what is Polymorphism?
---

in Generally Polymorphism means we can define one name in many forms.

Poly means Many
Morphism means Forms

the concept of defineing multiple logics/functionalites to perform a single action/operation,is known as a Polymorphism.

in generally,the Polymorphism can be categorized into two types,they are

1).Static/Compiletime Polymorphism

method overloading

2).Dynamic/Runtime Polymorphism

method overriding

method overloading
---

the concept of defineing multiple methods with same name but different no.of parameters with in the same class,is known as a method overloading.

ex1:
---

```
class test:
    def m1(self):
        print("hai")
    def m1(self,a):
        print("hello")
    def m1(self,a,b):
        print("good evening")
t1=test()
#t1.m1()
#t1.m1(4)
t1.m1(4,5)
```

output:
---

```
good evening
```

ex2:
---

```
class test:
    def m1(self,a):
        print("hello")
    def m1(self,a,b):
```

```
        print("good evening")
    def m1(self):
        print("hai")
t1=test()
t1.m1()
#t1.m1(4)
#t1.m1(4,5)
```

output:
---
hai

note:
---
python dont supporting method overloading concept because our python interpreter to recognize recent defined method only.

constructor overloading
---
        the concept of defineing multiple constructors with same name but different no.of parameters with in the same class,is known as a constructor overloading.

ex:
--
```
class test:
    def __init__(self):
        print("hai")
    def __init__(self,a):
        print("hello")
    def __init__(self,a,b):
        print("good evening")
#t1=test()
#t1=test(4)
t1=test(4,5)
```

output:
---
good evening

ex2:
---
```
class test:
    def __init__(self,a):
        print("hello")
    def __init__(self,a,b):
        print("good evening")
    def __init__(self):
        print("hai")
t1=test()
#t1=test(4)
```

```
#t1=test(4,5)
```

output:
---
hai

note:
---
python dont supporting constructor overloading concept also because the python interpreter to recognize last-defined constructor only.

Method Overriding?
---
the concept of defineing multiple methods with same name and same no.of parameters but we can define one method in super class and another method in sub class,is known as a Method Overriding.

ex:
---
```
class test:
    def m1(self):
        print("hai")
class demo(test):
    def m1(self):
        print("hello")
d1=demo()
d1.m1()
```

output:
---
hello

if we want to execute the super class methods in method overriding concept in that case we are using super()

ex2:
---
```
class test:
    def m1(self):
        print("hai")
class demo(test):
    def m1(self):
        super().m1()
        print("hello")
d1=demo()
d1.m1()
```

output:
---
hai

```
hello
```

ex3:

```
class sample:
    def m1(self):
        print("Good afternoon")
class test(sample):
    def m1(self):
        super().m1()
        print("hai")
class demo(test):
    def m1(self):
        super().m1()
        print("hello")
d1=demo()
d1.m1()
```

output:

```
Good afternoon
hai
hello
```

constructor Overriding:

the concept of defineing multiple constructors with the same name and same no.of parameters but we can define one constructor in super class and another constructor in sub class,is known as a constructor Overriding.

ex1:

```
class test:
    def __init__(self):
        print("hai")
class demo(test):
    def __init__(self):
        print("hello")
d1=demo()
```

output:

```
hello
```

in constructor overriding,by default the subclass constructor only executed,if we want to execute super class constructor in that case we are using super().

ex:

```
class test:
```

```
    def __init__(self):
        print("hai")
class demo(test):
    def __init__(self):
        super().__init__()
        print("hello")
d1=demo()

output:
------
hai
hello
```