THE CODE JOURNAL
PRESENTS

PYTHON SOLVED SAMPLE 2020

ALSO AVAILABLE FOR

PHP

MOBILE APPLICATIO DEVELOPMENT

WWW.THECODEJOURNAL.IN

Python Solved Sample 2020

Instructions:

(1) All questions are compulsory.

(2) Illustrate your answers with neat sketches wherever necessary.

(3) Figures to the right indicate full marks.

(4) Assume suitable data if necessary.

(5) Preferably, write the answers in sequential order.

Q1. Attempt any FIVE of the following. (10 Marks)

a) Name different modes of Python.

Answer: We can develop a python program in 2 different styles.

- Interactive Mode and
- Batch Mode.

Interactive Mode:

Interactive mode is a command line shell. If we write a python program in the command line shell.

Typically the interactive mode is used to test the features of the python, or to run a smaller script that may not be reusable.

The >>> indicates that the Python shell is ready to execute and send your commands to the Python interpreter. The result is immediately displayed on the Python shell as soon as the Python interpreter interprets the command.

To run your Python statements, just type them and hit the enter key. You will get the results immediately, unlike in script mode. For example, to print the text "Hello World", we can type the following:

```
>>> print("Hello World")
Hello World
>>>
```

Python Solved Sample 2020

Script Mode

If you need to write a long piece of Python code or your Python script spans multiple files, interactive mode is not recommended. Script mode is the way to go in such cases. In script mode, You write your code in a text file then save it with a .py extension which stands for "Python". Note that you can use any text editor for this, including Sublime, Atom, notepad++, etc.

b) List identity operators.

Answer: Identity operators compare the memory locations of two objects. There are two Identity operators as explained below −

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here is results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here is not results in 1 if id(x) is not equal to id(y).

c) Describe Dictionary.

Answer: Dictionary in Python is an unordered collection of data values, used to store data values like a map,

In Python, a Dictionary can be created by placing sequence of elements within curly **{}** braces, separated by ‘comma’. Dictionary holds a pair of values, one being the Key and the other corresponding pair element being its **Key:value**. Values in a dictionary can be of any datatype and can be duplicated, whereas keys can’t be repeated and must be *immutable*.

```
Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}
print("\nDictionary with the use of Integer Keys: ")
print(Dict)
```

d) State use of namespace in Python.

Answer: A namespace is a simple system to control the names in a program. It ensures that names are unique and won’t lead to any conflict.

Python implements namespaces in the form of dictionaries. It maintains a name-to-object mapping where names act as keys and the objects as values. Multiple namespaces may have the same name but pointing to a different variable..

- **Local Namespace**
  This namespace covers the local names inside a function. Python creates this namespace for every function called in a program. It remains active until the function returns.
- **Global Namespace**
  This namespace covers the names from various imported modules used in a project. Python creates this namespace for every module included in your program. It’ll last until the program ends.
- **Built-in Namespace**
  This namespace covers the built-in functions and built-in exception names. Python creates it as the interpreter starts and keeps it until you exit.

e) List different Object Oriented features supported by Python.

Answer: Python is also an object-oriented language since its beginning. Python is an object-oriented programming language. It allows us to develop applications using an Object Oriented approach. In Python, we can easily create and use classes and objects.

Major principles of object-oriented programming system are given below.

- Object
- Class
- Method
- Inheritance
- Polymorphism
- Data Abstraction

- Encapsulation

g) Describe Python Interpreter.

Answer: An interpreter is a program that reads and executes code. This includes source code, pre-compiled code, and scripts. Common interpreters include Perl, Python, and Ruby interpreters, which execute Perl, Python, and Ruby code respectively.

The Python interpreter is the application that runs your python script. Read the script line by line and converts that script into python byte code, and then writes the byte code into a pyc file. If your application has multiple files it creates a pyc file for every .py file. It is at this stage that syntax errors are generated.

Q.2) Attempt any THREE of the following. (12 Marks)

a) Explain two Membership and two logical operators in python with appropriate examples.

