python for Computational Problem Solving
- pCPS - OOP in python
Lecture Slides - Class #49_#50

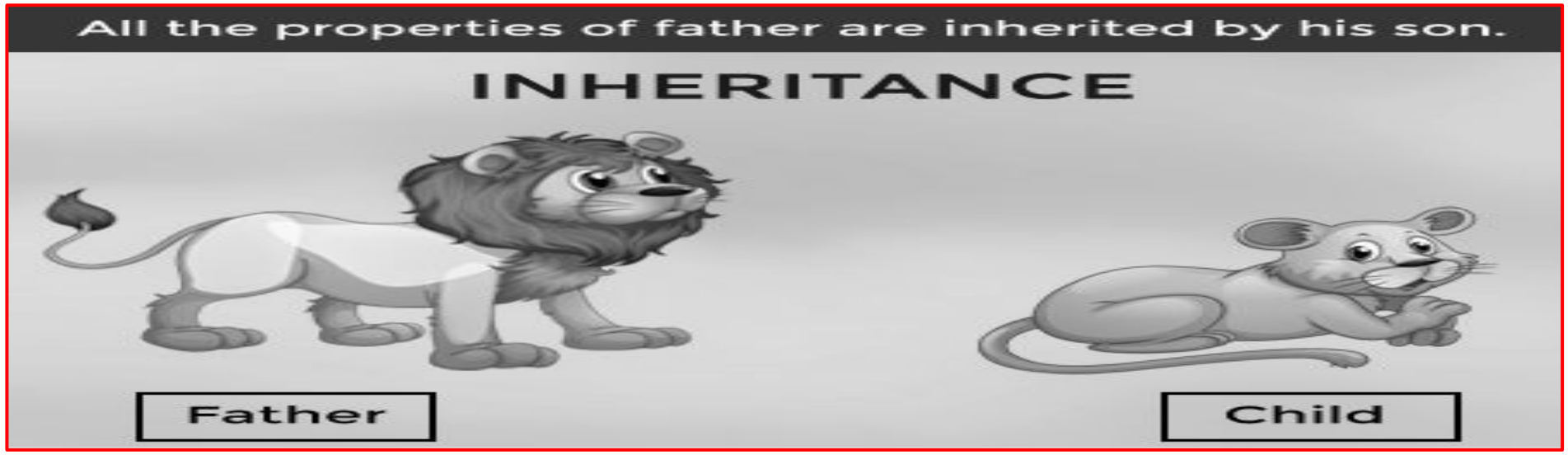
Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

python for Computational Problem Solving Syllabus

Unit V: Object Oriented Programming - 10 Hours

- OOP in python
- classes and objects
  - inheritance
  - polymorphism.
- Error handling & Exceptions - try, except and raise
- exception propagation

OOP - Inheritance in python



OOP - Inheritance in python

- **Object-oriented programming (OOP)** creates reusable patterns of code to curtail redundancy in development of python code / modules / projects.

