***Read all of the following information before starting the exam:***

* The exam is open book, open notes, open Python documentation, open internet, etc.
* You **MAY NOT** use any form of technology to communicate with, send to, or receive information from another person (e.g., classmates, other instructors, anonymous persons on the internet). HOWEVER, you are **encouraged** to submit written questions to the professor or TA by email and/or have a private help session with the instructor through ZOOM.
* **MODULES/PACKAGES:** You may **NOT** use any Python packages other than random, math and copy. However, you may use **any** Python packages you wish as a check of your answer, but they should not be a part of your submitted solution. You may use/reuse code (with proper attribution; e.g., “this function is copied from Dr. Smay’s Gauss\_Seidel.py file” or “this import is from my HW1 file”)
* **COMMENTS/DOCUMENTATION:** **ALL** of your functions should use docstrings and other comments inside the function as necessary.
* **SUBMISSION:** You must place all your .py files in a single folder called EX1SP22 and submit the zipped folder as a single .zip file to Canvas titled EX1SP22.zip.
* **GRADING:** When we grade your assignment, we will run your program with those given numerical values, looking for correct answers. Then we will change the numerical values (including changing the SIZES of the arrays) and look for correct answers for those modified values as well. We will only use numerical values, array sizes and functions that make sense. We will not be testing your program to see how it handles bad data. As you have seen in your homework, comments/documentation will count for about 20% of the grade on a problem.

Section 1 – short answers (25 pts), you should type your answers either in this Word document or as a separate text file such as created in notepad. Neatness and clarity matter in your answer.

# Given: B = (3.5, 3.7, 5.0, 10.6, 25.3, 50.6)

## What is wrong with this line of Python code: B[3] = 10.6 ?

**You cannot change change the values, only remove them. This will give an error.**

## What is printed to the console/terminal for this line of Python code:

**print('Slice sum = {mySum:06.2f}'.format(mySum=sum(B[-3:1:-1])))**

**Slice sum = 015.60**

## Write an equivalent line of code to the above print statement using all positive numbers for slicing.

**print(‘Slice sum = {mySum:06.2f}’.format(mySum = sum(B[2:4:1])))**

## Is a, b, c = B[-1:-3:-1] allowed? Why or why not?

**No. It will give you an error because there are not enough values to unpack in the tuple. It will give 2 when it needs 3 to complete the action.**

## If d. is allowed, what is the value of b? If d. is not allowed, correct the statement (by changing only one character) and tell the value of b.

**a, b = B[-1:-3:-1]**

**print(‘b = ‘, b)**

**b = 25.3**

# True or False: (regarding Python)

## A tuple is immutable but can be redefined within its original scope? \_\_\_\_\_**True**\_\_\_\_\_

## Python relies on indentation to define scope in the code? \_\_**True**\_\_\_\_\_\_\_

## def is a keyword used to identify a block of code that runs only when called? \_\_**False (“defines” the keyword, but is not the keyword to call)\_**

## A callback function cannot be defined inside of another function? \_ \_**True**\_\_\_\_\_

## A lambda function can only take one argument?\_\_\_**False**\_\_\_\_\_\_

# 

## Write a single line of code that will produce a 6×6 identity matrix (list of lists).

**identity\_matrix = [[1 if r == c else 0 for c in range(6)] for r in range(6)]**

## Write a single line of code to fill all the non-diagonal elements with a formatted string of the form ‘r,c’ where r is the row index and c is the column index in the matrix and the diagonal elements with 1.

**identity\_matrix = [[str(r)+','+str(c) if r != c else 1 for c in range(6)] for r in range(6)]**

Section 2 – solutions problems 4 – 6 should be written in separate Python files titled EX1-4.py, EX1-5.py, EX1-6.py.

# (25 points) An exercise in producing pseudorandom numbers, indexing, and data check/verification. Write a Python program that randomly fills a 10×10 matrix with the integers in the range of 0 to 99 but, that doesn’t use any integer more than once. Your code must verify that no duplicates numbers exist in the matrix.

Note:

We have used the random module in Homework 1 to produce pseudorandom numbers BUT, you **must** write your own implementation of the Wichmann-Hill algorithm to produce your random numbers

(see <https://en.wikipedia.org/wiki/Wichmann%E2%80%93Hill>).

“Seed” the pseudorandom number algorithm with [1234, 19857, 25000] to start.