Answer: Python's membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below −

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Example

```
#!/usr/bin/python

a = 10
b = 20
list = [1, 2, 3, 4, 5 ];

if ( a in list ):
   print "Line 1 - a is available in the given list"
else:
   print "Line 1 - a is not available in the given list"

if ( b not in list ):
   print "Line 2 - b is not available in the given list"
else:
   print "Line 2 - b is available in the given list"

a = 2
if ( a in list ):
   print "Line 3 - a is available in the given list"
else:
   print "Line 3 - a is not available in the given list"
```

When you execute the above program it produces the following result −

```
Line 1 - a is not available in the given list
Line 2 - b is not available in the given list
Line 3 - a is available in the given list
```

There are following **logical operators** supported by Python language. Assume variable a holds 10 and variable b holds 20 then −

Operator	**Description**	**Example**
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

b) Describe any four methods of lists in Python.

Answer: The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

The methods of list are as follows -:

1. cmp(list1,list2) -: Comapre elements of both lists

Syntax

```
cmp(list1, list2)
```

Parameters

- **list1** − This is the first list to be compared.
- **list2** − This is the second list to be compared.

```
list1, list2 = [123, 'xyz'], [456, 'abc']
print cmp(list1, list2)
print cmp(list2, list1)
list3 = list2 + [786];
print cmp(list2, list3)
```

When we run above program, it produces following result −

```
-1
1
-1
```

2. Python list method len() returns the number of elements in the *list*.

Syntax `len(list)`

Parameters

- **list** − This is a list for which number of elements to be counted.

Example

```
#!/usr/bin/python

list1, list2 = [123, 'xyz', 'zara'], [456, 'abc']
print "First list length : ", len(list1)
print "Second list length : ", len(list2)
```

When we run above program, it produces following result −

```
First list length :  3
Second list length :  2
```

3. Python list method max returns the elements from the *list* with maximum value.

Syntax

max(list)

Parameters

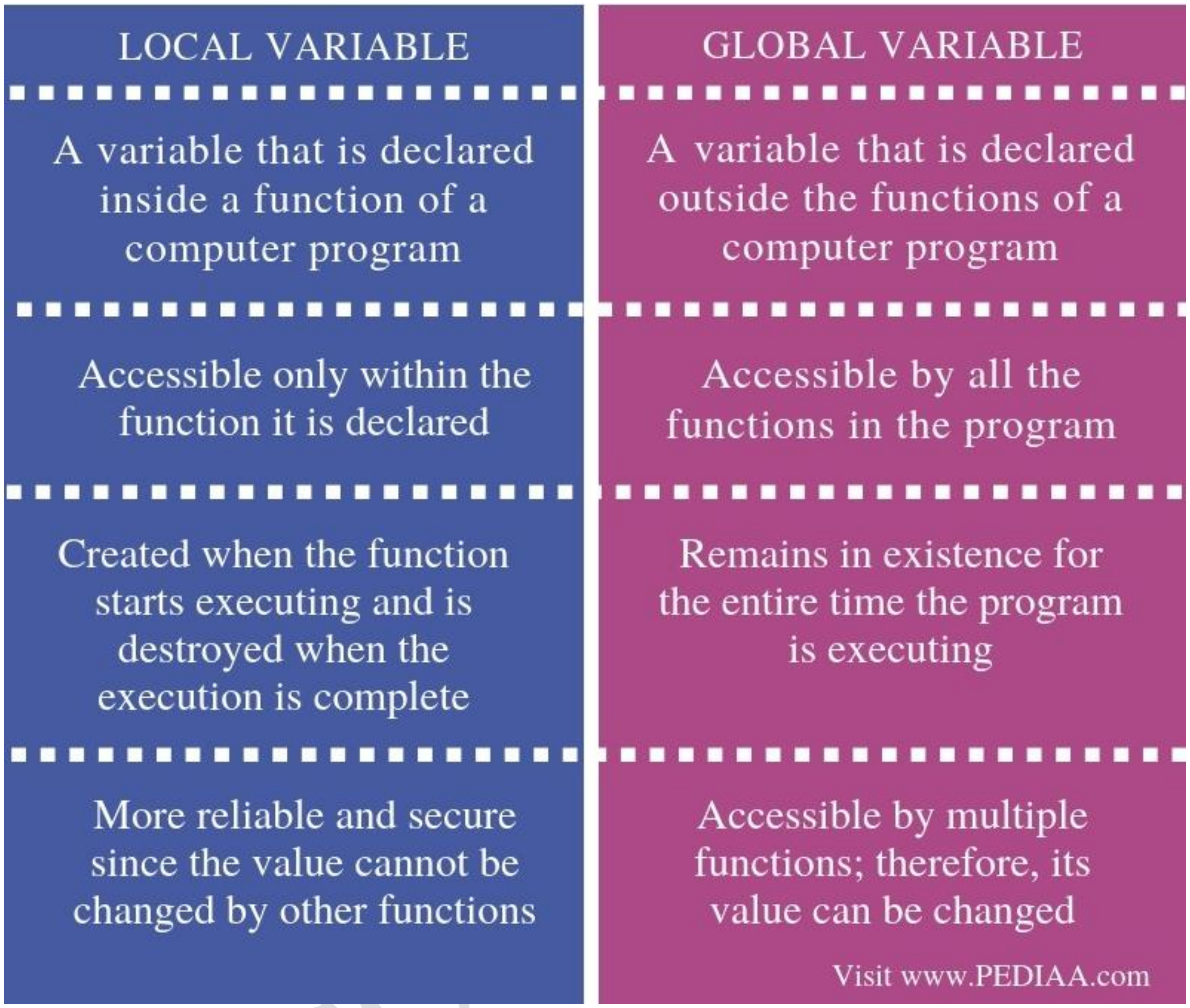
- **list** − This is a list from which max valued element to be returned.

