MONASH University

MONASH
INFORMATION
TECHNOLOGY

FIT9133 Semester 2 2019
Programming Foundations in Python

Week 9:
Python Standard Library and External Packages

Chunyang Chen
Gavin Kroeger

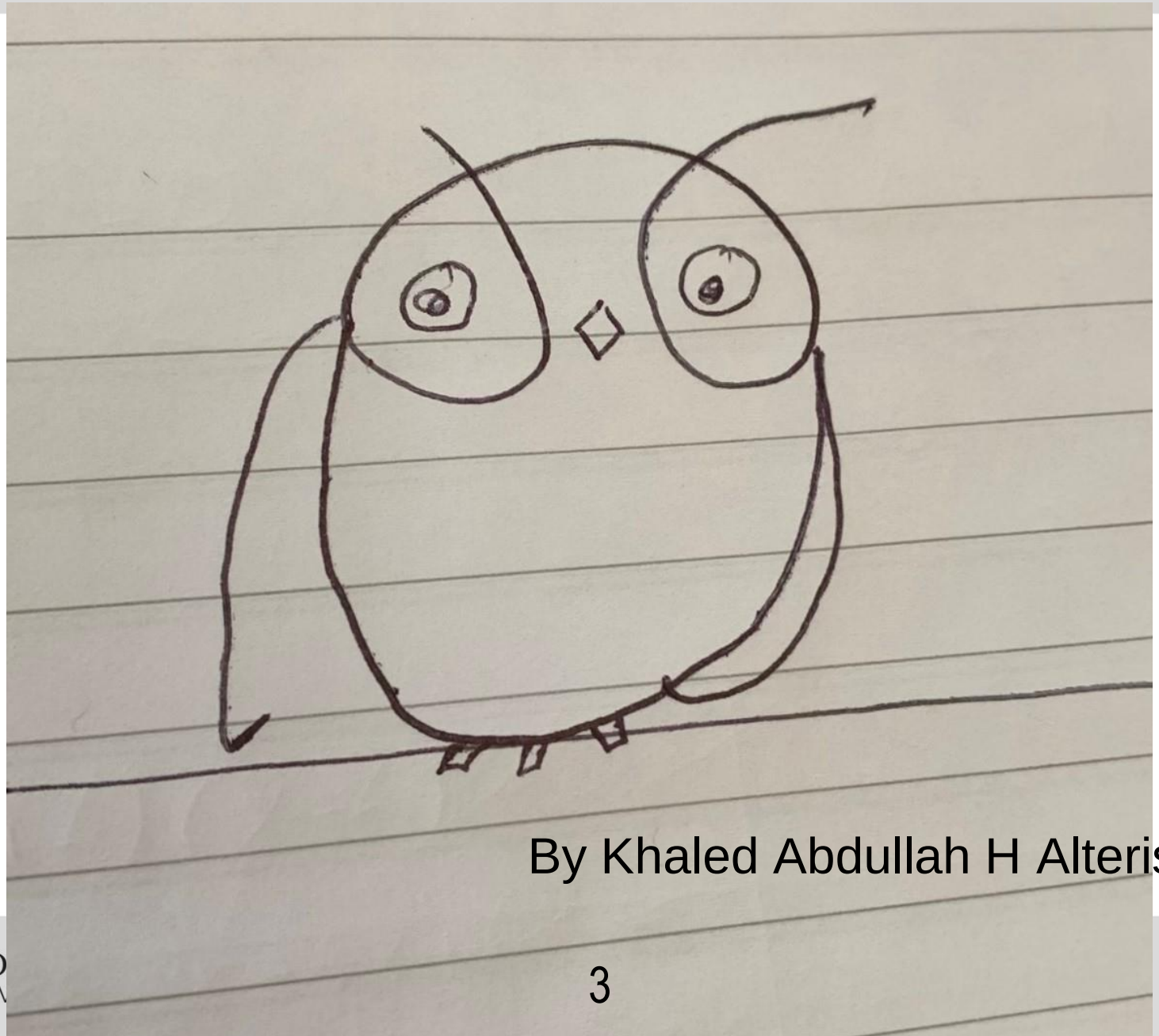
GROUP OF EIGHT AUSTRALIA

Update

- Assignment 2:
  - Have released and due date is 18th October, 2019, **11:55pm Sunday**.

