This assignment will be due on Wednesday, January 26 at 11:59 pm ET. To submit your work, please download this document as a .ipynb file and upload that to Canvas.

1. Add a text block below this, and add the names of your group members, as well as any other classmates you collaborated with for this assignment.

> Group 2: Rafael Piloto, Zee Kwong, Muyi Chen, Ginger Lau

2. Give three examples of problems that computers are well-suited to solve, and explain why it might be beneficial to have computers solve these problems.

> 1. Computers are well suited for predicting future events, which is beneficial for preperation. For example, models that predict the spread of COVID can help states prepare for incoming surges.
> 2. Computers are well suited for estimating system behavior, which is beneficial for understanding how a system might work. For example, models that simulate environments can be used to understand how an environment might behave in the real world.
> 3. Computers are well suited for testing hypotheses. Through the ability of prediction and simulation, we can verify hypotheses, which is beneficial because they are low-cost.

3. Explain what a model is, and how we might decide if a model is good or bad.

> A model is a simplified mathematical representation of a physical, biological, etc system, phenomenon, or process. A good model should capture some aspect of the system of interest.

4. Type and execute all snippets of code in Section 1.3 of the textbook.

```
import numpy
sqrt(2)
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call
last)
<ipython-input-1-0c1b4623f26e> in <module>()
      1 import numpy
----> 2 sqrt(2)

NameError: name 'sqrt' is not defined
```

```
import numpy
numpy.sqrt(2)
```

```
1.4142135623730951
```

```
from numpy import *
sqrt(2)
```

```
1.4142135623730951
```

```
import numpy as np
np.sqrt(2)
```

```
1.4142135623730951
```

```
from numpy import sqrt, exp
sqrt(2)
exp(3)
```

```
20.085536923187668
```

```
from numpy.random import random as rng
rng()
```

```
0.27790651885253004
```

```
import numpy as np
import matplotlib.pyplot as plt
```

```
import numpy as np, matplotlib.pyplot as plt
```

```
%reset
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

5. Read through Appendix D in the textbook, and then type each line of code in sections 1.4.2 in

```
(2)(3)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
last)
<ipython-input-12-05d88058fdf8> in <module>()
----> 1 (2)(3)

TypeError: 'int' object is not callable
```

> The error here is that `(2)(3)` is trying to call `(2)` as an object with parameter `(3)`. The expected behavior is 6. To get 6, we must use `*`.

```
a = 2; a(3)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
last)
<ipython-input-13-3ec61ef63de0> in <module>()
----> 1 a = 2; a(3)

TypeError: 'int' object is not callable
```

> The error is here the same. We assign the variable `a = 2`. `a` is an integer and therefore not callable. We must use `*` to achieve multiplication.

```
3a
```

```
  File "<ipython-input-15-ce83c9a3eeb6>", line 1
    3a
     ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

> This syntax is invalid. We must use the `*` to get mutiplication

```
3 a
```

```
  File "<ipython-input-16-640c4bc38171>", line 1
    3 a
      ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

> This syntax is invalid. We must use the `*` to get mutiplication.

6. What factors set the precision with which we can represent numbers on a computer?

> The amount of bits that a number can be stored with sets the precision with which we can represent numbers on a computer. Single precision numbers use 32 bits and double precision numbers use 64 bits. (Fun fact: This is why NASA only uses 15 digits of PI even though using at least 40 would provide significantly more precision).

7. Import the NumPy library and name it "np". Now execute a snippet of code defining a variable named "np.sqrt" to equal 2. Now try to take the square root of 100. What happens? What are two ways you can fix this problem?

```
import numpy as np
np.sqrt = 2
```

```
np.sqrt(100)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-18-f02dc2d40b6c> in <module>()
----> 1 np.sqrt(100)