Example

```
#!/usr/bin/python

list1, list2 = [123, 'xyz', 'zara', 'abc'], [456, 700, 200]
print "Max value element : ", max(list1)
print "Max value element : ", max(list2)

When we run above program, it produces following result −

Max value element :  zara
Max value element :  700
```

c) Comparing between local and global variable.

Answer:



LOCAL VARIABLE	GLOBAL VARIABLE
A variable that is declared inside a function of a computer program	A variable that is declared outside the functions of a computer program
Accessible only within the function it is declared	Accessible by all the functions in the program
Created when the function starts executing and is destroyed when the execution is complete	Remains in existence for the entire time the program is executing
More reliable and secure since the value cannot be changed by other functions	Accessible by multiple functions; therefore, its value can be changed

Visit www.PEDIAA.com

d) Write a python program to print Fibonacci series up to n terms.

Answer:

```
nterms = int(input("How many terms? "))


# first two terms
n1, n2 = 0, 1
count = 0


# check if the number of terms is valid
if nterms <= 0:
   print("Please enter a positive integer")
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
else:
   print("Fibonacci sequence:")
   while count < nterms:
       print(n1)
       nth = n1 + n2
       # update values
       n1 = n2
       n2 = nth
       count += 1
```

Q.3) Attempt any THREE of the following. (12 Marks)

a) Write a program to input any two tuples and interchange the tuple variable.

Answer:

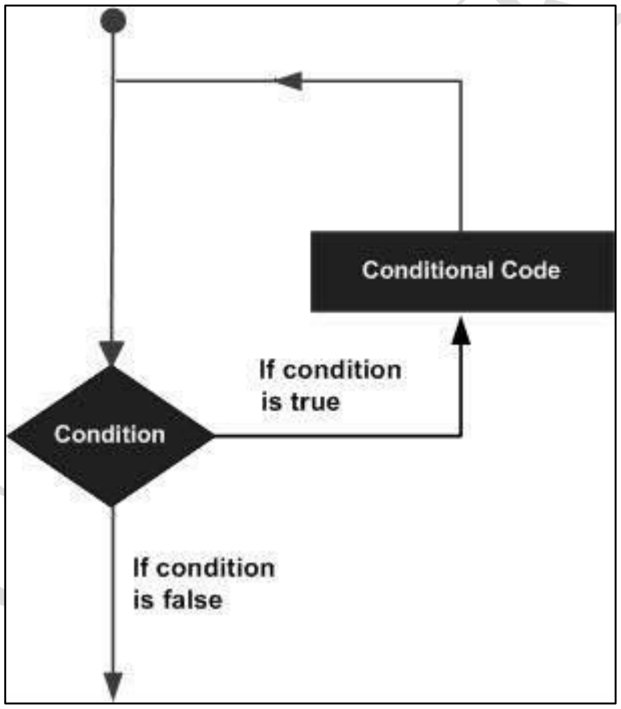
```
t1 = tuple( )
n = input(“Total number of values m first tuple”)
for i in range (n):
a = input(“Enter elements”)
 t2 = t2 + (a, )
print “First tuple”
 print t1
 print “Second tuple”
 print t2 t1, t2 = t2, t1
 print “After swapping”
 print “First tuple”
print t1
 print “Second tuple”
print t2
```

b) Explain different loops available in python with suitable examples.

