Royal Education Society's

College of Computer Science and Information Technology, Latur

Department of Computer Science

Academic Year(2022-23)

Choice based credit System(CBCS revised) Class: **B.Sc.(CS)-TY SEM-V**
Name of Paper: **Python BCS(502)** Prepared By: **Mr. R.S. Jadhav**

Instructions to the candidates:
1. All questions are Compulsory.
2. Figures to the right indicate full marks.
3. Assume suitable data, if required.

Q.1 Attempt any FIVE of the following (3 Marks each) 15

a) What is a boolean in Python?

b) What are some of the most commonly used built-in modules with examples in Python?

c) Explain variables with examples.

d) What are errors in Python? Explain with example.

e) What is dynamically typed language ?

f) What is pass keyword in Python ?

g) Write a program for database connectivity in Python with MySQL

Q. 2 Attempt any three of the following (5 Marks each) 15

a) What are the types inheritance in Python? Explain.

b) What are the common built-in data types in Python?

c) What are modules and packages in Python? Explain.

d) What is break, continue and pass in Python? Explain.

e) What is polymorphism ? Explain with example.

Q. 3 Attempt any three of the following (5 Marks each) 15

a) What is the difference between Python Arrays and lists?

b) What is Python? What are the benefits of using Python ?

c) How do you create class and object in Python?

d) Explain hybrid inheritance with example.

e) Explain modifying of string concept with examples.

Q. 4 Attempt any three of the following (5 Marks each) 15

a) What is set data type? Explain basic methods used umder set data type.

b) Explain multilevel inheritance with example.

c) Write a program for passing query to MySQL in Python.

d) Explain exception raising with example.

e) What is pickling and unpickling of data? Explain with example.

Q .5 Short notes on any three of the following (5 Marks each) 15

a) Comments in Python

b) Slicing in strings

c) Method overriding

d) File Handling in Python

e) Introduction to flask.

Q.1 Attempt any FIVE of the following (3 Marks each)

a) What is a boolean in Python?

Answer:

Python Boolean:

1. Booleans represent one of two values: True or False
2. In programming you often need to know if an expression is True or False.
3. You can evaluate any expression in Python, and get one of two answers, True or False.
4. When you compare two values, the expression is evaluated and Python returns the Boolean answer.

For ex:-

Code:-

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

Output:-

```
True
False
False
```

- The **bool()** function allows you to evaluate any value, and give you True or False in return,

For ex:-

```
print(bool("Hello"))
print(bool(15))
```

Output:-

```
True
True
```

b) What are some of the most commonly used built-in modules with examples in Python?

Answer:

Python Module:

1. A Python module is a file containing Python definitions and statements. A module can define functions, classes, and variables. A module can also include runnable code.
2. Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organized.
3. The most commonly available built-in modules are:
    - os
    - math
    - sys
    - random
    - re
    - datetime
    - JSON

The from-import Statement in Python:
Python's from statement lets you import specific attributes from a module without importing the module as a whole.

Importing specific attributes from the module
Here, we are importing specific sqrt and factorial attributes from the math module.

Code:-
```
# importing sqrt() and factorial from the
# module math
from math import sqrt, factorial

# if we simply do "import math", then
# math.sqrt(16) and math.factorial()
# are required.
print(sqrt(16))
print(factorial(6))
```

Output:
4.0
720

c) Explain variables with examples.

Answer:

1. Python Variables:
    - Variables are containers for storing data values.

2. Creating Variables:
    - Python has no command for declaring a variable.
    - A variable is created the moment you first assign a value to it.

3. The value stored in a variable can be changed during program execution.
4. A Python Variables is only a name given to a memory location, all the operations done on the variable effects that memory location.

Rules for creating variables in Python:

- A variable name must start with a letter or the underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
- Variable names are case-sensitive (name, Name and NAME are three different variables).
- The reserved words(keywords) cannot be used naming the variable.

For ex:

```
# An integer assignment
age = 45

# A floating point
salary = 1456.8

# A string
name = "John"

print(age)
print(salary)
print(name)
```

Output:

```
45
1456.8
John
```

d) What are errors in Python? Explain with example.

Answer:

Errors in Python:

1. Errors are the problems in a program due to which the program will stop the execution. On the other hand, exceptions are raised when some internal events occur which changes the normal flow of the program.
2. Two types of Error occurs in python.
    a) Syntax errors
    b) Logical errors (Exceptions)

a) Syntax errors

When the proper syntax of the language is not followed then a syntax error is thrown.

For Ex:

```
# initialize the amount variable
amount = 10000
# check that You are eligible to
#  purchase Dsa Self Paced or not
if(amount>2999)
   print("You are eligible to purchase Dsa Self Paced")
```

Output:

```
SyntaxError: invalid error
```

3. It returns a syntax error message because after the if statement a colon: is missing. We can fix this by writing the correct syntax.

e) What is dynamically typed language ?

Answer:

1. Before we understand a dynamically typed language, we should learn about what typing is. Typing refers to type-checking in programming languages.
2. In a strongly-typed language, such as Python, "1" + 2 will result in a type error since these languages don't allow for "type-coercion" (implicit conversion of data types).
3. On the other hand, a weakly-typed language, such as Javascript, will simply output "12" as result.
4. Type-checking can be done at two stages -
5. Static - Data Types are checked before execution.
6. Dynamic - Data Types are checked during execution.
7. Python is an interpreted language, executes each statement line by line and thus type-checking is done on the fly, during execution.

8. Hence, Python is a Dynamically Typed Language.

For ex: -

```
## assigning a value to a variable
x = [1, 2, 3]
## x is a list here
print(type(x))
## reassigning a value to the 'x'
x = True
## x is a bool here
print(type(x))
```

Output

```
<class 'list'>
<class 'bool'>
```

f) What is pass keyword in Python ?

Answer:

pass Keyword:

- The pass keyword represents a null operation in Python.
- It is generally used for the purpose of filling up empty blocks of code which may execute during runtime but has yet to be written.
- Without the pass statement in the following code, we may run into some errors during code execution.

