# Y Python

#### List

## Indexing

```
a_list = ["Python", "Data", "Science"]
a_list[0]

→ 'Python'
```

## ✓ List slicing

```
a_list[:2]

Typython', 'Data']
```

# Tuple

## Indexing

# Unpacking

→ ('Python', 'Data')

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
```

#### y 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
```

#### ✓ What the machine sees:

box3 darkblue

```
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

### ∨ 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()
Frive 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())
\rightarrow
               sex chest_pain rest_bp chol max_hr st_depr heart_disease
       age
        63 female
                             3
                                    145
                                           233
                                                  150
                                                            2.3
        37 female
                             2
                                    130
                                          250
                                                  187
                                                            3.5
                                                                             1
    1
    2
                             1
                                    130
                                          204
                                                  172
                                                            1.4
        41
              male
                                                                             1
        56 female
                             1
                                    120
                                           236
                                                  178
                                                            0.8
```

120

354

163

0.6

Start coding or generate with AI.

male

#### → Preview dataset

57

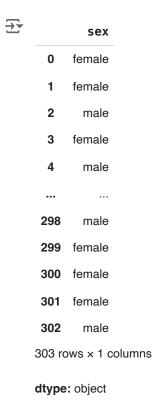
```
# show first 5 rows
df.head()
```

<b>→</b>		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	Next steps:		Generate code with df			/iew red	New interactive she		

#### Access a column

# Dictionary notation

df['sex']



## Unique values

```
df["sex"].unique()
array(['female', 'male'], dtype=object)
```

## Selection and Filtering

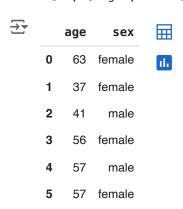
#### Column selection

df[['age', 'sex', 'heart\_disease']] # providing a list selects multiple columns **→**  $\blacksquare$ age sex heart disease 0 63 female 1 ılı 1 37 female 1 1 male 3 1 56 female 1 57 male 298 57 male 0 299 45 female 0 300 68 female 301 57 female 0 0 302 male 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"]]

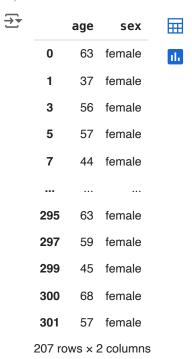


## Boolean row selection

```
df["sex"]=="female"
→
             sex
       0
            True
       1
            True
       2
           False
       3
            True
           False
      298
           False
      299
            True
      300
            True
      301
            True
      302 False
     303 rows x 1 columns
     dtype: bool
```

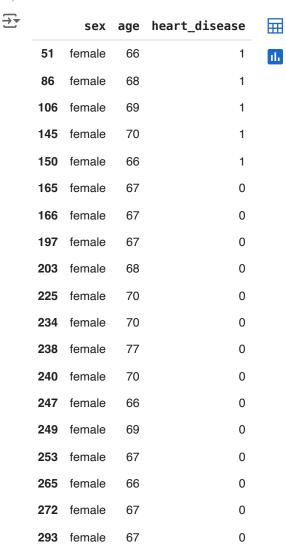
## Using boolean for row selection

```
# row selection (return the rows that are True), col selection
df.loc[df["sex"]=="female", ["age", "sex"]]
```



## 

 $\mbox{$\#$ row selection,} \qquad \mbox{$col selection}$   $\mbox{$df.loc[(df["sex"]=="female") \& (df["age"] > 65), ["sex", "age", "heart_disease"]]}$ 



## Binning

Convert a continuous or interval variable to a categorical variable.

0

df["age"].head(10)

300 female

68

```
\overline{2}
         age
      0
          63
          37
      1
                         # 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
         thirties
         forties
          fifties
      3
      4
          fifties
      5
          fifties
      6
          fifties
      7
          forties
      8
          fifties
          fifties
      9
     dtype: category
```

## Useful methods

## mean()

```
df["max_hr"].mean()
    149.64686468646866
```

## median()

```
df["max_hr"].median()
→ 153.0
```

. ^