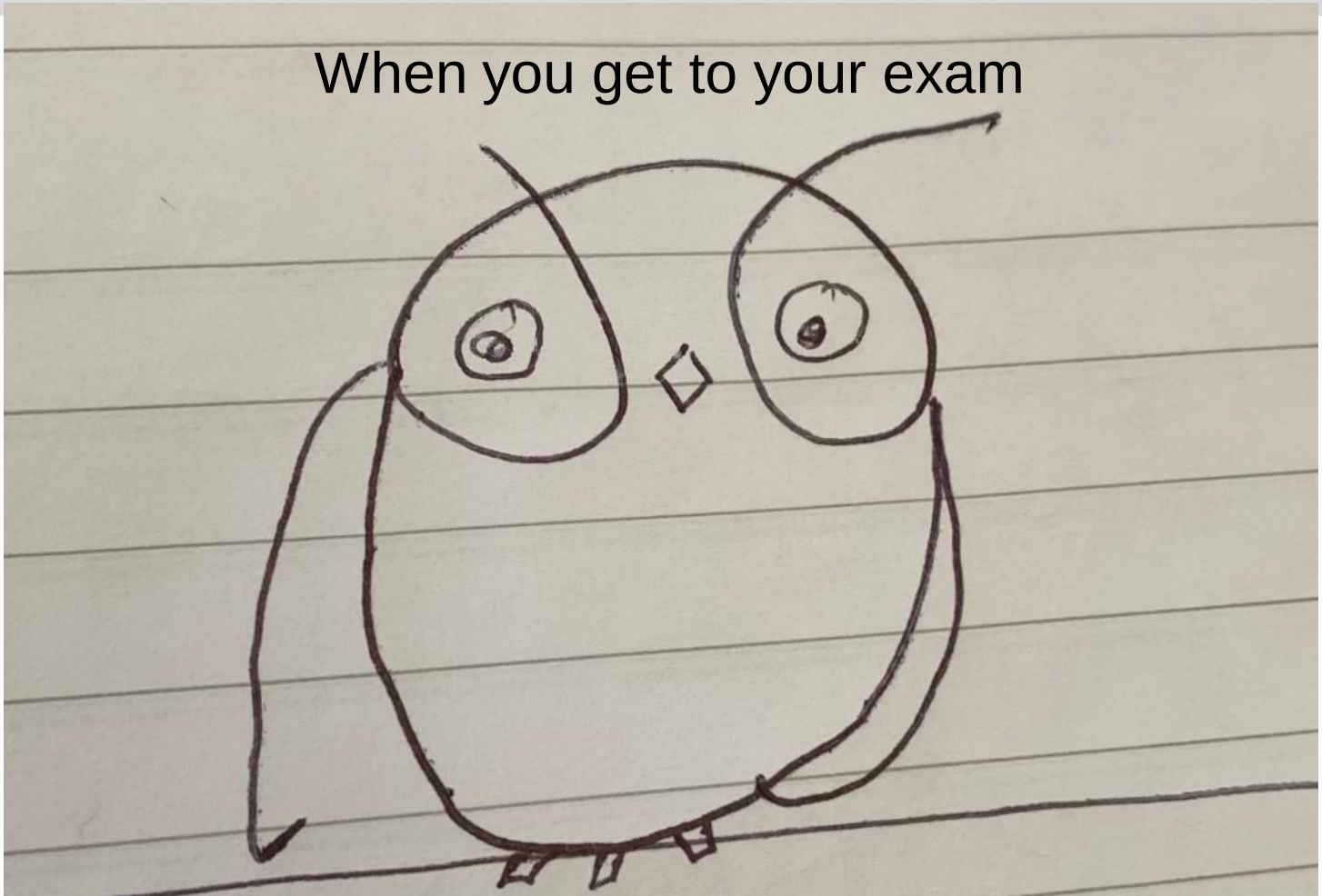
- Assignment 1:
  - We are going to release the results to you.
  - There are some problems with the marks. When its finished I will release.
  - If you do not receive it, there are two reasons:
    - You do not attend the interview
    - You are suspected for violating the academic integreity

- The best owl has been selected
  - That's right, it was a competition.



