Unit Testing

While we will not cover the unit testing library (https://docs.python.org/3/library/unittest.html) that python has, we wanted to introduce you to a simple way that you can test your code.

Unit testing is important because it the only way you can be sure that your code is do what you think it is doing.

Remember, just because ther are no errors does not mean your code is correct.

In [4]:

```
import numpy as np
import pandas as pd
import matplotlib as plt
pd.set_option('display.max_columns', 100) # Show all columns when looking at dataframe
```

In [5]:

```
# Download NHANES 2015-2016 data
df = pd.read_csv("nhanes_2015_2016.csv")
df.index = range(1,df.shape[0]+1)
```

In [6]:

```
df.head()
```

Out[6]:

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	DMDCITZN	DMDEDUC2	DMDMARTL	DMDHHSIZ	WTINT2YR	SDMVPS
1	83732	1.0	NaN	1.0	1	1	62	3	1.0	5.0	1.0	2	134671.37	1
2	83733	1.0	NaN	6.0	1	1	53	3	2.0	3.0	3.0	1	24328.56	1
3	83734	1.0	NaN	NaN	1	1	78	3	1.0	3.0	1.0	2	12400.01	1
4	83735	2.0	1.0	1.0	2	2	56	3	1.0	5.0	6.0	1	102718.00	1
5	83736	2.0	1.0	1.0	2	2	42	4	1.0	4.0	3.0	5	17627.67	2

Goal

We want to find the mean of first 100 rows of 'BPXSY1' when 'RIDAGEYR' > 60

```
In [7]:  # One possible way of doing this is:
         pd.Series.mean(df[df.RIDAGEYR > 60].loc[range(0,100), 'BPXSY1'])
         # Current version of python will include this warning, older versions will not

Out[7]:  139.57142857142858
```

```
In [8]:  df1 = df[df.RIDAGEYR > 60].loc[range(0, 100)]
         df1.head()

Out[8]:
```

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	DMDCITZN	DMDEDUC2	DMDMARTL	DMDHHSIZ	WTINT2YR	SDMVP
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	83732.0	1.0	NaN	1.0	1.0	1.0	62.0	3.0	1.0	5.0	1.0	2.0	134671.37	1.0
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	83734.0	1.0	NaN	NaN	1.0	1.0	78.0	3.0	1.0	3.0	1.0	2.0	12400.01	1.0
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
In [9]:  df1.shape

Out[9]:  (100, 28)
```

```
In [10]:  type(df1)

Out[10]:  pandas.core.frame.DataFrame
```

```
In [11]:  np.nanmean(df1['BPXSY1'])

Out[11]:  139.57142857142858
```

In [12]:

```
# test our code on only ten rows so we can easily check
test = pd.DataFrame({'col1': np.repeat([3,1],5), 'col2': range(3,13)}, index=range(1,11))
test
```

Out[12]:

	col1	col2
1	3	3
2	3	4
3	3	5
4	3	6
5	3	7
6	1	8
7	1	9
8	1	10
9	1	11
10	1	12

In [13]:

```
# pd.Series.mean(df[df.RIDAGEYR > 60].loc[range(0,5), 'BPXSY1'])
# should return 5

pd.Series.mean(test[test.col1 > 2].loc[range(0,5), 'col2'])
```

Out[13]:

```
4.5
```

What went wrong?

In [14]:
```
test[test.col1 > 2].loc[range(0,5), 'col2']
# 0 is not in the row index labels because the second row's value is < 2. For now, pandas defaults to filling this
# with NaN
```

Out[14]:
```
0    NaN
1    3.0
2    4.0
3    5.0
4    6.0
Name: col2, dtype: float64
```

In [15]:
```
# Using the .iloc method instead, we are correctly choosing the first 5 rows, regardless of their row labels
test[test.col1 >2].iloc[range(0,5), 1]
```

Out[15]:
```
1    3
2    4
3    5
4    6
5    7
Name: col2, dtype: int64
```

In [16]:
```
pd.Series.mean(test[test.col1 >2].iloc[range(0,5), 1])
```

Out[16]:
```
5.0
```

In [17]:
```
# We can compare what our real dataframe looks like with the incorrect and correct methods
df[df.RIDAGEYR > 60].loc[range(0,5), :] # Filled with NaN whenever a row label does not meet the condition
```

Out[17]:

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	DMDCITZN	DMDEDUC2	DMDMARTL	DMDHHSIZ	WTINT2YR	SDMVF
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	83732.0	1.0	NaN	1.0	1.0	1.0	62.0	3.0	1.0	5.0	1.0	2.0	134671.37	1.0
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	83734.0	1.0	NaN	NaN	1.0	1.0	78.0	3.0	1.0	3.0	1.0	2.0	12400.01	1.0
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [18]:

```
df[df.RIDAGEYR > 60].iloc[range(0,5), :] # Correct picks the first fice rows such that 'RIDAGEYR" > 60
```

Out[18]:

	SEQN	ALQ101	ALQ110	ALQ130	SMQ020	RIAGENDR	RIDAGEYR	RIDRETH1	DMDCITZN	DMDEDUC2	DMDMARTL	DMDHHSIZ	WTINT2YR	SDMVP
1	83732	1.0	NaN	1.0	1	1	62	3	1.0	5.0	1.0	2	134671.37	1
3	83734	1.0	NaN	NaN	1	1	78	3	1.0	3.0	1.0	2	12400.01	1
6	83737	2.0	2.0	NaN	2	2	72	1	2.0	2.0	4.0	5	11252.31	1
14	83754	2.0	1.0	1.0	2	2	67	2	1.0	5.0	1.0	7	10495.87	1
15	83755	1.0	NaN	3.0	2	1	67	4	1.0	5.0	2.0	1	14080.10	1

In [19]:

```
# Applying the correct method to the original question about BPXSY1
print(pd.Series.mean(df[df.RIDAGEYR > 60].iloc[range(0,100), 16]))

# Another way to reference the BPXSY1 variable
print(pd.Series.mean(df[df.RIDAGEYR > 60].iloc[range(0,100), df.columns.get_loc('BPXSY1')]))
```

```
136.29166666666666
136.29166666666666
```