wep
ENRICHMENT
FOR ALL

EDGE

JANUARY 8-19

Utrecht University

ENRICHMENT
FOR ALL

Machine Learning for Beginners

Hakim Qahtan

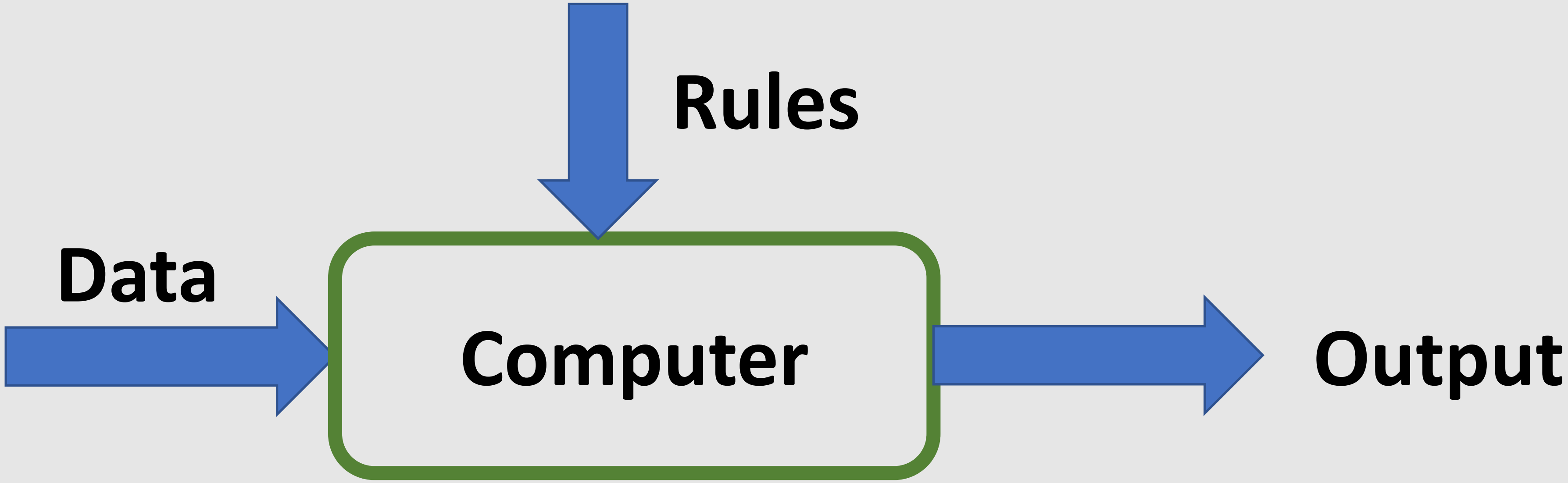
Department of Information and Computing Sciences

