Trees in Python

March 3, 2022

1 Trees in Python

1.1 Establishing the values of variables

[1]:
```
vars = {
        'x': 4,
        'y': 7
}
```

1.2 Class Declarations

1.2.1 Base Class Declarations

Creating the base class for expressions.

[2]:
```
class Exprs():
        pass
```

Defining the base class for Operators such as + and *.

[3]:
```
class Operators(Exprs):
        def __init__(self, l, r):
                self.l = l
                self.r = r

        def __str__(self):
                return f'({self.l} {self.symb} {self.r})'
```

1.2.2 Specific Operator Declarations

Defining specific classes for +, −, ∗ and /. Each of these classes has a seperate eval function.

[4]:
```
class Plus(Operators):
        def __init__(self, l, r):
                super().__init__(l, r)
                self.symb = '+'

        def eval(self, vars):
                return self.l.eval(vars) + self.r.eval(vars)
```

[5]:
```
class Minus(Operators):
        def __init__(self, l, r):
                super().__init__(l, r)
                self.symb = '-'

        def eval(self, vars):
                return self.l.eval(vars) - self.r.eval(vars)
```

[6]:
```
class Times(Operators):
        def __init__(self, l, r):
                super().__init__(l, r)
                self.symb = '*'

        def eval(self, vars):
                return self.l.eval(vars) * self.r.eval(vars)
```

[7]:
```
class Divide(Operators):
        def __init__(self, l, r):
                super().__init__(l, r)
                self.symb = '/'

        def eval(self, vars):
                return self.l.eval(vars) / self.r.eval(vars)
```

1.2.3 Constant and Variable Class Declarations

[8]:
```
class Const(Exprs):
        def __init__(self, value):
                self.value = value

        def __str__(self):
                return str(self.value)

        def eval(self, vars):
                return self.value
```

[9]:
```
class Var(Exprs):
        def __init__(self, name):
                self.name = name

        def __str__(self):
                return str(self.name)

        def eval(self, vars):
                return vars[self.name]
```

1.3 Object Creation

Creates the equation $5 * (4 + x)$ and display it in the correct form.

[10]:
```
e1 = Times(Const(5), Plus(Const(4), Var('x')))
```

[11]:
```
print(e1)
```

```
(5 * (4 + x))
```

Creates the equation $\frac{9+y}{8-x}$ and display it in the correct form.

[12]:
```
e2 = Divide(Plus(Const(9), Var('y')), Minus(Const(8), Var('x')))
```

[13]:
```
print(e2)
```

```
((9 + y) / (8 - x))
```

1.3.1 Evaluating Equations

Now, using the object function `eval()`, the equations can be evaluated, with variables taking their values from the `vars` dictionary.

[14]:
```
e1.eval(vars)
```

[14]:
```
40
```

[15]:
```
e2.eval(vars)
```

[15]:
```
4.0
```

1.4 Final Comments & Outcomes

This document outlines one use case of the tree abstract data structure and has expanded my knowledge and understanding of this topic.

Having implemented trees in Python, the next steps are as follows: - Address the issue of unnecessary brackets, see here - Explore other uses and aspects of trees - Including such information specifically pertaining to binary trees - Learn about ways to search trees

1.5 Attributions

Creator - Louis Stevens
Video referenced - Coding Trees in Python - Computerphile