

Day 14

```
def hello():
    print( " hello world")
invoke it with hello()
def welcome(name):
    print( f'Hello, {name} ')
invoke it with welcome ("amy")
def welcome_greeting(name, greeting):
    print(f' Hey{name}. {greeting} ')
invoke it with welcome_greeting(' Liz', "How are you?") --> Kwargs
def exponent(base,exponent):
    power = base ** exponent
    return power

num1 = 2
num2 = 3
answer = exponent( num1,num2)
print( answer)
print( exponent( 2,3))
def sum(*parameters): --> unlimited parameters
    total = 0
    for i in parameters:
        total = total + each_number
    return total

sum = sum(1, 2,3 ,4,5)
print(sum)
```

Day 16

```
Df = pd.read_csv(url)
print( df.to_string())
Null is null, a null is something = a special creation to indicate the absence of a value - its a made up value
Df.shape = not a method does not need ()
Look at only one column = df['School Name']
```

Day 16 (cont)

```
> print( df['School Name'].to_string())
Find the unique names= df['School Name'].unique()
type(unique_schools) shows the type.. This is not a data frame
Statistics:
df['Starting Salary'].max() or df['Starting Salary'].mean() or df['Starting Salary'].min()
Find the NAs = df['Starting Salary'].isna() then to count the trues = na_rows.sum()
Based on a condition
Df2 = df.query("Starting Salary > 75000 ")
```

Day 20

```
Change individual values
Df.loc[20, 'Starting Salary'] = ' '
Convert to numeric type = df['Starting Salary'] = pd.to_numeric( df['Starting Salary'] )
Df.loc [139, 'Starting Salary'] = 46000
#Convert starting salary to numeric FORCE CONVERT or "COERCE" conversion
Error_columns = pd.to_numeric( df['Starting Salary'], errors= 'coerce')
print( error_columns)
#find the NAs
Nas = error_columns.isna()
print(Nas)
Df[20:25]
#fix columns
Df.loc[70, 'Starting Salary'] = 42600
df[Nas]
Save it to the original by overwriting
df['Starting Salary'] = pd.to_numeric(df['Starting Salary'])
```



Day 15

```
import pandas as pd
data_list = [45,74,78]
series_of_numbers = pd.Series(data_list)
print(series_of_numbers[1]) --> 74
Years = [2021, 2022, 2023]
series_of_numbers = pd.Series(data_list, index=years)
print(series_of_numbers)
print(series_of_numbers[2021] --> 2021
grades = {'A': 34, 'B': 56}
grade_series = pd.Series(grades)
print(grade_series) or print(grade_series['A'])
quiz_scores = {
    'Quiz1': [32,56,56],
    'Quiz2': [78,34,32]}
df = pd.DataFrame(data=quiz_scores)
print(df)
# to overwrite -->
df = pd.DataFrame(data=quiz_scores, index=['Mike', 'Susan', 'Amy'])
df.head() --> top 5 rows
df.tail() --> bottom 5 rows
df[40:60] --> rows 40-59
```

Day 18

```
Find all the schools with the name Pitt
Df2 = df.query("School Name == 'Pitt' ")
df2.head()
Remove a column: df.drop(columns='Starting Salary', inplace=True)
Or df = df.drop(columns='Starting Salary')
Drop a row df.drop(index=2, inplace=True)
Delete entire row of data when one column had missing data df=df.dropna()
```

Day 19

```
Load descriptives for the df = df.describe()
Load tab-delimited file
Df2 = pd.read_csv(URL, sep='\t')
Replace function:
Df['School Name'].replace('-', '-', regex=True, inplace=True)
Fillna = df['Starting Salary'].fillna(0, inplace=True)
How many unique school names are there:
len(df['School Name'].unique())
Show only the rows in which df are duplicate:
Duplicates = df.duplicated(subset='School Name')
Boolean series = df[duplicates]
Df2 = df.drop_duplicates(subset='School Name', keep='first')
Find out schools with specified
PA_schools = df2['School Name'].str.contains('Pennsylvania')
Use a boolean series df2[PA_schools]
Overwrite instead on inplace
Df2 = df2.sort_values('Starting Salary', ascending=False)
Fix one bad value:
Df2.loc[2, 'Starting Salary'] = df2['Starting Salary'].mean()
```