Utrecht University

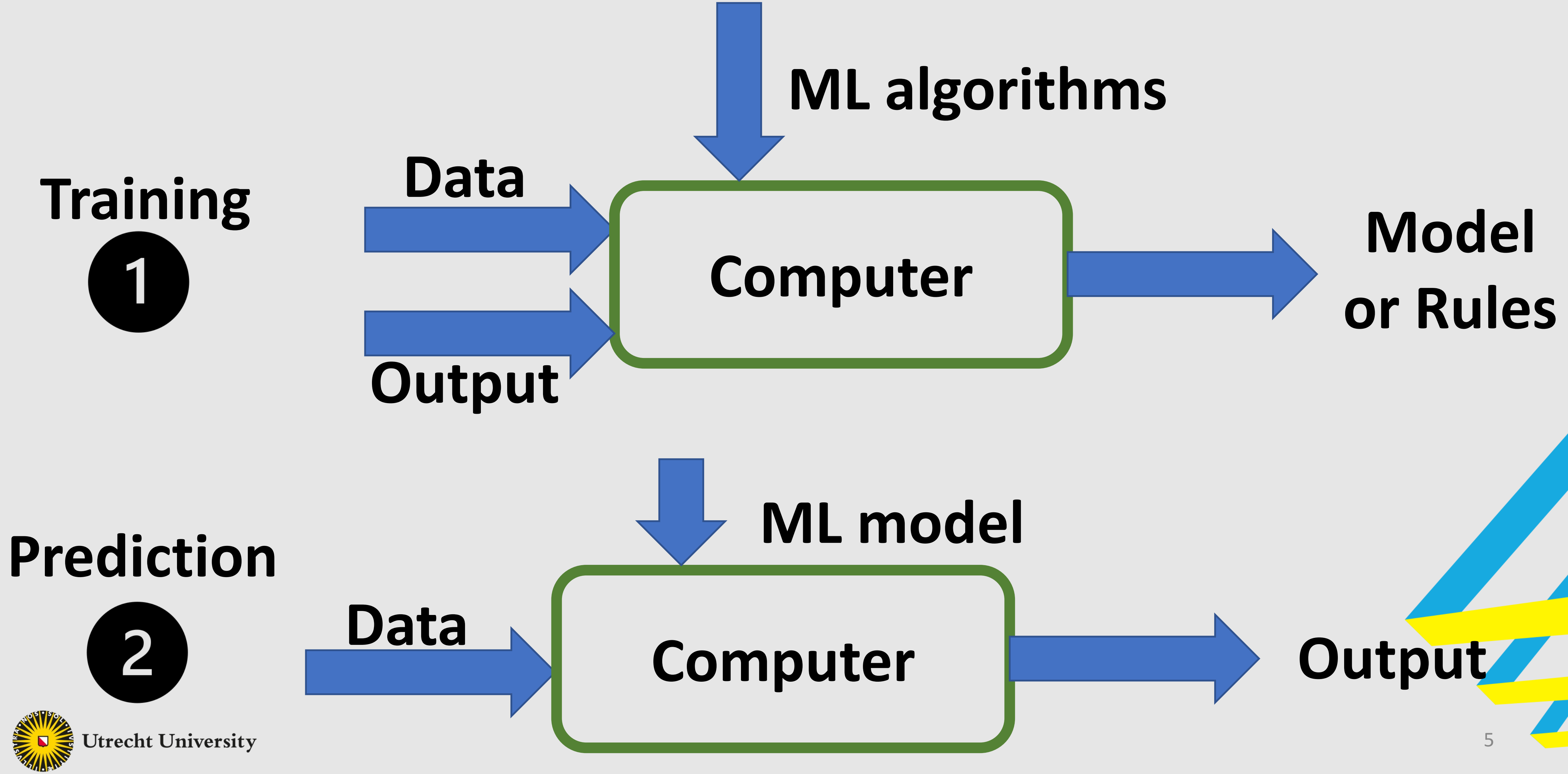
Utrecht University

Course Overview

Traditional programming



Supervised Learning



Training

❶

Data

Output

ML algorithms

Computer

Model or Rules

Prediction

❷

Data

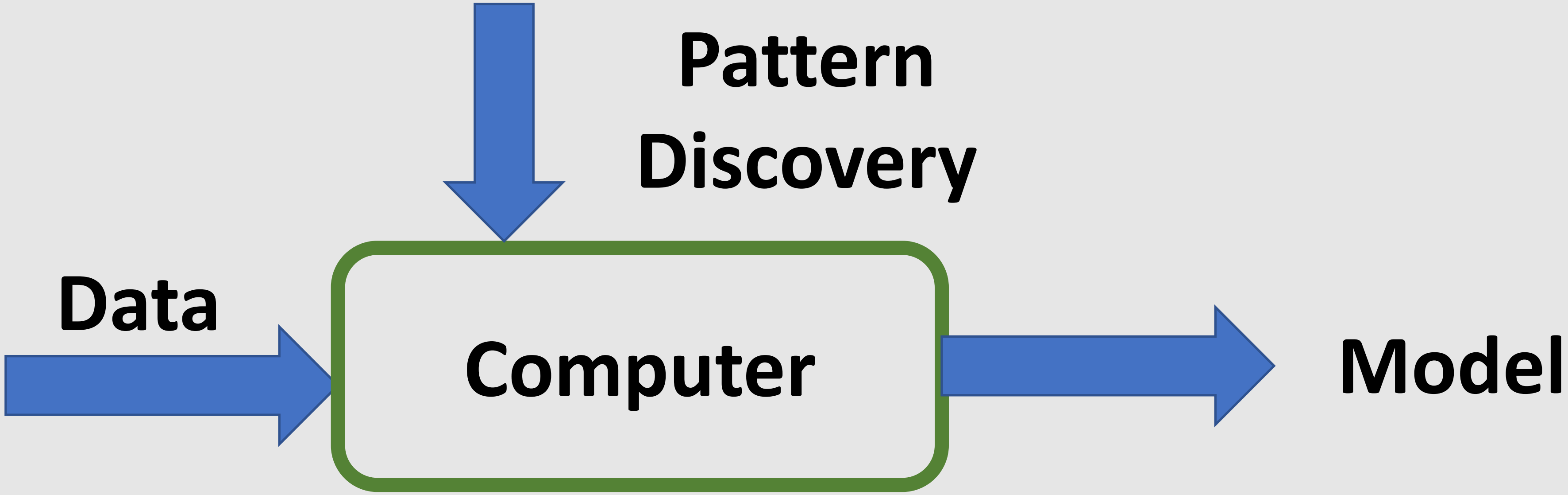
ML model

Computer

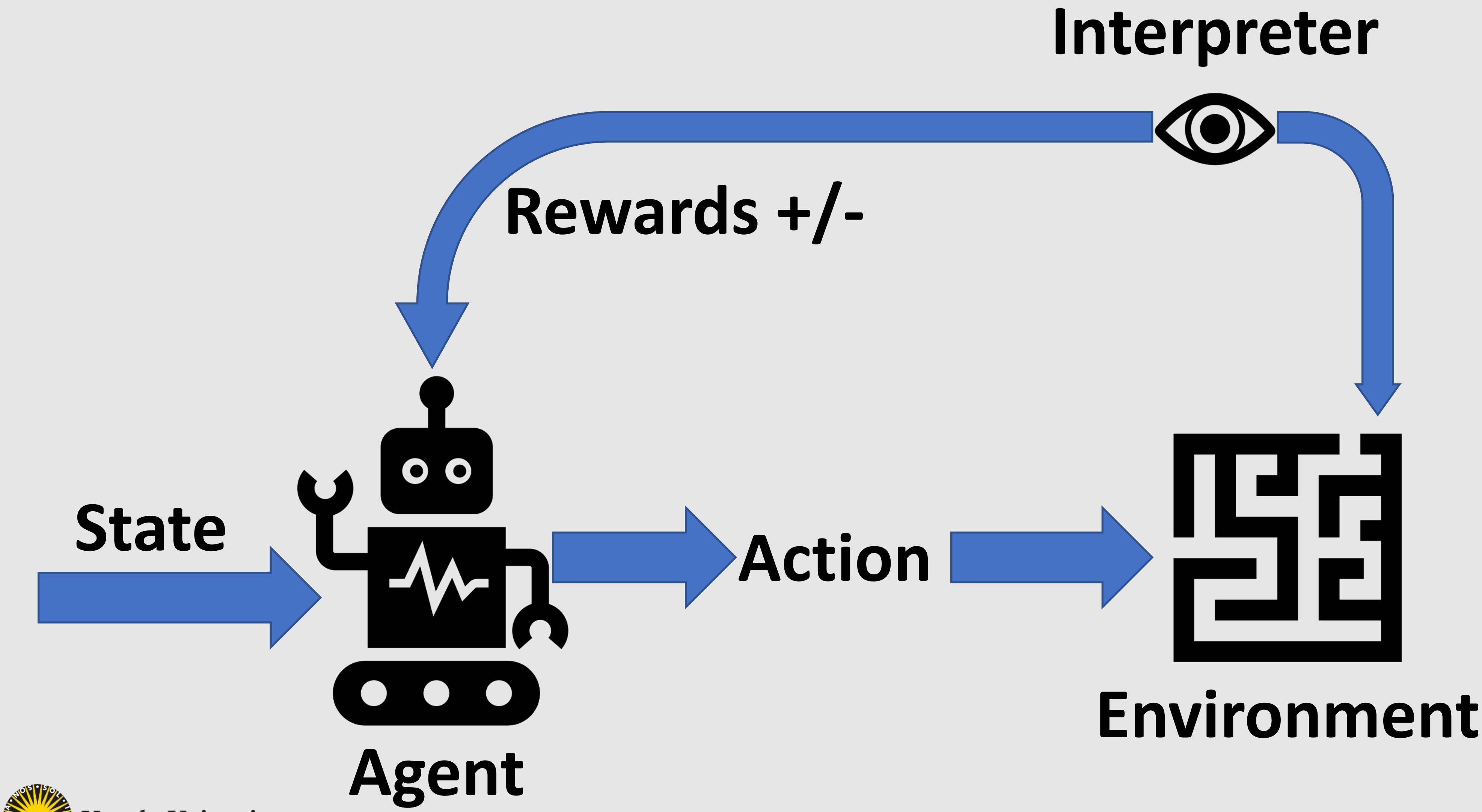
Output

Unsupervised Learning

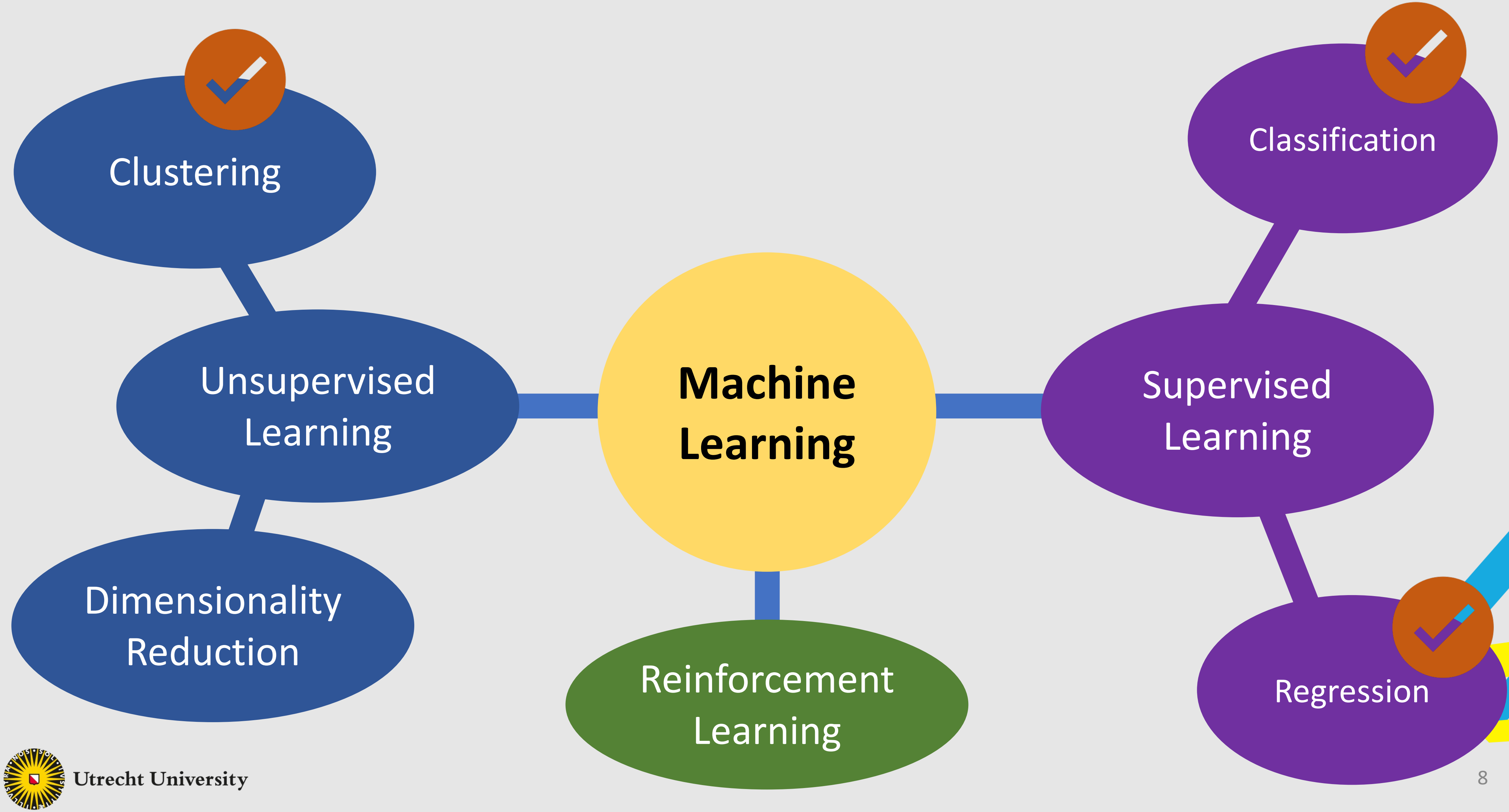
ENRICHMENT FOR ALL



Reinforcement Learning



Machine Learning



ML For Beginners

- The Basics of Python (Day 1)
- Supervised Learning
    - Regression and Demand Forecasting (Day 2)
    - Classification and Fairness (Day 3)
- Unsupervised Learning
    - Clustering (Day 4)

Today

- Wooclap
- Google Colab
- Introduction to Python

ENRICHMENT
FOR ALL

Introduction to Python

Part I : The Basics

Utrecht University

Python – The Basics

