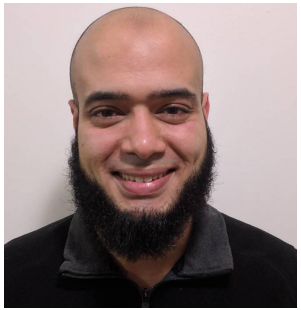
Python Programming

Special Methods Reflection

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Not Error!

- Sometimes python does a few trials to satisfy the operator BEFORE deciding no way and raises an error!
- We call it reflection
  - E.g. lt reflection is gt
    - A > B is same as B < A

```
class MyPair:
    def __init__(self, first, second):
        self.first = first
        self.second = second

    def __repr__(self):
        return f'({self.first}, {self.second})'

    def __lt__(self, other_pair):
        return self.first < other_pair.first and \
               self.second < other_pair.second


if __name__ == '__main__':

    p1 = MyPair(5, 10)
    p2 = MyPair(7, 13)

    print(p1 < p2)  # True
    print(p2 > p1)  # True:
    # Python calls automatically swaps and calls p1 < p2
```

An error!

- The add function returns NotImplemented value if the target object is not handled
- Hence it raises error
- So
  - MyPair + SingleValue Fails
- But if opposite is supported?
  - SingleValue + MyPair works!
- Python tries to swap and check out!

```
class MyPair:
    def __init__(self, first, second):
        self.first = first
        self.second = second

    def __add__(self, other):
        if not isinstance(other, MyPair):
            return NotImplemented
        return MyPair(self.first+other.first,
                      self.second+other.second)


class SingleValue:
    def __init__(self, val):
        self.val = val

if __name__ == '__main__':
    p = MyPair(2, 3)
    s = SingleValue(10)
    p = p + s
    # TypeError: unsupported operand type(s) for +:
    # 'MyPair' and 'SingleValue'
```

Reflection

- For **p+s** python: reflects IF __add__ is not defined or returned **NotImplemented**
- To reflect it
  - Swap them
  - Search for __radd__ but in the other class (SingleValue)
  - If exists, it calls it

```
class MyPair:
    def __init__(self, first, second):...

    def __repr__(self):...

    def __add__(self, other_pair):
        if not isinstance(other_pair, MyPair):
            return NotImplemented
        return MyPair(self.first + other_pair.first,
                      self.second + other_pair.second)

class SingleValue:
    def __init__(self, val):...

    def __radd__(self, mypair):
        if not isinstance(mypair, MyPair):
            return NotImplemented
        return MyPair(self.val + mypair.first,
                      self.val + mypair.second)

if __name__ == '__main__':
    p = MyPair(2, 3)
    s = SingleValue(10)
    p = p + s
    print(p)   # (12, 13)
```

Error Again

- s + p
  - Python searches for __add__ in Single Value
  - As it doesn’t exist, it searches for __radd__ in MyPair, which again doesn’t exist!
- So be careful from the flow
  - Python is systematic in searching

```
class MyPair:
    def __init__(self, first, second):...

    def __repr__(self):...

    def __add__(self, other_pair):...

class SingleValue:
    def __init__(self, val):...

    def __radd__(self, mypair):...

if __name__ == '__main__':
    p = MyPair(2, 3)
    s = SingleValue(10)
    p = s + p
    print(p)   # TypeError: unsupported
```

Other Operators for reflection

- __sub__ ⇒ __rsub__
- __mul__ ⇒ __rmul__
- __truediv__ ⇒ __rtruediv__
- __floordiv__ ⇒ __rfloordiv__
- __mod__ ⇒ __rmod__
- __pow__ ⇒ __rpow__
- __matmul__ ⇒ __rmatmul__

- **__lt__ ⇒ __gt__**
- **__gt__ ⇒ __lt__**
- __le__ ⇒ __ge__
- __ge__ ⇒ __le__
- **__ne__ ⇒ __eq__**
- **__eq__ ⇒ __ne__**

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."