Lecture 06: Inheritance extended

Swakkhar Shatabda

B.Sc. in Data Science
Department of Computer Science and Engineering
United International University

February 19, 2024

Two important methods

- isinstance and issubclass

```
class A:
    pass
class B(A):
    pass
print(issubclass(B,A))
b = B()
print(isinstance(b,A))
print(isinstance(b,B))
```

Object super class

```
class A:
    pass
dir(A)
```

```
['__class__', '__delattr__', '__dict__', '__dir__',
'__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__',  '__gt__', '__hash__', '__init__',
'__init_subclass__', '__le__','__lt__', '__module__',
'__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__', '__weakref__']
```

Object super class

- All classses are child of object superclass

```
class A(object):
    pass
a = A()
print(a.__class__)
```

```
<class '__main__.A'>
```

- When we are adding implementations of __str__(), we are actually overriding it.
- Thus it inherits all methods and attributes from object.

Operator Overloading

```
class Time:
    def __init__(self,h=0,m=0):
        self.hour = h
        self.min=m
    def __add__(self,other):
        return Time(self.hour+other.hour,self.min+other.min)
    def __str__(self):
        return f"{self.hour} hours {self.min} mins"
t1 = Time(10,30)
t2 = Time(5,10)
print(t1+t2)
```

Operator Overloading

Operator	Supporting Method
+	.__add__(self, other)
-	.__sub__(self, other)
*	.__mul__(self, other)
/	.__truediv__(self, other)
//	.__floordiv__(self, other)
%	.__mod__(self, other)
**	.__pow__(self, other[, modulo])

- Can you implement a - operator for the Time class?

Operator Overloading

```
class Time:
    def __init__(self,h=0,m=0):
        self.hour = h
        self.min=m
    def __le__(self,other):
        return self.min <= other.min if self.hour==other.hour
                else self.hour<other.hour
    def __str__(self):
        return f"{self.hour} hours {self.min} mins"
t1 = Time(5,10)
t2 = Time(5,5)
t3 = Time(4,5)
t4 = Time(6,10)
t5 = Time(5,11)
print(t2<=t1)
print(t3<=t1)
print(t4<=t1)
print(t5<=t1)
```

Problem with the other?

- What if the type of the self and other are different?

```
class Time:
    def __init__(self,h=0,m=0):
        self.hour = h
        self.min=m
    def __le__(self,other):
        return self.min <= other.min if self.hour==other.hour
                else self.hour<other.hour
    def __str__(self):
        return f"{self.hour} hours {self.min} mins"
t1 = Time(5,10)
print(t1<=5)
```

```
AttributeError: 'int' object has no attribute 'hour'
```

Operator Overloading

```
class Time:
    def __init__(self,h=0,m=0):
        self.hour = h
        self.min=m
    def __le__(self,other):
        return self.min <= other.min if self.hour==other.hour
                else self.hour<other.hour
    def __str__(self):
        return f"{self.hour} hours {self.min} mins"
t1 = Time(5,10)
t2 = Time(5,5)
t3 = Time(4,5)
t4 = Time(6,10)
t5 = Time(5,11)
print(t2<=t1)
print(t3<=t1)
print(t4<=t1)
print(t5<=t1)
```

Problem with the other?

- What if the type of the self and other are different?

```
class Time:
    def __init__(self,h=0,m=0):
        self.hour = h
        self.min=m
    def __add__(self,other):
        if isinstance(other,type(self)):
            return Time(self.hour+other.hour,self.min+other.min)
        elif isinstance(other,type(3)):
            return Time(self.hour+other,self.min)
        else:
            return None
    def __str__(self):
        return f"{self.hour} hours {self.min} mins"
t1 = Time(5,10)
print(t1+5)
```