DEPARTMENT OF COMPUTER SCIENCE

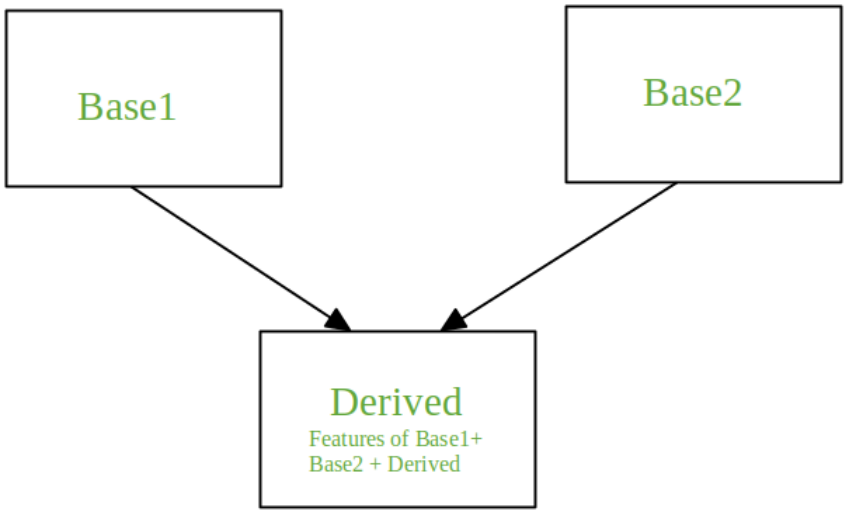
TAWANDA NDEMERA

H180270P

ASSIGNMENT 1

1(i) Multiple Inheritance

-When a class is derived from more than one base class it is called multiple Inheritance. The derived class inherits all the features of the base case.



Example of multiple Inheritance

```
# Parent class created
class Parent:
    parentname = ""
    childname = ""
def show_parent(self):
        print(self.parentname)
```

```
# Child class created inherits Parent class
class Child(Parent):
    def show_child(self):
        print(self.childname)


ch1 = Child() # Object of Child class
ch1.parentname = "Mark"   # Access Parent class attributes
ch1.childname = "John"
ch1.show_parent()   # Access Parent class method
ch1.show_child()    # Access Child class method
```

(ii)Single Inheritance

-When a child class inherits from only one parent class, it is called as single inheritance.

Example of Single inheritance in python

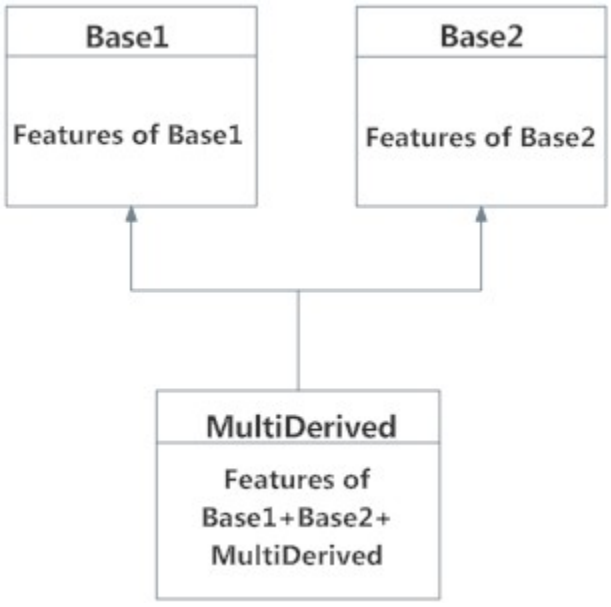
```
#Base class
class Parent_class(object):
# Constructor
def __init__(self, name, id):
self.name = name
self.id = id
# To fetch employee details
def Employee Details(self):
return self.id , self.name
# To check if this  is a valid employee
def Employee check(self):
if self.id > 500000:
return " Valid Employee "
else:
```

```
return " Invalid Employee "
# derived class or the sub class
class Child_class(Parent_class):
def End(self):
print( " END OF PROGRAM " )
# Driver code
Employee1 = Parent_class( "Employee1" , 600445)  # parent class object
print(Employee1.Employee_Details() , Employee1.Employee_check() )
Employee2 = Child_class( "Employee2" , 198754) # child class object
print(Employee2.Employee_Details() , Employee2.Employee_check() )
Employee2.End()
```

