Hello World

```
In [3]: # I can't believe we have to do this.... but whatever
        # Sean Kennedy
        # MSDS 7330 401
        # hello_world.py
        # 9-16-2019
        stupid_text_we_have_to_display = "Hello World"
        stupid_text_we_have_to_display

Out[3]: 'Hello World'
```

```
In [4]: # or if you're a complete muppet
        print(stupid_text_we_have_to_display)

        Hello World
```

ProTip

```
pip install jupytext and never worry about creating a .py file ever again!!!! :)
```

Tic Tac Toe

```
In [5]: tic_tac_toe_board = '''
                               |   |
                           -------------
                               |   |
                           -------------
                               |   |
                               '''
        print(tic_tac_toe_board)

                               |   |
                           -------------
                               |   |
                           -------------
                               |   |
```

Bonus:

```
Free lesson in f-strings and dictionaries!
```

In [6]:

```
def set_move_on_board(position, player):
    if position_dict[position] == '':
        position_dict[position] = player
        tic_tac_toe_board = f'''{position_dict['1']} | {position_dict['2']} |
{position_dict['3']}
---------
{position_dict['4']} | {position_dict['5']} | {position_dict['6']}
---------
{position_dict['7']} | {position_dict['8']} | {position_dict['9']}'''
    else:
        raise('That position is already taken!')
    return tic_tac_toe_board
```

In [7]:

```
position_dict = {
    '1':'',
    '2':'',
    '3':'',
    '4':'',
    '5':'',
    '6':'',
    '7':'',
    '8':'',
    '9':''
}

tic_tac_toe_board = f'''{position_dict['1']} | {position_dict['2']} | {positio
n_dict['3']}
---------
{position_dict['4']} | {position_dict['5']} | {position_dict['6']}
---------
{position_dict['7']} | {position_dict['8']} | {position_dict['9']}
                                          '''
print(tic_tac_toe_board)
```

```
  |  |
---------
  |  |
---------
  |  |
```

In [8]:

```
game_on = input('Would you like to play a game? Y/N')
```

```
Would you like to play a game? Y/NY
```

In [9]:

```
if(game_on) == 'Y':
    x_move = input('Great! You are Xs and I am Os, because Os are great and yo
u suck!\n Now make a move player X, choose a slot! (1-9)')
    tic_tac_toe_board = set_move_on_board(x_move, 'X')
```

```
Great! You are Xs and I am Os, because Os are great and you suck!
 Now make a move player X, choose a slot! (1-9)2
```

In [10]:
```
print(tic_tac_toe_board)
```

```
 | x |
---------
 |  |
---------
 |  |
```

- Could write some loops and more interactions here (and some checks to see when/if game is over - but I'm bored and have real work to do :)

In [12]:
```
#import the pandas library
import pandas as pd
```

DataSetExploration exercise

In [26]:
```
housing = pd.read_csv('data/AmesHousing.csv')
housing.describe()
```

Out[26]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	Yea
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.0
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.2
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.2
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.0
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.0
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.0
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.0
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.0

8 rows × 38 columns

In [28]:
```
housing.columns
```

Out[28]:
```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAd
d',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBat
h',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageTyp
e',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQua
l',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

Question 1 : Create a dataframe with the following columns : YearBuilt , HomePrice , LotArea , BedroomAbvGr and FullBaths

In [82]:
```
housing_reduced = housing[['YearBuilt', 'SalePrice', 'LotArea', 'BedroomAbvGr'
, 'FullBath']]
housing_reduced.describe()
```

Out[82]:

	YearBuilt	SalePrice	LotArea	BedroomAbvGr	FullBath
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	1971.267808	180921.195890	10516.828082	2.866438	1.565068
std	30.202904	79442.502883	9981.264932	0.815778	0.550916
min	1872.000000	34900.000000	1300.000000	0.000000	0.000000
25%	1954.000000	129975.000000	7553.500000	2.000000	1.000000
50%	1973.000000	163000.000000	9478.500000	3.000000	2.000000
75%	2000.000000	214000.000000	11601.500000	3.000000	2.000000
max	2010.000000	755000.000000	215245.000000	8.000000	3.000000

Question 2 : What is the avg price of single family homes

In [63]:

```
avg_prices_by_style = housing.groupby(['BldgType']).mean()[['SalePrice']].reset_index()
avg_prices_by_style
```

Out[63]:

	BldgType	SalePrice
0	1Fam	185763.807377
1	2fmCon	128432.258065
2	Duplex	133541.076923
3	Twnhs	135911.627907
4	TwnhsE	181959.342105

Question 3 : What is the mean home price of the single family homes built after 1950

In [62]:

```
avg_prices_by_style_after_1950 = housing[housing.YearBuilt > 1950].groupby(['BldgType']).mean()[['SalePrice']].reset_index()
avg_prices_by_style_after_1950
```

Out[62]:

	BldgType	SalePrice
0	1Fam	203555.156798
1	2fmCon	142827.777778
2	Duplex	136530.363636
3	Twnhs	135911.627907
4	TwnhsE	181959.342105

Question 4 : What is the median home price per number of bedrooms in the house?

In [61]:

```
median_prices_by_bedroom_count = housing.groupby(['BedroomAbvGr']).median()[['SalePrice']].reset_index()
median_prices_by_bedroom_count
```

Out[61]:

	BedroomAbvGr	SalePrice
0	0	202500.0
1	1	145250.0
2	2	137250.0
3	3	169945.0
4	4	193500.0
5	5	161500.0
6	6	141000.0
7	8	200000.0

Question 5 : What is the most expensive home in each Neighborhood

In [78]:

```
most_expensive_homes_by_neighborhood = housing.sort_values(['SalePrice']
                                                           , ascending=False).
groupby(['Neighborhood'])[['SalePrice']].first()
most_expensive_homes_by_neighborhood
```

Out[78]:

	SalePrice
Neighborhood	
Blmngtn	264561
Blueste	151000
BrDale	125000
BrkSide	223500
ClearCr	328000
CollgCr	424870
Crawfor	392500
Edwards	320000
Gilbert	377500
IDOTRR	169500
MeadowV	151400
Mitchel	271000
NAmes	345000
NPkVill	155000
NWAmes	299800
NoRidge	755000
NridgHt	611657
OldTown	475000
SWISU	200000
Sawyer	190000
SawyerW	320000
Somerst	423000
StoneBr	556581
Timber	378500
Veenker	385000

Question 6 :Sort homes by the year built

In [81]:
```
sort_by_year = housing.sort_values(['YearBuilt'], ascending=False)
sort_by_year
```

Out[81]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContou
378	379	20	RL	88.0	11394	Pave	NaN	Reg	Lv
157	158	60	RL	92.0	12003	Pave	NaN	Reg	Lv
644	645	20	FV	85.0	9187	Pave	NaN	Reg	Lv
762	763	60	FV	72.0	8640	Pave	NaN	Reg	Lv
412	413	20	FV	NaN	4403	Pave	NaN	IR2	Lv
...	...	...	...	...	...	...	...	...	.
747	748	70	RM	65.0	11700	Pave	Pave	IR1	Lv
1132	1133	70	RM	90.0	9900	Pave	NaN	Reg	Lv
630	631	70	RM	50.0	9000	Pave	Grvl	Reg	Lv
1137	1138	50	RL	54.0	6342	Pave	NaN	Reg	Lv
1349	1350	70	RM	50.0	5250	Pave	Pave	Reg	Lv

1460 rows × 81 columns