>> rainaverage([('Bombay',848),('Madras',103),('Bombay',923),('Bangalore',201),('Madras',128)])

[('Bangalore', 201.0), ('Bombay', 885.5), ('Madras', 115.5)]
```

Courtesy: NPTEL Course **Programming, Data Structures and Algorithms using Python** Jan 2019

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Problem Statement#6

John Snow is a very ambitious teenager and aspires to become the King of North someday. But his father Ned Stark is a very strict man and wants John to learn all traits before he becomes the King.

John is very weak in Mathematics, but he loves programming. Whenever his Mathematics teacher- Night King gives some homework, he would try to develop a program for same. Currently he is studying the Polynomials in mathematics and is stuck on the problem of polynomial arithmetic, you are John's best friend Samwell Talley both of you are now trying to solve following problem and trying to create a python program

Consider polynomials in a single variable x with integer coefficients: for instance, $3x^4 - 17x^2 - 3x + 5$. Each term of the polynomial can be represented as a pair of integers (coefficient, exponent). The polynomial itself is then a list of such pairs.

We have the following constraints to guarantee that each polynomial has a unique representation:

- Terms are sorted in descending order of exponent
- No term has a zero coefficient
- No two terms have the same exponent
- Exponents are always nonnegative

For example, the polynomial introduced earlier is represented as

[(3,4),(-17,2),(-3,1),(5,0)]

The zero polynomial, 0, is represented as the empty list [], since it has no terms with nonzero coefficients.

Write Python functions for the following operations:

addpoly(p1,p2)
multpoly(p1,p2)
that add and multiply two polynomials, respectively.

You may assume that the inputs to these functions follow the representation given above. Correspondingly, the outputs from these

functions should also obey the same constraints.

Hint: You are not restricted to writing just the two functions asked for. You can write auxiliary functions to "clean up" polynomials – e.g., remove zero coefficient terms, combine like terms, sort by exponent etc. Build a library of functions that can be combined to achieve the desired format.

You may also want to convert the list representation to a dictionary representation and manipulate the dictionary representation, and then convert back.

Some examples:

```
>>>addpoly([(4,3),(3,0)],[(-4,3),(2,1)])
  [(2, 1),(3, 0)]
```

Explanation: $(4x^3 + 3) + (-4x^3 + 2x) = 2x + 3$

```
>>>addpoly([(2,1)],[(-2,1)])
  []
```

Explanation: $2x + (-2x) = 0$

```
>>>multpoly([(1,1),(-1,0)],[(1,2),(1,1),(1,0)])
  [(1, 3),(-1, 0)]
```

Explanation: $(x - 1) * (x^2 + x + 1) = x^3 - 1$

Courtesy: NPTEL online course **Programming, data structures and algorithms using Python

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION & SUBMISSION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Problem Statement#8

Your long lost uncle Mukadar ka Sikander has spent his life in vegas, recentlyhewonafortuneanddecidedtoreturntoIndia.Afterallthe currency exchange he ran to deposit the amount in Bank and stoodin the queue for almost 4hours.

Tireduncledeclaredangrilythathewillopenhisownbankand name that as "Sikander'sBank".

He has seen lot of development, computerization in USA so decides to have a software prototype build to see the working. You are the only Engineer in family so your uncle approaches you and asks you for the favor and in turn he will name you heir in his fortune!!

Lucky you!

Now quick start with the program to impress Mukadar ka Sikander

	G.S. MANDAL'S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**
	LAB WORK INSTRUCTION & SUBMISSION SHEET
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CLASS: II YEAR B. TECH PART: 1 (2021-22)	**LAB:**	**SUBJECT:** CSE 226 Advanced Python Programming

Problem Statement#9

We all know "Sherlock Holmes" – the British Crime Detective.
Sherlock is offering an internship position to engineering college students who can apply technology to solve his cases. You are a BIG Sherlock Fan and are immediately excited to see the internship flyer.

But! Its Sherlock!! How can he make things simple? so it is required to aspiring interns to solve a case for sherlock. The case is connected to Sherlock's Brother Mycroft. Mycroft is missing from his office and has left a diary to the post box of Sherlock.
The diary is scribbled with jumbled words and is about 100 pages in all.

Sherlock found the key- in that diary each line has some information but words on each line are jumbled and the characters of each words are also jumbled.

Sherlock gave this puzzle in the internship add and asked all the aspiring detectives to give a technology solution for this case. You are Second Year student and you recently studied Python File Handling you decided to create program of file handling to decode Mycroft's diary.

Time to get **"SHERLOCKED"**