TypeError: 'int' object is not callable
```

SEARCH STACK OVERFLOW

> We are overriding the NumPy `sqrt` method and setting it to an integer 2. In order to fix this issue, we can use the `%reset` feature of the Notebooks to reset our environment. We can also navigate to Runtime > Restart runtime and get back the original NumPy method. We must be careful to remove the unwanted block of code to avoid the same happening again later.

8. Let's say we have a problem that produces the following equation: $a^2 = 1000e^{-a/25}$, and we want to solve for a.

(a) Are you able to analytically solve for the value of a that satisfies this equation? Explain.

> You can begin to solve the equation, but get stuck with either a log or e. The next step would be to try and expand the function using the Taylor Series, however this would provide us with an approximation rather than an exact answer.

(b) Describe in words how you might go identify a value for a that satisfies the equation using a plot. What steps would you take, and what features would you look for?

> Using a plot, we can plot a^2 and $1000e^{-a/25}$. Using the plot, we can then observe where the two functions intersect. The values of a where both plots intersect would be the solutions to the equation.

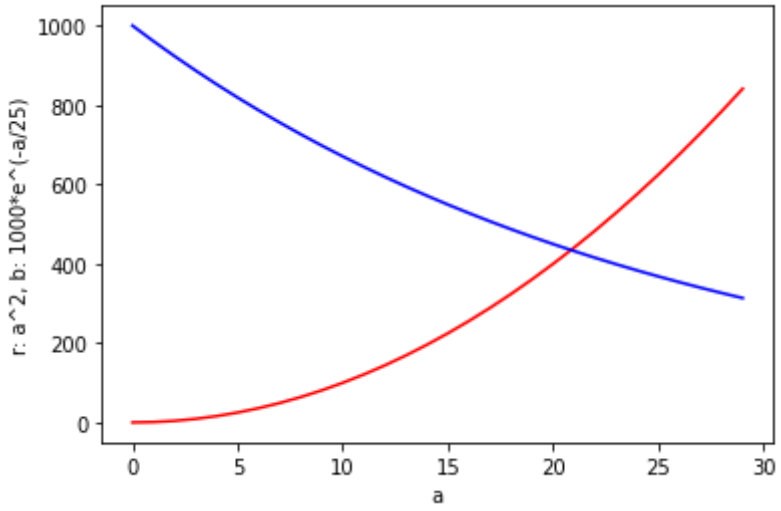
(c) Following your description in (b), create a plot that allows you to estimate the value of a that satisfies this equation. For full credit, be sure to label the axes on your plot. From visual inspection, estimate the value of a and explain your reasoning.

```
%reset
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

```
import math
import matplotlib.pyplot as plt


def left(a):
  return a ** 2


def right(a):
  return 1000 * math.exp(-a/25)

fig, ax = plt.subplots(1, 1)

x = [i for i in range(30)]

ax.plot(x, [left(a) for a in x], c='r')
ax.plot(x, [right(a) for a in x], c='b')

ax.set(xlabel="a", ylabel="r: a^2, b: 1000*e^(-a/25)")

plt.show()
```

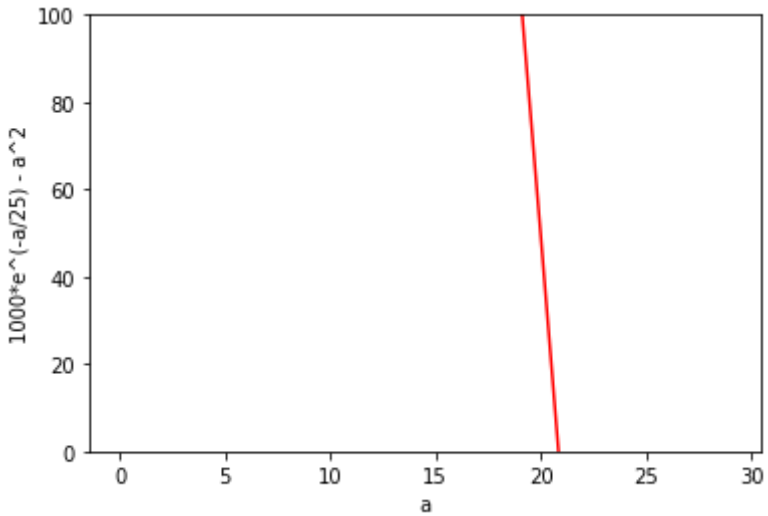
```
plt.show()
```



> a appears to be approximately 21

(d) Can you think of another way to use plots to estimate the value of a? Create this plot, and again, label the axes. From visual inspection, estimate the value of a and explain your reasoning.

```
fig, ax = plt.subplots(1, 1)

x = [i for i in range(30)]

ax.plot(x, [right(a) - left(a) for a in x], c='r')
ax.set(ylim=(0, 100), xlabel="a", ylabel="1000*e^(-a/25) - a^2")

plt.show()
```



> If we move the left side of the equation to the right side, then we get $0 = 1000e^{-a/25} - a^2$. Therefore, we can observe for what values of a does the

equation produce 0. Plotting this, we can see that a appears to be approximately 21.

✓ 0s completed at 2:29 PM