```
x = 34 - 23            # A comment.
y = "Hello"            # Another one.
z = 3.45
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + " World"   # String concat.
print (x)
print (y)
```

Python – The Basics

- Indentation matters to code meaning
- First assignment to a variable creates it
- Assignment is `=` and comparison is `==`
- For numbers `+ - * / %` are as expected
- Logical operators are words (`and, or, not`) *not* symbols
- The basic printing command is `print`

Python – Data Types

Integers (default for numbers)

```
x = 3
```

Floats

```
x = 3.456
```

Strings

Can use “” (double quotation) or ‘’ (single quotation) to specify strings

```
“abc”  == ‘abc’
```

Unmatched can occur within the string: `“matt’s”`

Python – The Basics

- Use a newline to end a line of code
  - Use \ when must go to next line prematurely
- indentations mark blocks of code
- Colons start of a new block in many constructs, e.g. function definitions, conditional clauses, loops
- Start comments with #, rest of line is ignored
- Use triple double quotation for comments over multiple lines

```
''' This comment goes
    over two lines '''
```

Python – The Basics

You can assign to multiple names at the same time

```
x, y = 2, 3
```

This makes it easy to swap values

```
x, y = y, x
```

Assignments can be chained

```
a = b = x = 2
```

Python – The Basics

Accessing a variable before it’s been properly created raises an error

```
print(w)
```

```
---------------------------------------------------------------------------
NameError Traceback (most recent call last) <ipython-input-35-ad11782dc618> in
<module> ----> 1 print (w) NameError: name 'w' is not defined
```

