Question **1**

Correct

Marked out of 1.00

Flag question

Question text

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 %)

class Exp:

def eval(self):

pass

class BinExp(Exp):

def \_\_init\_\_(self, left, operator, right):

self.left = left

self.operator = operator

self.right = right

def eval(self):

if self.operator == '+':

return self.left.eval() + self.right.eval()

elif self.operator == '-':

return self.left.eval() - self.right.eval()

elif self.operator == '\*':

return self.left.eval() \* self.right.eval()

elif self.operator == '/':

return self.left.eval() / self.right.eval()

else:

raise ValueError("Invalid operator")

class UnExp(Exp):

def \_\_init\_\_(self, operator, operand):

self.operator = operator

self.operand = operand

def eval(self):

if self.operator == '+':

return self.operand.eval()

elif self.operator == '-':

return -self.operand.eval()

else:

raise ValueError("Invalid operator")

class IntLit(Exp):

def \_\_init\_\_(self, value):

self.value = value

def eval(self):

return self.value

class FloatLit(Exp):

def \_\_init\_\_(self, value):

self.value = value

def eval(self):

return self.value

# Example usage:

x1 = IntLit(1)

x2 = FloatLit(2.0)

x3 = IntLit(2)

x4 = IntLit(-1)

x5 = BinExp(x1, '+', FloatLit(6.0))

x6 = UnExp('-', FloatLit(8.0))

Feedback

|  | **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 |  |
|  | print(x6.eval()) | -8.0 | -8.0 |  |

Passed all tests!

Question **2**

Correct

Marked out of 1.00

Flag question

Question text

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 %)

class Exp:

def printPrefix(self):

pass

class BinExp(Exp):

def \_\_init\_\_(self, left, operator, right):

self.left = left

self.operator = operator

self.right = right

def printPrefix(self):

return f"{self.operator} {self.left.printPrefix()} {self.right.printPrefix()}"

class UnExp(Exp):

def \_\_init\_\_(self, operator, operand):

self.operator = operator

self.operand = operand

def printPrefix(self):

return f"{self.operator}. {self.operand.printPrefix()}" # Include the period after the unary operator

class IntLit(Exp):

def \_\_init\_\_(self, value):

self.value = value

def printPrefix(self):

return str(self.value)

class FloatLit(Exp):

def \_\_init\_\_(self, value):

self.value = value

def printPrefix(self):

return str(self.value)

# Example usage:

x1 = IntLit(4)

x2 = IntLit(3)

x3 = IntLit(2)

# Creating the expression: -4 + 3 \* 2

unary\_minus = UnExp("-.", x1)

bin\_mult = BinExp(x2, "\*", x3)

bin\_plus = BinExp(unary\_minus, "+", bin\_mult)

Feedback

|  | **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!

Question **3**

Correct

Marked out of 1.00

Flag question

Question text

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 Exp:

def accept(self, visitor):

pass

class BinExp(Exp):

def \_\_init\_\_(self, left, operator, right):

self.left = left

self.operator = operator

self.right = right

def accept(self, visitor):

return visitor.visit\_binexp(self)

class UnExp(Exp):

def \_\_init\_\_(self, operator, operand):

self.operator = operator

self.operand = operand

def accept(self, visitor):

return visitor.visit\_unexp(self)

class IntLit(Exp):

def \_\_init\_\_(self, value):

self.value = value

def accept(self, visitor):

return visitor.visit\_intlit(self)

class FloatLit(Exp):

def \_\_init\_\_(self, value):

self.value = value

def accept(self, visitor):

return visitor.visit\_floatlit(self)

class Eval:

def visit\_binexp(self, exp):

if exp.operator == '+':

return exp.left.accept(self) + exp.right.accept(self)

elif exp.operator == '-':

return exp.left.accept(self) - exp.right.accept(self)

elif exp.operator == '\*':

return exp.left.accept(self) \* exp.right.accept(self)

elif exp.operator == '/':

return exp.left.accept(self) / exp.right.accept(self)

def visit\_unexp(self, exp):

if exp.operator == '+':

return exp.operand.accept(self)

elif exp.operator == '-':

return -exp.operand.accept(self)

def visit\_intlit(self, exp):

return exp.value

def visit\_floatlit(self, exp):

return exp.value

class PrintPrefix:

def visit\_binexp(self, exp):

return f"{exp.operator} {exp.left.accept(self)} {exp.right.accept(self)}"

def visit\_unexp(self, exp):

return f"{exp.operator}. {exp.operand.accept(self)}" # Include the period after the unary operator

def visit\_intlit(self, exp):

return str(exp.value)

def visit\_floatlit(self, exp):

return str(exp.value)

class PrintPostfix:

def visit\_binexp(self, exp):

return f"{exp.left.accept(self)} {exp.right.accept(self)} {exp.operator}"

def visit\_unexp(self, exp):

return f"{exp.operand.accept(self)} {exp.operator}."

def visit\_intlit(self, exp):

return str(exp.value)

def visit\_floatlit(self, exp):

return str(exp.value)

# Example usage:

x1 = IntLit(1)

x2 = FloatLit(2.0)

x3 = IntLit(2)

x4 = IntLit(-1)

x5 = BinExp(x1, '+', FloatLit(6.0))

# Creating the expression: -4 + 3 \* 2

x6 = BinExp(UnExp("-.", IntLit(1)), '+', BinExp(IntLit(4), '\*', FloatLit(2.0)))

Feedback

|  | **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 -. |  |
|  | 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!