Answer: In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement –



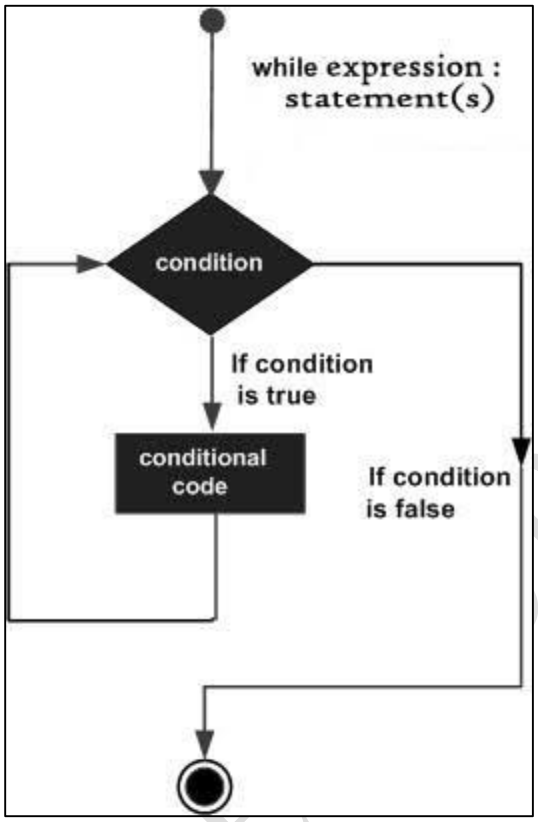
1. While Loop

A **while** loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax

The syntax of a **while** loop in Python programming language is −

```
while expression:
    statement(s)
```



Example

```
count = 0
while (count < 9):
   print 'The count is:', count
   count = count + 1

print "Good bye!"
```

2. For Loop

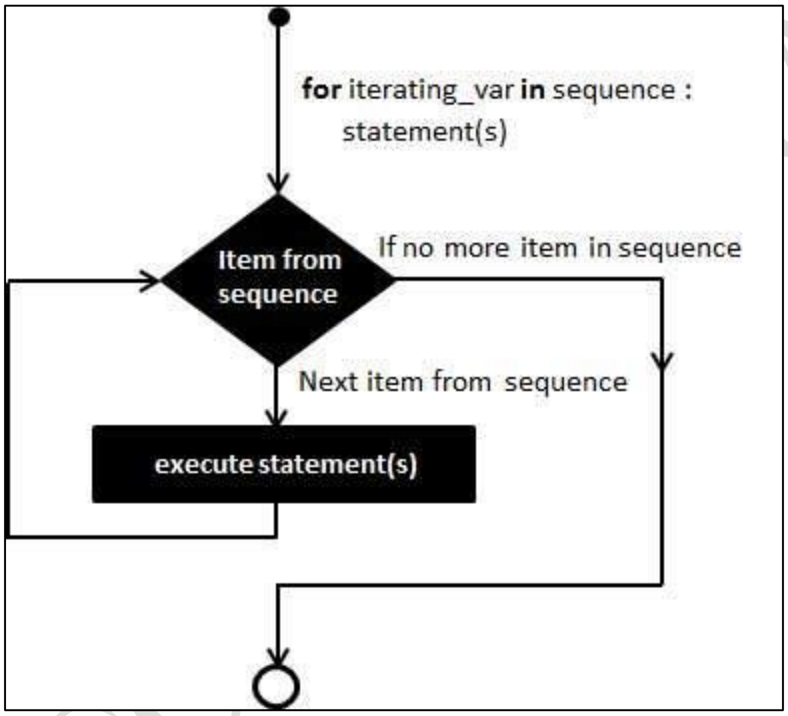
It has the ability to iterate over the items of any sequence, such as a list or a string.

Syntax

```
for iterating_var in sequence:
   statements(s)
```

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable iterating_var. Next, the statements block is executed. Each item in the list is assigned to iterating_var, and the statement(s) block is executed until the entire sequence is exhausted.

Flow Diagram



Example

```
#!/usr/bin/python

for letter in 'Python':     # First Example
   print 'Current Letter :', letter

fruits = ['banana', 'apple',  'mango']
for fruit in fruits:        # Second Example
   print 'Current fruit :', fruit

print "Good bye!"
```

3. Nested Loops

Python programming language allows to use one loop inside another loop. Following section shows few examples to illustrate the concept.

