Deakin University
School of IT
Teaching Team: SIT742 Teaching Team

SIT742: Modern Data Science

1 Introduction of the Assignment

1.1 Assignment Questions

There are total **2** parts in the assignment. The first part will focus on the basic python programming which includes the `data types`, the `control flow`, the `function and Class`, the `modules and library` from the workshop in **M02**. The second part will focus on more advanced python skills on the usage of `function`. This part will include the learning in **M02** and also **M03**, particular `numpy`.

1.2 What to Submit

In general, you will be required to submit your **notebook** from `Google Colab` and clearly list the answer for each question (you need to save your results of running as well). The expected format from your notebook will be like 1

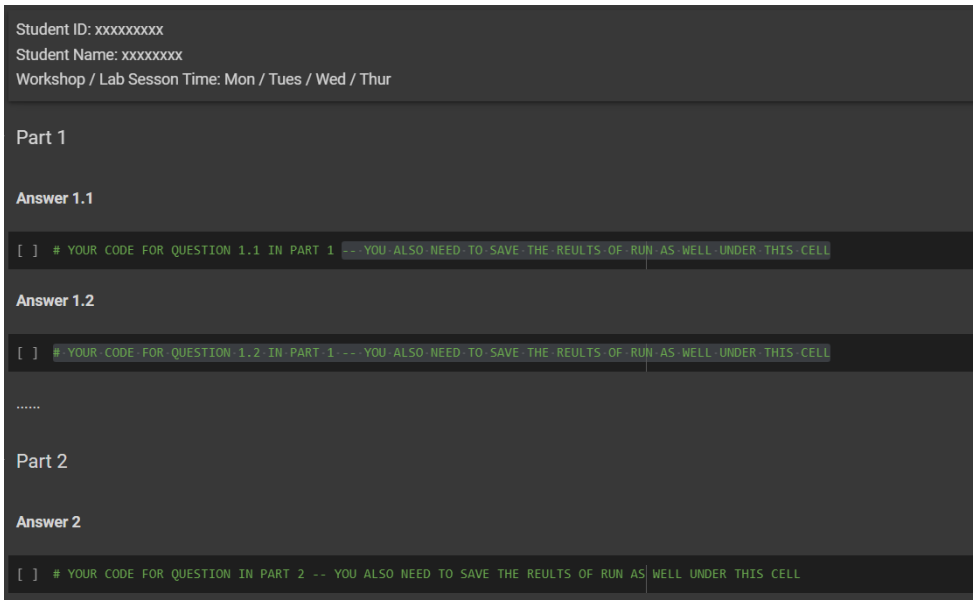

Figure 1: Notebook Format

Also, we would like you to make a **short video** from 5 to 10 mins to describe how you solve the questions on part 2 of the assignment (only if you would like to get **HD** for this assignment. If you did not submit the video, we will take penalty of your marks of part 2 by 50%) In the video, you could run the code you write line by line and detail the solutions you provide in the notebook. Part 2 of the assignment has two choices, for students who have the student ID on **odd** number, you will need to do the first choice, and for students who have the student ID on **even** number, you will need to do the second choice. If you did not choose the correct one for part 2, you will lose all the marks for part 2.

2 Part 1

There are **8** questions in this part for **80** marks and each of the question is **10** marks. You are required to use `Google Colab` to finish all the coding in the *code block cell* and also save the result of running as well. All assignment will be checked via `turnitin` and any plagiarism will be reported to unit chair and also school.

Question 1.1

ages = [5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]

- Could you find the median value of age (don't use numpy)
- Could you find the age which is larger than 90% of other ages (don't use numpy)

Question 1.2

Define a function `sum_test()`, the input n for `sum_test()` will calculate the $1+2+3+\cdots+n$ and print out the results when the $n = 12$

Question 1.3

You would like to design a score grade mechanism (control flow) which could allow you to:

- input a score to variable `score`
- return "F" when score < 60
- return "P" when score $>= 60$ and < 70
- return "C" when score $>= 70$ and score < 80
- return "D" when score $>= 80$ and score < 90
- return "HD" when *score* $>= 90$

You also need to judge whether the input for score is `int` type or not.

