PHY1112: Assignment 2

> Operator Overload

Assigned: January 16th, 2024

Due: January 23rd, 2024

Learning Objectives

1. Practice use of mathematical and logical operators.
2. Investigate how different data types behave with different operators
3. Practice type conversion
4. Practice manipulating lists

Grade Breakdown

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Part | 1 | 2 | 3 | 4 | Total |
| Points | 5 | 5 | 5 | 6 | 21 |
| Score |  |  |  |  |  |

**Question 1:**

1. Write a script that asks for a number input from the user and stores it to a variable.   
   **(1 mark)**
2. Multiply the variable by 30 and print out the result to the terminal. Describe what happens when you run your script and input a float number (*i.e.*, with a decimal point and fractional part).  
   **(1 mark)**
3. Convert the variable to a float, multiply it by 30 and print out the result. What happens now when you run your script and input a float number?  
   **(1 mark)**
4. Instead of converting the input to a float, convert it to an integer instead, and print out the result after conversion. Run your script and input a float number. What happens to the number you input after this conversion?  
   **(1 mark)**

**(5 marks total, 1 for file header/variable naming/comments)**

**Question 2: Varying “Degrees” of Frustration**

Like most programming languages, Python works exclusively in radians, not degrees. Forgetting that Python works like this can result in a great deal of frustration. Write a script that:

1. asks the user to input an angle in degrees and stores this into a variable. Note that the input function returns a string by default, and thus will need to be converted to a float.  
   **(1 mark)**
2. converts this angle to radians using , and prints this to the terminal.  
   **(1 mark)**

As many trigonometric functions are periodic with period radians, it is often useful to force angles to be in the range 0 to radians, by adding or subtracting multiples of . This can be accomplished using the modulo operator “%”.

1. In your script, use the “%” operator to convert the angle in radians to a value between 0 and radians, and print this answer to the terminal.

**(1 mark)**

1. Use your script to convert 960 degrees to an angle in radians between 0 and 2π. What does your script return? Do this calculation by hand to confirm your answer.  
   **(1 mark)**

**(5 marks total, 1 for file header/variable naming/comments)**

**Question 3: NOR and XOR**

`or`, `and`, `not` are logical operations that were discussed in class. Some others include:

**XOR**, a logical operation on two Boolean that returns a True only if exactly one of the (*not* both) inputs are True. **[[1]](#footnote-2)**

**NOR**, a logical operation on two Boolean that returns a True only if both inputs are False.

1. Write a Python script that asks the user for two Boolean inputs, then stores these in two Boolean variables, which are to be called `condition1` and `condition2`.

**(1 mark)**

Next, using combinations of Boolean operators `or`, `and`, `not`, determine in your script the following. (Output all results to the terminal)

1. `condition1` **NOR** `condition2`   
   **(1 mark)**
2. `condition1` **XOR** `condition2`   
   **(1 mark)**
3. Test your NOR and XOR output for all combinations of `condition1` and `condition2` and report your findings. The four possible combinations are:

|  |  |
| --- | --- |
| `condition1` | `condition2` |
| True | True |
| True | False |
| False | True |
| False | False |

**(1 mark)**

**(5 marks, 1 for file header/variable naming/comments)**

**Question 4: Were you list-ening?**

In class we discussed the various manipulations you can do to lists. This question will have you apply the various list manipulation methods.

Begin a new script by assigning a variable to the following example list:

[1, 2, 3, 1j, “2.0”]

Each part below is worth **0.5 marks.**

1. Using the `type` function, print the types of each of the elements.
2. Which function do you use to count the number of elements in a list?  
   Use this to print the total number of elements in the list.
3. Using the `append` method, add the integer values 4, 5, and 6 to the end of the list, and print the updated list.
4. Using the `extend` method, add the integer values 7, 8, and 9 to the end of the list, and print the updated list.
5. Using the `insert` method, insert the integer value 0 to the beginning of the list, and print the updated list.
6. Using the `index` method, find the index of 1j in the list, and print the updated list.
7. Using the `pop` method, remove the complex value 1j from the list, and print the updated list.
8. Using the `remove` method, remove the string “2.0” from the list, and print the updated list.
9. Using list spicing, create a new list that contains the 3rd to the 5th elements (inclusively, so there are 3 items) of the list from part ‘h’. Print this new list.
10. Using list splicing, and negative indices, create another new list that contains the 2nd element to the 2nd-last element (again inclusively) of the list from part ‘h’. Print this new list.

In your assignment word document, include a screenshot of all the results from parts a) to j)

**(6 marks total,** **1 for file header/variable naming/comments)**

**Code**

# Family name: Patrick Geraghty

# Student number:  300349204

# Course: PHY 1112

# Assignment: 2

# Year 2024

# Question 1

*def* question1():

    num = input("Enter a number: ")

    #End of part a

    print (num\*30)

    #End of part b. At this step, when question1 is run, the program will prompt the user for a string input. When any input is entered, the program attempts to multiply the input by 30. However, since the input is a string, the program will print the input 30 times.

    num = *float*(num)

    print (num\*30)

    #End of part c. At this step, when question1 is run, the program will prompt the user for a string input. When any input is entered, the program attempts to multiply the input by 30. However, since the input is a string, the program will print the input 30 times. Then, the input is converted to a float, and the program will print the input multiplied by 30. In the case that a decimal is entered, the number is multiplied by 30.

    num = *int*(num)

    print (num\*30)

    #End of part d. At this step, when question1 is run, the program will prompt the user for a string input. When any input is entered, the program attempts to multiply the input by 30. However, since the input is a string, the program will print the input 30 times. Then, the input is converted to an integer, and the program will print the input multiplied by 30. In the case that a decimal is entered, the number is rounded to the nearest whole number and the rounded value is multiplied by 30.

# Question 2

*def* question2():

    deg = *float*(input("Please enter an angle measured in degrees: "))

    rad = deg\*3.14159/180

    smallest\_rad = rad%(2\*3.14159)

    print(smallest\_rad)

    print(*str*(smallest\_rad/3.13159) + "\u03C0")

    #Upon testing with an angle of 960 degrees, the program prints 4.188786666666665. This is because the program is calculating the smallest radian measure of the angle, which is 4.188786666666665 radians (240 degrees).

# Question 3

*def* question3():

    condition1 = input("Enter a boolean value (T/F): ").lower()

    condition2 = input("Enter a boolean value (T/F): ").lower()

    if condition1 == "t":

        condition1 = True

    else:

        condition1 = False

    if condition2 == "t":

        condition2 = True

    else:

        condition2 = False

    nor = not (condition1 or condition2)

    print(nor)

    xor = condition1 ^ condition2

    print(xor)

    #Results of testing:

    #True True: False False

    #True False: False True

    #False True: False True

    #False False: True False

# Question 4

*def* question4():

    l = [1, 2, 3, 1*j*, "2.0"]

    for i in range (len(l)):

        print(*type*(l[i]))

    print(len(l))

    l.append(4)

    l.append(5)

    l.append(6)

    print(l)

    l.extend([7, 8, 9])

    print(l)

    l.insert(0, 0)

    print(l)

    l.index(1*j*)

    print(l)

    l.pop(l.index(1*j*))

    print(l)

    l.remove("2.0")

    print(l)

    l2 = l[2:5]

    print(l2)

    l3 = l[1:-1]

    print(l3)

1. Note that bitwise XOR is built into Python with the operator ^. [↑](#footnote-ref-2)