ME/MSE 241 Engineering Computations

Homework Assignment – 2

Instructions:

1) *Include comments for every few lines (or even each line) of code as relevant.*
2) *Use a consistent naming convention.*
3) *Using the notebook format (.ipynb) is encouraged. Use a single cell for each problem solution. Name the file <First name>_<Last name>.ipynb.*
4) *Do not hesitate to ask for help, there are no trivial questions!*

Problem 1 (20 pts)

Srinivasa Ramanujan discovered a series formula for π.

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{99^2} \sum_{i=0}^{\infty} \frac{(4i)!}{i!^4} \frac{1103 + 26390i}{396^{4i}}$$

Write a Python code using <u>just</u> the arithmetic operators (i.e., no use of any functions) that approximates the value of π using the first three terms of the above series. You should find that this approximation is exact up to the first 16 digits of π.

Hint: Most problems in math/programming can be solved by simply following a series of well-defined instructions. Such instructions constitute an *algorithm* (think recipe). To solve this and similar problems, here's a simple algorithm that uses no function calls:

1. Define a variable for each of the first three terms and assign each variable the Python *expression* that computes the value
2. Add the variables
3. Take the inverse and print the value; this is your output
4. Optionally, compare your output with the actual value of $\pi \approx 3.141592653589793$
    a. You cannot use comparison operators to compare *float* values reliably
    b. Instead, closeness of two *float* values is determined by comparing the square of their difference with a small arbitrary positive number, say 10^{-6}

Here's how a variation of this algorithm works mathematically with a different, simpler series that computes the value of e (≈ 2.7183).

$$e = \sum_{i=0}^{\infty} \frac{1^i}{i!} \approx \frac{1^0}{0!} + \frac{1^1}{1!} + \frac{1^2}{2!} = 2.5$$

Clearly, this is not as effective as Ramanujan's formula for π. Including more terms will improve the approximation in this case.

<u>P.S.</u>: You must follow this same strategy for solving all the problems in this course. Think of an algorithm to solve the given problem, implement it as a program, and verify the implementation. If you see incorrect results, check the code and/or the algorithm for any errors or logical flaws.

Problem 2 (20 pts)

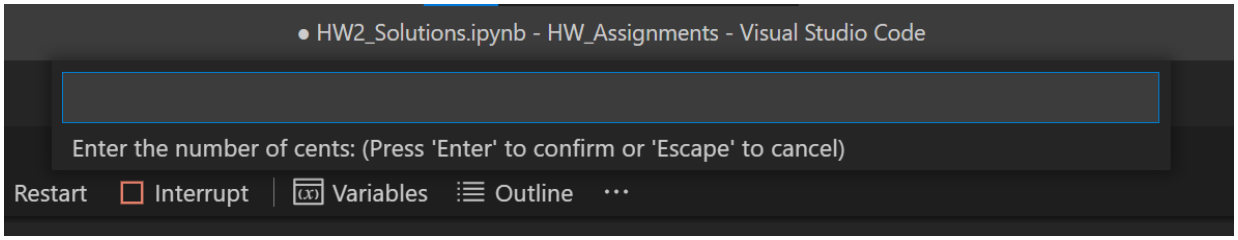
Given the number of cents, nickels, dimes, and quarters, write a program that prints the total amount of money in dollars.

The inputs (i.e., number of each coin type) should be obtained interactively from the user through the *input* function. Below example illustrates the usage of the *input* function.

```
>>> numCents = input("Enter the number of cents: ")
Enter the number of cents: 20
```

Note that the output of the *input* function is the entered text (20 in the above case) in the string format and needs to be converted to an appropriate datatype for calculating the total cash. Print the final computed value using the *print* function with appropriate text.

Note: With Python notebooks, the *input* function prompts for entering an input will appear at the top of the window right below the title bar (see the below image).



For more on the *input* function and its usage, refer to Chapter 3.1 of the textbook.

Problem 3 (20 pts)

Along the same lines as problem 2, write a program that calculates how much it costs to run an appliance over a 10-year period. Have the user enter the cost per kilowatt-hour in cents, the wattage of the appliance in Watts, and then the duration for which the appliance is used daily in hours. Assume that the input quantities would be of type *float*. Print the total cost in dollars.

Note: assume number of days per year to be 365.25 for problems 3 and 4.

Problem 4 (40 pts)

The module *time* provides a function *time* that computes the time since midnight 1st January 1970 (in the GMT time zone) (referred to as start of the epoch) in seconds (see below).

```
>>> import time
>>> time.time()
1643847032.9658227
```

Using the arithmetic operators, compute the number of whole years, weeks, days, hours, and minutes given the current time. Show the result using the following print function call:

```
print("Time since midnight 1st January 1970 is: {} years, {} weeks, {} days, {} hours, and {} minutes".format(years, weeks, days, hours, minutes))
```

In the above function call, *years*, *weeks*, *days*, *hours*, and *minutes* are variables that store the respective values.

This is the expect output when the program was run at 5:11 PM on 9/9/2022:

```
Time since midnight 1st January 1970 is: 52.0 years, 36.0 weeks, 0.0 days, 0.0 hours, and 11.0 minutes
```