Exercises - Lab 2

Try the following in lab, or on your own.

1. Consider the expression:

```
a or not b or (not a and b) and (b or a)
```

now consider every possible combination of `a` and `b`. Evaluate the above expression for each combination of values.

2. Consider the function `my_and(a,b)`, which returns True if and only if `a` and `b` are True. Note this is essentially recreating the already built in "`and`" operator. Implement the function `my_and(a,b)`, but only use the two operators: `not`, `or`. Hint: consider the expression `not(a and b)`, reason this out; is there an equivalent way to write it without the use of `and`?

3. Assume the following two functions have be run in the Python shell/interpreter:

```
def math1(x,y,z):
        z = a + 1
        c = x + 1
        y = c + 1
        return x + y + z

def math2(a,b,c):
        a = a + c
        b = math1(a,b,c)
        c = x + 1
        return a + b + c
```

For each of the following examples write what will be outputted to the screen in place of each "???", if anything at all. If at any point you think Python will error, state why.

a)

```
>>> math1(1,1,1)
???
```

b)

```
>>> a = 2
>>> c = 4
>>> b = math1(a,a,a)
>>> a
???
>>> b
???
>>> c
???
>>> z
???
```

c)

```
>>> a = 1
>>> x = 2
>>> math2(x,a,x)
???
>>> x
???
>>> a
???
```

Now consider adding the following two functions to the shell/interpreter (note they are both very similar to `math2`)

```
def math3(a,b,c):
        a = a + c
        b = math1(a,b,c)
        x = 5
        c = x + 1
        return a + b + c

def math4(a,b,c):
        a = a + c
        b = math1(a,b,c)
        c = x + 1
        x = 5
        return a + b + c
```

d)

```
>>> a = 1
>>> x = 1
>>> math3(x,a,x)
???
>>> x
???
```

e)

```
>>> a = 1
>>> x = 1
>>> math4(x,a,x)
???
>>> x
???
```

4. Write a function `get_circle_info(radius)` which computes the area and circumference of a circle.

For example:

```
>>> radius = 5
>>> get_circle_info(radius)
The area of the circle is: 78.54
The circumference of the circle is: 31.42
>>>
```

5. In the sample code below, how does Python know which namespace/instance of `num` to use/print?

Sample Code:

```
num = 1

def my_function():
        num = 5
        print(num, 'inside my_function')

print(num, 'in global/main')

my_function()
```

Output:

```
1 in global/main
5 inside my_function()
```

6. Write a function called in_range that takes an integer parameter and returns true if the value of the parameter is an integer from 1 to 3 inclusive. The only operators you are allowed to use in your code is arithmetic/comparison/Boolean.You are not allowed to use "if" statement or other concepts not covered in class.

Examples of use:

```
>>> in_range(1)
True
>>> in_range(2)
True
>>> in_range(3)
True
>>> in_range(0)
False
>>> in_range(3.4)
False
>>> in_range(2.5)
False
>>> in_range(1.0)
True
```

7. What is the following code print out?

```
def f1(x):
    return x == f4(x**2)

def f2(y):
    return y**2 + 100

def f3(x):
    return x**1 + 25

def f4(z):
    y = z**2
    return f2(z) + f3(y) + f5(y)

def f5(q):
    return 55

x = 20
q = 30
y = 40

final = (y and f1(12)) or (f3(q) == 55)
print(final)
```

8. If a dependent child is a person under 18 years of age who does not earn $10,000 or more a year, which expression would define a dependent child?

```
A. age < 18 and salary < 10000

B. age < 18 or salary < 10000

C. age <= 18 and salary <= 10000

D. age <= 18 or salary <= 10000
```

This question is mean't to help you translate an English condition to code. It was taken from a past exam.

9. Write a function that takes as input three lengths and returns true only when there is a **right**-triangle with these sides. **Do not** use an if-statement. Use the following function header.

```
def exists_triangle(x, y, z):
    """(float, float, float) -> bool
    Returns true only when there is a right-triangle with sides x, y, z.
    >>> exists_triangle(3,4,5)
    True
    >>> exists_triangle(5,3,4)
    True
    >>> exists_triangle(1,0,10)
    False
    """
```

Follow up: Consider that no hypotenuse among the sides was specified. However, the greatest length among the inputs can be assumed to be the hypotenuse. Can you write the above function (again without if-statements) that first determines the hypotenuse among x, y, and z?

10. Write a function that takes the coefficients from ax^2 + bx + c and

returns the largest solution to the quadratic equation. You may assume that input will always have real solutions. Use the following header

```
def solver(a, b, c):
    """(float, float, float) -> float
    Returns the largest x such that ax^2+bx+c=0
    >>> solver(2, -8, 6)
    3
    """
```

...More questions to come.

Challenge Question

1. The purpose of this question is to verify the associativity and commutativity of Python's standard algebraic operations `+ - * / //` and `**`, and the distributivity of `+` over `- * /` and `**`.

Recall: an operator **o** is said to be ***left associative*** if an expression **A o B o C** is interpreted as **(A o B) o C** and is said to be ***right associative*** if an expression **A o B o C** is interpreted as **A o (B o C)**. Additionally, an operator **o** is said to be ***commutative*** if **A o B = B o A** and an operator **o** and `#` are said to be ***distributive over*** `#` if **A # (B o C) = (A # B) o (A # C)**

You are to write a Python function which will test the associativity, commutativity, and distributivity of each of Python's standard algebraic operations.

The idea is to check if `(A + B) == (B + A)` then display that `+ is commutative`, otherwise display that `+ is not commutative`.

For example:

```
>>> a = 2
>>> b = 3
>>> c = 4
>>> my_associativity_checker(a,b,c)
+ is left associative
+ is commutative
...
+ is not distributive over /
+ is not distributive over **
>>>
```

#pin