Assignment 3

Due Mar 21th Wed, 5:00PM

Instructions:

* This is a computer-based assignment (Total points: 8). Please write your code in Python.

(1) Submission: Your codes/programs must be submitted electronically by email to sbai@fau.edu by 5:00 PM on the above due date. Use your **fau.edu email address** to send the code (unless you do not have one). Your email header/subject line should be:

MAD2502-HW3-Name

Include the Python code as an attachment with filename (put all your functions/codes in this single Python file):

MAD2502-HW3-Name.py

(2) Naming: A comment at the top of your program file should also identify yourself and the assignment that you are submitting:

```
# Assignment 3
# Name:
# Z-Number:
```

Please also follow the requirement(s) after each question to *name* your functions.

(3) Python commands/libraries: Do not use Python libraries unless it is mentioned in the question. Do not use any Python pre-defined commands/libraries that have not been discussed in this class. For example, do not use the Python built-in command that computes the sum or computes the square root.

(4) Comments/citation: Please comment your code. All code should be written by you. If any part of what you turn in is not your own work – you learn it from books, webpages etc – the source must be referenced.

(5) A random number

The following questions use your FAU Z-number (should be 8-digits). Please follow the instructions below to generate a random number t. Denote your Z-number by z.

- If your Z-number starts with 0, truncate the leading consecutive zeros, e.g. 00020202 becomes 20202. Then $z = 20202$.
- If your Z-number does not start with 0, keep all of it. E.g. $z = 22222222$.

In either case, you have a number z now. First, load the *random* module in your code (e.g. put the following line on top of your program),

```
import random
```

Initialize the random seed using your z and then generate a random number t.

```
### Replace 22222222 by your number z as described above.
random.seed(22222222)
# Use the following line. Do not change it.
t = random.uniform(0, 1)
```

In this example, the random numbers $t = 0.3964449878676$ was generated. It will be used in the following questions. You will have different t generated depending on your Z-number.

Question 1. (2 points)
Write a Python program to numerically approximate the value of

$$\sqrt{t+\sqrt{t+\sqrt{t+\sqrt{t+\cdots}}}} \tag{1}$$

where t is the random number generated previously.

Requirement: write a function with name "Approximate_Q1",

```
### Solution to Question 1
def Approximate_Q1(t):
        ... Your codes ...
```

Hints: approximate its value by computing $\sqrt{t}$, $\sqrt{t+\sqrt{t}}$, $\sqrt{t+\sqrt{t+\sqrt{t}}}$ and

$$\sqrt{t+\sqrt{t+\sqrt{t+\sqrt{t}}}}, \cdots,$$

successively.

You should not use the Python build-in command that computes the square root. Instead, use any algorithms/codes discussed in class to compute the square root.

Note that (1) actually equals to $\frac{1}{2}(1+\sqrt{4t+1})$. Your approximation of (1) should be accurate up to tolerance 0.000001 (compared to $\frac{1}{2}(1+\sqrt{4t+1})$).

Question 2. (2 points)
Question 2 in Assignment 2 shows how to use the area of rectangles to estimate the definite integral of a function. There, one uses the **midpoints** in the subintervals to compute the heights of the rectangle. In this question, we use **random** points in the subintervals. This is also known as Monte Carlo numerical integration.

Consider the definite integral,

$$\int_a^b f(x)\,\mathrm{d}x.$$

Divide the interval from $x = a$ to $x = b$ into n equal subintervals of length $\Delta x = \frac{b-a}{n}$ each. Let c_k be a **randomly generated point within** the k-th subinterval (respectively). Then the definite integral can be approximated again by

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} f(c_i)\Delta x = \Delta x \cdot [f(c_1) + f(c_2) + \cdots + f(c_n)]\,.$$

As before, take $f(x) = x^2 + \frac{1}{2}$ and end points $a = 0$, $b = 10$. Approximate

$$\int_0^{10} \left(x^2 + \frac{1}{2}\right)\,\mathrm{d}x.$$

using the above method.

Requirement: define a function with name "Approx_integral_Q2",

```
### Solution to Question 2
def Approx_integral_Q2(n):
        ... Your codes ...
```

The input n to the function is the number of subintervals used to approximated the definite integral. The function should return the above sum $\sum_{i=1}^{n} f(c_i)\Delta x$. You should **choose the number** n such that,

$$\left|\sum_{i=1}^{n} f(c_i)\Delta x - \int_0^{10} \left(x^2 + \frac{1}{2}\right)\,\mathrm{d}x\right| \leq 0.001.$$

Output the number n explicitly in the code.

You should use functions from the Python "random" module in this question.

Question 3. (2 points)
Write a Python program to re-implement the secant method using **recursion**. Use your program to find a root of the function with tolerance 0.0001.

$$f(x) = x^2 + \cos^7(x) + 2\sin(x).$$

Requirement: define a function with name "Secant_recursion_Q3",

```
### Solution to Question 3
def Secant_recursion_Q3():
        ... codes calls Secant_recursion_Q3() recursively...
```

For the starting points in secant's method, one may use 1 and 1.02. Note for efficiency, it is important that in each recursion call, only 1 function evaluation $f(x)$ is done.

Question 4. (2 points)
Write a Python program to simulate the following variant of the Monty Hall problem. Suppose that, in the beginning, the host does not know where the prize is located. After contestant picks a door, the host opens another door **uniformly at random** (between the two other doors). There are two situations:

- Aborted game: if the host happens to open a door with the prize, this game does not count and everything will be reshuffled. A new game will start after the reshuffling.
- Completed game: otherwise the game continues as usual. The host asks whether the contestant wants to change the door he/she picked. Now for the contestant, will changing-the-door increase his/her odd to win?

Write a Python program to simulate the contestant's winning odd (by running many games).
Requirement: define a function with name "simulate_game__Q4",

```
### Solution to Question 4
def simulate_game__Q4():
        ... Your codes ...
```

Your program should return the odds of winning for both strategies (changing door and not-changing door) over 10000 completed games.

You should use functions from the Python "random" module in this question.