G12_CS_OOP-4-AlgebraOfFunctions

January 17, 2020

1 Algebra of functions I : Pedestrian way

[3]:
```
#import re

class HigherFunc1:

    def __init__(self,code='x'):
        self.code = code
        self.f =  lambda x: eval(self.code)

    def __call__(self,x):
        return self.f(x)
    def __mul__(self,g):
        #return self.f(g.f(x))
        #nc = re.sub('x','('+g.code+')',self.code)
        nc = self.code.replace('x','('+g.code+')')
        return HigherFunc1(nc )
    def __add__(self,x):
        pass
    def __str__(self):
        return str('lambda(x){\n '+self.code+'\n}')
    def __repr__(self):
        return self.__str__()
    def __eq__(self, other):
        if isinstance(other, self.__class__):
            #return self.__dict__ == other.__dict__
            return self.code == other.code
        else:
            return False




f = HigherFunc1(' 2*x')
f(3)
```

[3]:
```
6
```

[11]:
```
f
```

[11]:
```
lambda(x){
  2*x
}
```

[4]:
```
g = HigherFunc1('x+1')
g(3)
```

[4]:
```
4
```

[5]:
```
h = HigherFunc1('x**2')
h(3)
```

[5]:
```
9
```

[6]:
```
h2 = h*(g*f)
h2(3)
```

[6]:
```
49
```

[7]:
```
h2b = (h*g)*f
h2b(3)
```

[7]:
```
49
```

[8]:
```
Id = HigherFunc1()
Id(3)
```

[8]:
```
3
```

[9]:
```
print(h2,h2b)
```

```
lambda(x){
 (( 2*x)+1)**2
} lambda(x){
 (( 2*x)+1)**2
}
```

[10]:
```
h2 == h2b
```

[10]:
```
True
```

[229]:
```
h2.__dict__
```

[229]:
```
{'code': '(( 2*x)+1)**2',
 'f': <function __main__.HigherFunc1.__init__.<locals>.<lambda>(x)>}
```

[230]:
```
h2b.__dict__
```

[230]:
```
{'code': '(( 2*x)+1)**2',
 'f': <function __main__.HigherFunc1.__init__.<locals>.<lambda>(x)>}
```

2 Algebra of Functions II

[234]:
```
class HigherFunc2:
    def __init__(self,f):
        self.f = f
    def __call__(self,*x,**kw):
        return self.f(*x,**kw)
    def __mul__(self,g):
        if not isinstance(g,HigherFunc2):
            return HigherFunc2(lambda *a, **kw: float(g)*self.f(*a,**kw))
        return HigherFunc2(lambda *a, **kw: self(g(*a, **kw)) )
    def __add__(self,g):
        if not isinstance(g,HigherFunc2):
            return HigherFunc2(lambda *a, **kw: self(*a,**kw)+float(g))
        return HigherFunc2(lambda *a,**kw: self(*a,**kw)+g.f(*a,**kw))
```

[235]:
```
def f2(x):
    return 2*x
def g2(x):
    return x+1
def hh(x):
    return x**2

F2 = HigherFunc2(f)
G2 = HigherFunc2(g)
H2 = HigherFunc2(hh)

I = (H2*(F2*G2))
I(3)
```

[235]:
```
64
```

[236]:
```
S2 = F2+G2
S2(3)
```

[236]:
```
10
```

[237]:
```
SM2 = F2*3
F2(3),SM2(3)
```

[237]:
```
(6, 18.0)
```

[238]:
```
SS2 = F2+4
SS2(3)
```

[238]:
```
10.0
```

3 Function (aka. Non Object-Oriented) Version

[17]:
```
def compose(r,s):
    def t(x):
        return r(s(x))
    return t
ch = compose(f,g)
ch2 = compose(g,f)
ch(3),ch2(3)
```

[17]:
```
(8, 7)
```

3.1 Problem: Can we use HigherFunc2 to define functions with 2 parameters and still have composition and/or addition available as they are?

[240]:
```
hf = HigherFunc2(lambda x,y: (x+1)*y)
hg = HigherFunc2(lambda x: (1,x) )
hc = hf*hg
a,b=hg(3)
print('a/b:',a,b)
hc(3)
```

```
a/b: 1 3
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-240-1e70603f2461> in <module>
  4 a,b=hg(3)
  5 print('a/b:',a,b)
----> 6 hc(3)


<ipython-input-234-e28e74c53a36> in __call__(self, *x, **kw)
  3         self.f = f
  4     def __call__(self,*x,**kw):
----> 5         return self.f(*x,**kw)
  6     def __mul__(self,g):
  7         if not isinstance(g,HigherFunc2):


<ipython-input-234-e28e74c53a36> in <lambda>(*a, **kw)
  7         if not isinstance(g,HigherFunc2):
```

```
        8             return HigherFunc2(lambda *a, **kw: float(g)*self.
↪f(*a,**kw))
   ----> 9         return HigherFunc2(lambda *a, **kw: self(g(*a, **kw)) )
       10     def __add__(self,g):
       11         if not isinstance(g,HigherFunc2):


        <ipython-input-234-e28e74c53a36> in __call__(self, *x, **kw)
          3         self.f = f
          4     def __call__(self,*x,**kw):
   ----> 5         return self.f(*x,**kw)
          6     def __mul__(self,g):
          7         if not isinstance(g,HigherFunc2):


        TypeError: <lambda>() missing 1 required positional argument: 'y'
```

[154]:
```
(lambda x,y: (x+1)*y)(*[1,3])
```

[154]:
```
6
```

[196]:
```
type(float(3))
```

[196]:
```
float
```

[205]:
```
(F2).__class__
```

[205]:
```
__main__.HigherFunc2
```

[207]:
```
type(F2) == type(G2)
```

[207]:
```
True
```

[210]:
```
isinstance(F2,(HigherFunc2))
```

[210]:
```
True
```

[242]:
```
a = [1,2,3]
b=[4,5]
a += b
a
```

[242]:
```
[1, 2, 3, 4, 5]
```

[ ]:
```
c = list(a)
```