Output:

Single Inheritance in Python 1-2

(iii) Multi-Level Inheritance



```
class Family:
    def show_family(self):
        print("This is our family:")
 # Father class inherited from Family

class Father(Family):
    fathername = ""
    def show father(self):
        print(self.fathername)

# Mother class inherited from Family
class Mother(Family):
    mothername = ""

    def show mother(self):
        print (self. mothername)
 # Son class inherited from Father and Mother classes

class Son (Father, Mother):
    def show parent(self):
        print("Father :", self.fathername)
        print("Mother :", self.mothername)
 s1 = Son()  # Object of Son class
s1.fathername = "Mark"
s1.mothername = "Sonia"
s1.show_family()
s1.show_parent()
```

2(i) Abstraction in python is the process of hiding the real implementation of an application from the user and emphasizing only on usage of it.

Example of Abstraction

```
# Python program showing
# abstract base class work
 from abc import ABC, abstractmethod
class Animal(ABC):

   def move(self):
       pass
 class Human(Animal):
   def move(self):
       print("I can walk and run")
 class Snake(Animal):
   def move(self):
       print("I can crawl")

class Dog(Animal):
 def move(self):
       print("I can bark")

class Lion(Animal):

   def move(self):
       print("I can roar")

# Driver code
R = Human()
R.move()
```

```
K = Snake()
K.move()

R = Dog()
R.move()

K = Lion()
K.move()
```

(ii) Polymorphism

-The word polymorphism means having many forms. In programming, polymorphism means same function name (but different signatures) being uses for different types.

Example of Polymorphism

```
class India():
    def capital(self):
        print("New Delhi is the capital of India.")
    def language(self):
        print("Hindi is the most widely spoken language of India.")

    def type(self):
        print("India is a developing country.")

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

    def language(self):
        print("English is the primary language of USA.")
```

```
    def type(self):
        print("USA is a developed country.")


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

3) **Python Dictionary** are defined into two elements keys and values. Keys will be a single element. Values can be a list or list within a list, numbers, etc.

Example

```
Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}
del Dict ['Charlie']
print(Dict)
```

Output

When you run this code, it should print the dictionary list without Charlie.

We used the code del Dict

When code executed, it has deleted the Charlie from the main dictionary

4) **Dictionary Methods and their examples**

(i) len ()

-The len () function returns the number of items in an object.

-When the object is a string, the len () function returns the number of characters in the string

Example

Return the number of characters in a string:

```
mylist = "Hello"
x = len(mylist)
```

(ii) str ()

- The str () function converts the specified value into a string.

Example

Convert the number 3.5 into a string:

```
x = str (3.5)
```

(iii) max ()

-It returns the largest item in an iterable or the largest of two or more arguments.

Example

```
list1 = [1, 2, 4, 5, 54]
print(max(list1)) # Returns 54 as 54 is the largest value in the list
```

(iv) min ()

-It returns the smallest item in an iterable, or the smallest of two or more arguments.

Example

```
list1 = [1, 2, 4, 5, -54]
print(min(list1)) # Returns -54 as -54 is the smallest value in the list
```

(v) copy ()

-A copy is sometimes needed so one can change one copy without changing the other.

Example

```
# initializing list 1
li1 = [1, 2, [3,5], 4]
  # using copy for shallow copy
li2 = copy.copy(li1
```

vi. get ()

-is a conventional method to access a value for a key.

Example

```
dic = {"A":1, "B":2}
print(dic.get("A"))
print(dic.get("C"))
```

```
print(dic.get("C","Not Found ! "))
```

vii. keys ()

-returns a view object that displays a list of all the keys in the dictionary.

Example

```
Dictionary1 = {'A': 'Geeks', 'B': 'For', 'C': 'Geeks'}
  # Printing keys of dictionary
print (Dictionary1.keys())
  # Creating empty Dictionary
empty_Dict1 = {}
  # Printing keys of Empty Dictionary
print(empty_Dict1.keys())
```

viii. **values ()**

-The values () method returns a view object.

Example

```
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = car.values()
print(x)
```

ix. items ()

-The items () method returns a view object. The view object contains the key-value pairs of the dictionary, as tuples in a list.

Example

```
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

```
x = car.items ()
car["year"] = 2018
print(x)
```

x. clear ()

-The clear () method removes all the elements from a dictionary.

Example

```
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
car.clear()
print(car)
```