For ex:

```
def myEmptyFunc():
  # do nothing
  pass
myEmptyFunc()
# nothing happens without the pass keyword
# File "<stdin>", line 3
# IndentationError: expected an indented block
```

g) Write a program for database connectivity in Python with MySQL.

Answer:

MySQL:

1. MySQL is an Open-Source database and one of the best type of RDBMS (Relational Database Management System).

2. Install MySQL Connector Library for Python
3. Here is how to connect MySQL with Python:
    a. For Python 2.7 or lower install using pip as:
        pip install mysql-connector

    b. For Python 3 or higher version install using pip3 as:
        pip3 install mysql-connector
4. Syntax to access MySQL with Python:

```
import mysql.connector
db_connection = mysql.connector.connect(
host="hostname",
user="username",
passwd="password"
)
```

For Ex:

```
import mysql.connector
db_connection = mysql.connector.connect(
host="localhost",
user="root",
passwd=""
)
print(db_connection)
```

Output:

<mysql.connector .connection.MySQLConnection object at 0x000002338A4C6B00>

Q. 2 Attempt any three of the following (5 Marks each)

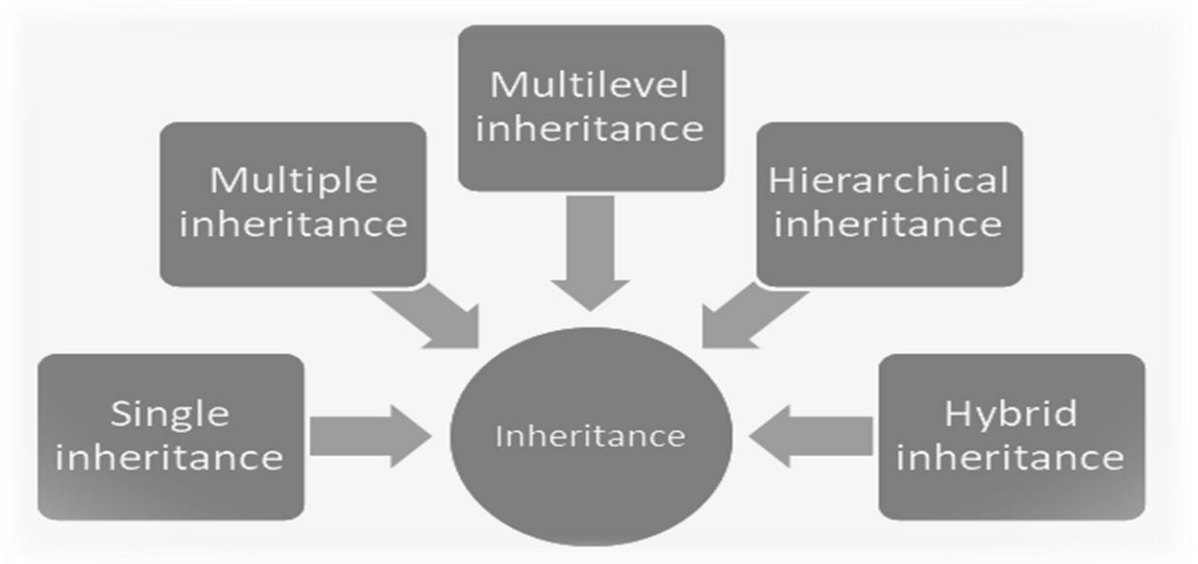
a) What are the types of inheritance in Python? Explain.

Answer:

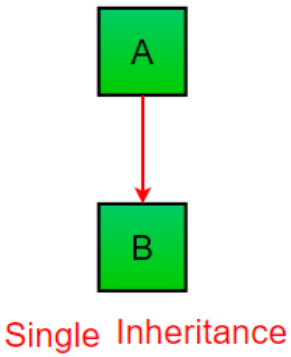
Types of inheritance Python:

1. Inheritance is the capacity of a particular class to obtain or inherit properties from another class and then use them when required. Inheritance has the following characteristics:

- It is an excellent representation of relationships in the real world.
- It allows code reuse. It doesn't require us to create the same code repeatedly and again.
- It also allows us to add options to an existing class without having to modify the existing code.

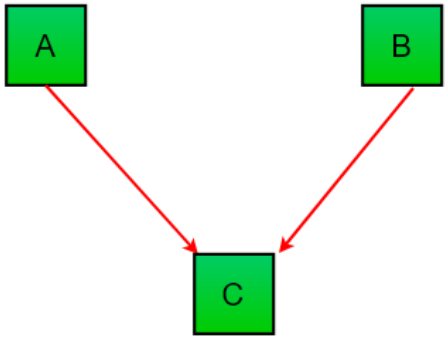


Single Inheritance:
Single inheritance enables a derived class to inherit properties from a single parent class, thus enabling code reusability and the addition of new features to existing code.



Multiple Inheritance:

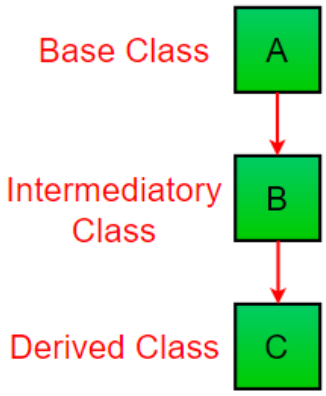
When a class can be derived from more than one base class this type of inheritance is called multiple inheritances. In multiple inheritances, all the features of the base classes are inherited into the derived class.



Multiple Inheritance

Multilevel Inheritance :

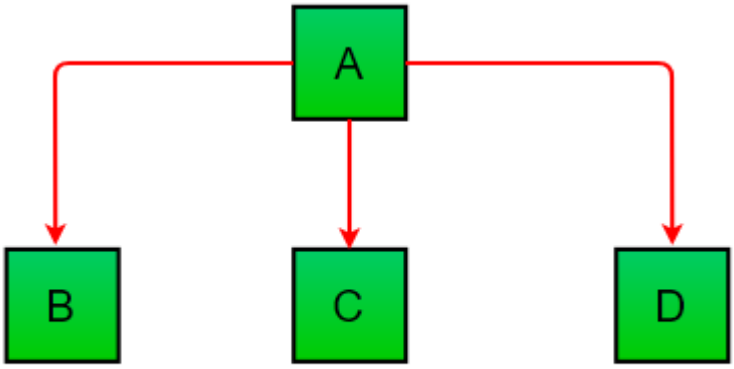
In multilevel inheritance, features of the base class and the derived class are further inherited into the new derived class. This is similar to a relationship representing a child and a grandfather.



