LECTURER: TAI LE QUY

Introduction to Programming with Python

INTRODUCTION TO PROGRAMMING WITH PYTHON
TOPIC OUTLINE

Modules and Packages

6

UNIT 6

Modules and packages

STUDY GOALS

- Understand Python namespaces and their usage
- Learn how to use Python documentation
- Learn about various important data science packages in Python

EXPLAIN SIMPLY

1. What are Python namespaces and why are they important?
2. What are the two ways of documenting Python codes?
3. What are the five major data science packages in Python?

Why modules?	A module is a self-contained set of codes that performs a specific set of tasks e.g., logging, statistical analysis, graphs/charts plotting etc. Modules are convenient in that they can be reused in multiple applications.

A module is a Python file with .py as file extension. To use a module in an application, one simply imports the module:

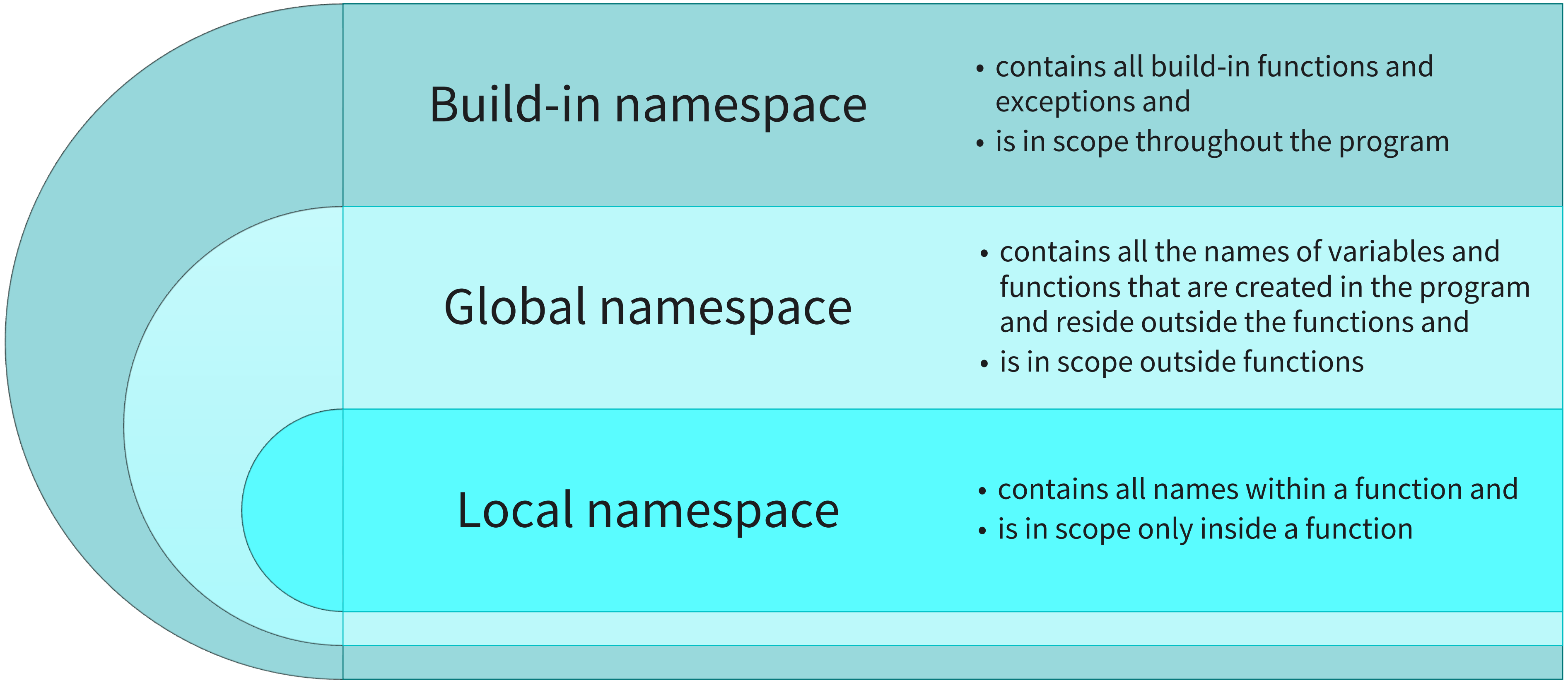
```
import <file name without the .py extension>
```

Or to use a particular function within a module, one uses the following syntax:

```
from <file name without the .py extension> import <function name>
```

Types of namespaces

Namespace is simply a collection of names. There are three namespaces in Python: build-in, global and local namespaces.