Syntax

```
for iterating_var in sequence:
   for iterating_var in sequence:
      statements(s)
   statements(s)
```

The syntax for a **nested while loop** statement in Python programming language is as follows –

```
while expression:
   while expression:
      statement(s)
   statement(s)
```

Example

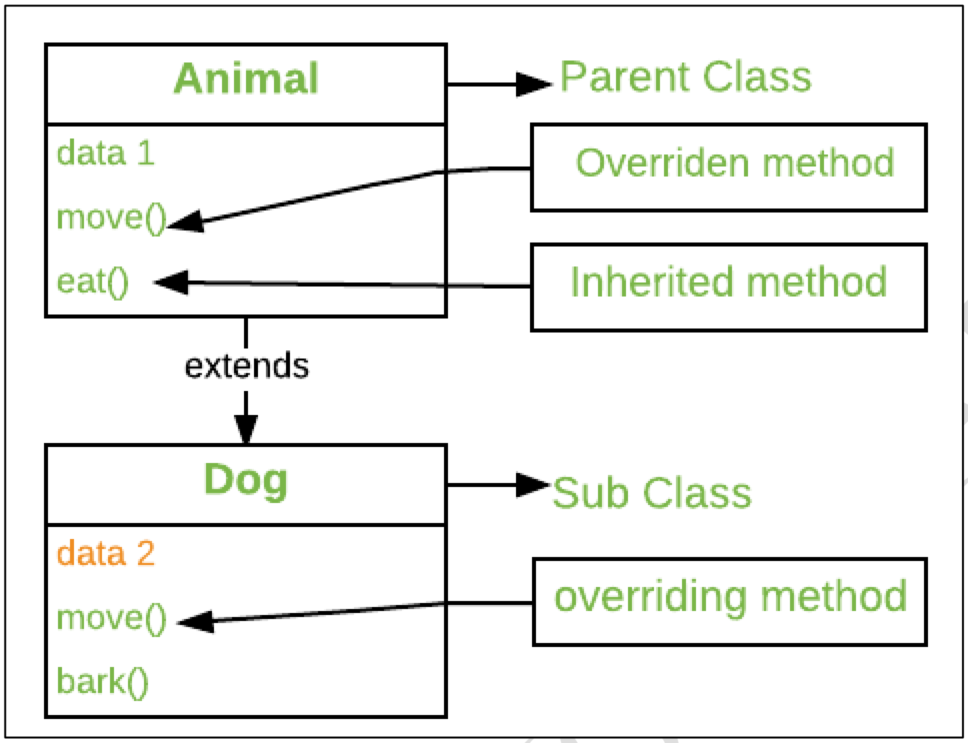
The following program uses a nested for loop to find the prime numbers from 2 to 100 –

```
#!/usr/bin/python

i = 2
while(i < 100):
   j = 2
   while(j <= (i/j)):
      if not(i%j): break
      j = j + 1
   if (j > i/j) : print i, " is prime"
   i = i + 1

print "Good bye!"
```

d) Illustrate the use of method overriding? Explain with example.

Answer: Method overriding is an ability of any object-oriented programming language that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. When a method in a subclass has the same name, same parameters or signature and same return type (or sub-type) as a method in its super-class, then the method in the subclass is said to **override** the method in the super-class.

Python Solved Sample 2020



Example -:

```
# Python program to demonstrate
# method overriding


# Defining parent class
class Parent():

    # Constructor
    def __init__(self):
        self.value = "Inside Parent"

    # Parent's show method
    def show(self):
        print(self.value)

# Defining child class
class Child(Parent):

    # Constructor
    def __init__(self):
        self.value = "Inside Child"

    # Child's show method
```

```
   def show(self):
      print(self.value)


# Driver's code
obj1 = Parent()
obj2 = Child()

obj1.show()

obj2.show()
```

Output:

```
Inside Parent
Inside Child
```

Q.4) Attempt any THREE of the following. (12 Marks)

a) Use of any four methods of tuple in python?

Answer: A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

1. LEN() Function -:

Python tuple method **len()** returns the number of elements in the tuple.

Syntax

```
len(tuple)
```

Example -:

```
tuple1, tuple2 = (123, 'xyz', 'zara'), (456, 'abc')
print "First tuple length : ", len(tuple1)
print "Second tuple length : ", len(tuple2)
```

2. MAX() Function -:

Python Solved Sample 2020

Python tuple method **max()** returns the elements from the tuple with maximum value.

