Lecture 27
Polymorphism
Class Method vs Abstract Method
Invoking Classes

OOP – Spring 2022 (Python)

Polymorphism

Function Overloading

Function Overriding

Polymorphism - Polymorphism with Function and Objects

```
class A:
    def f1(self):
        print('A F1')
    def f2(self):
        print('A F2')
class B:
    def f1(self):
        print('B F1')
    def f2(self):
        print('B F2')

def func(obj):
    obj.f1()
    obj.f2()
```

Output:

```
A F1
A F2
B F1
B F2
```

```
obj_A = A()
obj_B = B()
func(obj_A)
func(obj_B)
```

Polymorphism - Polymorphism with Class Methods

```
class A:
    def f1(self):
        print('A F1')
    def f2(self):
        print('A F2')
class B:
    def f1(self):
        print('B F1')
    def f2(self):
        print('B F2')
```

Output:

```
A F1
A F2
B F1
B F2
```

```
obj_a = A()
obj_b = B()
for obj in (obj_a,
obj_b):
    obj.f1()
    obj.f2()
```

Polymorphism - Polymorphism with Inheritance

```
class Batsman:
 def intro(self):
  print('Types of batsman')
def play(self):
  print('Batsman play good')

class Opener(Batsman):
 def play(self):
  print('Opener face new ball')

class MiddleOrder(Batsman):
 def play(self):
  print('Middle Order face old ball')

class HardHitter(Batsman):
 def play(self):
  print('Hard Hitter hit boundaries')
```

Polymorphism - Polymorphism with Inheritance

```
obj_batsman = Batsman()
obj_openner = Openner()
obj_middle_order =
MiddleOrder()
obj_hard_hitter =
HardHitter()

obj_batsman.intro()
obj_batsman.play()

obj_openner.intro()
obj_openner.play()

obj_middle_order.intro()
obj_middle_order.play()

obj_hard_hitter.intro()
obj_hard_hitter.play()
```

Output:

```
Types of batsman
Batsman play good
Types of batsman
Opener face new ball
Types of batsman
Middle Order face old ball
Types of batsman
Hard Hitter hit boundaries
```