1.Use the following to set default parameters:

Double-click (or enter) to edit

```
import numpy as np
import pandas as pd
print("Pandas Version:", pd.__version__)
pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 500)
```

```
Pandas Version: 1.5.3
```

2.In pandas, we can create data structures in two ways: series and dataframes Check the following snippet to understand how we can create a dataframe from series, dictionary, and n-dimensional arrays The following code snippet shows how we can create a dataframe from a series:

Double-click (or enter) to edit

```
series = pd.Series([2, 3, 7, 11, 13, 17, 19, 23])
print(series)
# Creating dataframe from Series
series_df = pd.DataFrame({
  'A': range(1, 5),
  'B': pd.Timestamp('20190526'),
  'C': pd.Series(5, index=list(range(4)), dtype='float64'),
  'D': np.array([3] * 4, dtype='int64'),
  'E': pd.Categorical(["Depression", "Social Anxiety", "Bipolar Disorder", "Eating Disorder"]),
  'F': 'Mental health',
  'G': 'is challenging'
 })
print(series_df)
```

```
0     2
1     3
2     7
3    11
4    13
5    17
6    19
7    23
dtype: int64
   A          B    C  D                 E              F               G
0  1 2019-05-26  5.0  3        Depression  Mental health  is challenging
1  2 2019-05-26  5.0  3    Social Anxiety  Mental health  is challenging
2  3 2019-05-26  5.0  3  Bipolar Disorder  Mental health  is challenging
3  4 2019-05-26  5.0  3   Eating Disorder  Mental health  is challenging
```

The following code snippet shows how to create a dataframe for a dictionary:

```
# Creating dataframe from Dictionary
dict_df = [{'A': 'Apple', 'B': 'Ball'},{'A': 'Aeroplane', 'B':'Bat', 'C': 'Cat'}]
dict_df = pd.DataFrame(dict_df)
print(dict_df)
```

```
           A     B    C
0      Apple  Ball  NaN
1  Aeroplane   Bat  Cat
```

4.The following code snippet shows how to create a dataframe **bold text** from n-dimensional **arrays:**

```
# Creating a dataframe from ndarrays
```

```
sdf = {
       'County':['Østfold', 'Hordaland', 'Oslo', 'Hedmark', 'Oppland', 'Buskerud'],
       'ISO-Code':[1,2,3,4,5,6],
       'Area': [4180.69, 4917.94, 454.07, 27397.76, 25192.10, 14910.94],
       'Administrative centre': ["Sarpsborg", "Oslo", "City of Oslo", "Hamar", "Lillehammer", "Drammen"]
            }
sdf = pd.DataFrame(sdf)
print(sdf)
```

```
      County  ISO-Code      Area Administrative centre
0    Østfold         1   4180.69             Sarpsborg
1  Hordaland         2   4917.94                  Oslo
2       Oslo         3    454.07          City of Oslo
3    Hedmark         4  27397.76                 Hamar
4    Oppland         5  25192.10           Lillehammer
5   Buskerud         6  14910.94               Drammen
```

3. Now, let's load a dataset from an external source into a pandas DataFrame. After3. that, let's see the first 10 entries:

```
columns = ['age', 'workclass', 'fnlwgt', 'education','education_num','marital_status', 'occupation', 'relationship', 'ethnicity', '
df = pd.read_csv('/content/adult_csv.csv')
df.head(10)
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationsh
0	2	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-fam
1	3	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husba
2	2	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-fam
3	3	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husba
4	1	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	W
5	2	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	W
6	3	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-fam
7	3	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husba
8	1	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-fam
9	2	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husba

4.The following code displays the rows, columns, data types, and memory used by the dataframe:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29530 entries, 0 to 29529
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             29530 non-null  int64  
 1   workclass       27875 non-null  object 
 2   fnlwgt          29530 non-null  int64  
 3   education       29530 non-null  object 
 4   education-num   29530 non-null  int64  
 5   marital-status  29530 non-null  object 
 6   occupation      27870 non-null  object 
 7   relationship    29530 non-null  object 
 8   race            29529 non-null  object 
 9   sex             29529 non-null  object 
 10  capitalgain     29529 non-null  float64
 11  capitalloss     29529 non-null  float64
 12  hoursperweek    29529 non-null  float64
 13  native-country  28998 non-null  object 
 14  class           29529 non-null  object 
dtypes: float64(3), int64(3), object(9)
memory usage: 3.4+ MB
```

5.Let's now see how we can select rows and columns in any dataframe:

```
# Selects a row
df.iloc[10]
# Selects 10 rows
df.iloc[0:10]
# Selects a range of rows
df.iloc[10:15]
# Selects the last 2 rows
df.iloc[-2:]
# Selects every other row in columns 3-5
df.iloc[::2, 3:5].head()
```

	education	education-num
0	Bachelors	13
2	HS-grad	9
4	Bachelors	13
6	9th	5
8	Masters	14