Syntax

```
max(tuple)
```

Example -:

```
tuple1, tuple2 = (123, 'xyz', 'zara', 'abc'), (456, 700, 200)
print "Max value element : ", max(tuple1)
print "Max value element : ", max(tuple2)
```

3. MIN() Function -:

Python tuple method **min()** returns the elements from the tuple with minimum value.

Syntax

```
min(tuple)
```

Example -:

```
tuple1, tuple2 = (123, 'xyz', 'zara', 'abc'), (456, 700, 200)
print "min value element : ", min(tuple1)
print "min value element : ", min(tuple2)
```

4. CMP() Function -:

Python tuple method **cmp()** compares elements of two tuples.

Syntax

```
cmp(tuple1, tuple2)
```

Example -:

```
tuple1, tuple2 = (123, 'xyz'), (456, 'abc')
print cmp(tuple1, tuple2)
print cmp(tuple2, tuple1)
tuple3 = tuple2 + (786,);
print cmp(tuple2, tuple3)
```

b) Write a python program to read contents of first.txt file and write same content in second.txt file.

```
with open("test.txt") as f:
    with open("out.txt", "w") as f1:
        for line in f:
            f1.write(line)
```

c) Show how try…except blocks is used for exception handling in Python with example

Answer: The try block lets you test a block of code for errors. The except block lets you handle the error.

When an error occurs, or exception as we call it, Python will normally stop and generate an error message. These exceptions can be handled using the try statement:

How try() works?

- First **try** clause is executed i.e. the code between **try** and **except** clause.
- If there is no exception, then only **try** clause will run, **except** clause is finished.
- If any exception occured, **try** clause will be skipped and **except** clause will run.
- If any exception occurs, but the **except** clause within the code doesn't handle it, it is passed on to the outer **try** statements. If the exception left unhandled, then the execution stops.
- A **try** statement can have more than one **except** clause.

```
try:
  print(x)
except NameError:
  print("Variable x is not defined")
except:
  print("Something else went wrong")
```

Another Example -:

```
def divide(x, y):
    try:
        # Floor Division : Gives only Fractional Part as Answer
        result = x // y
        print("Yeah ! Your answer is :", result)
    except ZeroDivisionError:
        print("Sorry ! You are dividing by zero ")
```

d) Write the output for the following if the variable fruit='banana':

>>>fruit[:3]

>>>fruit[3:]

>>>fruit[3:3]

>>>fruit[:]

Answer: The output for the following code are as follows -:

>>>fruit[:3]	ban
>>>fruit[3:]	ana
>>>fruit[3:3]	n
>>>fruit[:]	banana

Q.5) Attempt any TWO of the following. (12 Marks)

a) Determine various data types available in Python with example.

Answer: A variable can hold different types of values. For example, a person's name must be stored as a string whereas its id must be stored as an integer.

Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.

1. Number	2. String
3. List	4. Tuple
5. Dictionary	

Python Solved Sample 2020

Numbers

Number stores numeric values. Python creates Number objects when a number is assigned to a variable. For example;

a = 3 , b = 5 #a and b are number objects

Python supports 4 types of numeric data.

```
int (signed integers like 10, 2, 29, etc.)

long (long integers used for a higher range of values like 908090800L, -0x1929292L, etc.)

float (float is used to store floating point numbers like 1.9, 9.902, 15.2, etc.)

complex (complex numbers like 2.14j, 2.0 + 2.3j, etc.)
```

String

The string can be defined as the sequence of characters represented in the quotation marks.

The following example illustrates the string handling in python.

```
str1 = 'hello javatpoint' #string str1

str2 = ' how are you' #string str2

print (str1[0:2]) #printing first two character using slice operator

print (str1[4]) #printing 4th character of the string

print (str1*2) #printing the string twice

print (str1 + str2) #printing the concatenation of str1 and str2
```

Output:

```
he
o
hello javatpointhello javatpoint
hello javatpoint how are you
```

List

Lists are similar to arrays in C. However; the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].

We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (*) works with the list in the same way as they were working with the strings.

example.

```
l  = [1, "hi", "python", 2]

print (l[3:]);

print (l[0:2]);

print (l);

print (l + l);

print (l * 3);
```

Output:

```
[2]
[1, 'hi']
[1, 'hi', 'python', 2]
[1, 'hi', 'python', 2, 1, 'hi', 'python', 2]
```

```
[1, 'hi', 'python', 2, 1, 'hi', 'python', 2, 1, 'hi', 'python', 2]
```

Tuple

A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses ().