Build-in namespace

- contains all build-in functions and exceptions and
- is in scope throughout the program

Global namespace

- contains all the names of variables and functions that are created in the program and reside outside the functions and
- is in scope outside functions

Local namespace

- contains all names within a function and
- is in scope only inside a function

Example of namespaces

Note that each module has a self-contained namespace i.e., two modules can use the same variable names without causing name clash.

Example of namespaces

```
def say_hello(_name):
   print('Hello',_name)

name = input('Type your name: ')
say_hello(name)
print('Goodbye', name)
```

`print`: build-in namespace
`_name`: local namespace

`name`: global namespace
`say_hello`: global namespace
`print`: build-in namespace

DOCUMENTATION

Comments and docstrings

It is a good practice to leave comments in the program explaining what various parts of the code do. There are two ways to do this: using comments or docstrings.

Example:

```
def say_hello(_name):
    """
    Function to print greeting message
    """
    print('Hello’,_name)

# Prompt the user for input
name = input('Type your name: ')
say_hello(name)
print('Goodbye', name)
```

The text in between the two triple double-quotes """ is a docstring.

The text to the right of the # symbol is a comment.

Advantage of doctrings	Although docstrings and comments seem to be doing the same thing, the advantage of using docstrings is that docstrings are tied into the Python documentation help library and thus can be retrieved outside the program. So, while the user may not have access to the code in a module, it is possible to see what a particular function within the module does, if the module uses docstrings.

Example:

To retrieve the docstring of `say_hello` function (see preceding slide), one uses `help`:

```
>>> help(say_hello)
Help on function say_hello in module __main__:

say_hello(_name)
    Function to print greeting message
```

NumPy

NumPy is a scientific computing package in Python. It provides fast array and matrix operations as well as basic linear algebra, statistical and random sampling functions.

Installation:

`$ conda install numpy` (or `$ pip install numpy`)

Example of usage: Using numpy to calculate the mean.

```
import numpy
# Create an array of temperatures
temepratures = numpy.array([10, 19, 18, 16, 15, 10, 5, 3])
# Calculate the mean (average) temperature
mean_temp = temperatures.mean()
```

Matplotlib	Matplotlib is a library that provides a wide range of flexible, customizable, and easy-to-use plotting functions for displaying data. It includes histograms, bar charts, scatterplots, and more.

Installation:

`$ conda install matplotlib` (or `$ pip install matplotlib`)

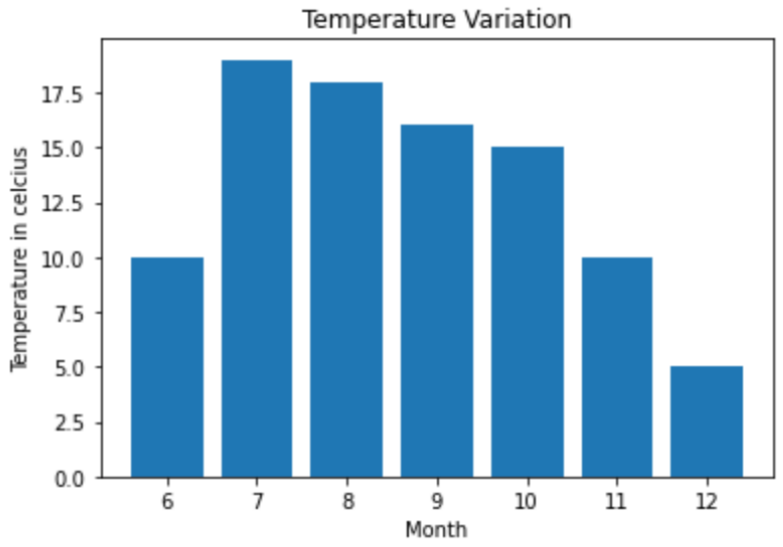
Example of usage: Using matplotlib to draw a histogram.

```
import numpy
import matplotlib.pyplot as plt
# Create an array of temperatures
temperatures = numpy.array([10, 19, 18, 16, 15, 10, 5])
months = numpy.array([6, 7, 8, 9, 10, 11, 12])
```