Multilevel Inheritance

Hierarchical Inheritance:

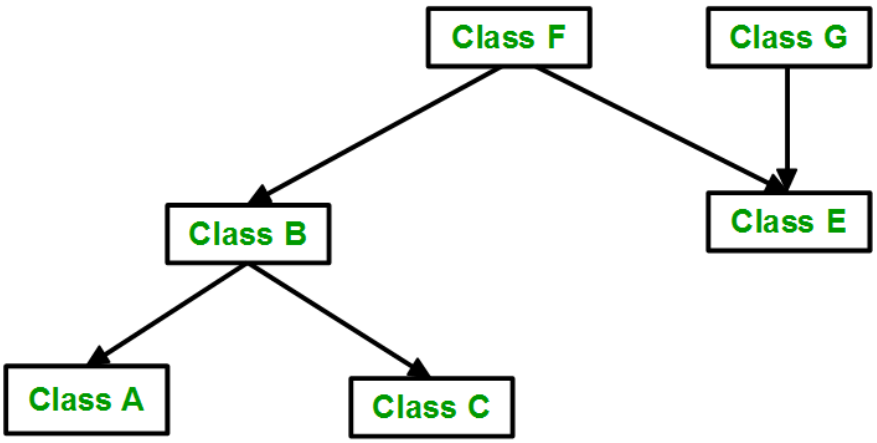
When more than one derived class are created from a single base this type of inheritance is called hierarchical inheritance. In this program, we have a parent (base) class and two child (derived) classes.



Hierarchical Inheritance

Hybrid Inheritance:

Inheritance consisting of multiple types of inheritance is called hybrid inheritance.



b) What are the common built-in data types in Python?

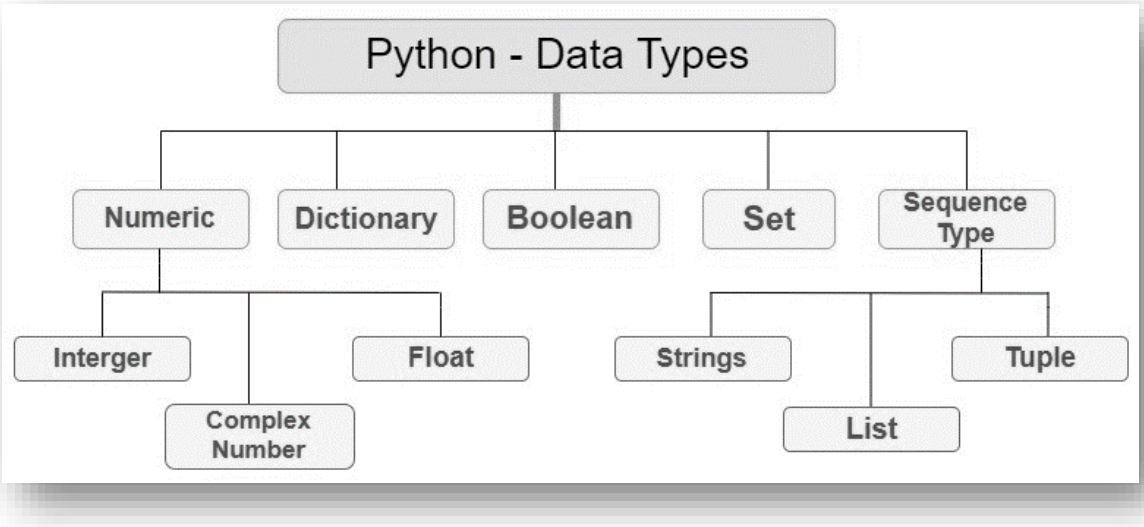
Answer:

Python Data Types

1. Data types are the classification or categorization of data items.
2. It represents the kind of value that tells what operations can be performed on a particular data.

Following are the standard or built-in data type of Python:

- Numeric
- Sequence Type
- Boolean
- Set
- Dictionary



Numeric Data Types

1. In Python, numeric data type represent the data which has numeric value. Numeric value can be integer, floating number or even complex numbers. These values are defined as int, float and complex class in Python.
2. Note – type() function is used to determine the type of data type.

Code:

```
a = 5
print("Type of a: ", type(a))
b = 5.0
print("\nType of b: ", type(b))
```

```
c = 2 + 4j
print("\nType of c: ", type(c))
```

Output:
Type of a: <class 'int'>
Type of b: <class 'float'>
Type of c: <class 'complex'>

Sequence Type

1. In Python, sequence is the ordered collection of similar or different data types.
2. Sequences allows to store multiple values in an organized and efficient fashion.
3. There are several sequence types in Python –
    - String
    - List
    - Tuple

Code:

```
a = "Hello"
print("Type of a:",type(a))
b = [10,'a',20,'b']
print("Type of b:",type(b))
c = {10,20,30}
print("Type of c:",type(c))
```

Output:

Type of a: <class 'str'>
Type of b: <class 'list'>
Type of c: <class 'set'>

Boolean

Data type with one of the two built-in values, True or False.
Code:

```
print(type(True))
print(type(False))
```

Output:
<class 'bool'>
<class 'bool'>

Set
In Python, Set is an unordered collection of data type that is iterable, mutable and has no duplicate elements.

Code:
```
a = set()
print("Type of a:",type(a))
b = {10,'a',20,'b'}
print("Type of b:",type(b))
```

Output:
Type of a: <class 'set'>
Type of b: <class 'set'>

Dictionary
Dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair.

Code:
```
a = dict()
print("Type of a:",type(a))
b = {10:'a',20:'b'}
print("Type of b:",type(b))
```

Output:
Type of a: <class 'dict'>
Type of b: <class 'dict'>

c) What are modules and packages in Python? Explain.