A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.

Example -:

```
t  = ("hi", "python", 2)

print (t[1:]);

print (t[0:1]);

print (t);

print (t + t);

print (t * 3);

print (type(t))

t[2] = "hi";
```

Output:

```
('python', 2)
('hi',)
('hi', 'python', 2)
('hi', 'python', 2, 'hi', 'python', 2)
('hi', 'python', 2, 'hi', 'python', 2, 'hi', 'python', 2)
<type 'tuple'>
Traceback (most recent call last):
  File "main.py", line 8, in <module>
    t[2] = "hi";
TypeError: 'tuple' object does not support item assignment
```

Dictionary

Dictionary is an ordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. Key can hold any primitive data type whereas value is an arbitrary Python object.

The items in the dictionary are separated with the comma and enclosed in the curly braces {}.

example. -:

```
d = {1:'Jimmy', 2:'Alex', 3:'john', 4:'mike'};

print("1st name is "+d[1]);

print("2nd name is "+ d[4]);

print (d);

print (d.keys());

print (d.values());
```

Output:

```
1st name is Jimmy
2nd name is mike
{1: 'Jimmy', 2: 'Alex', 3: 'john', 4: 'mike'}
[1, 2, 3, 4]
['Jimmy', 'Alex', 'john', 'mike']
```

b) Write a python program to calculate factorial of given number using function.

Answer: The factorial of a number is the product of all the integers from 1 to that number.

For example, the factorial of 6 is 1*2*3*4*5*6 = 720. Factorial is not defined for negative numbers and the factorial of zero is one, 0! = 1.

```
def recur_factorial(n):
   if n == 1:
       return n
   else:
       return n*recur_factorial(n-1)

num = 7

# check if the number is negative
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   print("The factorial of", num, "is", recur_factorial(num))
```

Output

```
The factorial of 7 is 5040
```

Python Solved Sample 2020

c) Show the output for the following:

1. >>> a=[1,2,3] >>>b=[4,5,6] >>> c=a+b	2. >>>[1,2,3]*3	3. >>>t=['a','b','c','d','e','f'] >>>t[1:3]=['x','y'] >>>print t

Answer:

1. [1,2,3,4,5,6]	2. [1,2,3,1,2,3,1,2,3]	3. ['a','x','y','d','e','f']

Q.6) Attempt any TWO of the following. (12 Marks)

a) Describe Set in python with suitable examples.

Answer: A Set is an unordered collection data type that is iterable, mutable and has no duplicate elements. Python's set class represents the mathematical notion of a set. The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set. This is based on a data structure known as a hash table.

If Multiple values are present at the same index position, then the value is appended to that index position, to form a Linked List. In, Python Sets are implemented using dictionary with dummy variables, where key beings the members set with greater optimizations to the time complexity.

```
Set = set(["a", "b", "c"])

print("Set: ")
print(Set)

# Adding element to the set
Set.add("d")
```

```
print("\nSet after adding: ")

print(Set)
```

Output:

```
Set:
set(['a', 'c', 'b'])

Set after adding:
set(['a', 'c', 'b', 'd'])
```

b) Illustrate class inheritance in Python with an example.

Answer: Inheritance enable us to define a class that takes all the functionality from parent class and allows us to add more. In this article, you will learn to use inheritance in Python.

It refers to defining a new class with little or no modification to an existing class. The new class is called **derived (or child) class** and the one from which it inherits is called the **base (or parent) class**.

Python Inheritance Syntax

```
class BaseClass:
  Body of base class
class DerivedClass(BaseClass):
  Body of derived class
```

Example -:

```
class Polygon:
    def __init__(self, no_of_sides):
        self.n = no_of_sides
        self.sides = [0 for i in range(no_of_sides)]

    def inputSides(self):
        self.sides = [float(input("Enter side "+str(i+1)+" : ")) for i in range(self.n)]

    def dispSides(self):
        for i in range(self.n):
            print("Side",i+1,"is",self.sides[i])
```

c) Design a class Employee with data members: name, department and salary. Create suitable methods for reading and printing employee information.