Matplotlib (cont.)

Example of usage (cont.):

```
# Plot the temperature histogram
fig, ax = plt.subplots()
ax.set_xlabel('Month')
ax.set_ylabel('Temperature in celcius’)
ax.set_title('Temperature Variation')
ax.bar(months, temperatures)
```



SciPy	The SciPy package provides most (but not all) of NumPy's functions in its own namespace. It also contains more advanced scientific computation including calculus, differential equations, signal processing to name but a few.
	
	Installation:
	
	`$ conda install scipy` (or `$ pip install scipy`)
	
	Example of usage: Using scipy to fit a curve.
	
	`import numpy`
	`from scipy.optimize import curve_fit`
	`import matplotlib.pyplot as plt`
	`# Create an array of temperatures`
	`temperatures = numpy.array([10, 19, 18, 16, 15, 10, 5, 3])`
	`months = numpy.array([6, 7, 8, 9, 10, 11, 12])`

SciPy (cont.)

Example of usage (cont.):

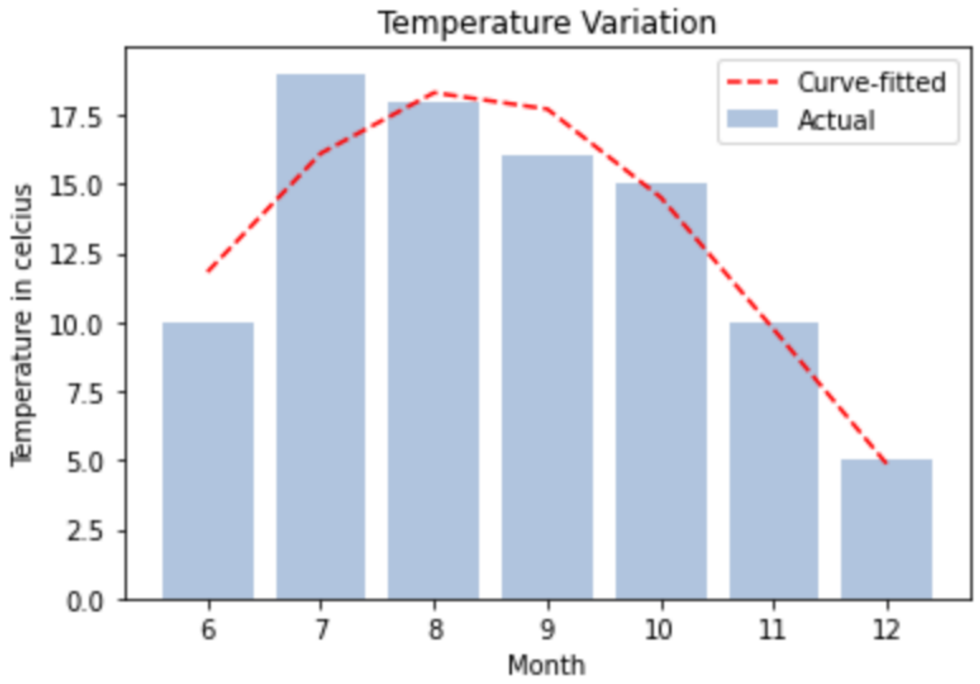
```
# Define the objective function
def objective(x, a, b, c):
    return a*numpy.sin(b*x) + c

# Curve fit - fit the temperatures with the objective function
popt, pcov = curve_fit(objective, months, temperatures)
a, b, c = popt
# Plot the histogram of temperatures
fig, ax = plt.subplots()
ax.set_xlabel('Month')
ax.set_ylabel('Temperature in celcius')
ax.set_title('Temperature Variation')
ax.bar(months, temperatures, color='lightsteelblue', label='Actual')
```

SciPy (cont.)