Instead

```
w = 3
print(w)
```

Python – Tuples, Lists, Strings, and Arrays

- Tuple: `t = (‘john’, 32, [CMSC])`
- Strings: `s = “John Smith” or s = ‘John Smith’`
- List: `l = [1, 2, ‘john’, (‘up’, ‘down’)]`
- Arrays: requires importing the numpy library
  - `import numpy as np`
  - `_1d = np.array([1, 2, 3])`
  - `_2d = np.array ([[1, 2, 3], [4, 5, 6]])`

Python – Matrices and Sets

- A matrix is 2-dimensional array
  - `import numpy as np`
  - `mat1 = np.matrix ([[1, 2, 3], [4, 5, 6]])`
  - `mat2 = np.matrix (‘1, 2, 3; 4, 5, 6’)`
- A set is list with no repetitions of the elements
  - `set1 = set([1, 1, 2, 3, 3, 4])`
  - The contents of `set1` will be `{1, 2, 3, 4}`

Python – Computing Statistical Quantities

- Mean value
    - `import numpy as np`
    - `np.mean([1, 2, 3, 4, 5, 6]) # OR`
    - `np.array([1, 2, 3, 4, 5, 6]).mean()`
- Standard deviation
    - `np.std([1, 2, 3, 4, 5, 6]) # OR`
    - `np.array([1, 2, 3, 4, 5, 6]).std()`

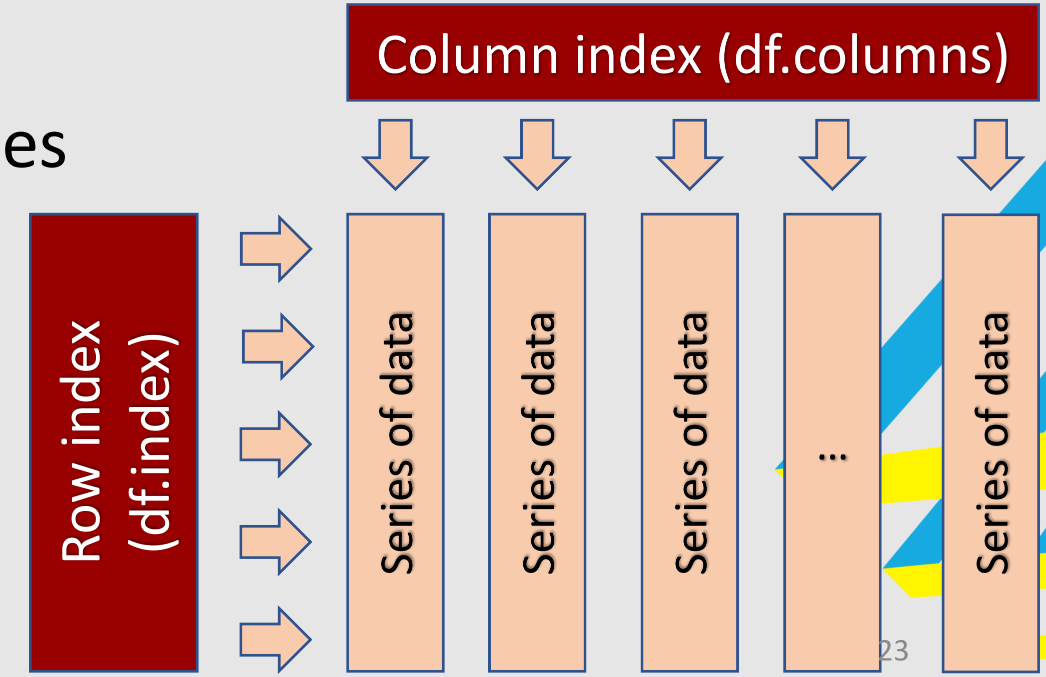
Coffee Break

Introduction to Python

Part II: Pandas DataFrames

Pandas Dataframes

- The most popular way to handle data tables in Python is using Pandas dataframes
- DataFrame: a rectangular table of data and contains an ordered collection of columns, each of which can be a different value type (numeric, string, boolean, etc.)
- Has columns and rows indexes
- Columns are made up of pandas series



Column index (df.columns)

Row index (df.index)