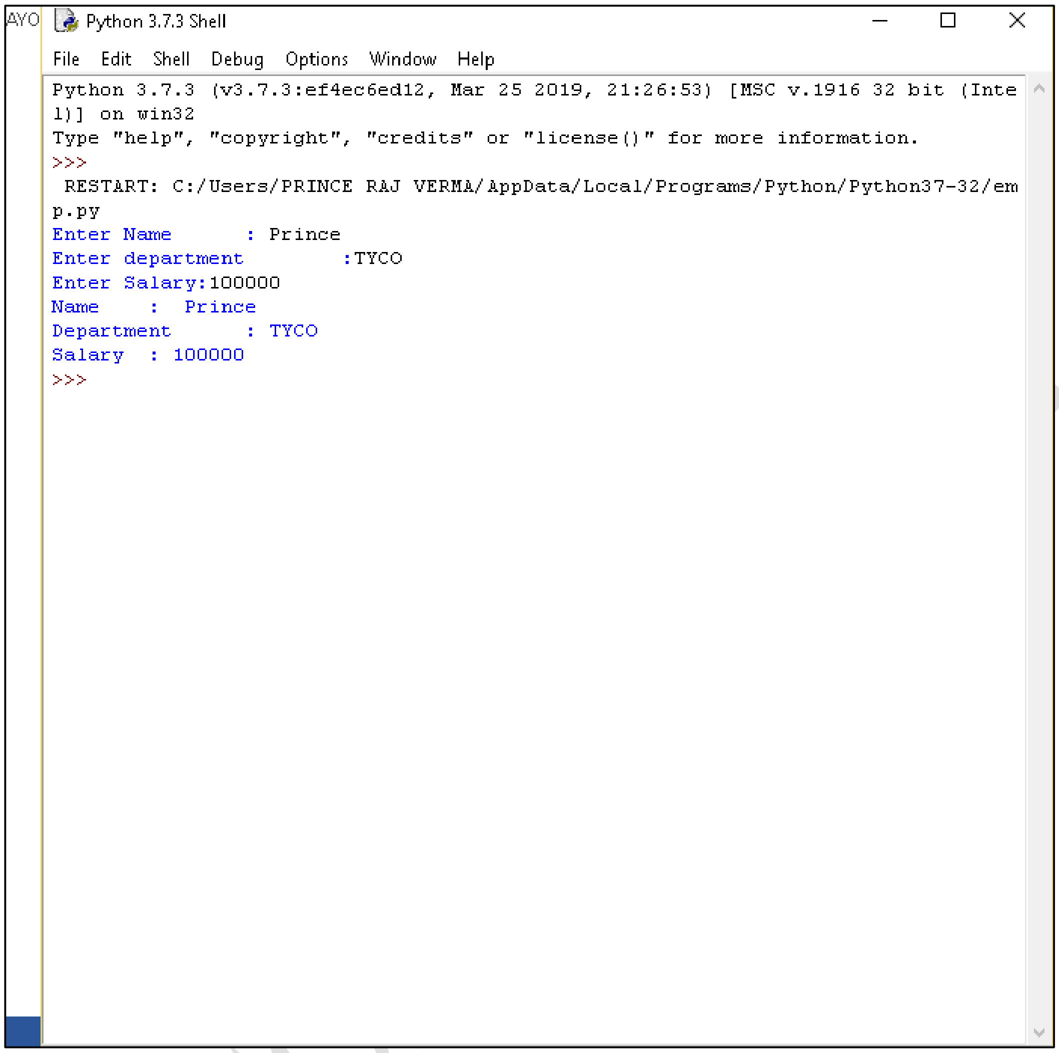
Answer:

```
class Employee:

     __name=""

    __dep=""

     __salary=0

    def setData(self):

         self.__name = input("Enter Name\t:")

        self.__dep = input("Enter department\t:")
```

Python Solved Sample 2020

```
self.__salary = int(input("Enter Salary:"))

    def showData(self):

         print("Name\t:", self.__name)

         print("Department\t:", self.__dep)

        print("Salary\t:", self.__salary)

def main():

    #Employee Object

    emp=Employee()

    emp.setData()

    emp.showData()



if __name__=="__main__":

    main()
```

Output -:



```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
l)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/PRINCE RAJ VERMA/AppData/Local/Programs/Python/Python37-32/em
p.py
Enter Name       : Prince
Enter department        :TYCO
Enter Salary:100000
Name     :  Prince
Department       : TYCO
Salary  : 100000
>>>
```