Example of usage (cont.):

```
# Plot the curve-fitted result
y_line = objective(months, a, b, c)
ax.plot(months, y_line, '--', color='red', label='Curve-fitted')
ax.legend()
```



Pandas	Pandas provides a variety of data structures for use in scientific applications. It reads csv files into DataFrame structures which can then be efficiently processed in Pandas.

Installation:

`$ conda install pandas` (or `$ pip install pandas`)

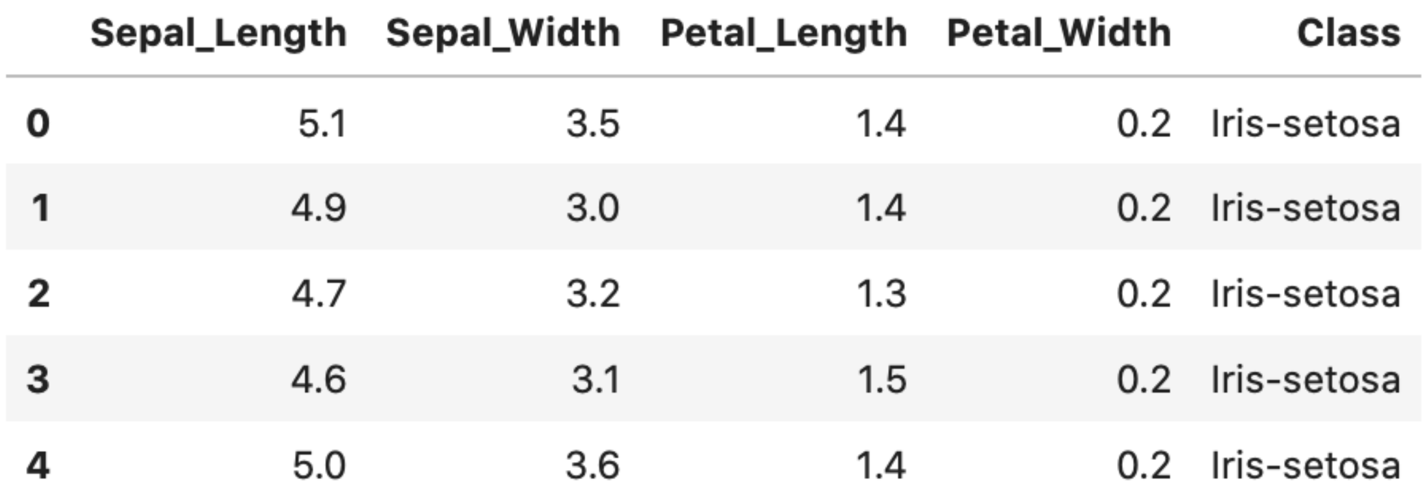
Example of usage: Using pandas to read a csv file.

```
import pandas
# Read csv from URL
csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data’
```

Pandas (cont.)

Example of usage (cont.):

```
# Set the column names
col_names =
['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Class']
iris_dataframe = pandas.read_csv(csv_url, names=col_names)
iris_dataframe.head()
```



	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Scikit-learn	Scikit-learn is a machine learning package. It includes regression (linear, logistic, Poisson), classification, clustering, dimensionality-reduction, feature extraction etc.

