CS-UY 1114: Lab 3

Nested Conditionals, While Loops

You must get checked out by your lab CA prior to leaving early. **If you leave without being checked out, you will receive 0 credits for the lab.**

Restrictions

The Python structures that you use in this lab should be restricted to those you have learned in lecture so far. Please check with your teaching assistants in case you are unsure whether something is or is not allowed!

Create a new python file for each of the following problems.

Your files should be named *lab[num]_q[num].py* similar to homework naming conventions.

Problem 1: What Does It Print? + While Loops!

Solve this problem by hand (or a txt file). State the final values, and what should be printed after each loop. If there is an infinite loop, state that.

(a) What would be printed if y = 15?

```
x = 0
y = 15
if y % 2 == 0:
    if y > 5:
        x = y ** 2
    else:
        x += 1
else:
    if y > 5:
        y = y ** 2
    else:
        y += 1
print(x)
```

(b)

```
num = 1
while num < 28:
    print(num)
    num *= 3
```

(c)

```
num = 13
while num < 23:
    if num % 2 == 0:
        print(num)
    else:
        print("fizz")
    num += 1
```

(d)

```
num = 1
while num < 100:
    if num > 50:
        if num % 10 == 0:
            print(“FIZZ”)
        elif num % 5 == 0:
            print(“BUZZ”)
    else:
        if num % 10 == 0:
            print("fizz")
        elif num % 5 == 0:
            print("buzz")
    num += 1
```

Problem 2: It Pays to be Lucky

Try your luck in pulling one single Pokémon card from the new trading card pack! Pokémon trading cards each have a picture of a Pokémon along with its respective stats and attacks. Each card can come in different design variations in different rarities that can increase its resell value. Let's say the prices increase according to 3 categories, each with subcategories:

- **Rarity:** *Uncommon* cards receive a $5 price increase, and *rare* cards receive a $15 price increase.
- **Holographic-ness:** *Reverse holos* receive a $10 price increase, *holos* receive a $15 price increase, and *full holos* receive a $20 price increase.
- **Special Pokémon:** *Starters* receive a $5 price increase and *legendaries* receive a $30 price increase.

All Pokémon cards start at a $5 value.

Create a program that asks the user yes or no questions about their card. Your program should:

- Prompt **at least 3** inputs about the card's categories: if the card is of a special rarity, if the card is holographic, and if the card is a special Pokémon.
- For every category the user answers yes to, prompt the user with yes or no questions about the subcategories for that category
- Output the card's final resale price

The following are examples of possible outputs:

```
Welcome to the Pokémon card price calculator! Is your card of a
special rarity? (Y/N) N
Is your card holographic? (Y/N) N
Is your card of a special Pokémon? (Y/N) N
Your card has a final resale price of: $5
```

```
Welcome to the Pokémon card price calculator! Is your card of a
special rarity? (Y/N) Y
Is your card uncommon? (Y/N) Y
Is your card holographic? (Y/N) Y
Is your card a reverse holo? (Y/N) N
Is your card a holo? (Y/N) Y
Is your card of a special Pokémon? (Y/N) Y
Is your card of a starter? (Y/N) N
Is your card of a legendary? (Y/N) Y
Your card has a final resale price of: $55
```

Use this final output as a guideline to all the possible questions your program *could* ask the user:

```
Welcome to the Pokémon card price calculator! Is your card of a
special rarity? (Y/N) Y
Is your card uncommon? (Y/N) N
Is your card rare? (Y/N) N
Is your card holographic? (Y/N) Y
Is your card a reverse holo? (Y/N) N
Is your card a holo? (Y/N) N
Is your card a full holo? (Y/N) N
Is your card of a special Pokémon? (Y/N) Y
Is your card of a starter? (Y/N) N
Is your card of a legendary? (Y/N) N
Your card has a final resale price of: $5
```

Problem 3: Python and Friends

If you are reading this, you are probably really familiar with python now. A lot of people hate them, they can be difficult to work with, they can get very long, and they are very dangerous, sometimes even able to kill humans. They are some of the world's largest snakes after all. Wait, what were you thinking about when I mentioned python?

Anyways, there are many species of snakes in the world, some are dangerous and others are relatively safe. Therefore, it is important to classify the types of snakes. This is where you come in. Write a program that asks for information regarding a snake species, classifies what kind of snake it is, and prints out the snake along with fun facts about it.

The information you would need to ask from the user are whether the snake is venomous, whether the snake is small, and whether the snake is aggressive. You can assume the user will only answer `"yes"` or `"no"` for each question. A sample execution for collecting input may look like this:

```
Is the snake venomous? no
Is the snake small? yes
Is the snake aggressive? no
```

Next, identify what the type of snake is from the inputs. **You are NOT allowed to use AND and OR in your solutions. Instead, you should be using nested if statements (if statements inside another if statement).**
The possible types of snakes are:

Python Python

- Fun Facts: One of the largest and most famous snakes. However, they are pretty slow, and are commonly used to introduce people to learning about snakes.
- Characteristics:
  - Non-Venomous
  - Large
  - Non-Aggressive

Assembly Anaconda

- Fun Facts: Many people hate learning about these snakes, as they look very intimidating. In the Totally Official CS1114 Snake Register, they are said to love being in low level altitudes.
- Characteristics:
    - Non-Venomous
    - Large
    - Aggressive

Java Kingsnake

- Fun Facts: Very befitting of their name, they are objectively the most sophisticated snake species. One may even say they are very classy.
- Characteristics:
    - Non-Venomous
    - Small
    - Non-Aggressive

Javascript Treesnake

- Fun Facts: Despite its name, they are completely different from the Java Kingsnake. They like to lay on the nodes of a tree to browse through nearby animals for their next meal.
- Characteristics:
    - Non-Venomous
    - Small
    - Aggressive

C Serpent

- Fun Facts: Can be found in the sea. Has the ability to control their memory, being able to allocate parts of their brain for certain tasks and permanently delete information from their memories.
- Characteristics:
    - Venomous
    - Large
    - Non-Aggressive

C++ Cobra

- Fun Facts: Evolved from the C Serpents many years ago. Reports show they have the weird habit of pointing at objects with their tails.
- Characteristics:
    - Venomous
    - Large
    - Aggressive

Verilog Viper

- Fun Facts: Many people first see these snakes around architectures. Reports claim that they like to chew on circuit wires.
- Characteristics:
    - Venomous
    - Small
    - Non-Aggressive

Matlab Mamba

- Fun Facts: Commonly used to introduce mech-ies to working with snakes. They often hatch plots to catch their prey and enjoy graphical images.
- Characteristics:
    - Venomous
    - Small
    - Aggressive

Finally, print out the identified snake along with its fun fact. A full sample execution would look like this:

```
Is the snake venomous? no
Is the snake small? no
Is the snake aggressive? no
That is a Python Python. One of the largest and most famous
snakes. However, they are pretty slow, and are commonly used to
introduce people to learning about snakes.
```

Problem 4: Multiple Use Calculator

This problem is a brief extension on last week's single use calculator (Lab 3 Problem 4). This extension will let the user use the calculator as many times as they want.

1. Begin by copying your solution from the *Single Use Calculator* into a new file.
2. Now make the additions necessary to have the calculator run until the user inputs Q to quit. Otherwise they should just hit enter to continue making calculations.

The following is the expected output of the program:

```
This is a four operation calculator.
Hit enter to continue and Q to quit calculator:
Enter your first number: 2
Enter the operation (+, -, *, /): +
Enter your second number: 3
2.0 + 3.0 = 5.0
Hit enter to continue and Q to quit calculator:
Enter your first number: 3
Enter the operation (+, -, *, /): *
Enter your second number: 9
3.0 * 9.0 = 27.0
Hit enter to continue and Q to quit calculator:
Enter your first number: 3
Enter the operation (+, -, *, /): 2
Enter your second number: 3
3.0 2 3.0 is an invalid operation.
Hit enter to continue and Q to quit calculator: Q
Goodbye!
```

Problem 5: Multiple Use Calculator

Blackjack is a card game played between the player and a dealer where the goal is to get the sum of your cards as close to 21 as possible. We'll create a modified version of the game according to these rules:

- At the start of the round, the dealer will receive a random card sum within the range [2, 21]. The dealer cannot pick up more cards after this point
- At the start of the round, the player will receive a random card of value [1,11]
- A player may "DEAL" to receive another random card of value [1,11] or "STAND" to stay with the cards they currently have.

The win conditions are as follows:

- The player can win only if the sum of their carts is less than or equal 21 and of their cad sum is closer to 21 than the dealer's
- The player can tie if the sum of their cards is equal to the dealer's sum.
- Otherwise, the player loses.

Your program must:

1. Print a welcome message to the user
2. Each round, print the user's card sum and prompt the user for an action
3. Display to the user whether they won, lost or tied

Here are some examples outputs:

```
Welcome to Blackjack!
Your current card sum is: 6
What would you like to do next?: (DEAL, STAND) DEAL
Your current sum is: 16
What would you like to do next?: (DEAL, STAND) STAND
You won! Your card sum was 16 and the dealer's was 15
```

```
Welcome to Blackjack!
Your current card sum is: 4
What would you like to do next?: (DEAL, STAND) DEAL
Your current sum is: 12
```

```
What would you like to do next?: (DEAL, STAND) DEAL
Your current sum is: 14
What would you like to do next?: (DEAL, STAND) DEAL
Your current sum is: 23
What would you like to do next?: (DEAL, STAND) DEAL
You lost! Your card sum was 23 and the dealer’s was 14
```

HINT: Think about what type of loop to use for this and how to structure the loop before you start coding