Python

List

Indexing

```
a_list = ["Python", "Data", "Science"]
```

```
a_list[0]
```

```
'Python'
```

List slicing

```
a_list[:2]
```

```
['Python', 'Data']
```

Tuple

Indexing

```
a_tuple = ("Python", "Data", "Science")
```

```
a_tuple[0]
```

```
'Python'
```

slicing

```
a_tuple[:2]
```

```
('Python', 'Data')
```

Unpacking

```
item_0, item_1, item_2 = ("Python", "Data", "Science")

item_0
```

```
'Python'
```

```
item_0, (item_1, item_2) = "Python", ("Data", "Science")

item_1
```

```
'Data'
```

```
fig, (plot1, plot2) = "Figure", ("Plot1", "Plot2")

plot1
```

```
'Plot1'
```

```
fig, ((plot1, plot2),(plot3, plot4)) = "Figure", (("Plot1", "Plot2"), ("Plot3", "Plot4"))

plot2
```

```
'Plot2'
```

Dictionaries

Create a dictionary

```
d = {"color": "mediumblue", "linestyle": "dashed"}
```

Access a dictionary

```
d["color"]
```

```
'mediumblue'
```

---

for Loops

```
for item in ["item 1", "item 2", "item 3"]:
    print(item)
```

```
item 1
item 2
item 3
```

zip

```
# zip combines the respective items from each list as a tuple

list(zip(["Data", "Machine", "Artificial"], ["Science", "Learning", "Intelligence"]))
```

```
[('Data', 'Science'), ('Machine', 'Learning'), ('Artificial', 'Intelligence')]
```

Unpacking zipped items

```
# Unpacking the zipped tuples as we loop through

for item1, item2 in zip(["Data", "Machine", "Artificial"], ["Science", "Learning", "Intelligence"]):
    print(item1, item2)
```

```
Data Science
Machine Learning
Artificial Intelligence
```

What the machine sees:

```
                  # after being zipped into a list of tuples...
for item1, item2 in [('Data', 'Science'), ('Machine', 'Learning'), ('Artificial', 'Intelligence')]:
    print(item1, item2)
```

```
Data Science
Machine Learning
Artificial Intelligence
```

Example

```
# Unpacking the created list of tuples as we loop through each zipped item

for box, color in zip(["box1", "box2", "box3"], ["lightblue", "mediumblue", "darkblue"]):
    print(box, color)
```

```
box1 lightblue
box2 mediumblue
box3 darkblue
```

What the machine sees:

```
for box, color in [('box1', 'light blue'), ('box2', 'medium blue'), ('box3', 'darkblue')]:
    print(box, color)
```

```
box1 light blue
box2 medium blue
box3 darkblue
```

Function

Define the function

```
def my_plot():
    return "my data viz..."
```

Call the function

```
# No parameters

my_plot()
```

```
'my data viz...'
```

Using parameters

```
def my_plot(plot_type, color):
    return f"My {color} {plot_type}"
```

Call the function

```
# Providing two required values

my_plot("histogram", "skyblue")
```

```
'My skyblue histogram'
```

Using default values

```
def my_plot(plot_type, color="mediumblue"):
    return f"My {color} {plot_type}"


# Using the default value for color

my_plot("histogram")
```

```
'My mediumblue histogram'
```

Providing an optional color value

```
my_plot("histogram", "lightblue")
```

```
'My lightblue histogram'
```

Pandas

```
from google.colab import drive
import os

drive.mount('/content/drive')
os.chdir('/content/drive/MyDrive/')
for item in os.listdir():
  print(item)
print("--------------")

os.chdir('/content/drive/MyDrive/cloud/GitHub/AdvDataViz/Notebooks/')
for item in os.listdir():
  print(item)

print("--------------")

notebooks = "/content/drive/MyDrive/cloud/GitHub/AdvDataViz/Notebooks"
print(os.listdir(notebooks))
print("--------------")

file = "heart-disease.csv"
file_path = os.path.join(notebooks, file)
with open(file_path, "r") as f:
  contents = f.read()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/con
learningStore
healthyCar
startup
Artificial Intelligence
cloud
--------------
heart-disease.csv
student_performance.csv
churn.csv
employee_attrition_.csv
Top 50 US Tech Companies.csv
01 Python_Pandas.ipynb
02 Matplotlib.ipynb
03 Matplotlib  - Exercise.ipynb
03 Matplotlib - Exercise Solutions.ipynb
04 Continuous Variables - Histogram .ipynb
05 Continuous Variables - Histogram - Exercise Solutions.ipynb
05 Continuous Variables - Histogram - Exercise .ipynb
```

```
06 Continuous Variables - Boxplot.ipynb
07 Continuous Variables - Boxplot - Exercise .ipynb
07 Continuous Variables - Boxplot - Exercise Solutions.ipynb
08 Continuous Variables - Scatterplot.ipynb
09 Continuous Variables - Scatterplot - Exercise .ipynb
09 Continuous Variables - Scatterplot - Exercise Solutions.ipynb
11 Pandas Data Visualization.ipynb
12 Seaborn.ipynb
13 Seaborn - Exercise .ipynb
13 Seaborn - Exercise Solution.ipynb
myplotlib.py
10 Categorical Variables - Bar_Pie.ipynb
14 Functions.ipynb
15 Custom Modules.ipynb
--------------
['heart-disease.csv', 'student_performance.csv', 'churn.csv', 'employee_attrition_.csv', 'Top 5
--------------
```

```
import pandas as pd
```

