CSCI 220L

Lab 3

Learning Objectives

- Develop Python programs that read input, produce output, and do arithmetic with loops.
- Develop Python programs that use the accumulator pattern.

Activities

Complete the following activities in the provided Python file. Make sure to include meaningful comments, input prompts, and output messages to receive full credit.

1. Gross pay

   Write a function, `calc_pay()`, that asks the user to enter their number of hours worked and pay rate, and outputs their total pay. The total pay should be rounded to two digits. For example:

   ```
   Enter hours worked: 35
   Enter pay rate: 2.75
   ----------------------------
   Total pay: $96.25
   ```

2. Finding a power

   You have found yourself stranded on a deserted island with a bad version of Python that does not support the ** exponentiation operator (or the `math.pow()` function). Use your knowledge of loops and the accumulator pattern to create a new function, `power()`, that asks the user to input a base and an exponent, computes the power, and displays it in the following format:

   ```
   4 ^ 3 = 64
   ```

   Your solution should not contain the ** operator or the `math.pow()` function.

3. Newton's method

   Newton's method (also called the Babylonian method) is a technique for approximating the square root of a number. Given a number *x* and an initial approximation *guess*, you can find a simple estimate of $\sqrt{x}$ with:

   $$guess_{next} = \frac{guess + \frac{x}{guess}}{2}$$

   By repeating this calculation, each time using the result as the next guess, you will get increasingly accurate guesses. You can use $x/2$ as the initial guess.

   Write a function, `approx_sqrt()`, that uses this method to approximate a square root. The function should ask the user for *x* and the number of times to refine the guess. The function's output should include the initial number and the final guess:

   ```
   Enter a number to square root: 9
   Enter a number of times to refine the approximation: 10
   The square root of 9 is approximately 3.0
   ```

4. Alternating sequences

   Write a function, `alternate()`, that asks the user for a quantity, and then displays that many terms from the following sequence. Note that a term is a single number, not a pair of numbers!

   0 6 0 6 0 6 0 6 ...

   For example, if the user enters 3, the function should display "0 6 0". If you know about conditionals, your solution cannot use them.

   Hint: If you're stuck on how to generate the repeating pattern, think about the different arithmetic operators that Python offers (`+ - * / // % **`). Each one creates a certain pattern; for instance, you know that `f(x) = x + 3` makes a linear pattern, while `f(x) = x**2` makes a parabolic pattern. Try experimenting with the operators you are less familiar with to see what kinds of patterns they create.

   Hint 2: You can use `print()`'s keyword argument `end` to modify or remove the automatic line breaks that Python adds after each print statement.

5. Drawing with loops

   Write a function, `flag()`, that draws the following simplified American flag using loops:

```
* * * * * * ==================================
* * * * * * ==================================
* * * * * * ==================================
* * * * * * ==================================
* * * * * * ==================================
=============================================
=============================================
=============================================
=============================================
=============================================
```

   (The flag is 45 characters wide and 10 lines tall.)

6. Drawing with loops, part 2

   Write a function, `triangle()`, that draws the following triangle using loops:

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *
```

   Hint: If you're having trouble, try drawing a solid square first. I like to start by writing a loop that displays a single row of the square, and then "wrapping" that loop in an outer loop that makes the row appear to repeat multiple times, thus creating the square.

7. Shopping cart

   a. Write a function, `receipt()`, that asks the user to enter a grocery item, price, and quantity, and then displays an itemized receipt as shown below:

```
Enter item: hot dog
Enter price: 2.00
Enter quantity: 5
RECEIPT 5 hot dog @ $2.00 = $10.00
--------------------------------
TOTAL: $10.00
```

   b. (Optional, but may help with the next part.) Extend your function to ask for a second item, and display the second item's receipt output immediately after it is entered. Remember to update the total cost at the end.

   c. Extend the function to allow the user to enter the number of groceries purchased at the beginning, and then output an itemized receipt with each item.

Upload `lab3.py` to the OAKS dropbox before the deadline. Make sure you have most of the exercises completed before your lab meeting.