Assignment 1

Write your name and nyu email

Please write names below

- Name: David Shen
- Email: ds6870@nyu.edu

Write the names of people you worked with to complete this assignment

- Name:
- Name:
- Name:

NOTE:

1. You can edit this cell by double clicking in it and hitting `Ctrl` + `enter` for it to render
2. This is a Mardown cell, where we type text. For more info on how to do formating on mardown please see here https://ingeh.medium.com/markdown-for-jupyter-notebooks-cheatsheet-386c05aeebed, but you don't need to

```
Start coding or generate with AI.
```

Part A

Exercise 1: Variables

Assume you deposit 2000 at a bank at the beginning of 2019. The annual return is 5% from 2019 to 2024. How much is your 2000 worth at the end of 2024?

Assign values to the variables a, b and c by subsituting `__` by the appropriate values in this example. You have to look at the expression in the coding cell under "4.Expression" to know what is appropriate

Tip: `**` is the exponentiation operation `3**2=9`

```
# this a coding cell, you run the code by hitting shift+enter or pressing the "play buttom"
#on the top

#Try running the code before doign anything to see what a coding error look like

a=__
b=__
c=__
```

Exercise 2: Expressions

Once you assigned the values to a, b, and c run the cell below to get the value of your networth

```
a*(1+b)**c
```

```
155624547606000
```

Exercise 3: Understanding sequencing

Your first two pieces of code run succefuly (I very much hope!) . You first created three variables and you then use these variables to compute an expression which returned the value of your wealth in 2024. (yay!).

Now in the cell below I want you to change the variable that represents the annual bank return from 5% to 10%

```
# Write down your code below

a=6000
b=10
c=10
```

Now that you have done that, go back to the code cell in expressions, the one that says `a*(1+b)**c` and recompile it by hitting shift-enter.

What did you find? write this number in the cell below

```
8063.498276064734
```

```
8063.498276064734
```

1. Now run the coding cell under variables and the coding cell under expressions again.
2. What you should have just experienced is a very important feature of how Jupyter Notebooks work
3. The order of the coding cells in the notebook are IRRELEVANT for coding pourposes!

What matters is the sequence that they were called. When you change the interest rate in the cell above this variable assignment substitute the earlier assignment, so when you run the expression cell again it used the most recent variable assignment.

- One way to understand what is going on is to look at the little numbers in between the squares in the left of each coding cell.They tell us the sequence that each was called.
- This way you know the order things were called.

Exercise 4: One important allert about the little numbers

Go back to the coding cell under "3.Variable" and change the interest to 1000%.

Don't run the cell. What did you notice about the number in the squares? Explain in the cell below how this can be a source of confusion.

Exercise 5: Creating a cell

Which cell below?

-> to answer the question above you need to create a markdown cell (since your answer will be text)

there two way you can do this

1. On the top of the notebook, click on the plus signal while this cell is selected, this will create a cell below. This cell will be by default a coding cell. Now to trasnform in a Markdown cell (to write text), while having the new cell selected click on the top buttom that says "code" and select "Markdown". There!
2. In colab you see that you can that you can directly choose which cell to add (it says "+Code" or "+Text"). In Colab you can also simply hover above/below the cell to see the same buttoms. Press the above to add to a cell above, or presse the buttons below (what we want!) to add the cell below
    - If the cursor is activated you can hit `esc` to get out of the cell. You can also just hitting `Shift`+`enter` or simply clickling on the white area on the left of the cell

The data in the code cell is now inconsistent with the other cells that depend on the data. It has been edited but not run, so the number in the square still shows that it was run before the subsequent cells, even though the subsequent cells are using the old data. This may lead to confusion resulting from inconsistent data that has not updated. Users can try to avoid this by running cells often.

Exercise 6: Compouding returns

Say you have 100 dollar and you invest in a stock that earned 15% in the first year and -14% in the second year.

Modify the code below so that it gives the correct answer to this question

```
r1=0.15
r2=-0.14
100*(1+r1)*(1+r2)
```

```
98.89999999999999
```

Exercise 7: Deleting a cell

Now that you did all and have all your cell properly compiled you should save by clickling on the save buttom and then import it to the class blackboard website in the Assigments tab on the left.

But before you do this, there is one more thing to do! Do you see this cell below? IT is ugly and we don't want it there. Lets delete it!

Again two ways of doing it

1. Select it and click on scissors buttom at the top of the notebook (in jupyter) or the trach can in the top right of the cell
2. short-cut: in jupyter click `d` twice while having the cell selected buy not active. in colab click 'Ctrl+M'+ d

Part c

Exercise 1

What do you think the value of `z` is after running the code below?

```
z = 3
z = z + 4
print("z is", z)
```

Answer here: z is 7

Exercise 2

Read about what the `len` function does (by running `len?`).

What will it produce if we give it the variable `x`?

Answer here: return the length of x which is 14

Check whether you were right by running the code `len(x)`.

```
x = "something else"
# your code here -- notice the comment!
len(x)
```

```
14
```

Exercise 3

We can use our introspection skills to investigate a package's contents.

In the cell below, use tab completion to find a function from the `time` module that will display the **local** time.

