```
In [9]: import math
```

Data Types in Python

The following data types can be used in base python:

- **boolean**
- **integer**
- **float**
- **string**
- **list**
- **None**
- complex
- object
- set
- dictionary

We will only focus on the **bolded** ones

Let's connect these data types to the the variable types we learned from the Variable Types video (https://www.coursera.org/learn/understanding-visualization-data/lecture/iDodZ/variable-types).

Numerical or Quantitative (taking the mean makes sense)

- Discrete
  - Integer (int) #Stored exactly
- Continuous
  - Float (float) #Stored similarly to scientific notation. Allows for decimal places but loses precision.

```
 In [1]: type(4)
Out[1]: int
```

```
 In [2]: type(0)
Out[2]: int
```

```
 In [3]: type(-3)
Out[3]: int
```

```
 In [4]: #try taking the mean
         numbers = [2, 3, 4, 5]
         print(sum(numbers)/len(numbers))
         type(sum(numbers)/len(numbers)) #In Python 3 returns float, but in Python 2 would return int

         3.5
Out[4]: float
```

Floats

```
 In [5]: 3/5
Out[5]: 0.6
```

```
 In [6]: 6*10**(-1)
Out[6]: 0.6000000000000001
```

```
 In [7]: type(3/5)
Out[7]: float
```

```
 In [10]: type(math.pi)
Out[10]: float
```

```
 In [11]: type(4.0)
Out[11]: float
```

```
In [12]: # Try taking the mean
         numbers = [math.pi, 3/5, 4.1]
         type(sum(numbers)/len(numbers))

Out[12]: float
```

Categorical or Qualitative

- Nominal
  - Boolean (bool)
  - String (str)
  - None (NoneType)
- Ordinal
  - Only defined by how you use the data
  - Often important when creating visuals
  - Lists can hold ordinal information because they have indices

Boolean

```
In [13]: # Boolean
         type(True)

Out[13]: bool
```

```
In [19]: # Boolean
         if 6 < 5:
             print("Yes!")
```

```
In [20]: myList = [True, 6<5, 1==3, None is None]
         for element in myList:
             print(type(element))

         <class 'bool'>
         <class 'bool'>
         <class 'bool'>
         <class 'bool'>
```

```
In [21]: print(sum(myList)/len(myList))
         type(sum(myList)/len(myList))

         0.5

Out[21]: float
```

String

```
In [22]: type("This sentence makes sense")

Out[22]: str
```

```
 In [ ]: type("Makes sentense this sense")
```

```
In [23]: type("math.pi")

Out[23]: str
```

```
In [24]: strList = ['dog', 'koala', 'goose']
         sum(strList)/len(strList)

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-24-b0bd059010c7> in <module>()
               1 strList = ['dog', 'koala', 'goose']
         ----> 2 sum(strList)/len(strList)

         TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Nonetype

```
In [25]: # None
         type(None)

Out[25]: NoneType
```

```
In [26]: # None
         x = None
         type(x)

Out[26]: NoneType
```

In [27]:

```
noneList = [None]*5
sum(nonList)/len(nonList)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-27-08e0974f29ad> in <module>()
      1 noneList = [None]*5
----> 2 sum(nonList)/len(nonList)

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

Lists

A list can hold many types and can also be used to store ordinal information.

In [ ]:

```
# List
myList = [1, 1.1, "This is a sentence", None]
for element in myList:
    print(type(element))
```

In [ ]:

```
sum(myList)/len(myList)
```

In [ ]:

```
# List
myList = [1, 2, 3]
for element in myList:
    print(type(element))
sum(myList)/len(myList) # note that this outputs a float
```

In [ ]:

```
myList = ['third', 'first', 'medium', 'small', 'large']
myList[0]
```

In [ ]:

```
myList.sort()
myList
```

There are more datatypes available when using different libraries such as Pandas and Numpy, which we will introduce to you as we use them.