ITEC 1150 Ch 2.1 Lab Projects

CONDITIONAL STATEMENTS, MORE FORMATTING & START ON VALIDATION

Program Development Plan (PDP)

This is a step-by-step process for successfully constructing an application. Follow these steps and repeat until you have successfully completed the program.

This is a reminder to use the PDP process.

You do not need to turn in your PDP document, but going through the process will help you design your programs.

PDP template -

1. Problem definition
2. Analysis of variables & functions
3. Steps in algorithm
4. Code (separate .py file with added comments)
5. Screen snips from testing
6. Keep track of any wanted or needed improvements for a future version

General Requirements

All assignments must meet the following requirements:

The program must start with header at top and include appropriate comments throughout. Header example:

```
"""
Author: Erik Granse
Date: 2024-09-02
Description: Calculate and display student's average grades
"""
```

- Ensure the output is *information*; it needs to be a statement which explains the value being displayed (for example, “The average grade is 12.34”). Simply outputting “12.34” is not sufficient.

General Requirements (cont.)

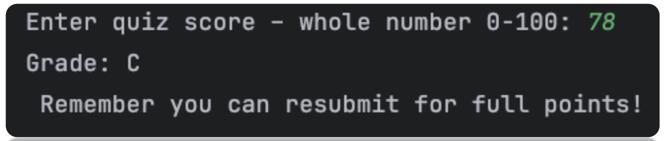
All assignments must meet the following requirements:

- The data in the program must be stored in variables.
- The output **must** come from variables in the program
  - Do not simply hard code the output value in the `print()` statement.
  - Some data will be given to you, and some will be user input—any calculations that need to happen must be in your program. Don't calculate somewhere else and enter the value into your program.

Lab Section 1: Grade Calculator

IMPORTANT: you should complete the PDP thinking process for every program. You do not need to turn in one.

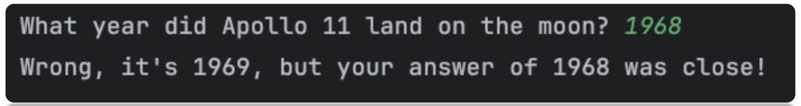
- Complete the grade.py program from the lesson slide 20. The program must:
  - Ask the user to input a whole number score from 0 to 100.
  - Display an error message if the value entered is not a number.
  - Display a letter grade for any numeric value entered.
  - Give additional feedback of your choosing for each grade (for example, “Grade: C. Remember you can resubmit for full points!”)
- Remember to run your program several times to test ALL the possible options.



```
Enter quiz score - whole number 0-100: 78
Grade: C
 Remember you can resubmit for full points!
```

Lab Section 2: Apollo Question

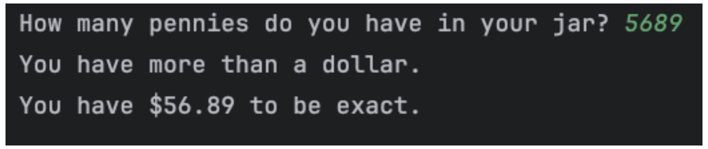
- Enhance the apollo.py program from slides 18-19 in the lesson.
- The program must:
  - Ask the user what year Apollo 1 landed on the moon
  - If the user answers correctly (1969), display a success message including the user's answer
  - If the user answers either 1968 or 1970, display a message including the user's answer and the correct answer, which says that their guess was close, but wrong
  - If the user answers more than 1 year away from 1969, display a message including the user's answer and the correct answer, which says that their guess was wrong



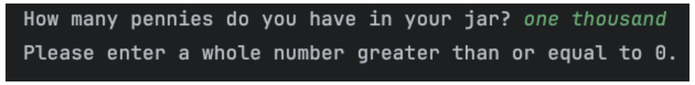
```
What year did Apollo 11 land on the moon? 1968
Wrong, it's 1969, but your answer of 1968 was close!
```

Lab Section 3: Penny Counter

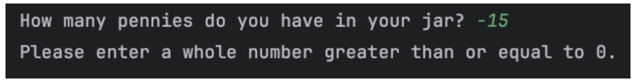
- Create a program named penny_counter.py, using the grade.py program as a model. The program must:
- Ask the user to enter the number of pennies in a jar (an integer).
- The program should give an appropriate message, and end if the user provides unexpected input (letters, negative numbers, etc.). Hint: adapt the validation code from the lesson, slide 20.
- If the user input passes validation, the program should use a conditional logic block to give them one of these three custom messages, depending on the number of pennies they enter:
  - You have less than a dollar OR
  - You have more than a dollar OR
  - You have exactly one dollar.
- End with a sentence, telling the user how much their pennies are worth in dollars and cents.



```
How many pennies do you have in your jar? 5689
You have more than a dollar.
You have $56.89 to be exact.
```



```
How many pennies do you have in your jar? one thousand
Please enter a whole number greater than or equal to 0.
```



```
How many pennies do you have in your jar? -15
Please enter a whole number greater than or equal to 0.
```

