14

```
Define a function =
def hello():
print( 'Hello World' )
Then invoke it with = hello()
Define a function with a PARAMETER (argument)
Def welcom e(n ame):
Print ( f ' Hello, {name} ' )
Invoke it with = welcome( 'Amy' )
Two required Parame ters:
Def welcom e_g ree tin g(name, greeti ng_ text):
print( f ' Hey, {name}. {greet ing _text} ')
Invoke it with = welcom e_g ree ting( 'Liz' , 'How
are you?' ) - these are known as KWARG
Define a method to do a calcul ation
Def expone nt( base, exponent):
Power = base ** exponent
Return power
Num1 = 2
Num2 = 3
Answer = expone nt( num1, num2)
print( answer)
print( expone nt( 2,3))
Def sum_of _nu mbers( *param eters):
Total = 0
For each_n umber in parame ters:
Total = total + each_n umber
Return total
Sum = sum_of _nu mbe rs( 1,2 ,3, 4,5,6)
print(sum
```

16

```
Df = pd.read_csv(url)
print( df.t o_ str ing())
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']
print( df[ 'School Name' ].to_s tring( ))
```

16 (cont)

> 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['St-arting 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 ")

20

```
Change individual values
Df.loc[20, 'Starting Salary'] = ' '
Convert to numeric type = df['St arting Salary'] =
pd.to_ num eric( df['St arting Salary'] )
Df.loc [139, 'Starting Salary'] = 46000
#Convert starting salary to numeric FORCE CONVERT
or "COERCE" conversion
Error_ columns = pd.to_ num eric( df['St arting
Salary'], errors= 'coerce')
print( err or_ col umns)
#find the NAs
Nas = error_ col umn s.i sna()
print(Nas)
Df[20:25]
#fix columns
Df.loc[70, 'Starting Salary'] = 42600
df[Nas]
Save it to the original by overwr iting
df['St arting Salary'] = pd.to_ num eri c(d f[' -
Sta rting Salary'])
```

15

```
Import pandas as pd
Data_list = [45, 74, 78]
Series _of _nu mbers = pd.ser ies (da ta_ list)
print( ser ies _of _nu mbe rs[1])
Years = [2021, 2022, 2023]
Create series with labels and use KWARG
Series _of _nu mbers = pd.ser ies (da ta= dat -
a_list, index= years)
print( ser ies _of _nu mbers)
Show me the value for 2021
```

15 (cont)

> print(series_of_numbers[2021])
Create a series with integrated data labels
Grade_distribution = {'A' : 34, 'B' : 56}
Convert the dictionary ^ to a series = grade_series = pd.series(data=-grade_distribution)
print(grade_series) or print(grade_series['A'])
2 dimensional data - in multiple lists
Quiz_scores = {
'Quiz1' : [32, 56, 56] ,
'Quiz2' : [78, 34, 32]}
Df = pd.DataFrame(data=quiz_scores)
print(df)
Overwrite the df like this:
Df = pd.DataFrame(data=quiz_scores, index=['Mike' , 'Susan', 'Amy']
df.head() = top 5 rows df.tail() = bottom 5 rows df[40:60] = select row

18

```
Find all the schools with the name Pitt
Df2 = df.query( " School Name == 'Pitt' ")
df2.head()
Remove a column: df.dro p(c olu mns ='S tarting
Salary', inplace = True)
Or df = df.dro p(c olu mns ='S tarting Salary')
Drop a row df.dro p(i ndex=2, inplace = True)
Delete entire row of data when one column had
missing data df=df.d ro pna()
```

19

```
Load descriptives for the df = df.describe()
Load tab-de lim itted file
Df2 = pd.rea d_c sv(URL, sep='\t' )
Replace function:
Df[ 'School Name'].re pla ce('-', ' -', regex= -
True, inplac e=True)
Fillnas = df['St arting Salary '].f il lna(0,
inplac e=True)
How many unique school names are there:
len( df['School Name'].un ique())
Show only the rows in which df are duplicate:
Duplicates = df.dup lic ate d(s ubset= 'School
Name')
Boolean series = df[dup lic ates]
Df2 = df.dro p_d upl ica tes (su bse t=' School
Name', keep=' first')
```

19 (cont)

> 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()

19

```
Load descriptives for the df = df.describe()
Load tab-de lim itted file
Df2 = pd.rea d_c sv(URL, sep='\t' )
Replace function:
Df[ 'School Name'].re pla ce('-', ' -', regex= -
True, inplac e=True)
Fillnas = df['St arting Salary '].f il lna(0,
inplac e=True)
How many unique school names are there:
len( df['School Name'].un ique())
Show only the rows in which df are duplicate:
Duplicates = df.dup lic ate d(s ubset= 'School
Name')
Boolean series = df[dup lic ates]
Df2 = df.dro p_d upl ica tes (su bse t=' School
Name', keep=' first')
Find out schools with specified
PA_schools = df2['S chool Name'].st r.c ont ain -
s(' Pen nsy lva nia')
Use a boolean series df2[PA _sc hools]
Overwrite instead on inplace
Df2 = df2.so rt_ val ues ('S tarting Salary',
ascending = False)
Fix one bad value:
Df2.loc[2, 'Starting Salary'] = df2['S tarting
Salary '].m ean()
```