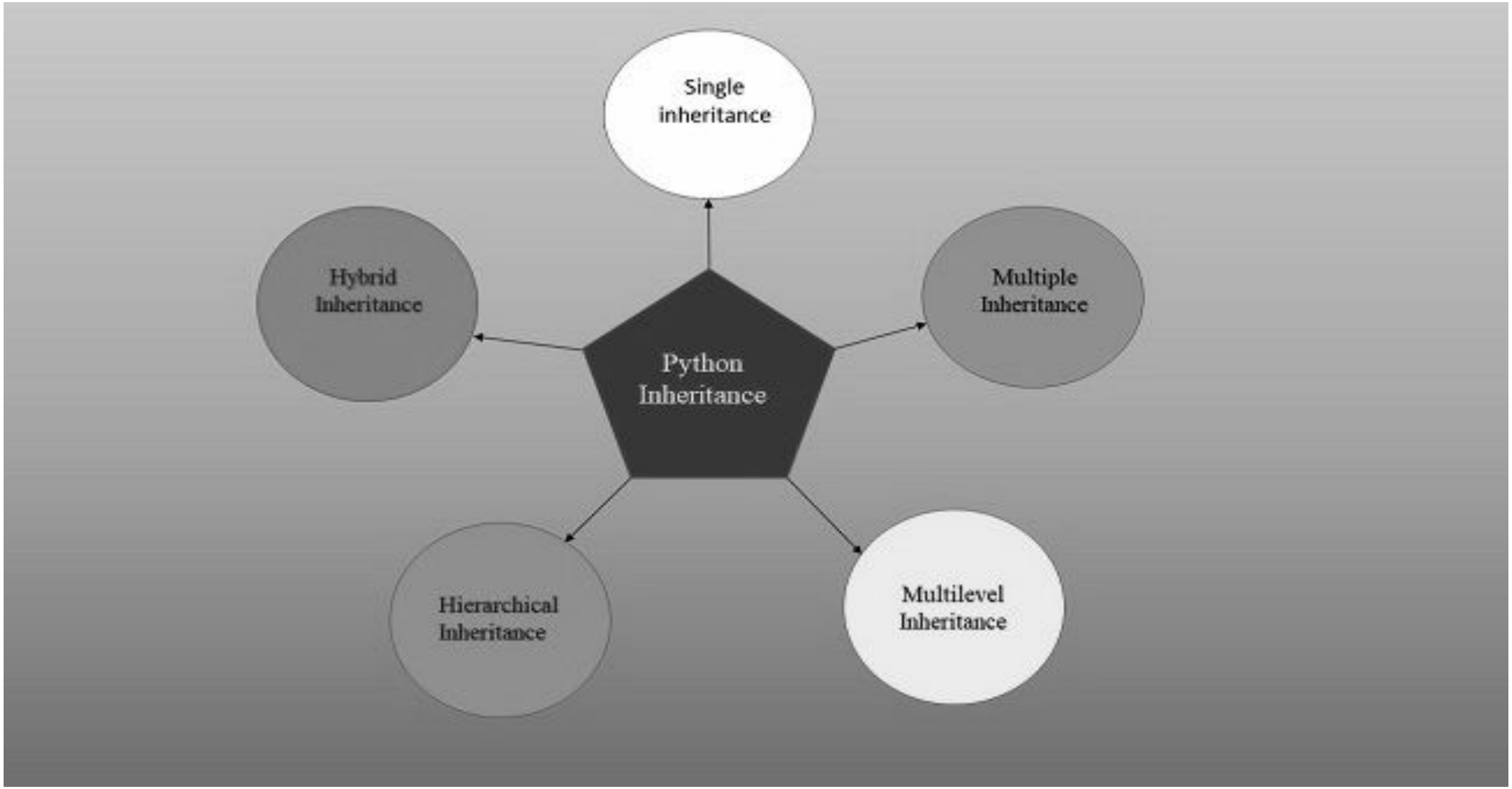
- One of the way that **OOP** achieves **recyclable** code is through **inheritance**, when one **subclass** can **leverage** code from another **base class**

- **Inheritance** is **when** a **class** uses **code constructed** within **another class**.

- **Classes** called **child** classes or **subclasses inherit methods** and **variables** from **parent** or **base classes**.