Series of data

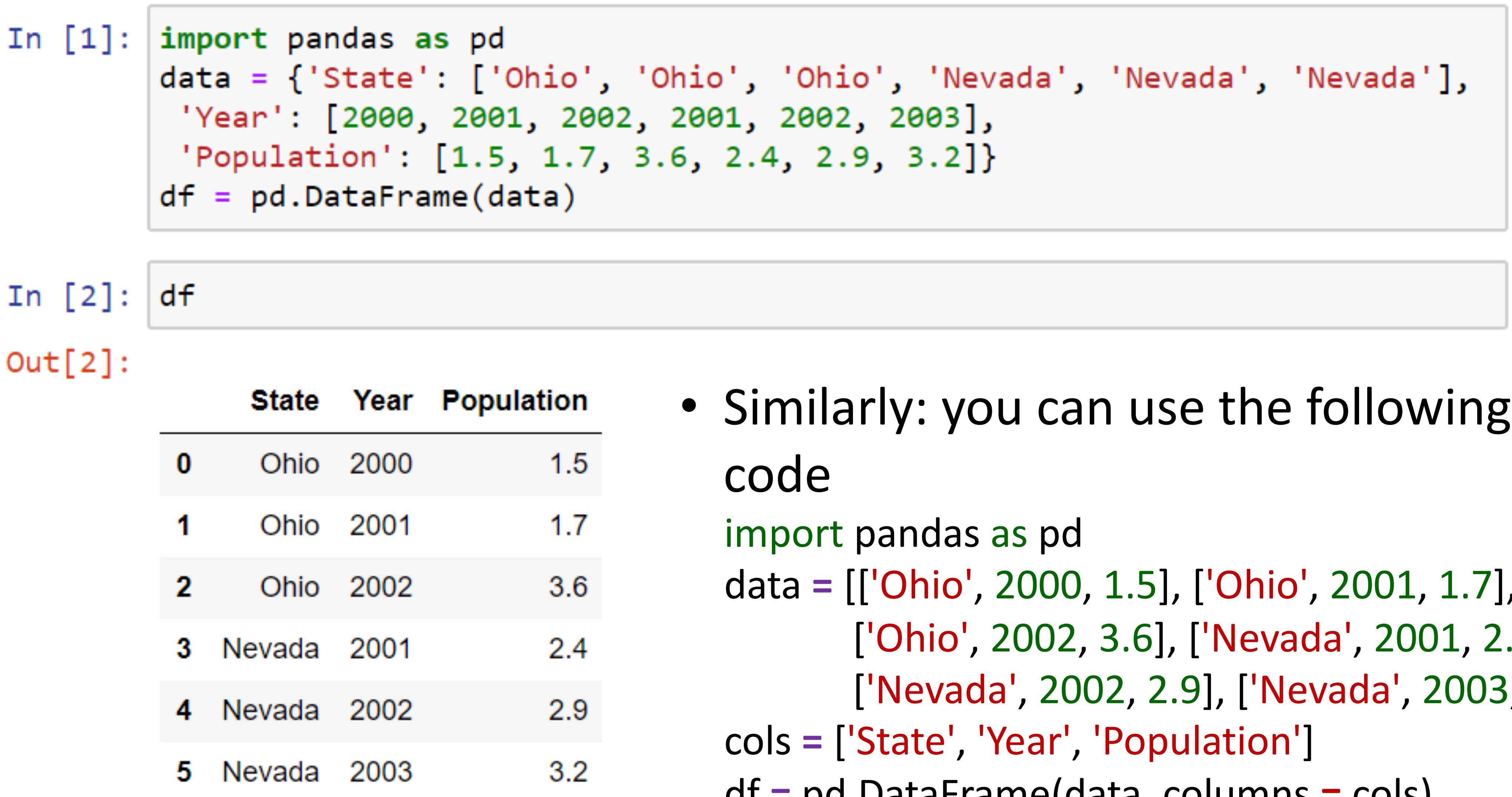
Series of data

Series of data

…

Series of data

Creating DataFrame



```
In [1]: import pandas as pd
        data = {'State': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada', 'Nevada'],
         'Year': [2000, 2001, 2002, 2001, 2002, 2003],
         'Population': [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
        df = pd.DataFrame(data)
```

```
In [2]: df
```

```
Out[2]:
```

	State	Year	Population
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9
5	Nevada	2003	3.2

- Similarly: you can use the following code

```
import pandas as pd
data = [['Ohio', 2000, 1.5], ['Ohio', 2001, 1.7],
        ['Ohio', 2002, 3.6], ['Nevada', 2001, 2.4],
        ['Nevada', 2002, 2.9], ['Nevada', 2003, 3.2]]
cols = ['State', 'Year', 'Population']
df = pd.DataFrame(data, columns = cols)
```

Load DataFrame from CSV Files

- The simplest way is:

```
df = pd.read_csv('file.csv’)     # often works
```

- More options can be added when loading a csv file into a dataframe

```
df = pd.read_csv('movies.csv', header=0,
    index_col=0, quotechar='"',sep=",",
    na_values = ['na', '-', '.', ‘’])
```

- More options can be found in Pandas documentation
- Remeber to import the pandas library as pd

Load DataFrame from EXCEL Files

- Each Excel sheet in a Pandas dataframe

```
workbook = pd.ExcelFile('movies.xlsx’)

df = workbook.parse(workbook.sheet_names[0])
```

- The parse() method takes many arguments like read_csv().
- Refer to the pandas documentation

Working with Dataframes

- Consider the movies dataset extracted from imdb dataset
- Start by reading the csv file

```
df = pd.read_csv(filepath_or_buffer = 'movies.csv', delimiter=',',
        doublequote=True, quotechar='"',na_values = ['na', '-', '.', ''],
        quoting=csv.QUOTE_ALL, encoding = "ISO-8859-1")
```

- Extract sub-table of the dataframe

```
df.info()                    # index & data types
n = 4
dfh = df.head(n)             # get first n rows
dft = df.tail(n)             # get last n rows
dfs = df.describe()          # summary stats cols
top_left_corner_df = df.iloc[:5, :5]
```

Extracting Data from Dataframes

- Extarct row number 0

```
row1 = df.iloc[0,:]     #You may ignore adding the :
row1 = df.iloc[0]
```

- Extract the column with the names of directors

```
df.director_name     #  OR
df["director_name"]
```

Extracting Data from Dataframes (Cont.)

