POLYMORPHISM IN PYTHON

What is 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 inbuilt polymorphic functions:

```
# Python program to demonstrate in-built poly-

# morphic functions


# len() being used for a string

print(len("geeks"))


# len() being used for a list

print(len([10, 20, 30]))
```

Output:

```
5

3
```

Examples of used 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:

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

```
class India():

    def capital(self):

        print("New Delhi is the capital of India.")


    def language(self):

        print("Hindi the primary 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()
```

Output:

```
New Delhi is the capital of India.
Hindi the primary language of India.
India is a developing country.
Washington, D.C. is the capital of USA.
English is the primary language of USA.
USA is a developed country.
```

Polymorphism with Inheritance:

In Python, Polymorphism lets us define methods in the child class that have the same name as the methods in the parent class. In inheritance, the child class inherits the methods from the parent class. However, it is possible to modify a method in a child class that it has inherited from the parent class. This is particularly useful in cases where the method inherited from the parent class doesn’t quite fit the child class. In such cases, we re-implement the method in the child class. This process of re-implementing a method in the child class is known as Method Overriding.

```
class Bird:

def intro(self):

        print("There are many types of birds.")


def flight(self):

        print("Most of the birds can fly but some cannot.")


class sparrow(Bird):

def flight(self):

        print("Sparrows can fly.")


class ostrich(Bird):

def flight(self):

        print("Ostriches cannot fly.")


obj_bird = Bird()

obj_spr = sparrow()
```

```
obj_ost = ostrich()


obj_bird.intro()
obj_bird.flight()


obj_spr.intro()
obj_spr.flight()


obj_ost.intro()
obj_ost.flight()
```

Output:

```
There are many types of birds.
Most of the birds can fly but some cannot.
There are many types of birds.
Sparrows can fly.
There are many types of birds.
Ostriches cannot fly.
```

Polymorphism with a Function and objects:

It is also possible to create a function that can take any object, allowing for polymorphism. In this example, let's create a function called "func()" which will take an object which we will name "obj". Though we are using the name 'obj', any instantiated object will be able to be called into this function. Next, lets give the function something to do that uses the 'obj' object we passed to it. In this case lets call the three methods, viz., capital(), language() and type(), each of which is defined in the two classes ‘India’ and ‘USA’. Next, let’s create instantiations of both the ‘India’ and ‘USA’ classes if we don’t have them already. With those, we can call their action using the same func() function:

Code : Implementing Polymorphism with a Function

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

    def language(self):
        print("Hindi the primary 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.")

def func(obj):
        obj.capital()
        obj.language()
        obj.type()

obj_ind = India()
obj_usa = USA()

func(obj_ind)
func(obj_usa)
```

Output:

```
New Delhi is the capital of India.
Hindi the primary language of India.
India is a developing country.
Washington, D.C. is the capital of USA.
English is the primary language of USA.
USA is a developed country.
```