6.Let's combine NumPy and pandas to create a dataframe as follows:

```
import pandas as pd
import numpy as np
np.random.seed(24)
dFrame = pd.DataFrame({'F': np.linspace(1, 10, 10)})
dFrame = pd.concat([df,pd.DataFrame(np.random.randn(10, 5),columns=list('EDCBA'))],axis=1)
dFrame.iloc[0, 2] = np.nan
dFrame
```

	F	E	D	C	B	A	E	D	
0	1.0	1.329212	NaN	-0.316280	-0.990810	-1.070816	1.329212	-0.770033	-0.
1	2.0	-1.438713	0.564417	0.295722	-1.626404	0.219565	-1.438713	0.564417	0.
2	3.0	0.678805	1.889273	0.961538	0.104011	-0.481165	0.678805	1.889273	0.

7.Let's style this table using a custom rule. If the values are greater than zero, we change the color to black (the default color); if the value is less than zero, we change the color to red; and finally, everything else would be colored green. Let's define a Python function to accomplish that:

6	7.0	-0.385684	0.519818	1.686583	-1.325963	1.428984	-0.385684	0.519818	1.

```
# Define a function that should color the values that are less than 0
def colorNegativeValueToRed(value):
  if value < 0:
     color = 'red'
  elif value > 0:
    color = 'black'
  else:
   color = 'green'
  return 'color: %s' %color
```

8.Now, let's pass this function to the dataframe. We can do this by using the style8. method provided by pandas inside the dataframe:

```
s = df.style.applymap(colorNegativeValueToRed, subset=['A','B','C','D','E'])
s
```

	F	E	D	C	B	A
0	1.000000	1.329212	nan	-0.316280	-0.990810	-1.070816
1	2.000000	-1.438713	0.564417	0.295722	-1.626404	0.219565
2	3.000000	0.678805	1.889273	0.961538	0.104011	-0.481165
3	4.000000	0.850229	1.453425	1.057737	0.165562	0.515018
4	5.000000	-1.336936	0.562861	1.392855	-0.063328	0.121668
5	6.000000	1.207603	-0.002040	1.627796	0.354493	1.037528
6	7.000000	-0.385684	0.519818	1.686583	-1.325963	1.428984
7	8.000000	-2.089354	-0.129820	0.631523	-0.586538	0.290720
8	9.000000	1.264103	0.290035	-1.970288	0.803906	1.030550
9	10.000000	0.118098	-0.021853	0.046841	-1.628753	-0.392361

9.Now, let's go one step deeper. We want to scan each column and highlight the 9. maximum value and the minimum value in that column:

```
def highlightMax(s):
  isMax = s == s.max()
  return ['background-color: orange' if v else '' for v in isMax]
def highlightMin(s):
  isMin = s == s.min()
  return ['background-color: green' if v else '' for v in isMin]
```

```
df.style.apply(highlightMax).apply(highlightMin).highlight_null(null_color='red')
```

```
<ipython-input-51-21252515042f>:1: FutureWarning: `null_color` is deprecated: use
  df.style.apply(highlightMax).apply(highlightMin).highlight_null(null_color='red'
```

```
          F          E          D          C          B          A
0  1.000000   1.329212        nan  -0.316280  -0.990810  -1.070816
1  2.000000  -1.438713   0.564417   0.295722  -1.626404   0.219565
2  3.000000   0.678805   1.889273   0.961538   0.104011  -0.481165
3  4.000000   0.850229   1.453425   1.057737   0.165562   0.515018
4  5.000000  -1.336936   0.562861   1.392855  -0.063328   0.121668
5  6.000000   1.207603  -0.002040   1.627796   0.354493   1.037528
```

10.Are you still not happy with your visualization? Let's try to use another Pythonlibrary called seaborn and provide a gradient to the table:

```
import seaborn as sns
colorMap = sns.light_palette("pink", as_cmap=True)
styled = df.style.background_gradient(cmap=colorMap)
styled
```

```
           F          E          D          C          B          A
0   1.000000   1.329212        nan  -0.316280  -0.990810  -1.070816
1   2.000000  -1.438713   0.564417   0.295722  -1.626404   0.219565
2   3.000000   0.678805   1.889273   0.961538   0.104011  -0.481165
3   4.000000   0.850229   1.453425   1.057737   0.165562   0.515018
4   5.000000  -1.336936   0.562861   1.392855  -0.063328   0.121668
5   6.000000   1.207603  -0.002040   1.627796   0.354493   1.037528
6   7.000000  -0.385684   0.519818   1.686583  -1.325963   1.428984
7   8.000000  -2.089354  -0.129820   0.631523  -0.586538   0.290720
8   9.000000   1.264103   0.290035  -1.970288   0.803906   1.030550
9  10.000000   0.118098  -0.021853   0.046841  -1.628753  -0.392361
```

✓ 0s completed at 1:35 PM ● ✕