- Extract set of rows

```
Rows_set1 = df.iloc[[5:10], ]         # Extracts rows 5,6,7,8, and 9
Rows_set2 = df.iloc[[5,6,8,10], ]     # Extracts rows 5,6,8, and 10
```

- Extract set of columns

```
cols_set1 = df[df.columns[5:10]][:]          # Extracts columns 5,6,7,8, and 9
cols_set2 = df[df.columns[[5,7,9]]][:]     # Extracts columns 5,7, and 9
col_set3 = df[['actor_3_facebook_likes', 'actor_1_facebook_likes', 'content_rating']]
```

- Note that: df.columns is a vector that contains the attributes’ names

Extracting Data from Dataframes (Cont.)

- Extract set of rows with a condition

```
df.loc[df['content_rating'] == 'PG-13', ['actor_1_facebook_likes',
             'actor_3_facebook_likes', 'budget']]
```

- You can do the same thing using iloc

```
df.iloc[(df['content_rating'] == 'PG-13').values, [1, 3]]
```

- Note that: iloc requires numerical values for the indexes

Profiling the Dataframes

- Display number of columns

```
print(len(df.columns))
```

- Display number of rows

```
print(len(df))      # OR print(len(df[df.columns[0]])
```

- Find the number of non-null values in each column (attribute)

```
df.count()
```

Profiling the Dataframes

- Display number of distinct values in an attribute

```
for col in  df.columns:
    print(col, ' has (', len(df[col].unique()), ') unique values')
```

- Display the data type of each attribute

```
dataTypeSeries = df.dtypes
for col_idx in range(len(df.columns)):
            print(df.columns[col_idx], 'has type (', dataTypeSeries[col_idx], ')')
```

Profiling the Dataframes – Computing Statistical Quantities

- Find max, min, and average of numerical attributes

```
for col in df.columns:
    if (not (df.dtype[col] == 'object')):
       print(col, 'has Min = ', df[col].min(),
           'Max = ', df[col].max(),
           'Average = ', df[col].mean())
```

- If the number of digits after the decimal point is large, use ‘round(n)’

Coffee Break

Introduction to Python

Part III: Visualization

ENRICHMENT FOR ALL

Utrecht University

Python – Matplotlib – Scatter Plot

- Emulates MATLAB

```
import matplotlib.pyplot as plt


xs = df.num_voted_users
ys = df.cast_total_facebook_likes
plt.scatter(xs, ys)
plt.show()
```

Need to install matplotlib

Scatter plot

Python – Matplotlib – Line Plot

```
import matplotlib.pyplot as plt

xs = [1,2,3,4,5]
ys = [x**2 for x in xs]

plt.plot(xs, ys)            # OR
plt.plot(xs, ys, linewidth = 5, color = 'r’)
plt.show()
```

Python – Matplotlib – Bar Plot

```
import matplotlib.pyplot as plt

xs = [1, 2, 3, 4, 5]
ys = [3, 2, 4, 2, 8]
colors = ['b', 'k', 'r', 'g', 'c']
plt.bar(xs, ys, color = colors, edgecolor = ”black")
plt.savefig('barPlot.pdf', bbox_inches = 'tight')
plt.show()
```

Python – Matplotlib – Pie Chart

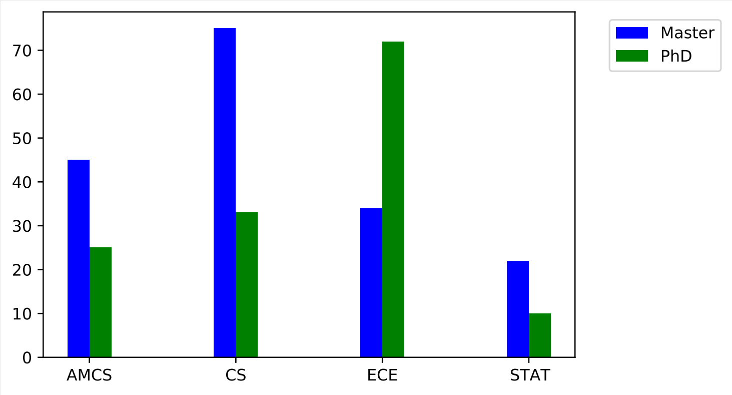
```
import matplotlib.pyplot as plt

xs = ['AMCS', 'CS', 'EE', 'B', 'CBRC']
ys = [10, 20, 50, 15, 5]
plt.pie(ys, labels = xs, autopct='%1.1f%%')
plt.savefig('pieChart.pdf', bbox_inches = 'tight')
plt.show()
```

