In [ ]:

Write (or just say out loud to yourself) a brief description of all the following Object Types and Data Structures we have learned about.

Numbers: Python has various types of numbers (numeric literals). Primarily integers and floating point numbers.

Integers are just whole numbers, positive or negative. For example 2 and -2 are examples of integers

Floating point numbers in Python are notable because they have a decimal point in them, or use an exponetial (e) to define the number. For example 2.0 and -2.1 are examples of floating numbers. 4E2 (4 times 10 to the power of 2) is also an example of a floating point number in Python.

Strings: Strings are used in Python to record text information, such as name. Strings in Python are actuall a sequence, which basically means Python keeps track of every element in the string as a sequence. For example, Python understands the string"hello" to be a sequence of letters in a specific order. This means we will be able to use indexing to grab particular letters (like the first letter, or the last letter.)

Creating a String:

Single Word

'hello'

Lists: Earlier when discussing strings we introduced the concept of a sequence in Python. Lists can be thought of the most general version of a sequence in Python. Unlike strings, lists are mutable, meaning the elements inside a list can be changed.

Example: my_list = [1,2,3]

Tuples: In Python tuples are very similar to lists, however, unlike lists they are immutable meaning they can not be changed. You would use tuples to present things that shouldnt be changed, such as days of the week, or dates on a calendar.

t = (1,2,3) t = ('one',2)

Dictionaries: We have been learning about sequences in Python but now we are going to switch gears and learn about mappings in Python.

Example: my_dict = {'key1': 'value1', 'key2': 'value2'}

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Numbers

Write an equatrion that uses multiplications, division, an exponent, addition, and subtraction that is equal to 100.25

In [1]:
```
100 + .25
```

Out[1]:
```
100.25
```

In [2]:
```
(60 + (10 ** 2) / 4 * 7) - 134.75
```

Out[2]:
```
100.25
```

Answer these 3 questions:

What is the value of the expression 4 * (6+5)?

In [3]:
```
4 * (6+5)
```

Out[3]:
```
44
```

What is the value of the expression 4 * 6 + 5?

In [4]:
```
4 * 6 + 5
```

Out[4]:
```
29
```

What is the value of the expression 4 + 6 * 5?

In [5]:
```
4 + 6 * 5
```

Out[5]:
```
34
```

What is the type of the result of the expression 3 + 1.5 + 4?

Floating Point Number

What would you use to find a numbers square root, as well as its square?

In [7]:
```
100 ** 0.5
```

Out[7]:
```
10.0
```

In [8]:
```
10 ** 2
```

Out[8]:
```
100
```

Given the string 'hello' give an index command that returns 'e'

In [9]:
```
s = 'hello'
```

In [10]:
```
s[1]
```

Out[10]:
```
'e'
```

Reverse the string "hello" using slicing:

In [11]:
```
s = 'hello'
```

In [12]:
```
s[::-1]
```

Out[12]:
```
'olleh'
```

Given the string "hello", give two methods of producing the letter 'o' using indexing:

In [13]:
```
s = 'hello'
```

In [15]:
```
#Method 1
s[-1]
```

Out[15]:
```
'o'
```

In [16]:
```
#Method 2
s[4]
```

Out[16]:
```
'o'
```

Lists

Build the list [0,0,0] two separate ways:

In [17]:
```
#Method 1
[0]*3
```

Out[17]:
```
[0, 0, 0]
```

In [18]:
```
#Method 2
list2 = [0,0,0]
list2
```

Out[18]:
```
[0, 0, 0]
```

Reassign 'hello' in this nested list to say 'goodbye' instead:

In [19]:
```
list3 = [1,2,[3,4,'hello']]
```

In [20]:
```
```

In [21]:
```
list3
```

Out[21]:
```
[1, 2, [3, 4, 'goodbye']]
```

Sort the list below:

In [22]:
```
list4 = [5,3,4,6,1]
```

In [23]:
```
# Method 1:
sorted(list4)
```

Out[23]:
```
[1, 3, 4, 5, 6]
```

In [25]:
```
#Method 2
list4.sort()
list4
```

Out[25]:
```
[1, 3, 4, 5, 6]
```

In [ ]:
```

```

In [ ]:
```

```

Dictionaries Using keys and indexing, grab the 'hello' from the following dictionaries:

In [26]:
```
d = {'simple_key':'hello'}
#Grab 'hello'

d['simple_key']
```

Out[26]:
```
'hello'
```

In [27]:
```
d = {'k1':{'k2':'hello'}}
# Grab 'hello'

d['k1']['k2']
```

Out[27]:
```
'hello'
```

In [32]:
```
#Getting a little tricker
d = {'k1':[{'nest_key':['this is deep',['hello']]}]}
```

In [33]:
```
# This is harder than I expected...

d['k1'] [0] ['nest_key'] [1][0]
```

Out[33]:
```
'hello'
```

Question: Can you sort a dictionary? Why or why not?

Answer: No, because normal dictionaries are mappings no a sequence

In [ ]:

In [ ]:

In [ ]:

Tuples

What is the major difference between tuples and lists?

Tuples are immuntable

How do you create a tuple?

In [35]:
```
t = (1,2,3)
```

In [ ]:

In [ ]:

In [ ]:

Sets

Question: What is unique about a set?

Answer: They do not allow for duplicate items

Use a set to find the unique values of the list below:

In [37]:
```
list5 = [1,2,2,33,4,4,11,22,3,3,2]
```

In [38]:
```
set (list5)
```

Out[38]:
```
{1, 2, 3, 4, 11, 22, 33}
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [40]:
```
Booleans


Operator   Description
==         If the values of two operands are equal, then the condition becomes true
!=         If values of two operands are not equal, then condition becomes true
>          If the value of left operand is greater than the value of right operand, then
<          If the value of left operand is less than the value of right operand, then con
>=         If the value of left operand is greater than or equal to the value of right op
<=         If the value of left operand is less than or equal to the value of right opera
```

```
  File "C:\Users\Keegz\AppData\Local\Temp/ipykernel_4040/3000008914.py", line 4
    Operator   Description
Example
               ^
SyntaxError: invalid syntax
```

In [41]:
```
2 > 3
```

Out[41]:
```
False
```

In [42]:
```
3 <= 2
```

Out[42]:
```
False
```

In [43]:
```
3 == 2.0
```

Out[43]:
```
False
```

In [44]:
```
3.0 == 3
```

Out[44]:
```
True
```

In [45]:
```
4**0.5 !=2
```

Out[45]:
```
False
```

In [48]:
```
#two nested lists
l_one = [1,2,[3,4]]
l_two = [1,2,{'k1':4}]

#True or False?
l_one[2][0] >= l_two[2]['k1']
```

Out[48]:
```
False
```

In [ ]: