Đã bắt đầu vào lúc	Tuesday, 7 September 2021, 2:03 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Tuesday, 7 September 2021, 4:15 PM
Thời gian thực hiện	2 giờ 12 phút
Điểm	3,00/3,00
Điểm	**10,00** của 10,00 (**100**%)

Câu hỏi **1**

Chính xác

Điểm 1,00 của 1,00

To express an arithmetic expression, there are 5 following classes:

Exp: general arithmetic expression

BinExp: an arithmetic expression that contains one binary operators (+,-,*,/) and two operands

UnExp: an arithmetic expression that contains one unary operator (+,-) and one operand

IntLit: an arithmetic expression that contains one integer number

FloatLit: an arithmetic expression that contains one floating point number

Define these classes in Python (their parents, attributes, methods) such that their objects can response to eval() message by returning the value of the expression. For example, let object x express the arithmetic expression 3 + 4 * 2.0, x.eval() must return 11.0

Answer: (penalty regime: 0 %)

```
29
30      def eval(self):
31          if self.op == "+":
32              return self.l+self.r
33          if self.op == "-":
34              return self.l-self.r
35          if self.op == "*":
36              return self.l*self.r
37          if self.op == "/":
38              return self.l/self.r
39
40  class UnExp(Exp):
41
42      def __init__(self,op:str,e):
43          self.op = op
44          self.e  = e
45
46      def eval(self):
47          if self.op == "+":
48              return +self.e.eval()
49          if self.op == "-":
50              return -self.e.eval()
51
52
```

	Test	Expected	Got	
✔	`print(x1.eval())`	`1`	`1`	✔
✔	`print(x2.eval())`	`2.0`	`2.0`	✔
✔	`print(x3.eval())`	`2`	`2`	✔
✔	`print(x4.eval())`	`-1`	`-1`	✔
✔	`print(x5.eval())`	`7.0`	`7.0`	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **2**

Chính xác

Điểm 1,00 của 1,00

Extend the contents of classes Exp, BinExp, UnExp, IntLit, FloatLit such that they can response to printPrefix() message to return the string corresponding to the expression in prefix format. Note that, unary operator +/- is printed as +./-. in prefix format and there is a space after each operator or operand. For example, when receiving message printPrefix(), the object expressing the expression -4 + 3 * 2 will return the string "+ -. 4 * 3 2 "

Answer: (penalty regime: 0 %)

```
            return self.l_val+self.r_val
        if self.op == "-":
            return self.l_val-self.r_val
        if self.op == "*":
            return self.l_val*self.r_val
        if self.op == "/":
            return self.l_val/self.r_val

class UnExp(Exp):

    def __init__(self,op:str,e):
        self.op = op
        self.e  = e

    def eval(self):
        if self.op == "+":
            return +self.e.eval()
        if self.op == "-":
            return -self.e.eval()

    def printPrefix(self):
        _e = self.e.printPrefix()
        return self.op + ". " + str(_e)

```

	Test	Expected	Got	
✔	`print(x1.printPrefix())`	`1`	`1`	✔
✔	`print(x2.printPrefix())`	`2.0`	`2.0`	✔
✔	`print(x3.printPrefix())`	`+ 1 1`	`+ 1 1`	✔
✔	`print(x4.printPrefix())`	`-. 1`	`-. 1`	✔
✔	`print(x5.printPrefix())`	`+ -. 1 * 4 2.0`	`+ -. 1 * 4 2.0`	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **3**

Chính xác

Điểm 1,00 của 1,00

As in the previous question, when a task is added into expression classes, new methods are added into these classes. Please change the way these classes are implemented in such a way that these classes do not change their contents when new tasks are added into these classes:

- Define class Eval to calculate the value of an expression

- Define class PrintPrefix to return the string corresponding to the expression in prefix format

- Define class PrintPostfix to return the string corresponding to the expression in postfix format

Let x be an object expressing an expression, x.accept(Eval()) will return the value of the expression x, x.accept(PrintPrefix()) will return the expression in prefix format and x.accept(PrintPostfix()) will return the expression in postfix format.

Be careful that you are not allowed to use **type(), isinstance()** when implementing this exercise

Tip: Use Visitor pattern.

Answer: (penalty regime: 0 %)

```
class Eval:
    def visitBinExp(self, exp):
        if exp.op == "+":
            return exp.l.accept(self)+exp.r.accept(self)
        if exp.op == "-":
            return exp.l.accept(self)-exp.r.accept(self)
        if exp.op == "*":
            return exp.l.accept(Eval())*exp.r.accept(Eval())

    def visitUnExp(self, exp):
        if exp.op == "+":
            return +exp.e.accept(self)
        if exp.op == "-":
            return -exp.e.accept(self)

    def visitLit(self,lit):
        return lit.val


class PrintPrefix:
    def visitBinExp(self, exp):
        _l = exp.l.accept(self)
        _r = exp.r.accept(self)
        return exp.op + " " + _l + " " + _r
```

	Test	Expected	Got	
✔	print(x1.accept(Eval())) print(x1.accept(PrintPrefix())) print(x1.accept(PrintPostfix()))	1 1 1	1 1 1	✔
✔	print(x2.accept(Eval())) print(x2.accept(PrintPrefix())) print(x2.accept(PrintPostfix()))	2.0 2.0 2.0	2.0 2.0 2.0	✔
✔	print(x3.accept(Eval())) print(x3.accept(PrintPrefix())) print(x3.accept(PrintPostfix()))	2 + 1 1 1 1 +	2 + 1 1 1 1 +	✔
✔	print(x4.accept(Eval())) print(x4.accept(PrintPrefix())) print(x4.accept(PrintPostfix()))	-1 -. 1 1 -.	-1 -. 1 1 -.	✔

	Test	Expected	Got	

✔	`print(x5.accept(Eval()))`
	`print(x5.accept(PrintPrefix()))`
	`print(x5.accept(PrintPostfix()))`
		`7.0`
		`+ -. 1 * 4 2.0`
		`1 -. 4 2.0 * +`
			`7.0`
			`+ -. 1 * 4 2.0`
			`1 -. 4 2.0 * +`
				✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

◄ OOP Quiz

Chuyển tới...

Link Video of session 7/9/2021 ►