Answer:

Modules:

1. A Python module is a file containing Python definitions and statements.
2. A module can define functions, classes, and variables.
3. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use.
4. It also makes the code logically organized.

Create a simple Python module
Let’s create a simple calc.py in which we define two functions, one add and another subtract.

Code:

```
# A simple module, calc.py
def add(x, y):
   return (x+y)

def subtract(x, y):
   return (x-y)
```

- Import Module in Python

1. We can import the functions, and classes defined in a module to another module using the import statement in some other Python source file.

2. When the interpreter encounters an import statement, it imports the module if the module is present in the search path.
3. A search path is a list of directories that the interpreter searches for importing a module.
4. For example, to import the module calc.py, we need to put the following command at the top of the script.

- Syntax of Python Import

```
import module
```

Note: This does not import the functions or classes directly instead imports the module only.
To access the functions inside the module the dot(.) operator is used.

- Importing modules in Python

Now, we are importing the calc that we created earlier to perform add operation.

```
# importing  module calc.py
import calc

print(calc.add(10, 2))
```

Output:
12

The from-import Statement in Python:

1. Python's from statement lets you import specific attributes from a module without importing the module as a whole.
2. Importing specific attributes from the module:
3. Here, we are importing specific sqrt and factorial attributes from the math module.

Code:

```
# importing sqrt() and factorial from the
# module math
from math import sqrt, factorial

# if we simply do "import math", then
# math.sqrt(16) and math.factorial()
# are required.
print(sqrt(16))
print(factorial(6))
```

Output:

```
4.0
720
```

Packages:

1. The package is a simple directory having collections of modules.
2. This directory contains Python modules and also having __init__.py file by which the interpreter interprets it as a Package.
3. The package is simply a namespace. The package also contains sub-packages inside it.

Examples of Packages:
Numpy
Pandas

Example:

```
Student(Package)
| __init__.py (Constructor)
| details.py (Module)
| marks.py (Module)
| collegeDetails.py (Module)
```

d) What is break, continue and pass in Python? Explain.

Answer:

1. Loop control statements change execution from its normal sequence.
2. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.
3. Python supports the following control statements.

- Break statement
- Continue statement
- Pass statement

Break statement

- The break statement is used to terminate the loop or statement in which it is present.

Syntax:
break

Code:

```
numbers = [10, 40, 120, 230]
for i in numbers:
   if i > 100:
      break
   print('current number', i)
```

Output:
curret number 10
curret number 40

Continue statement
Continue is also a loop control statement just like the break statement.
continue statement is opposite to that of break statement, instead of terminating the loop, it forces to execute the next iteration of the loop.

Syntax:
continue

Code:

```
numbers = [2, 3, 11, 7]
for i in numbers:
    print('Current Number is', i)
    # skip below statement if number is greater than 10
    if i > 10:
        continue
    square = i * i
    print('Square of a current number is', square)
```

Output:

```
Current Number is 2
Square of a current number is 4
Current Number is 3
Square of a current number is 9
Current Number is 11
Current Number is 7
Square of a current number is 49
```

Pass statement

1. As the name suggests pass statement simply does nothing.
2. The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

Syntax:

```
pass
```

Code:

```
months = ['January', 'June', 'March', 'April']
for mon in months:
    pass
print(months)
```

Output:

```
['January', 'June', 'March', 'April']
```

e) What is polymorphism ? Explain with example.

Answer:

Polymorphism:

1. The word polymorphism means having many forms. In programming, polymorphism means the same function name (but different signatures) being used for different types.
2. The key difference is the data types and number of arguments used in function.

Example of inbuilt polymorphic functions:

```
# Python program to demonstrate in-built poly-
# morphic functions
# len() being used for a string
print(len("hello"))

# len() being used for a list
print(len([10, 20, 30]))
```

Output:

5

3

Examples of user-defined polymorphic functions:

```
# A simple Python function to demonstrate
# Polymorphism
def add(x, y, z = 0):
    return x + y+z

# Driver code
print(add(2, 3))
print(add(2, 3, 4))
```

Output:

5

9

Polymorphism with class methods:

1. The below code shows how Python can use two different class types, in the same way.
2. We create a for loop that iterates through a tuple of objects.
3. Then call the methods without being concerned about which class type each object is.
4. We assume that these methods actually exist in each class.

```
class India():
    def capital(self):
        print("New Delhi is the capital of India.")

class USA():
    def capital(self):
        print("Washington, D.C. is the capital of USA.")

obj_ind = India()
obj_usa = USA()
for country in (obj_ind, obj_usa):
    country.capital()
```

Output:
New Delhi is the capital of India.
Washington, D.C. is the capital of USA.

Q. 3 Attempt any three of the following (5 Marks each)

a) What is the difference between Python Arrays and lists?

Answer:

Arrays:

1. An array is defined as a collection of items that are stored at contiguous memory locations.
2. It is a container which can hold a fixed number of items, and these items should be of the same type.

Array Representation

- An array can be declared in various ways and different languages. The important points that should be considered are as follows:
    - Index starts with 0.
    - We can access each element via its index.
    - The length of the array defines the capacity to store the elements.

Syntax:

```
from array import *
arrayName = array(typecode, [initializers])
```

Code:

```
from array import *
var = array('i',[1,2,3,4,5])
print("var: ",var)
#Accessing element from var
x = var[0]
print("The value of x: ",x)
#Updating var
var[0] = 6
print("Updated var: ",var)
#Finding length of var
y = len(var)
print("The value of y: ",y)
#Adding single element to var
var.append(10)
print("Updated var: ",var)
```

```
#Removing element from var
#pop method removes element using index number
var.pop(2)
print("Removed from var: ",var)
```

Output:

```
var:  array('i', [1, 2, 3, 4, 5])
The value of x:  1
Updated var:  array('i', [6, 2, 3, 4, 5])
The value of y:  5
Updated var:  array('i', [6, 2, 3, 4, 5, 10])
Removed from var:  array('i', [6, 2, 4, 5, 10])
```

List:

1. Lists are used to store multiple items in a single variable.
2. Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.
3. Lists are created using square brackets:

Code:

```
var = [1,2,3,4,5]
print("var: ",var)
#Accessing element from var
x = var[0]
print("The value of x: ",x)
#Updating var
var[0] = 6
print("Updated var: ",var)
#Finding length of var
y = len(var)
print("The value of y: ",y)
#Adding single element to var
var.append(10)
print("Updated var: ",var)
#Removing element from var
#pop method removes element using index number
var.pop(2)
print("Removed from var: ",var)
```

Output:
var: [1, 2, 3, 4, 5]
The value of x: 1
Updated var: [6, 2, 3, 4, 5]
The value of y: 5
Updated var: [6, 2, 3, 4, 5, 10]
Removed from var: [6, 2, 4, 5, 10]

b) What is Python? What are the benefits of using Python ?