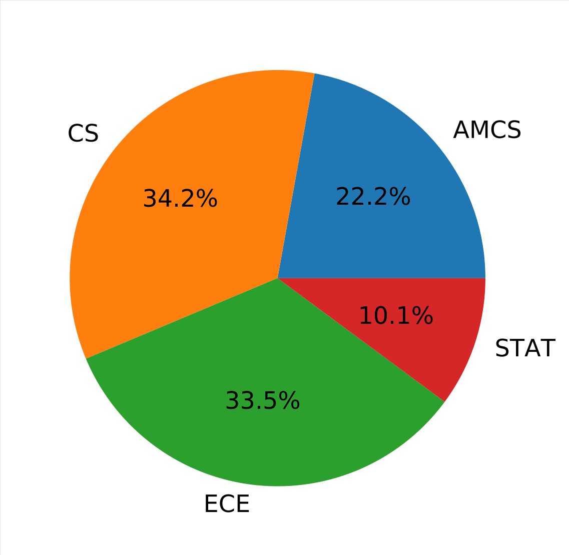
Python – Exercise

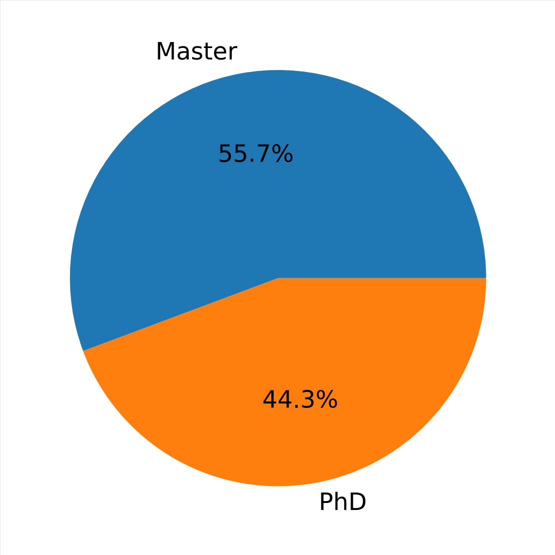
- Consider the following data for the number of students in different programs

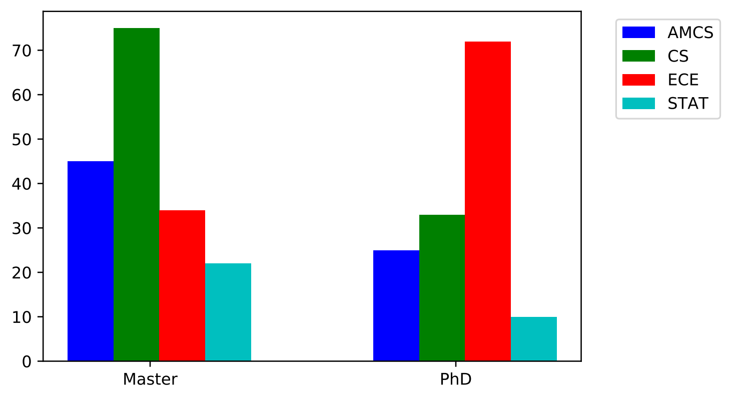
program	AMCS	CS	ECE	STAT
Master	45	75	34	22
PhD	25	33	72	10

- Draw the data as bar plot and pie chart









Python – Exercise – Examples of Figures

Wrap-Up

- Summerize what you learned today in 2-minutes

ENRICHMENT
FOR ALL

Thank You

Utrecht University

Extra

For interested students

Set Operations on Dataframes

- Assume the following dataframes

```
dd1 = pd.DataFrame( { 'id': ['1', '2', '3', '4', '5'], 'Feature1': ['A', 'C', 'E', 'G', 'I’],
          'Feature2': ['B', 'D', 'F', 'H', 'J’]})
```

```
dd2 = pd.DataFrame( { 'id': ['1', '2', '6', '7', '8'], 'Feature1': [‘A', ‘C', 'O', 'Q', 'S’],
          'Feature2': [‘B', ‘D', 'P', 'R', 'T’]})
```

- The *concat* function concatenates the dataframes allowing repetition

```
union_df = pd.concat([dd1, dd2])                    # concatenate row-wise (default)
union_df = pd.concat([dd1, dd2], axis = 1)          # concatenate column-wise
```

Join Operation on Dataframes

- The *merge* function joins dataframes on selected attribute

```
df_merge_col = pd.merge(dd1, dd2, on='id')
```

- If the joining attribute has different names in both dataframes

```
df_merge_col = pd.merge(dd1, dd2, left_on='att_dd1’, right_on = 'att_dd2')
```