

# PYTHON OOPs

Polymorphism : Polymorphism is an object-oriented programming concept that refers to the ability of a variable, function or object to take on multiple forms.

- A language that features polymorphism allows developers to program in the general rather than program in the specific.
- Polymorphism can be carried out through inheritance, with subclasses making use of base class methods or overriding them.
- Polymorphism allows for flexibility and loose coupling so that code can be extended and easily maintained over time.

## Creating Polymorphic classes :

- Create two distinct classes to use with two distinct objects.
- These distinct classes need to have an interface that is in common so that they can be used.
- Make different methods with same name in two classes.

### Example :

```
class Ruling:

    def name(self):

        print ("Bharatiya Janata Party")

    def president(self):

        print ("Amit Shah")

    def year(self):

        print ("Founded on 6th April 1980")

class Opposition():

    def name(self):

        print ("Indian National Congress")
```

```

def president(self):
    print ("Sonia Gandhi")

def year(self):
    print ("Founded on 28th December 1885")

rule = Ruling()
rule.president()

opposite = Opposition()
opposite.president()

```

Output :

Amit Shah

Sonia Gandhi

- In the example we created two classes with three methods which are same in both classes , but there functionality is different.
- We instantiated these classes to two objects and same method from both the classes is called.

### Polymorphism with class methods :

- Python can use each of these different class types in the same way, we can first create a for loop that iterates through a tuple of objects.
- Calling the methods without being concerned about which class type each object is and we assume these methods exist in each class.

Example :

- For the above example of the political parties and using for to iterate over each method is shown below:

```

rule = Ruling()
opposite = Opposition()

```

```
for details in (rule,opposite):
```

```
    details.name()
```

```
    details.president()
```

```
    details.year()
```

Output :

Bharatiya Janata Party

Amit Shah

Founded on 6th April 1980

Indian National Congress

Sonia Gandhi

Founded on 28th December 1885

- Here First for loop iterates through rule object of Ruling class and then to Opposition class, so the methods of Ruling are executed first .

- By this python is using these methods in a way without considering what class type each of these objects is.

Polymorphism with functions :

- Create a function that can take any object for polymorphism.
- is\_politics() is the function and random which takes in the objects in example.

Example:

```
def is_politics(random):
```

```
    random.year()
```

```
rule = Ruling()
```

```
opposite = Opposition()
```

```
is_politics(rule)
```

```
is_politics(opposite)
```

Output :

Founded on 6th April 1980

Founded on 28th December 1885

### Example Explanation :

- In the example we created a function called `is_politics()` and `random` to take the object which are called.
- We will give some functionality to do that uses `random` object we passed to it i.e, `year()` method ,which is defined in both classes.
- We will create instantiations of both classes with these we can call their action using same `is_politics()`.
- Finally, though we passed a `random` object (`random`) into the `is_politics()` function when defining it, we were still able to use it effectively for instantiations of the `Ruling` and `Opposition` classes.
- The rule called the `year()` method defined in the `Ruling` class, and the `opposite` object called the `year()` method defined in the `Opposition` class.

**Note:** Without using polymorphism, a type check may be required before performing an action on an object to determine the correct method to call.