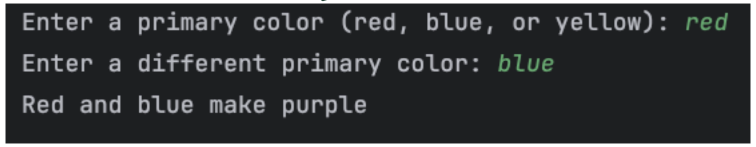
Lab Section 4: Color Mixer *or* Coin Counter

- Choose one of the following two python projects to complete:
  - color_mixer.py or
  - coin_counter.py:
- Those are shown on the next two slides

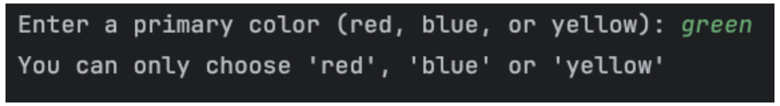
Lab Section 4: Color Mixer

- Choose 1 of color_mixer or coin_counter – you do not need to do both
- Create a program named **color_mixer.py**. When you mix red paint with blue, you get purple; when you mix red paint with yellow, you get orange; when you mix blue paint with yellow, you get green.
  - Write a program that asks the user to enter a 1st color choice from a set of options: red, yellow or blue, and then asks them to enter a 2nd color choice from the remaining 2.
  - Use conditional logic to control the flow of the program, figure out the mixed color, and report it back to the user in a nice-looking display.
  - Include feedback to handle unexpected input, such as entering the same color twice, or mis-spellings. If the user makes an error, the program can end after displaying the message.

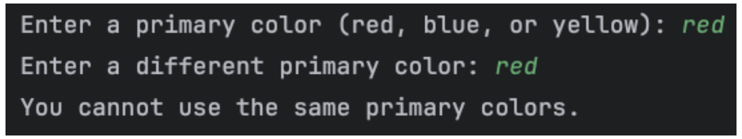
It's tricky to get this to work!
Solutions that are *close* are accepted



```
Enter a primary color (red, blue, or yellow): red
Enter a different primary color: blue
Red and blue make purple
```



```
Enter a primary color (red, blue, or yellow): green
You can only choose 'red', 'blue' or 'yellow'
```

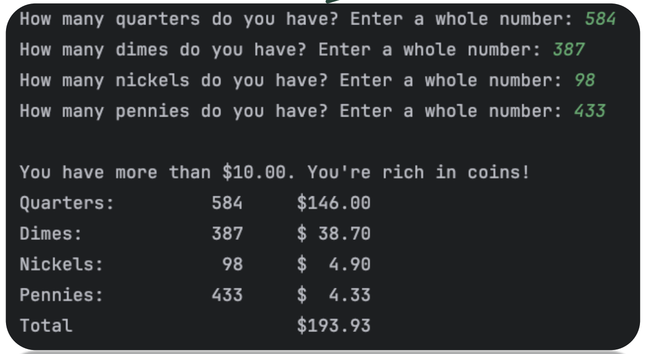


```
Enter a primary color (red, blue, or yellow): red
Enter a different primary color: red
You cannot use the same primary colors.
```

Lab Section 4: Coin Counter

- Choose 1 of color_mixer or coin_counter – you do not need to do both
- Create a program named **coin_counter.py**: A coin jar usually has mixed coins. The program must:
  - Ask the user how many quarters (worth $0.25), dimes (worth $0.10), nickels (worth $0.05), and pennies (worth $0.01) they have in their coin jar. (This means asking the user 4 questions.)
  - Add up the coins by both type and grand total.
  - Use a conditional logic block to display a custom message telling the user if they have more than $10, less than $10 or exactly $10.
  - Print all the data in a table showing the amount in each coin & total. Note: number columns align right!

You do not need to validate user input for this program.



```
How many quarters do you have? Enter a whole number: 584
How many dimes do you have? Enter a whole number: 387
How many nickels do you have? Enter a whole number: 98
How many pennies do you have? Enter a whole number: 433

You have more than $10.00. You're rich in coins!
Quarters:          584     $146.00
Dimes:             387     $ 38.70
Nickels:            98     $  4.90
Pennies:           433     $  4.33
Total                      $193.93
```

Upload!

- Files due
  - grade.py
  - apollo.py
  - penny_counter.py
  - AND <u>either</u> color_mixer.py <u>OR</u> coin_counter.py
  - Run ALL programs several times to prove that ALL the possible options work.
- Remember – all programs, comments and output must be informational & look organized! Be consistent with naming conventions, capitalization, spacing, alignment, single quote ' vs. double " etc.
- Next – submit your lab assignment on time & do all the prep for next class!
- Questions? Let me know!