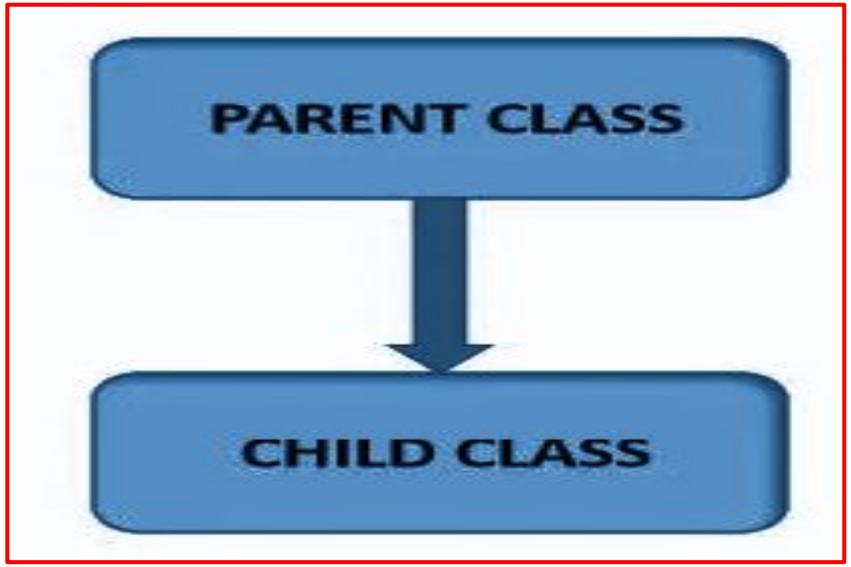
- The **Child** subclass is **inheriting** from the **Parent** base class, the **Child** class can **reuse** the **code** of **Parent**, allowing the **programmer** to use **fewer** lines of code **(loc)** and **decrease redundancy**

OOP - Inheritance in python



OOP - Single Inheritance in python

- **Object-oriented programming (OOP)** creates reusable patterns of code to curtail redundancy in development python code / modules / projects.
- **Single inheritance** enables **derived** class to call **parent** class **method** and also to **override parent** class's existing **methods**
- **Each** of these **classes** has its **own code** block.
- Per **single inheritance**, **every** element **present** in the **parent** class's code block can be wisely **used** in the **child** class.




```
class Parent_class_Name:

#Parent_class code block

class Child_class_Name(Parent_class_name):

#Child_class code block
```

OOP - Single Inheritance in python

- **Object-oriented programming (OOP)** creates reusable patterns of code to curtail redundancy in development python code / modules / projects.
- **Single inheritance** enables **derived** class to call **parent** class **method** and also to **override parent** class's existing **methods**
- **Each** of these **classes** has its **own code** block.
- Per **single inheritance**, **every** element **present** in the **parent** class's code block can be wisely **used** in the **child** class.

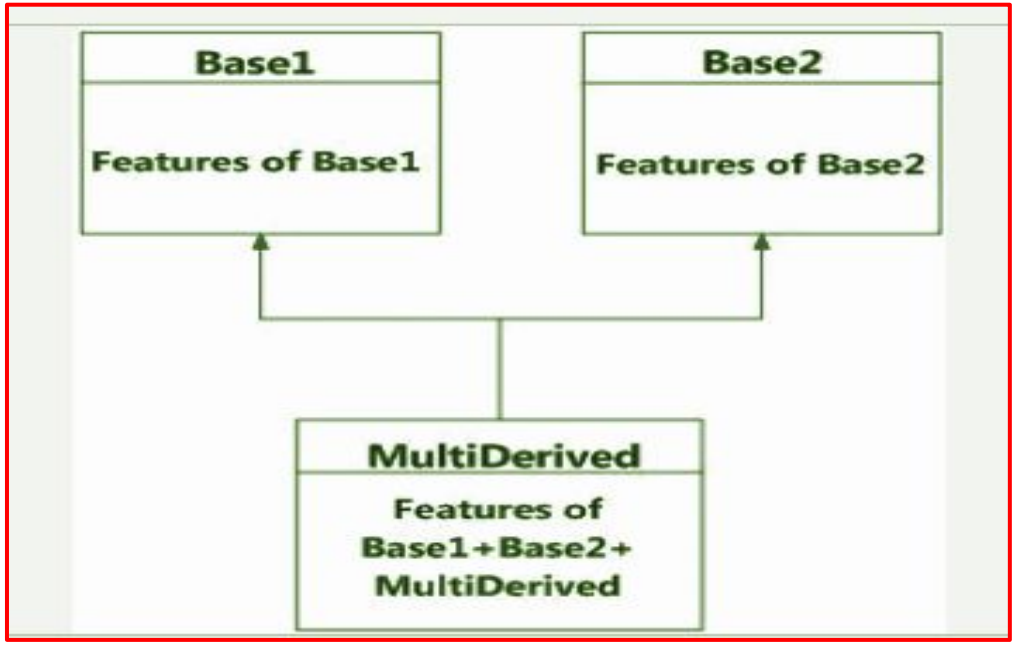
```
# Single Inheritance Example
# Parent or Base Class
class University:
    University = 'PES University'
    Program = 'B.Tech First Semester'
    Session = 'September 2021 - March 2022'

    def Info(self):
        print(self.University)
        print(self.Program)
        print(self.Session)

# Child or Derived or Sub Class
class Section(University):

    def __init__(self,Name, Strength):
        self.Name = Name
        self.Strength = Strength

    def AdditionalInfo(self):
        print(self.Name)
        print(self.Strength)

Generic = University()
P = Section('P Section',70)
Q = Section('Q Section',65)
Generic.Info()
P.AdditionalInfo
print('----------------------------------------')
Q.Info()
Q.AdditionalInfo()
```

```
PES University
B.Tech First Semester
September 2021 - March 2022
----------------------------------------
PES University
B.Tech First Semester
September 2021 - March 2022
Q Section
65
```