Data

DataFrame object

```
#df = pd.read_csv("heart-disease.csv")

df = pd.read_csv(file_path)
print(df.head())
```

```
   age     sex  chest_pain  rest_bp  chol  max_hr  st_depr  heart_disease
0   63  female           3      145   233     150      2.3              1
1   37  female           2      130   250     187      3.5              1
2   41    male           1      130   204     172      1.4              1
3   56  female           1      120   236     178      0.8              1
4   57    male           0      120   354     163      0.6              1
```

Start coding or generate with AI.

Preview dataset

```
# show first 5 rows

df.head()
```

	age	sex	chest_pain	rest_bp	chol	max_hr	st_depr	heart_disease
0	63	female	3	145	233	150	2.3	1
1	37	female	2	130	250	187	3.5	1
2	41	male	1	130	204	172	1.4	1
3	56	female	1	120	236	178	0.8	1
4	57	male	0	120	354	163	0.6	1

Next steps: Generate code with `df` View recommended plots New interactive sheet

Access a column

```
# Dictionary notation

df['sex']
```

	sex
0	female
1	female
2	male
3	female
4	male
...	...
298	male
299	female
300	female
301	female
302	male

303 rows × 1 columns

dtype: object

Unique values

```
df["sex"].unique()
```

```
array(['female', 'male'], dtype=object)
```

Selection and Filtering

Column selection

```
df[['age', 'sex', 'heart_disease']]    # providing a list selects multiple columns
```

	age	sex	heart_disease
0	63	female	1
1	37	female	1
2	41	male	1
3	56	female	1
4	57	male	1
...	...	...	...
298	57	male	0
299	45	female	0
300	68	female	0
301	57	female	0
302	57	male	0

303 rows × 3 columns

Row and Column selection with loc

Allows you to select a subset of the rows and columns using the label/name of the row/column

```
# loc (selection is inclusive) implies the name/label of the row, column

df.loc[:5, ["age", "sex"]]
```

	age	sex
0	63	female
1	37	female
2	41	male
3	56	female
4	57	male
5	57	female

Boolean row selection

```
df["sex"]=="female"
```

	sex
0	True
1	True
2	False
3	True
4	False
...	...
298	False
299	True
300	True
301	True
302	False

303 rows × 1 columns

dtype: bool

Using boolean for row selection

```
# row selection (return the rows that are True),  col selection

df.loc[df["sex"]=="female", ["age", "sex"]]
```

	age	sex
0	63	female
1	37	female
3	56	female
5	57	female
7	44	female
...	...	...
295	63	female
297	59	female
299	45	female
300	68	female
301	57	female

207 rows × 2 columns

& (and)

```
                  # row selection,                      col selection

df.loc[(df["sex"]=="female") & (df["age"] > 65), ["sex", "age", "heart_disease"]]
```

	sex	age	heart_disease
51	female	66	1
86	female	68	1
106	female	69	1
145	female	70	1
150	female	66	1
165	female	67	0
166	female	67	0
197	female	67	0
203	female	68	0
225	female	70	0
234	female	70	0
238	female	77	0
240	female	70	0
247	female	66	0
249	female	69	0
253	female	67	0
265	female	66	0
272	female	67	0
293	female	67	0
300	female	68	0

Binning

Convert a **continuous or interval** variable to a **categorical** variable.

```
df["age"].head(10)
```

	age
0	63
1	37
2	41

```
# bounds: (29, 39], (39, 49], (49, 59],(59, 69],(69, 79]

df["age"] = pd.cut(df["age"], [29, 39, 49, 59, 69, 79], labels=["thirties","forties","fifties","si›
df["age"].head(10)
```

	age
0	sixties
1	thirties
2	forties
3	fifties
4	fifties
5	fifties
6	fifties
7	forties
8	fifties
9	fifties

dtype: category

Useful methods

mean()

```
df["max_hr"].mean()
```

```
149.64686468646866
```

median()

```
df["max_hr"].median()
```

```
153.0
```