Question 1.4

you would use `while statement` to print out below * mark in **9** lines (must use `while statement`).

```
*
**
***
****
*****
****
***
**
*
```

```
num = 1
while num>0:
    print("*"*num)
    #continue to write your code in below#
```

Question 1.5

Given variable `test = "aAsmr3idd4bgs7Dlsf9eAF"`, find out all the numerical number in the string variable `test` and store those numerical number into another string variable `result`, then print `result`.

Question 1.6

Define a function `find_all`, the function could find the **first index** of substring `"hello"` in the input string `"helloworldhellopythonhelloc++hellojava"`, the return value of the function is a list of the **first index** , such as $[0, 10, 21, 29]$.

Question 1.7

Define a class `Person` with two variables on `name` and `age`. In the class `Person`, there are two methods:

- one is `Get_age()`,
- another one is `Set_age()`.

Therefore, we could achieve below:

```
daniel = Person('Daniel',50)
print(daniel)
#The result of above print should be: name: Daniel, age:50#
daniel.Set_age(60)
print(daniel.Get_age())
#The result of above print should be: 60#
print(daniel)
#The result of above print should be: name: Daniel, age:60#
```

Question 1.8

Given the array `nums` with integers, return all the possible permutations of the array. You can return the answer in any order by defining the function `permute`.

Details as below:
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]

Input: nums = [1]
Output: [[1]]
All the integers of `nums` are `unique`

3 Part 2

There are **2** questions in this part. You only need to choose one question to answer (based on your student ID, if **odd number** then first question, if **even number** then second question.) The part 2 will be **20** marks. You are required to use `Google Colab` to finish all the coding in the *code block cell* and also save the result of running as well. All assignment will be checked via `turnitin` and any plagiarism will be reported to unit chair and also school. Also, a short video is needed to explain your code (according to 1.2).

3.1 Check Student ID

In here, the code of checking your student ID is provided:

```
def check_studentid(x):
    if x % 2 == 0:
        print('%d is even' % x)
    else:
        print('%d is odd' % x)

check_studentid(#your student ID)
```

You need to copy this code to your notebook and run the function with your student ID. You will also need to print/save the result of the code running.

3.2 Odd number Question

Question 2 Find area – For odd number

First import below libraries in `Google Colab`:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Then run below code:

```
def f1(x):
    a = (-x**2)+2*x
    return a

fig, ax = plt.subplots()
ax.set_title('-x^2+2x')
ax.plot(np.linspace(0, 2, 100), f1(np.linspace(0, 2, 100)),
        label='-x^2+2x')
ax.fill_between(np.linspace(0, 2, 100), f1(np.linspace(0, 2, 100)))
ax.legend()
plt.show()
```

You will have a visualization as below Figure 2:

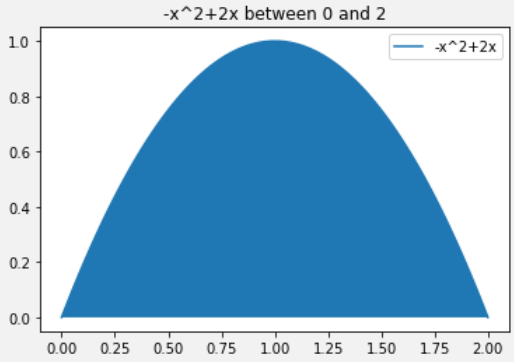

Figure 2: $-x^2 + 2x$ function from $x = 0$ to $x = 2$

Interpretation of the visualization:

- This is the $-x^2 + 2x$ function from defined `f1`,
- The x axis is limited for 100 points from 0 to 2
- The y axis is the output of `f1(x)` for each x value.
- Therefore, you have the plot for `f1(x)` function.