OOP - Multiple Inheritance in python

- In **multiple inheritance**, the features of all the **base classes** are **inherited** into the **derived** class.

- The **syntax** for **multiple** inheritance is **similar** to **single** inheritance

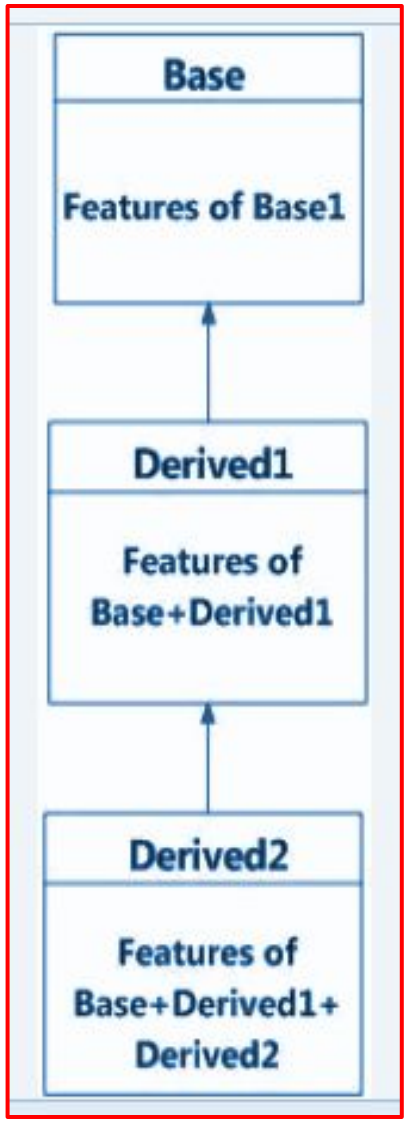



```
# Multiple Inheritance Example
# Class 1
class University:
    University = 'PES University'
    Program = 'B.Tech First Semester'
    Session = 'September 2021 - March 2022'


# Class 2
class Section:
    Name = 'P Section'
    Strength = 70

# Class 3
class Student(University,Section):

    def __init__(self,SRN, SName):
        self.University = 'PESU'
        self.SRN = SRN
        self.SName = SName

S1 = Student('PES1', 'Test1')

print(S1.University)
print(S1.Name)
print(S1.SRN)
print(S1.SName)
```

```
PESU
P Section
PES1
Test1
```

OOP - Multilevel Inheritance in python

- We can also **inherit** from a **derived** class. This is called **multilevel inheritance**.
- It can be of **any depth** in **Python**.
- In **multilevel inheritance**, **features** of the **base** class and the **derived** class are **inherited** into the **new derived** class.