By Khaled Abdullah H Alterish

When you get to your exam



And you can’t remember your own name

Module 4 Synopsis

- Module 4 is aimed to introduce you with:
  - Python library and packages
    - Standard packages: Math and Random
    - External packages: NumPy, SciPy, Matplotlib, Pandas
  - Searching algorithms
    - Linear Search
    - Binary Search
  - Sorting algorithms
    - Bubble Sort
    - Selection Sort
    - Insertion Sort

Module 4 Learning Objectives

- Upon completing this module, you should be able to:
  - Utilise a number of useful Python packages for scientific computation and basic data analysis
  - Recognise a suitable algorithm for solving a particular computational problem
  - Contrast different algorithms for searching and sorting

Python Standard Library: Math and Random

Math (`math` module)

Almost all the return values are of type `float`

- Mathematical functions:
  - Power and logarithmic: `exp(x)`, `log(x,base)`, `pow(x,y)`, …
  - Trigonometric: `sin(x)`, `cos(x)`, `tan(x)`, …
  - Hyperbolic: `sinh(x)`, `cosh(x)`, `tanh(x)`, …
- Mathematical constants:
  - `math.pi` (3.141592…)
  - `math.e` (2.718281…)
  - `math.tau` (6.283185…)
- Python documentation on `math`:
  - https://docs.python.org/3/library/math.html

Random (`random` module)

- Usage:
  - Generate pseudo-random numbers
- Basic function: `random()`
  - Return a random floating point number in the range `[0.0,1.0)`
- Random function for “integers”:
  - `randint(a,b)`: return a random integer in the range `[a,b]`

(More on) Random

- Random functions for “sequences”:
  - `choice(seq)`: return a random element from the sequence `seq`
  - `shuffle(seq)`: shuffle the sequence `seq` in place
  - `sample(seq,k)`: return a `k` length list of unique elements from the sequence `seq`

- Python documentation on `random`:
  - https://docs.python.org/3/library/random.html

Python Standard Library

- Standard libraries
  - Text processing services:
    - `string, re, difflib`
  - Functional programming modules
    - `itertools, operator`
  - File and directory access
    - `os.path, shutil`
  - Data compression and archiving
    - `zlib, gzip`
  - General operating system service
    - `os, io, time, logging`
  - …

- Python documentation on standard library:
  - https://docs.python.org/3/library/

Python External Packages: NumPy, SciPy, Matplotlib, Pandas

NumPy

- **`numpy`**:
  - Useful for scientific computing
  - Import statement: `import numpy as np`
  - "Basis" of many other scientific computing and data analysis packages (e.g. SciPy, Pandas)

- Data structure: n-dimensional (homogeneous) "arrays"
  - 1-dimensional array: `an_array = np.array([1,2,3])`
  - 2-dimensional array:

```
an_array = np.array([[1,2],[3,4]])
an_array = np.array([[1,2,3],[4,5,6],[7,8,9]])
an_array = np.arange(1,10).reshape(3,3)
```

- NumPy reference guide:
  - https://docs.scipy.org/doc/numpy/reference/

SciPy

- **`scipy`**:
  - Extension from NumPy
  - Provide a collection of algorithms and functions for scientific computing

- Sub-packages (different scientific domains):
  - Linear algebra
  - Integration and differentiation
  - Statistical distribution
  - Optimisation
  - Clustering
  - Image processing
  - Signal processing