Question 2.1 What is the maximum value of `f1(x)` function? Can you use python programming to code a function `find_max(n)` to find the max value of `f1(x)`? you would like to try $n = 100000$ samples of the x value from -100 to 100 (use np.linspace(-100, 100, n)) and initialize the maximum of `f1(x)` with variable `max_value`, on each value from `f1(x)`, you would like to compare the current `f1(x)` with `max_value`, if the current is larger, then you will update the `max_value` with current `f1(x)`. You will run with all 100000 samples and round your result of `max_value` to integer.

Question 2.2

The problem is how to calculate the area of the `f1(x)` when x is from 0 to 2? There is one method to calculate the area of given shape – `monte carlo method` as below:

- You will need to obtain the `max_value` from the result of **Question 2.1**,
- You will need to sample points within the rectangle r.In the rectangle, the first point on bottom left is [0,0], second point on bottom right is [2,0], the third point on top left is [0, `max_value`], the fourth point on top right is [2, `max_value`],
- You need to find how many sampled points are within the area of `f1(x)` where x is from 0 to 2,
- You need to use the area of r multiply the ratio of points in area of f1(x),
- Then the area of `f1(x)` could be calculated.

You are required to define the function `find_area(sample_num,max_value)` for this problem, and you will need to run the `find_area(sample_num=100000,max_value)` and print / save the results.

3.3 Even number Question

Question 2 Hill climb on linear regression – For even number

First import below libraries in `Google Colab`:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Then run below code:

```
X = np.linspace(0, 3, 30)
Y = 2.5 * np.linspace(0, 3, 30) + 5 * np.random.rand(30)
fig, ax = plt.subplots()
ax.set_title('Regression line to fit')
ax.set_ylabel('Y')
ax.set_xlabel('X')
ax.plot(X, Y, label='line to fit')
ax.legend()
```

```
plt.show()
```

You will have a visualization as below Figure 3:

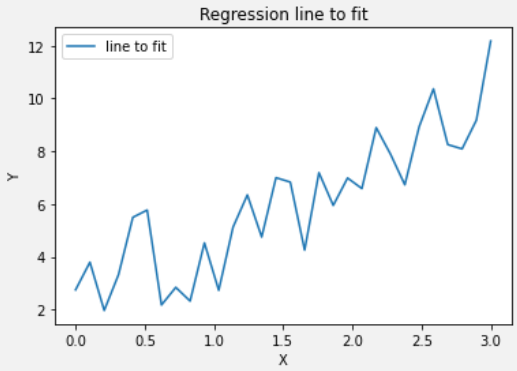

Figure 3: Regression line to fit

You will need to fit a linear regression to this line plot and the regression equation is like $y_{fit} = a * X$ To fit the line, you will need to find the optimal a here and you could follow `hill climb` as below steps:

- defining the total rounds of run n,randomly giving value to a in first round,
- calculating the error $(y_{fit} - Y)$,
- adjusting the a for little as a_{adjust}, and get new $y_{fit} = a_{adjust} * X$,
- calculating the $error_{new}$ $(y_{fit} - Y)$,
- if the $error_{new} < error$, then $a = a_{adjust}$ and $error = error_{new}$.
- Finishing all n rounds

Question 2.1

You will need to define function `find_error(a,X,Y)` for `root mean square error` as Equation 1 (you can only use numpy to do this question)

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(y_{fit}^{i} - Y^{i})^2} \tag{1}$$

where $y_{fit} = a * X$

Question 2.2

You are required to define the `hill climb` function `fit_regression(n,X,Y)` to find optimal a. You will also need to run below code after you have found the optimal a:

```
fig, ax = plt.subplots()
ax.set_title('Regression line to fit')
ax.set_ylabel('Y')
ax.set_xlabel('X')
ax.plot(X, Y, label='line to fit')
ax.plot(X, a * X, label='fitted line')
ax.legend()
plt.show()
```