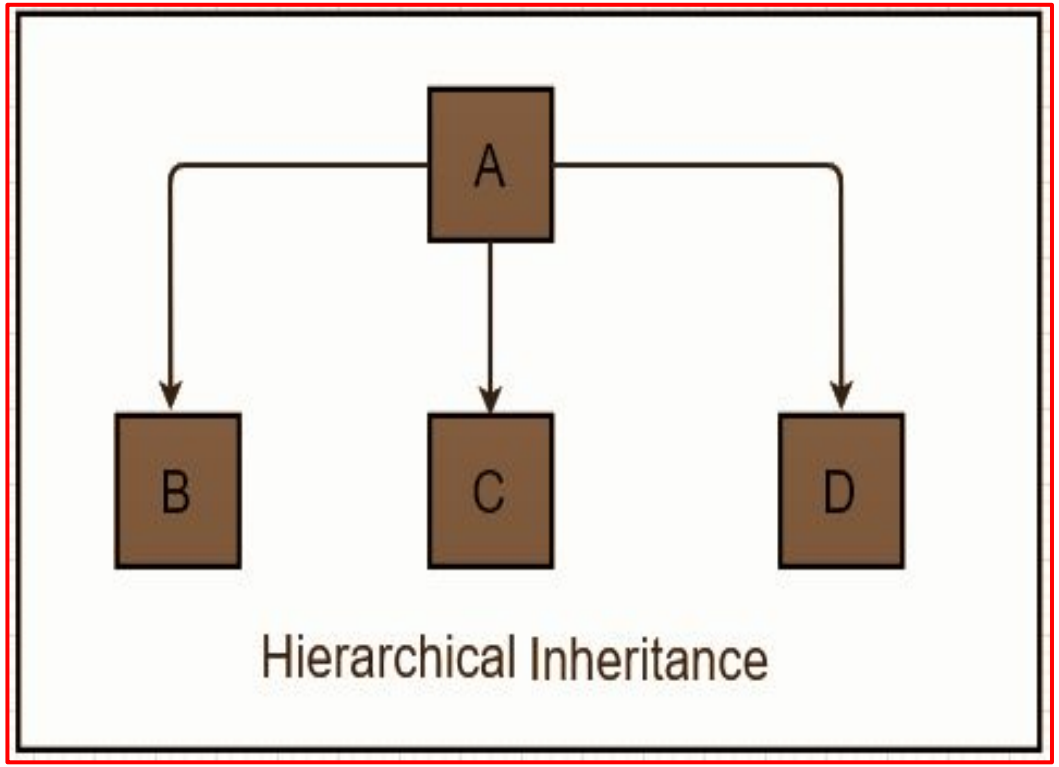



```
# Multilevel Inheritance Example
# Class 1
class University:
    University = 'PES University'
    Program = 'B.Tech First Semester'
    Session = 'September 2021 - March 2022'

# Class 2
class Section(University):
    def __init__(self,Name,Strength):
        self.Name = Name
        self.Strength = Strength


class Student(Section):
    def __init__(self,SRN, SName):
        self.P = Section('P',70)
        self.University = 'PESU'
        self.SRN = SRN
        self.SName = SName

Q = Section('Q',65)
S1 = Student('PES1', 'Test1')
print(Q.University)
print(S1.Session)
print(S1.P.Name)
print(S1.SRN)
print(S1.SName)
```

```
PES University
September 2021 - March 2022
P
PES1
Test1
```

OOP - Hierarchical Inheritance in python

- When we **derive** or **inherit more** than **one child** class from **one(same)** parent class **results** in **inheritance** called **hierarchical** inheritance