- SciPy reference guide:
  - https://docs.scipy.org/doc/scipy/reference/

Matplotlib

- **`matplotlib`**:
  - 2D and 3D plotting library
  - Generate plots, histograms, bar charts, scatterplots, …
- Simple plotting module: **`pyplot`**
  - Import statement: `import matplotlib.pyplot as plt`
  - Basic commands:
    - `plt.plot()`, `plt.scatter()`, `plt.bar()`, `plt.axis()`, `plt.show()`, …
- Matplotlib documentation:
  - http://matplotlib.org/contents.html

Pandas

- **`pandas`**:
  - Useful for data structuring and data analysis
  - Import statement: **`import pandas as pd`**
  - (Often) import NumPy together: **`import numpy as np`**

- Data structure: "data frames" (tabular-like)
  - Create from a dictionary:

```
a_dataframe = pd.DataFrame(
                {'name':['Alice','Bob','Charles'],
                 'age':[25, 23, 34],
                 'gender':['female','male','male']})
```

  - Create from an NumPy array:

```
a_dataframe = pd.DataFrame(np.arange(16).reshape((4,4)),
                index = ['x1','x2','x3','x4'],
                columns = ['y1','y2','y3','y4'])
```

(More on) Pandas

- Pandas documentation:
  - http://pandas.pydata.org/pandas-docs/stable/
- 10 Minutes to Pandas:
  - http://pandas.pydata.org/pandas-docs/stable/10min.html
- Pandas Cheat Sheet:
  - https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf

MONASH University

Review Exercise:

Is the following a valid statement for creating a Numpy array?

```
>>> import numpy as np
>>> an_array = np.array([[1,2,3],
                         (4,5,6),
                         [7,8,9]])
```

A. Yes
B. No
C. Not sure

Given the NumPy matrices, which of the following will result in the product of the two matrices as given below?

```
>>> import numpy as np
>>> matrix_1 = np.array([[1,1],
                         [1,1]])
>>> matrix_2 = np.array([[0,1],
                         [2,3]])
>>> dot_matrix = ???
>>> print(dot_matrix)
>>> array([[1,1],
           [5,5]])
```

A. matrix_1.dot(matrix_2)
B. matrix_2.dot(matrix_1)
C. np.dot(matrix_1, matrix_2)
D. Not sure

What would be the output for the given program?

```
def is_equal(a, b):
        if a == b:
            return True
        else:
            return False

>>> from scipy import vectorize
>>> is_equal_vec = vectorize(is_equal)
>>> list_1 = [1,2,3]
>>> list_2 = [3,2,1]
>>> print(is_equal_vec(list_1,list_2))
```

A. Some error occurred
B. [True, False, True]
C. [False, True, False]
D. Not sure

Which of the following is a valid statement to produce the given data frame in Pandas?

```
>>> import pandas as pd
>>> nested_dict = ???
>>> df = pd.DataFrame(nested_dict)
```

	red_wine	white_wine
1998	1	3
1999	1	2
2000	2	0

A. {'white_wine': {1998:1, 1999:1, 2000:2},
'red_wine': {1998:3, 1999:2, 2000:0}}

B. {'red_wine': {1998:1, 1999:1, 2000:2},
'white_wine': {1998:3, 1999:2, 2000:0}}

C. {'white_wine': {1998:3, 1999:2, 2000:0},
'red_wine': {1998:1, 1999:1, 2000:2}}

D. Not sure

Review Question 5

Given the data frame in Question 4 (called `df`), which of the following can be applied to retrieve the content of the first row?

	red_wine	white_wine
1998	1	3
1999	1	2
2000	2	0

A. df[0]
B. df[0:]
C. df[0:1]
D. df.ix[1998]
E. Not sure

Review Question 6

Given the data frame in Question 4 (called `df`), which of the following can be applied to add a new column 'table_wine'?

	red_wine	white_wine	table_wine
1998	1	3	1
1999	1	2	1
2000	2	0	1

A. df['table_wine'] = 1,1,1
B. df['table_wine'] = [1,1,1]
C. df['table_wine'] = 1
D. All of the above