Python Mickey's Multiple Unit Converter

Contents

Time required: 120 minutes

- Comment each line of code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Pseudocode

1. Write pseudocode for the exercise
2. Submit with the assignment

utils.py

We are going to build a utils file for Python that we can use in our future programs.

Create or add to an existing module file named **utils.py** You can add commonly used functions to this as we work with Python.

Enter the following code.

```python
"""
    Name: utils.py
    Author:
    Created:
    Purpose: A utilty module with commonly used functions
"""

def get_title(program_title):
    """
        Takes in a string argument
        returns a string with ascii decorations
    """
    # Get the length of the statement
    text_length = len(program_title)

    # Create the title string by concatenation

    title_string = "+--" + "-" * text_length + "--+\n"
    title_string = title_string + "|  " + program_title + "  |\n"
    title_string = title_string + "+--" + "-" * text_length + "--+"

    # Return the contatenated title string
    return title_string
```

```python
def get_float(prompt):
    """
        Get a float from the user with try catch
        The prompt string parameter is used to ask the user
        for the type of input needed
    """
    # Declare local variable
    num = 0

    # Ask the user for an input based on the what parameter
    num = input(prompt)

    # If the input is numeric, convert to float and return value
    try:
        return float(num)

    # If the input is not numeric,
    # Inform the user and ask for input again
    except ValueError:
        print(f'You entered: {num}, which is not a number.')
        print(f"Let's try that again.\n")

        # Call function from the beginning
        # This is a recursive function call
        return get_float(prompt)
```

Minimum Requirements

Create a Python program named **unit_converter.py**

The following pseudocode will get you started. Finish the pseudocode before you start coding. Comment the program to show the flow of code after you get it working.

```
# unit_converter.py
import utils
import convert

# Display menu

# Get menu input from user to determine which convert function to call

if menu_choice == 1:
    # Call utils.get_float() which will return a float

    # Call convert.cm_to_inches(cm) to return a formatted string
```

- From a menu, the user can choose from the conversions as shown in the example run.
- Get input from the user
- Call appropriate conversion functions from convert.py
- The results will be returned to unit_converter.py as a formatted string and displayed
- Use the **get_float()** function to get a float input from the user and return a float.
- Otherwise, the program should convert the length and print out the result.

```
# convert.py
cm_to_inches(input)
    # Pass float user input to function
    convert
    return formatted string

inches_to_cm(input)
    convert
    return formatted string

km_to_miles(input)
    convert
    return formatted string

miles_to_km(input)
    convert
    return formatted string
```

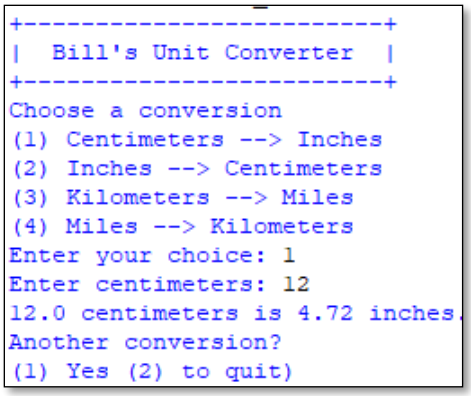
Add the following to your **convert.py file**

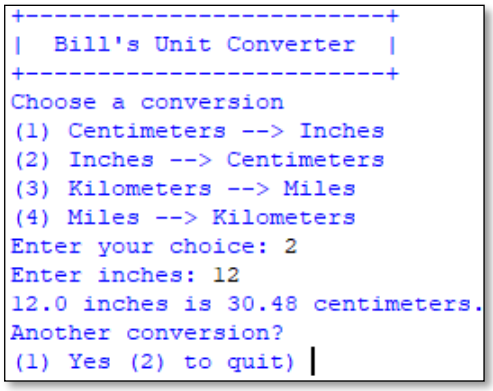
- Each function will be in the convert.py file.
- Each conversion will be a separate function with a float input argument.
- Round the results to 2 decimal places. Include thousands separator.
- Return the results as a formatted string.

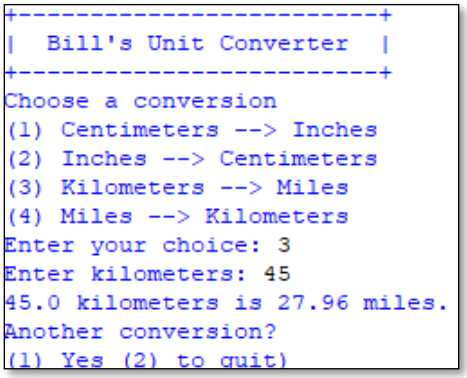
This example code shows how to import the **utils** module and use the **utils.get_title()** function. You would use a similar concept to use the **convert.py** module functions.

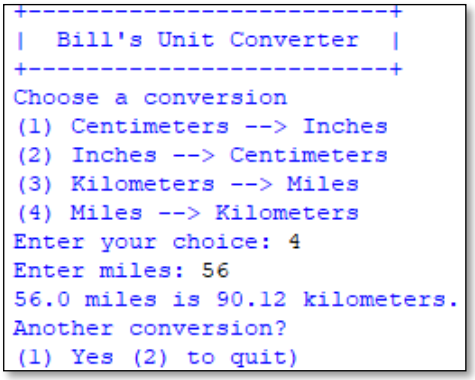
```
1  """
2      Name: unit_converter.py
3      Author:
4      Created:
5      Purpose: Convert from one measurement to another
6      Do not allow negative numbers
7  """
8  import utils
9
10 def main():
11     # Print the title of the program
12     print(utils.get_title("Unit Converter"))
13
```

Example runs:









Assignment Submission

1. Attach the pseudocode.
2. Attach all program files. unit_converter.py, utils.py, converter.py
3. Attach screenshots showing the successful operation of the program.
4. Submit in Blackboard.