CptS 111 — Lab #4

Conditionals and Relational & Boolean Operators

Learning Objectives:

- Be comfortable using `if-(elif)-else` constructs
- Be comfortable using relational/comparison operators
- Be comfortable using Boolean/logical operators

Prerequisites:

- `if-(elif)-else` constructs
- Relational/comparison operators, e.g, `>, <=, ==`
- Boolean/logial operators, e.g, `and, or, not`
- Combination of relational and Boolean expressions

Task 1: Practicing conditionals in the IDLE Shell window

Recall that = is the assignment operator. It is used to assign an lvalue to the expression to its right, which may be a literal, a function, an equation, or something else. When we need to ascertain whether values are equal in Python, we can't use the assignment operator so we use two equal signs **==** rather than one. If we want to know whether a value isn't equal to something, we use **!=** which means not equal. If we want to know whether a value is larger or smaller than another value, we use **>** or **<**, respectively.

For this task, open your IDLE Shell window, and follow the directions given. As usual, results haven't been included. Be sure to read and understand all the comments, and as always, type all the boldface text. **Try to determine the output before you press the return key!**

```
>>> # Let's begin by using if statements in the two examples that follow.
>>> answer = input('Have you attended a WSU sporting event [y/n]? ')
Have you attended a WSU sporting event [y/n]? n
>>> if answer == 'y':
        print('Hope it was fun.')
>>> if answer == 'n':
        print('You should give it a try!')
>>>
>>> num = int(input('Enter an integer: ')
Enter an integer: 42
>>> if num < 0:
        print(num, 'is less than zero.')
>>> if (num % 2) == 0:         # If modulo 2 is zero, then even.
        print(num, 'is even.')
>>> if (num % 2) != 0:         # If modulo 2 not zero, then odd.
        print(num, 'is odd.')
>>>
```

```
>>> # We see that using if statements works perfectly fine.  However,
>>> # they're not efficient in the examples above.  In addition, using
>>> # successive if statements can lead to wrong results.  Consider
>>> # the following alternatives.
>>> answer = input('Have you attended a WSU sporting event [y/n]?')
Have you attended a WSU sporting event [y/n]? y
>>> if answer == 'y':
        print('Hope it was fun.')
    else:
        print('You should give it a try!')
>>>
>>> num = int(input('Enter an integer: ')
Enter an integer: 42
>>> if num < 0:
        print(num, 'is less than zero.')
    elif (num % 2) == 0:
        print(num, 'is even.')
    else:
        print(num, 'is odd.')
>>>
>>> # In the two examples above, Python stops executing statements as
>>> # soon as one is true.  Thus, it ignores the last two statements in
>>> # each of these examples which is more efficient than previously.
>>>
>>> # Note that the parentheses used with (num % 2) aren't necessary,
>>> # but they can help with readability.
>>>
>>> # Recall that almost everything in Python evaluates to True.  The only
>>> # objects that don't are False, 0, None, and anything that's empty.
>>> if 'SpongeBob':
        print('A string gives True!')
>>> if 42:
        print('An integer gives True!')
>>> if 1.618:
        print('A float gives True!')
>>> if 0:
        print('0 gives True!')      # Not!
>>> if '':
        print('\'\' gives True!')   # Not!
>>> if []:
        print('[] gives True!')     # Not!
>>>
>>> # Feel free to test more values to see whether they evaluate to
>>> # True or False!
```

After you've completed this task, show your work to your TA to get credit.