Answer:

1. Python is a high-level dynamic programming language interpreted and focuses on code readability. There are fewer steps compared to Java and C. In 1991, it was founded by Guido Van Rossum, a developer.
2. Python is one of the most popular and rapidly growing programming languages. Python is a programming language that is powerful, adaptable, and easy to learn. Python also has a vibrant community.
3. Because it supports a wide range of programming paradigms, it is widely found in various businesses. It also automatically manages memory.

Advantages of Python

1. Simple to Use and Understand

- For newcomers, Python is simple to understand and use. It's a highly developed programming language with an English-like syntax.
- The language is simple to adapt as a result of these factors. Because of its simplicity, Python's fundamentals can be implemented faster than those in other programming languages.

2. Free and Open-Source

- Python is distributed under an open-source license approved by the Open-Source Initiative (OSI).
- As a result, users can work on it and distribute it. Users can download the source code, modify it, and even distribute their Python version.
- Companies that wish to modify a specific behavior and build their version

will benefit.

3. Productivity has Increased

- Users can create new kinds of applications using the Python programming language.
- Because of its versatility, this language permits the operator to try new things.
- Because of the language, the user is not prevented from trying something new.
- Python is favored in these scenarios since other programming languages lack the flexibility and freedom that Python does.

4. Interpreted Language

- It is an interpreted language, implying that the code is implemented line by line. This is one of the features that makes it simple to use. In the event of an error, it halts the process and reports the problem.
- Python only shows one error, even if the program has multiple errors. This makes debugging easier.

5. Extensive library

- Python includes a huge number of libraries that the user can use. The standard library in Python is immense, and it includes almost every function imaginable.
- Large and supportive communities, as well as corporate sponsorship, have contributed to this. When working with Python, users do not need to use external libraries.

6. Dynamically Typed

- Until we run the program, Python has no idea what kinds of parameter we're talking about.
- It allocates the data type automatically during execution. Variables and their data types do not need to be declared by the programmer.

7. Portability

- Many other languages, including C/C++, demand that user must change their code to run on different platforms.
- Python, on the contrary, is not equivalent to other programming languages. It only needs to be written once, and then it can be run anywhere.
- However, the user should avoid involving any system-dependent features.

8. Supportive community

- Python is a programming language generated many years ago and has a large community that can assist programmers of all experience levels, from rookies to specialists.
- Python's community has helped it grow quickly in comparison to other languages. The Python programme
-
- ng language comes with many guides, instructional videos, and highly understandable documentation to help developers learn the language faster and more effectively.

c) How do you create class and object in Python?

Answer:

1. In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming.
2. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.
3. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.

Main Concepts of Object-Oriented Programming (OOPs)

- Class
- Objects
- Polymorphism
- Encapsulation
- Inheritance

Python Classes and Objects:

1. A class is a user-defined blueprint or prototype from which objects are created.
2. Classes provide a means of bundling data and functionality together.
3. Creating a new class creates a new type of object, allowing new instances of that type to be made.

Syntax: Class Definition

```
class ClassName:
    # Statement
```

Syntax: Object Definition

```
obj = ClassName()
print(obj.atrr)
```

4. Class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.
5. A class is like a blueprint for an object.

Some points on Python class:

- Classes are created by keyword class.
- Attributes are the variables that belong to a class.
- Attributes are always public and can be accessed using the dot (.) operator.
- Eg.: Myclass.Myattribute

An object consists of :

- State: It is represented by the attributes of an object. It also reflects the properties of an object.
- Behaviour: It is represented by the methods of an object. It also reflects the response of an object to other objects.
- Identity: It gives a unique name to an object and enables one object to interact with other objects.

Code:

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("ABC", 36)
print(p1.name)
print(p1.age)
```

Output:

```
ABC
36
```

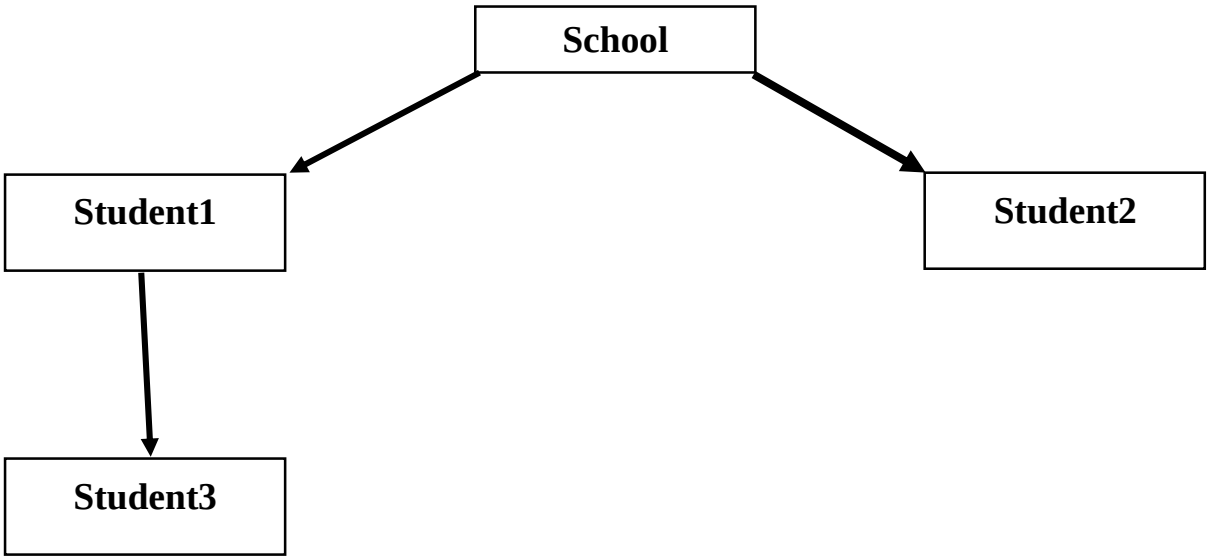
The __init__() function is called automatically every time the class is being used to create a new object.

d) Explain hybrid inheritance with example.

Answer:

Hybrid Inheritance:

Inheritance consisting of multiple types of inheritance is called hybrid inheritance.



Code:

```
# Python program to demonstrate hybrid inheritance
class School:
                    def func1(self):
                     print("This function is in school.")
class Student1(School):
                    def func2(self):
                     print("This function is in student 1. ")
class Student2(School):
                    def func3(self):
                     print("This function is in student 2.")
class Student3(Student1, School):
                    def func4(self):
                     print("This function is in student 3.")
```

```
# Driver's code
object = Student3()
object.func1()
object.func2()
```

Output:

This function is in school.

This function is in student 1.

- Hybrid Inheritance is a blend of more than one type of inheritance. The class is derived from the two classes as in the multiple inheritance.
- However, one of the parent classes is not the base class. It is a derived class. This feature enables the user to utilize the feature of inheritance at its best.
- This satisfies the requirement of implementing a code that needs multiple inheritances in implementation.

e) Explain modifying of string concept with examples.

Amswer:

Modifying String:-

- Python strings are immutable
- Python recognize as strings everything that is delimited by quotation marks
(” ” or ‘ ‘).

CODE:-

```
a = "Your_Name_is_displayed_on_TV."
print(a)
print(len(a)) # length of string
# Modifying string
print(a.upper()) # upper the letters
print(a.lower()) # lower the letters
print(a.capitalize()) # capital the first letter of total
string
print(a.title()) # capital the first letter of word in
total string
print(a.swapcase()) # capital to small and small to
capital
print(a.count('o')) #count 'o' in string
print(a.count('_')) # count '_' in string
print(a.find('Name')) #find the position in string
print(a.find('T')) #find particular one value in string
print(a.index('on')) # find the letters on in string
print(a.split('is')) # split from 'is' word
print(a.split('dis')) #split from 'dis' word
print(a.startswith('Y')) # If string starts with 'Y' then
print true,otherwise returns false.
print(a.endswith('.'))# If string ends with '.' then print
true,otherwise returns false.
print('.'*30) # repeating of string
a.replace('Your','Our')
print(a) # replacing the word from existing word
print(" ".join(a)) #add a whitespace between every char
print("+".join(a)) #add + whitespace between every char
```

```
# String Cocatenation
b = 'ABC'
c = 'DEF'
print(b+c)
```

OUTPUT:-
Your_Name_is_displayed_on_TV.
29
YOUR_NAME_IS_DISPLAYED_ON_TV.
your_name_is_displayed_on_tv.
Your_name_is_displayed_on_tv.
Your_Name_Is_Displayed_On_Tv.
yOUR_nAME_IS_DISPLAYED_ON_tv.
2
5
5
26
23
['Your_Name_', '_d', 'played_on_TV.']
['Your_Name_is_', 'played_on_TV.']
True
True
.....
Your_Name_is_displayed_on_TV.
Y o u r _ N a m e _ i s _ d i s p l a y e d _ o n _ T V .
Y+o+u+r+_+N+a+m+e+_+i+s+_+d+i+s+p+l+a+y+e+d+_+o+n+_+T+V+.
ABCDEF

Q. 4 Attempt any three of the following (5 Marks each)

f) What is set data type? Explain basic methods used umder set data type.

Answer:

Sets:

A Python set is the collection of the unordered items. Each element in the set must be unique, immutable, and the sets remove the duplicate elements.
Sets are mutable which means we can modify it after its creation.
Python's set class represents the mathematical notion of a set.

CODE:-

```
# Python program to demonstrate sets

# Same as {"a", "b", "c"}
myset = set(["a", "b", "c"])
print(myset)
```

OUTPUT:-

```
{'c', 'a', 'b'}
```

Frozen Sets:

Frozen sets in Python are immutable objects that only support methods and operators that produce a result without affecting the frozen set or sets to which they are applied.

CODE:

```
# A frozen set
frozen_set = frozenset(["e", "f", "g"])

print("\nFrozen Set")
print(frozen_set)
```

OUTPUT:

```
Frozen Set
```

frozenset({'g', 'f', 'e'})

Methods for Sets:

Code:

```
#Methods in Sets:
# Adding Single item to set
a = {10,20,30}
print("Value of a:",a)

a.add(40)
print("Updated value of a:",a)

#Removing elements from set
a.remove(10)
print("After removing item a:",a)
```

Output:

```
Value of a: {10, 20, 30}
Updated value of a: {40, 10, 20, 30}
After removing item a: {40, 20, 30}
```

Set Operations:

```
#Set Operations
a = {10,20,30,5,6,4}
print("Value of a:",a)

b = {40,50,60,5,6}
print("value of b:",b)

#Union operation
print("Union Operation",a.union(b))
print("Union Operation",a|b)

#Intersection operation
print("Intersection Operation",a.intersection(b))
print("Intersection Operation",a&b)
```