Installation:

`$ conda install scikit-learn` (or `$ pip install -U scikit-learn`)

Example of usage: Using scikit-learn to do classification.

```
import pandas
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

Scikit-learn (cont.)

Example of usage (cont.):

```
# Read csv from URL
csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data'
col_names =
['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Class']
iris_dataframe = pandas.read_csv(csv_url, names=col_names)
# X = Measurements, Y = Classification
X = iris_dataframe.iloc[:,:-1]
Y = iris_dataframe.iloc[:,-1]
# Converting into numpy array with sepal length and sepal width
X = X.to_numpy()[:,(0,1)]
Y = Y.replace({'Iris-setosa':0, 'Iris-versicolor':1, 'Iris-
virginica':2})
```

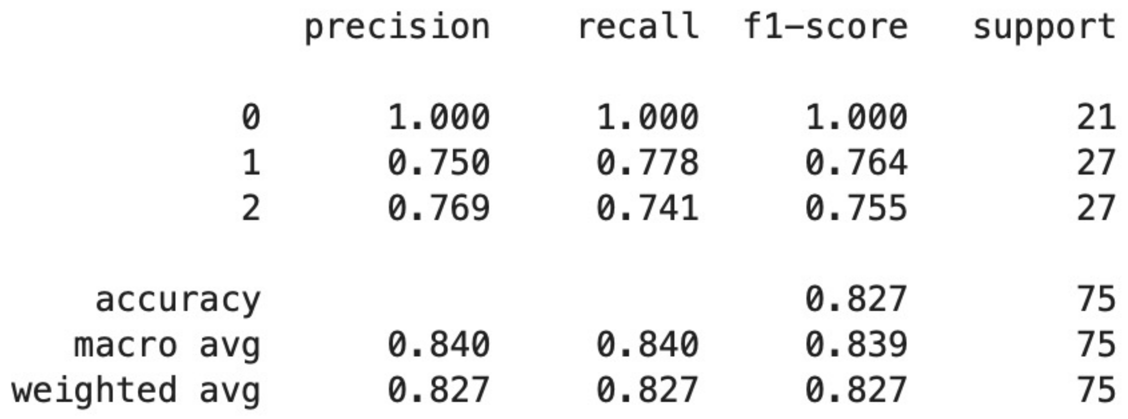
Scikit-learn (cont.)

Example of usage (cont.):

```
# Splitting into train and test sets
X_train, X_test, Y_train, Y_test =
train_test_split(X,Y,test_size=0.5, random_state=42)
# Run logistic regression
log_reg = LogisticRegression()
log_reg.fit(X,Y)
# Testing the logistic regression model
training_prediction = log_reg.predict(X_train)
test_prediction = log_reg.predict(X_test)
print(metrics.classification_report(y_train, training_prediction,
digits=3))
```

Scikit-learn (cont.)

Example of usage (cont.):



```
              precision    recall  f1-score   support

           0      1.000     1.000     1.000        21
           1      0.750     0.778     0.764        27
           2      0.769     0.741     0.755        27

    accuracy                          0.827        75
   macro avg      0.840     0.840     0.839        75
weighted avg      0.827     0.827     0.827        75
```

- Understand Python namespaces and their usage
- Learn how to use Python documentation
- Learn about various important data science packages in Python

SESSION 6

TRANSFER TASK

Use matplotlib to plot the iris dataset

`https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data`

Two scatter plots are required: one with sepal length vs sepal width and the other with petal length vs petal width. The plots should contain a title, axis labels and a legend (with blue for Setosa, green for Versicolor and red for Virginia).

TRANSFER TASKS
PRESENTATION OF THE RESULTS

Please present your results.
The results will be discussed in plenary.

1. Which of the following will import a specific function, add_numbers, from a library, adding_library?
   a. `import adding_library`
   b. `import add_numers from adding_library`
   c. `import adding_library.adding_numbers`
   d. `from adding_library import add_numbers`

LEARNING CONTROL QUESTIONS

2. Python's `int( )` function will take an argument and convert it to an integer. In which namespace does `int( )` exist?

3. Which best describes the difference between a comment and a docstring in Python?

How did you like the course?

HOW DID YOU LIKE THE COURSE?

😞 😐 🤩

LIST OF SOURCES

Lutz, M. (2013). Introducing Python Object Types. *Learning Python* (5th ed.). O’Reilly.