Output your 10×10 matrix to the console in the format:

A=

[82, 18, 74, 50, 33, 16, 8, 92, 60, 85]

[23, 78, 5, 75, 6, 1, 66, 32, 19, 54]

[91, 55, 28, 90, 36, 52, 0, 65, 30, 14]

[79, 9, 63, 94, 58, 96, 24, 71, 64, 37]

[35, 42, 17, 84, 98, 80, 10, 70, 7, 29]

[59, 25, 99, 11, 62, 87, 27, 51, 83, 40]

[81, 4, 41, 3, 2, 76, 26, 39, 73, 88]

[86, 45, 21, 48, 97, 61, 67, 68, 93, 89]

[77, 46, 38, 57, 13, 22, 72, 34, 47, 31]

[95, 44, 12, 53, 69, 15, 49, 20, 56, 43]

Seed values = [1234, 19857, 25000]

Verified 0 duplicated numbers!

# (25 pts)

## Use your pseudorandom number generator from question 4 to generate a list of 1000 uniformly-distributed, random, floating-point numbers *between* 0 and 1 (duplicates are okay). Consider these values to represent probability levels pi that can be found by integrating the Gaussian probability density function (PDF) between

x=μ-5⋅σ to x=xi such that pi=P(x<xi|N(μ,σ)) where μ=175, σ=15.

## Using your Simpson function (nPoints=50) to integrate the Gaussian PDF combined with your Secant *method* for root finding, find the set of x values compatible with your list of 1000 probabilities. Note: you will need to use callback functions for this process.

## Finally, calculate the estimates of the population parameters μ and σ2 using the unbiased sample estimators from you set of x values. Compare your values to those from N(175, 15).

Output the values for your population estimators like:

Population mean estimate = y.yy

Population variance estimate = z.zz

Note: you may need to use a clamp function to confine your probabilities to fall within the lower limit associated with the lower limit for integration.

(i.e., make sure pi≥P(x≤μ-5⋅σ|N(μ,σ))=Simpson(GPDF, (μ, σ), μ-10\*σ, μ-5\*σ))

# (25 pts) The following describes a tiny part of the airplane design process – Choosing the engine size needed to meet a take-off distance requirement (STO). STO is the distance an airplane will roll on the runway before it is able to lift off into the air. Calculating STO is performed using a sequence of five equations shown below. Those equations require seven parameters that control takeoff performance. The engine thrust parameter is a major factor in determining take-off distance. The graph shows how take-off distance gets shorter as engine thrust is increased.

Write a Python program that includes and calls the following ***three*** functions:

a) def STO(thrust):

thrust: the value of the engine thrust. Important, this is the only argument to be passed to this function. All of the ***other*** required airplane parameters (weight, S, CLmax, CD, rho, gc) may be assumed constant. Local variables for those parameters may be defined and assigned values inside the STOfunction. Use the airplane parameter values given in the figure above.

***The function returns:*** the airplane take-off distance, calculated using the five equations given above. The first four of those equations calculate the value of three constants to be used in the fifth equation (VTO, A and B). The fifth equation requires the use of numerical integration. Use your Simpson function to perform the integration.

b) def ThrustNeededForTakeoff(distance):

distance: the required take-off distance. Important, this is the only argument to be passed to this function. All of the ***other*** required airplane parameters may be assumed constant. Local variables for those parameters may be defined and assigned values inside the ***ThrustNeededForTakeoff*** function.

***The function returns:*** the engine thrust needed to allow the airplane to take-off in the specified distance. Hint: The ThrustNeededForTakeofffunction behaves as the ***inverse of the*** STOfunction. Therefore it must use a root-finding method to find the ***value of thrust*** that causes: STO(thrust)–distance = 0***.*** Use your Secant function to solve for this value of thrust.

c) def main():

main() ***has no arguments and no return value.***

main() ***does the following:***

***i)*** calls STO(13000)to calculate the take-off distance for an engine thrust of 13000 pounds. Print the answer with one decimal place, using a nice text label. The nice text label should include the thrust value of 13000 pounds.

***ii)*** calls ThrustNeededForTakeoff(1500)to calculate the thrust needed to allow takeoff in 1500 feet. Print the answer with two decimal places, using a nice text label. The nice text label should include the takeoff distance of 1500 feet.

***ii)*** calls ThrustNeededForTakeoff(1000)to calculate the thrust needed to allow takeoff in 1000 feet. Print the answer with two decimal places, using a nice text label. The nice text label should include the takeoff distance of 1000 feet.