```
#Difference operation
print("Difference operation",a.difference(b))
print("Difference operation",a-b)
```

OUTPUT:

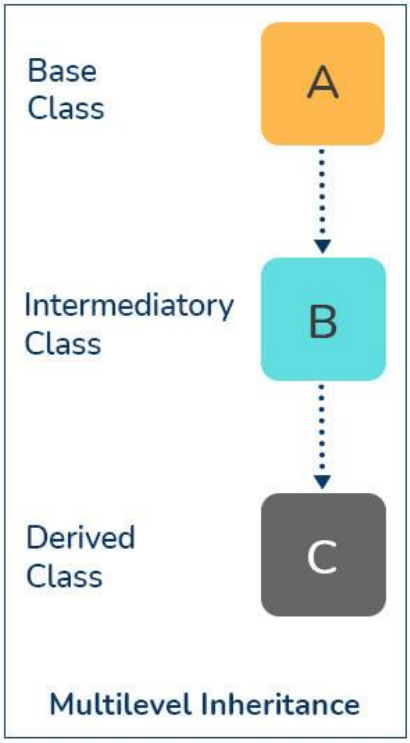
```
Value of a: {4, 5, 6, 10, 20, 30}
value of b: {5, 6, 40, 50, 60}
Union Operation {4, 5, 6, 40, 10, 50, 20, 60, 30}
Union Operation {4, 5, 6, 40, 10, 50, 20, 60, 30}
Intersection Operation {5, 6}
Intersection Operation {5, 6}
Difference operation {10, 4, 20, 30}
Difference operation {10, 4, 20, 30}
```

g) Explain multilevel inheritance with example.

Answer:

Multilevel Inheritance:

1. In Multi-level inheritance derived class inherits from another
2. derived class. There is no limit for level.
3. The members of the parent class, A, are inherited by child class which is then inherited by another child class, B.
4. The features of the base class and the derived class are further inherited into the new derived class, C.
5. Here, A is the grandfather class of class C.



CODE:

```
# Parent class
class A:
  def __init__(self, a_name):
     self.a_name = a_name
```

```
# Intermediate class
class B(A):
  def __init__(self, b_name, a_name):
    self.b_name = b_name
    # invoke constructor of class A
    A.__init__(self, a_name)

# Child class
class C(B):
  def __init__(self,c_name, b_name, a_name):
    self.c_name = c_name
    # invoke constructor of class B
    B.__init__(self, b_name, a_name)

  def display_names(self):
    print("A name : ", self.a_name)
    print("B name : ", self.b_name)
    print("C name : ", self.c_name)

#  Driver code
obj1 = C('child', 'intermediate', 'parent')
print(obj1.a_name)
obj1.display_names()
```

OUTPUT:

```
parent
A name: parent
B name: intermediate
C name: child
```

h) Write a program for passing query to MySQL in Python.

Answer:

CODE:-

```
import mysql.connector
mydb = mysql.connector.connect(host='localhost',
                    user='root',
                    password="")
print(mydb)
```

OUTPUT:-

```
<mysql.connector.connection.MySQLConnection object at
0x000001593860AE60>
```

Creating a Database

To create a database in MySQL, use the "CREATE DATABASE" statement:

CODE:-

```
import mysql.connector
mydb = mysql.connector.connect(host='localhost',
                    user='root',
                    password="")
mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE student")
```

Check if Database Exists

You can check if a database exist by listing all databases in your system by using the "SHOW DATABASES" statement:

CODE:-

```
import mysql.connector
```

```
mydb = mysql.connector.connect(host='localhost',
                    user='root',
                    password="")

mycursor = mydb.cursor()
mycursor.execute("SHOW DATABASES")
for x in mycursor:
  print(x)
```

OUTPUT:-

```
('information_schema',)
('mysql',)
('performance_schema',)
('phpmyadmin',)
('test',)
```

Creating a Table

- To create a table in MySQL, use the "CREATE TABLE" statement.
- Make sure you define the name of the database when you create the connection

CODE:-

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="", db="student")
print(mydb)
mycursor = mydb.cursor()
mycursor.execute("CREATE TABLE if not exists customers (name
VARCHAR(255), address VARCHAR(255))")
mycursor.execute("show tables")
for i in mycursor: print(i)
mycursor.close()
```

OUTPUT:
<mysql.connector.connection.MySQLConnection object at
0x0000020EA203DC30>
('customers',)

i) Explain exception raising with example.

Answer:
Raise an exception

- As a Python developer you can choose to throw an exception if a condition occurs.
- To throw (or raise) an exception, use the raise keyword.

CODE:

```
x = -1
if x < 0:
  raise Exception("Sorry, no numbers below zero")
```

OUTPUT:

```
Traceback (most recent call last):
  File "demo_ref_keyword_raise.py", line 4, in <module>
    raise Exception("Sorry, no numbers below zero")
Exception: Sorry, no numbers below zero
```

CODE:

```
>>> raise KeyboardInterrupt
Traceback (most recent call last):
...
KeyboardInterrupt

>>> raise MemoryError("This is an argument")
Traceback (most recent call last):
...
MemoryError: This is an argument

>>> try:
...     a = int(input("Enter a positive integer: "))
...     if a <= 0:
...         raise ValueError("That is not a positive number!")
```

```
... except ValueError as ve:
...     print(ve)
Enter a positive integer: -2
That is not a positive number!
```

j) What is pickling and unpickling of data? Explain with example.

Answer:

Ppickling of data:

1. Python pickle module is used for serializing and de-serializing python object structures.
2. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling.

Unpickling:

- We can converts the byte stream (generated through pickling) back into python objects by a process called as unpickling.
- In real world sceanario, the use pickling and unpickling are widespread as they allow us to easily transfer data from one server/system to another and then store it in a file or database.
- It is advisable not to unpickle data received from an untrusted source as they may pose security threat. However, the pickle module has no way of knowing or raise alarm while pickling malicious data.
- Only after importing pickle module we can do pickling and unpickling. Importing pickle can be done using the following command −

import pickle

Code for Pickling:

```
import pickle
a = open("raw.pickle","wb")
mylist = ['a', 'b', 'c', 'd']
pickle.dump(mylist,a)
```

Output:

```
□□]□(□a□□b□□c□□d□e.
```

In the above code, list – “mylist” contains four elements (‘a’, ‘b’, ‘c’, ‘d’). We open the file in “wb” mode instead of “w” as all the operations are done using bytes in the current working directory.

A new file named “raw.pickle” is created, which converts the mylist data in the byte stream.

Code for unpickling:

```
import pickle

a = open("raw.pickle","rb")

b = pickle.load(a)

print(b)
```

Output: On running above scripts, you can see your mylist data again as output.

```
['a', 'b', 'c', 'd']
```

- Pickling is a way to convert a python object (list, dict, etc.) into a character stream.
- The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

Q .5 Short notes on any three of the following (5 Marks each)

a) Comments in Python

Answer:

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.