Use `time.FUNC_NAME?` (where `FUNC_NAME` is replaced with the function you found) to see information about that function and then call the function. (Hint: look for something to do with the word `local`).

```
import time
# your code here -- notice the comment!
time.localtime()
```

```
time.struct_time(tm_year=2026, tm_mon=1, tm_mday=23, tm_hour=18, tm_min=19, tm_sec=57, tm_wday=4, tm_yday=23, tm_isdst=0)
```

Exercise 4

Try running `import time as t` in the cell below, then call the function you identified above and see if it returns the local time

Does it work?

```
# your code here!
import time as t
t.localtime()
```

```
time.struct_time(tm_year=2026, tm_mon=1, tm_mday=23, tm_hour=18, tm_min=19, tm_sec=57, tm_wday=4, tm_yday=23, tm_isdst=0)
```

Exercise 5

Create the following variables:

- `D`: A floating point number with the value 10,000
- `r`: A floating point number with value 0.025

- `T`: An integer with value 30

We will use them in a later exercise.

Use `type` function on each variable together with print to show the type of each variable

```
# your code here!
D = 10000.0
r = 0.025
T = 30
print(type(D))
print(type(r))
print(type(T))
```

```
<class 'float'>
<class 'float'>
<class 'int'>
```

Exercise 6

Remember the variables we created earlier?

Let's compute the present discounted value of a payment (D) made in T years assuming an interest rate of 2.5%. Save this value to a new variable called `PDV` and print your output.

Hint: The formula is

$$\text{PDV} = \frac{D}{(1+r)^T}$$

```
# your code here
D / (1 + r) ** T
```

```
4767.426851809713
```

Exercise 7

Verify the "trick" where the percent difference ($\frac{x-y}{y}$) between two numbers close to 1 can be well approximated by the difference between the log of the two numbers ($\log(x) - \log(y)$).

Use the numbers `x` and `y` below. (Hint: you will want to use the `math.log` function)

Use print so both results are displayed

```
x = 1.05
y = 1.02
# your code here!
import math
print((x - y) / y)
print(math.log(x) - math.log(y))
```

```
0.029411764705882377
0.02898753687325232
```

Exercise 8

The code below is invalid Python code

```
x = 'What's wrong with this string'
```

Can you fix it?

Hint: Try creating a code cell below and testing things out until you find a solution.

```
# your code here!
x = "What's wrong with this string"
```

Exercise 9

Using the variables `x` and `y`, how could you create the sentence `Hello World`?

Hint: Think about how to represent a space as a string.

```
x = "Hello"
y = "World"
# your code here!
x + ' ' + y
```

```
'Hello World'
```

Exercise 10

One of our favorite (and most frequently used) string methods is `replace`.

It substitutes all occurrences of a particular pattern with a different pattern.

For the variable `test` below, use the `replace` method to change the `c` to a `d`.

Hint: Type `test.replace?` to get some help for how to use the method replace.

```
test = "abc"
# your code here!
test.replace('c', 'd')
```

```
'abd'
```

Exercise 11

Suppose you are working with price data and encounter the value `"$6.50"`.

We recognize this as being a number representing the quantity "six dollars and fifty cents."

However, Python interprets the value as the string `"$6.50"`. (Quiz: why is this a problem? Think about the examples above.)

In this exercise, your task is to convert the variable `price` below into a number.

Hint: Once the string is in a suitable format, you can call write `float(clean_price)` to make it a number.

```
price = "$6.50"
# your code here
clean_price = price[1:]
float(clean_price)
```

```
6.5
```

Exercise 12

Without typing the commands, determine whether the following statements are true or false.In particular write a list of 6 items of the type `[True,False,...]` with the results you beleive are correct

Your answer: [False, True, False, True, True, True]

```
x = 2
y = 2
z = 4

# Statement 1
x > z

# Statement 1
x == y

# Statement 3
(x < y) and (x > y)

# Statement 4
(x < y) or (x > y)

# Statement 5
(x <= y) and (x >= y)
```

```
# Statement 6
True and ((x < z) or (x < y))
```

Once you have evaluated whether the command is `True` or `False`, run the code in Python.

```
# your code here!
x = 2
y = 2
z = 4

# Statement 1
print(x > z)

# Statement 1
print(x == y)

# Statement 3
print((x < y) and (x > y))

# Statement 4
print((x < y) or (x > y))

# Statement 5
print((x <= y) and (x >= y))

# Statement 6
print(True and ((x < z) or (x < y)))
```

```
False
True
False
False
True
True
```

Exercise 13

For each of the code cells below, think carefully about what you expect to be returned *before* evaluating the cell.

Then evaluate the cell to check your intuitions.

NOTE: For now, do not worry about what the `[` and `]` mean -- they allow us to create lists which we will learn about in an upcoming lecture.

```
all([True, True, True])
```

```
True
```

```
all([False, True, False])
```

```
False
```

```
all([False, False, False])
```

```
False
```

```
any([True, True, True])
```

```
True
```

```
any([False, True, False])
```

```
True
```

```
any([False, False, False])
```

```
False
```

Submission

1. Go in file/print in the top left. Choose save as a PDF
2. Go in file/Download. Choose ".ipynb"
3. Make sure to download to a local folder in your machine designated for class
4. Go to Assignment 1 in Brightspace and upload both files.