Jim Crivello's Module 4 Project - Part 2

Task 1. Create DataFrame

In [37]:

```
import pandas as pd
```

In [38]:

```
grades_dict = {'Wally': [87, 96, 70], 'Eva': [100, 87, 90], 'Sam': [94, 77, 90], 'Kati
```

In [39]:

```
grades = pd.DataFrame(grades_dict)
```

In [40]:

```
grades
```

Out[40]:

	Wally	Eva	Sam	Katie	Bob
0	87	100	94	100	83
1	96	87	77	81	65
2	70	90	90	82	85

Task 2. Custom Index

In [41]:

```
pd.DataFrame(grades_dict, index=['Test1', 'Test2', 'Test3'])
```

Out[41]:

	Wally	Eva	Sam	Katie	Bob
Test1	87	100	94	100	83
Test2	96	87	77	81	65
Test3	70	90	90	82	85

In [42]:

```
grades.index = ['Test1', 'Test2', 'Test3']
```

In [43]:

```
grades
```

Out[43]:

	Wally	Eva	Sam	Katie	Bob
Test1	87	100	94	100	83
Test2	96	87	77	81	65
Test3	70	90	90	82	85

In [44]:

```
grades['Eva']
```

Out[44]:

```
Test1    100
Test2     87
Test3     90
Name: Eva, dtype: int64
```

In [45]:

```
grades.Sam
```

Out[45]:

```
Test1    94
Test2    77
Test3    90
Name: Sam, dtype: int64
```

Task 3. Assessing Rows (loc, iloc)

In [46]:

```
grades.loc['Test1']
```

Out[46]:

```
Wally     87
Eva      100
Sam       94
Katie    100
Bob       83
Name: Test1, dtype: int64
```

In [47]:

```
grades.iloc[1]
```

Out[47]:

```
Wally    96
Eva      87
Sam      77
Katie    81
Bob      65
Name: Test2, dtype: int64
```

In [48]:

```
grades.loc['Test1' : 'Test3']
```

Out[48]:

	Wally	**Eva**	**Sam**	**Katie**	**Bob**
Test1	87	100	94	100	83
Test2	96	87	77	81	65
Test3	70	90	90	82	85

In [49]:

```
grades.iloc[0:2]
```

Out[49]:

	Wally	**Eva**	**Sam**	**Katie**	**Bob**
Test1	87	100	94	100	83
Test2	96	87	77	81	65

In [50]:

```
grades.loc[['Test1', 'Test3']]
```

Out[50]:

	Wally	**Eva**	**Sam**	**Katie**	**Bob**
Test1	87	100	94	100	83
Test3	70	90	90	82	85

In [51]:

```
grades.iloc[[0,2]]
```

Out[51]:

	Wally	Eva	Sam	Katie	Bob
Test1	87	100	94	100	83
Test3	70	90	90	82	85

In [52]:

```
##   Answer to question -- My preference would be the grades.iloc[[0,2]] approach
```

Task 3. Accessing Subsets (at, iat)

In [53]:

```
grades.loc['Test1':'Test2', ['Eva', 'Katie']]
```

Out[53]:

	Eva	Katie
Test1	100	100
Test2	87	81

In [54]:

```
grades.iloc[[0,2], 0:3]
```

Out[54]:

	Wally	Eva	Sam
Test1	87	100	94
Test3	70	90	90

Task 4. Describe (By Column)

In [55]:

```
grades.describe()
```

Out[55]:

	Wally	Eva	Sam	Katie	Bob
count	3.00	3.00	3.00	3.00	3.00
mean	84.33	92.33	87.00	87.67	77.67
std	13.20	6.81	8.89	10.69	11.02
min	70.00	87.00	77.00	81.00	65.00
25%	78.50	88.50	83.50	81.50	74.00
50%	87.00	90.00	90.00	82.00	83.00
75%	91.50	95.00	92.00	91.00	84.00
max	96.00	100.00	94.00	100.00	85.00

In [56]:

```
pd.set_option("display.precision", 2)
```

In [57]:

```
grades.describe()
```

Out[57]:

	Wally	Eva	Sam	Katie	Bob
count	3.00	3.00	3.00	3.00	3.00
mean	84.33	92.33	87.00	87.67	77.67
std	13.20	6.81	8.89	10.69	11.02
min	70.00	87.00	77.00	81.00	65.00
25%	78.50	88.50	83.50	81.50	74.00
50%	87.00	90.00	90.00	82.00	83.00
75%	91.50	95.00	92.00	91.00	84.00
max	96.00	100.00	94.00	100.00	85.00

In [58]:

```
grades.mean()
```

Out[58]:

```
Wally     84.33
Eva       92.33
Sam       87.00
Katie     87.67
Bob       77.67
dtype: float64
```

Task 5. Transpose (rows <--> columns)

In [59]:

```
grades.T
```

Out[59]:

	Test1	Test2	Test3
Wally	87	96	70
Eva	100	87	90
Sam	94	77	90
Katie	100	81	82
Bob	83	65	85

In [60]:

```
grades.T.describe()
```

Out[60]:

	Test1	Test2	Test3
count	5.00	5.00	5.00
mean	92.80	81.20	83.40
std	7.66	11.54	8.23
min	83.00	65.00	70.00
25%	87.00	77.00	82.00
50%	94.00	81.00	85.00
75%	100.00	87.00	90.00
max	100.00	96.00	90.00

In [61]:

```
grades.T.mean()
```

Out[61]:

```
Test1    92.8
Test2    81.2
Test3    83.4
dtype: float64
```

Task 6. Sort

In [62]:

```
grades.sort_index(ascending=False)
```

Out[62]:

	Wally	Eva	Sam	Katie	Bob
Test3	70	90	90	82	85
Test2	96	87	77	81	65
Test1	87	100	94	100	83

In [63]:

```
grades.sort_index(axis=1)
```

Out[63]:

	Bob	Eva	Katie	Sam	Wally
Test1	83	100	100	94	87
Test2	65	87	81	77	96
Test3	85	90	82	90	70

In [64]:

```
grades.sort_values(by='Test1', axis=1, ascending=False)
```

Out[64]:

	Eva	Katie	Sam	Wally	Bob
Test1	100	100	94	87	83
Test2	87	81	77	96	65
Test3	90	82	90	70	85

In [65]:

```
grades.loc['Test1'].sort_values(ascending=False)
```

Out[65]:

```
Eva       100
Katie     100
Sam        94
Wally      87
Bob        83
Name: Test1, dtype: int64
```

In [66]:

```
grades.sort_index(inplace=True)
```

In [67]:

```
grades
```

Out[67]:

	Wally	Eva	Sam	Katie	Bob
Test1	87	100	94	100	83
Test2	96	87	77	81	65
Test3	70	90	90	82	85