Task 2: Continuing with conditionals of greater complexity

Let's continue practicing conditionals in the IDLE Shell window. Again, read the comments, and enter the boldface text. Results are printed so you can check your work.

```
>>> # We'll begin by considering an algorithm using nested conditionals
>>> # to determine leap years.
>>> year = int(input('Enter a year [4 digits]: '))
Enter a year [4 digits]: 1600
>>> if (year % 4) != 0:       # Must be divisible by 4
        if year < 2020:
            print(year, ``wasn't a leap year.'')
        else:
            print(year, ``isn't a leap year.'')
    elif (year % 100) != 0:   # Can't be divisible by 100
        if year < 2020:
            print(year, 'was a leap year!')
        else:
            print(year, 'is a leap year!')
    elif (year % 400) != 0:   # Must be divisible by 400
        if year < 2020:
            print(year, ``wasn't a leap year.'')
        else:
            print(year, ``isn't a leap year.'')
    else:
        if year < 2020:
            print(year, 'was a leap year!')
        else:
            print(year, 'is a leap year!')

1600 was a leap year!
>>>
>>> # Sometimes we can't use an if-(elif)-else construct but must use
>>> # multiple if statements.  Consider, e.g., the following.
>>> bonus = 0
>>> answer = input('All-Star Game [y/n]: ')
All-Star Game [y/n]: n
>>> if answer == 'y':
        bonus += 25000
>>> answer = input('MVP [y/n]: ')
MVP [y/n]: y
>>> if answer == 'y':
        bonus += 75000
>>> answer = input('Gold Glove award [y/n]: ')
Gold Glove award [y/n]: n
>>> if answer == 'y':
        bonus += 50000
>>> answer = input('Home run champ [y/n]: ')
Home run champ [y/n]: y
>>> if answer == 'y':
        bonus += 25000
```

```
>>> answer = input('Batting average champ [y/n]: ')
Batting average champ [y/n]: n
>>> if answer == 'y':
        bonus += 25000
>>> print(f``This baseball player's bonus was ${bonus:,}.'')

This baseball player's bonus was $100,000.
>>>
>>> # In the example above, the player' bonus depended on how many if
>>> # statements were true!  Note how we can use a comma after the colon
>>> # in the replacement field and the result of doing so.  Pretty
>>> # cool, eh?  Let's look at one more example, this time using a
>>> # combination of relative and Boolean operators.
>>> age = int(input('Enter your age in years: '))
Enter your age in years: 29
>>> citizen = int(input('Enter number of years as a US citizen: '))
Enter number of years as a US citizen: 29
>>> if (age >= 30) and (citizen >= 9):
        print(``You're eligible to run for the House and Senate.'')
    elif (age >= 25) and (citizen >= 7):
        print(``You're eligible to run for the House but not the Senate.'')
    else:
        print(``Sorry.  You're ineligible at this time.'')

You're eligible to run for the House but not the Senate.
>>>
>>> # Note that we didn't have to use parentheses in the test statements,
>>> # but it makes the code easier to read.
```

After you've completed this task, show your work to your TA to get credit.

Task 3: BMI redux

In Lab #2, you wrote a program to calculate a user's body mass index (BMI) using the equation

$$\frac{703 \times weight}{(height)^2}$$

where $weight$ is weight (or, more precisely, mass) in pounds and $height$ is height in inches. In this task, you're going to extend this program, making it a little more useful. The table below gives classifications for BMIs.[1]

BMI	Classification
< 18.5	Underweight
18.5 to < 25	Normal Weight
≥ 25	Overweight

Open the program you saved in Lab #2, `lab2_t3.py`, and using the BMI you calculated, the table above, and an `if-elif-else` construct, extend your program to give the results shown in the

[1] These classifications are used in the medical field, but use here does not constitute agreement.

examples below (use appropriate messages for the three classifications not used in the examples). Don't forget to change the header information, and save your file in a Lab4 folder as `lab4_t3.py` before running your code. Don't just save it or you'll overwrite the code you wrote in Lab #2.

```
Enter weight [pounds]: 110
Enter height [inches]: 65
Your BMI is 18.3.
You are classified as underweight.

Enter weight [pounds]: 139.5
Enter height [inches]: 63.5
Your BMI is 24.3.
Your weight is classified as normal.

Enter weight [pounds]: 311
Enter height [inches]: 72
Your BMI is 42.2.
You are classified as overweight.
```

After your program is working properly, show it to your TA and demonstrate that it does what it should to get credit.

Task 4: Income tax bracket

Income tax rates for 2023 have been adjusted for inflation. For a person filing as a single taxpayer, marginal tax rates are given below, e.g., the marginal tax rate for someone making *more than* $95,375 is 24%.

```
      $0 -  $11,000: 10%
 $11,000 -  $44,725: 12%
 $44,725 -  $95,375: 22%
 $95,375 - $182,100: 24%
$182,100 - $231,250: 32%
$231,250 - $578,125: 35%
         > $578,125: 37%
```

In an IDLE Editor window, write a program (`lab4_t4.py`) that prompts a user for their income and uses an `if-elif-else` construct and either a combination of relational and Boolean operators or else operator chaining (the latter is unique to Python). See the examples below for the correct output.

```
Enter your annual income: 11000
Your marginal tax rate is 10%.

Enter your annual income: 44725
Your marginal tax rate is 12%.

Enter your annual income: 44726
Your marginal tax rate is 22%.
```

When your program is working properly, show it to your TA for credit.

That's all there is for this lab. Great job!