Creating a Comment

1. Comments starts with a #, and Python will ignore them:

Example
#This is a comment
print("Hello, World!")

2. Comments can be placed at the end of a line, and Python will ignore the rest of the line:

Example
print("Hello, World!") #This is a comment

3. A comment does not have to be text that explains the code, it can also be used to prevent Python from executing code:

Example
#print("Hello, World!")
print("Cheers, Mate!")

Multi Line Comments

- Python does not really have a syntax for multi line comments.
- To add a multiline comment you could insert a # for each line:

Example

#This is a comment

#written in

#more than just one line

```
print("Hello, World!")
```

- Or, not quite as intended, you can use a multiline string.
- Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

Example

```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

- As long as the string is not assigned to a variable, Python will read the code, but then ignore it, and you have made a multiline comment.

b) Slicing in strings
Answer:

String Slicing

• To access a range of characters in the String, the method of slicing is used. Slicing in a String is done by using a Slicing operator (colon).

CODE:-

```
# Python Program to
# demonstrate String slicing

# Creating a String
String1 = "HELLOCOCSITLATUR"
print("Initial String: ")
print(String1)
#Printing start index value
print("Start Index:")
print(String1[5:])
```

```
#Printing End index value
print("End Index:")
print(String1[:9])
# Printing 3rd to 12th character
print("\nSlicing characters from 3-12: ")
print(String1[3:12])

# Printing characters between
# 3rd and 2nd last character
print("\nSlicing characters between " +
    "3rd and 2nd last character: ")
print(String1[3:-2])
```

OUTPUT:-
Initial String:
HELLOCOCSITLATUR
Start Index:
COCSITLATUR
End Index:
HELLOCOCS

Slicing characters from 3-12:
LOCOCSITL

Slicing characters between 3rd and 2nd last character:
LOCOCSITLAT

c) Method overriding
Answer:
Overriding

- Override means having two methods with the same name but doing different tasks.
- It means that one of the methods overrides the other.
- The concept of Method overriding allows us to change or override the Parent Class
- function in the Child Class.

In Python, to override a method, you have to meet certain conditions

- You can’t override a method within the same class. It means you

have to do it in the child class using the Inheritance concept.

- To override the Parent Class method, you have to create a method in the Child class with the same name and the same number of parameters.

Example:

```
# Python Method Overriding
class Employee:
        def message(self):
            print('This message is from Employee Class')
class Company(Employee):
        def message(self):
            print('This Company class is inherited from Employee')
emp = Employee()
emp.message()
comp = Company()
comp.message()
```

Output:

'This message is from Employee Class'
'This Company class is inherited from Employee'

d) File Handling in Python

Answer:

FILE HANDLING

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but like other concepts of

Working of open() function

Before performing any operation on the file like read or write, first we have to open that file. For this, we should use Python's inbuilt function open()

Syntax:

variable = open('filename', 'mode')

Where the following mode is supported:

- r: open an existing file for a read operation.
- w: open an existing file for a write operation. If the file already contains some data then it will be overridden.

- a: open an existing file for append operation. It won't override existing data.
- r+: To read and write data into the file. The previous data in the file will not be deleted.
- w+: To write and read data. It will override existing data.
- a+: To append and read data from the file. It won't override existing data.

Working of read() mode

There is more than one way to read a file in Python. If you need to extract a string that contains all characters in the file then we can use file.read(). The full code would work like this:

CODE:-

```
a = open('file.txt','r')
print(a.read())
```

OUTPUT:-

Hello_World!

Another way to read a file is to call a certain number of characters like in the following code the interpreter will read the first five characters of stored data and return it as a string:

CODE:-

```
a = open('file.txt','r')
print(a.read(1))
print(a.read(2))
```

OUTPUT:-

H
el

Creating a file using write() mode

- Let's see how to create a file and how write mode works:
- To manipulate the file, write the following in your Python environment:
- The close() command terminates all the resources in use and frees the system of this particular program.

CODE;-

```
a = open('file.txt','w')
```

```
a.write("This is write function")
a.write('''This will override
data which is
stored in our file''')
a.close()
```

e) Introduction to flask.

Answer:

Flask:

- Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher.
- Flask's framework is more explicit than Django's framework and is also easier to learn because it has less base code to implement a simple web-Application.
- A Web-Application Framework or Web Framework is the collection of modules and libraries that helps the developer to write applications without writing the low-level codes such as protocols, thread management, etc.
- Flask is based on WSGI(Web Server Gateway Interface) toolkit and Jinja2 template engine.

Getting Started With Flask:

- Python 2.6 or higher is required for the installation of the Flask. You can start by import Flask from the flask package on any python IDE.
- For installation on any environment, you can click on the installation link given below.
- To test that if the installation is working, check out this code given below.

```
# an object of WSGI application
from flask import Flask
app = Flask(__name__)   # Flask constructor

# A decorator used to tell the application
# which URL is associated function
@app.route('/')
def hello():
    return 'HELLO'

if __name__=='__main__':
```

app.run()

- '/' URL is bound with hello() function. When the home page of the webserver is opened in the browser, the output of this function will be rendered accordingly.
- The Flask application is started by calling the run() function.
- The method should be restarted manually for any change in the code.
- To overcome this, the debug support is enabled so as to track any error.

CODE:

```
app.debug = True
app.run()
app.run(debug = True)
```

Routing:

- Nowadays, the web frameworks provide routing technique so that user can remember the URLs.
- It is useful to access the web page directly without navigating from the Home page.
- It is done through the following route() decorator, to bind the URL to a function.

CODE:

```
# decorator to route URL
@app.route('/hello')

# binding to the function of route
def hello_world():
  return 'hello world'
```

- If a user visits http://localhost:5000/hello URL, the output of the hello_world() function will be rendered in the browser.
- The add_url_rule() function of an application object can also be used to bind URL with the function as in above example.

CODE:

```
def hello_world():
  return 'hello world'
app.add_url_rule('/', 'hello', hello_world)
```