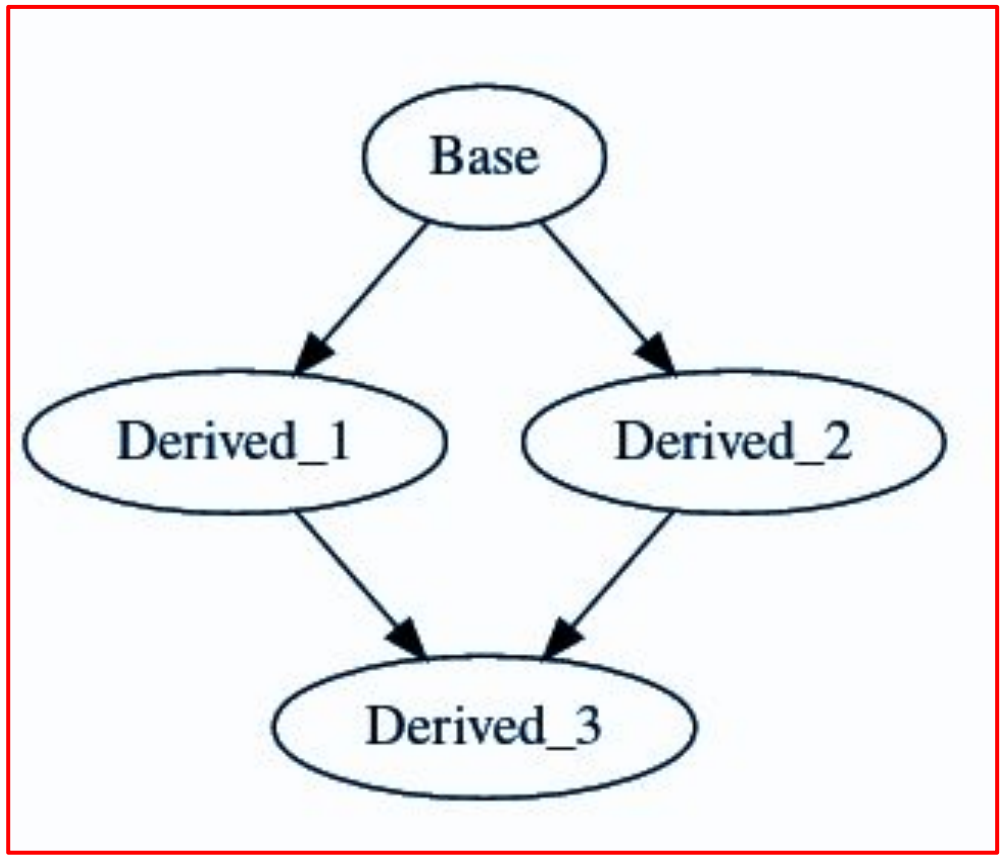



```
# Hierarchical Inheritance Example
# Class 1
class University:
    University = 'PES University'
    Program = 'B.Tech First Semester'
    Session = 'September 2021 - March 2022'

class Department(University):
    DName = 'EC'
    Location = 'B-Block 3 and 4 Floor'

class Section(University):
    def __init__(self,SName, Strength):
        self.SName = SName
        self.Strength = Strength

D = Department()
P = Section('P Section',70)

print(D.University)
print(D.Location)

print(P.Program)
print(P.SName)
print(P.Strength)
```

```
PES University
B-Block 3 and 4 Floor
B.Tech First Semester
P Section
70
```

OOP - Hybrid Inheritance in python

- **Features** of **more** than **one** type of **inheritance** are **mixed** to form **Hybrid** Inheritance.




```
# Hybrid Inheritance Example
# Class 1
class University:
    UName = 'PES University'
    Program = 'B.Tech First Semester'
    Session = 'September 2021 - March 2022'

class Department(University):
    DName = 'EC'
    Location = 'B-Block 3 and 4 Floor'

class Section(University):
    def __init__(self,SName, Strength):
        self.SName = SName
        self.Strength = Strength

class Student(Department,Section):
    def __init__(self,SObject,SRN, SName):
        self.University = 'PESU'
        self.SRN = SRN
        self.SName = SName
        self.SObject = SObject

U = University()
D = Department()
P = Section('Q',65)
S = Student(Section('P',70),'PES1','Test1')

print(U.UName)
print(D.Location)
print(P.Strength)
print(S.SName,S.SRN,S.SObject.SName,S.SObject.Strength)
```

```
PES University
B-Block 3 and 4 Floor
65
Test1 PES1 P 70
```

PES UNIVERSITY ONLINE

PES UNIVERSITY ONLINE

End of class #49, #50

Thank you

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University
nitin.pujari@pes.edu

For Course Digital Deliverables visit www.pesuacademy.com