Deakin University
School of IT
Unit Team: SIT742

Trimester 2, 2022
Assessment Task 1

SIT742: Modern Data Science

Extension Request Students with difficulty in meeting the deadline for proper reasons such as illness, etc. must apply for an assignment extension with supporting evidence, no later than 8:00pm on 10/08/2021 (Wednesday). Apply via '**CloudDeakin**', the menu item '**Extension Request**' under the '**Assessment**' drop-down menu.

Academic Integrity All assignment will be checked for plagiarism, and any academic misconduct will be reported to unit chair and university.

Instructions

Assessment Task 1 Questions

There are total **2** parts in the assessment task:

Part 1 The first part will focus on the basic Python programming skills which includes the `data types`, the `control flow`, the `function and Class`, the `modules and library` from **M02**.

Part 2 The second part is for those who are aiming to achieve '`High Distinction`' (HD) for this assessment task, and it will focus on more advanced Python programming skills for data science. This part will require the knowledge covered in **M02** and also **M03**, in particular, `numpy` and beyond.

What to Submit?

You are required to submit the following completed files to the corresponding *Assignment* (Dropbox) in CloudDeakin:

`SIT742Task1.ipynb` The completed notebook with all the run-able code on all requirements.

In general, you need to complete, save the results of running, download and submit your **notebook** from Python platform such as `Google Colab`. You need to clearly list the answer for each question, and the expected format from your notebook will be like in Figure 1.

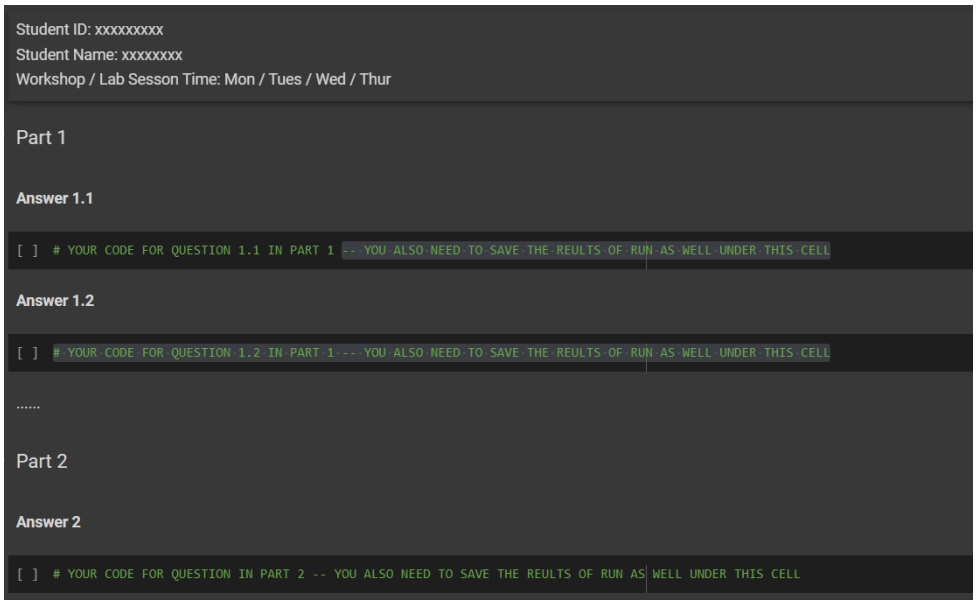

Figure 1: Notebook Format

`SIT742Task1Part2.avi` If you are aiming to achieve '`High Distinction`' (HD) and choose to work on *Part 2* of this assessment task, one important submission is a **short video**, in which *You* are required to orally present the solutions that you provide in your submitted notebook and to illustrate the running of code line by line.

The length of video demonstration should be between 5 and 10 minutes, and the file format can be common ones, such as '`MKV`', '`WMV`', '`MOV`' etc.

Part I

Python Programming

There are **8** questions in this part for **80** marks, and each question is **10** marks.

You are required to use `Google Colab` to finish all the coding in the *code block cell*, provide sufficient coding comments, and also save the result of running as well.

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]

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

Question 1.2

- define a function `sum_test()`, which is summation of a sequence numbers. It takes the argument n. For example, when $n = 5$, the result should be $1 + 2 + 3 + 4 + 5 = 15$;
- call the function and print out the results when the $n = 12$.

Question 1.3

You are required to design the code (use while statement) could achieve the 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* $>= 90$

In the code, you also need to judge whether the input for score is `int` type or not.

```
while True:
    try:
            score = int(input("please_fill_your_score:"))
        #continue to write your code in below#
```

Question 1.4

You are required to use `while statement` to print out below ***** mark in **9** lines (must use `while statement`).

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

```
num = 1
while num>0:
    # print * mark if not above 5
    print("*"*num)
    #continue to write your code in below#
```

Question 1.5

Given a string variable `test`, for example, `test = "aAsmr3idd4bgs7Dlsf9eAF"`, using code with `for` statement to find out all the digits (for example, here it is "3479") in the string variable `test`, and store those digits (here it is "3479") into another string variable `result`, then print `result`.

You can implement it as a function.

Question 1.6

Define a function `find_all`, which could find the **first index** of substring `"hello"` in the input string such as `"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 for you to implement:

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

So that your code can 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 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]]

You code should also check and ensure that all the integers in `nums` are `unique`.

Part II

Python Foundations for Data Science

This part is for students who are aiming to achieve '`High Distinction`' (HD) for this assessment task.

There are **2** versions of Question 2 in this part for **20** marks: **10** marks for coding, and **10** marks for video presentation `SIT742Task1Part2.avi` (as in '**What to Submit**'). You should only work on one version based on your own enrollment details as in below Section 1, and working on the wrong one will result in zero for this question.

For your question, you are required to use `Google Colab` to finish all the coding in the *code block cell*, provide sufficient coding comments, and also save the result of running.

1 Which version of Question 2 for you?

The code of determining your Q2 version is provided:

```
def sum_digits(n):
    r = 0
    while n:
        r, n = r + n % 10, n // 10
    return r

def check_studentid(studentid):
    x = sum_digits(studentid)
    if x % 2 == 0:
       print('version I')
    else:
       print('version II')

check_studentid(9876543210)
#replace the value by 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.

2 Question 2 (Version-I)

Find the area

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 in Figure 2.

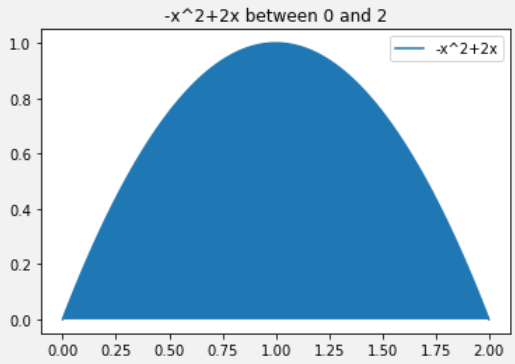

Figure 2: $-x^2 + 2x$ function from $x = 0$ to $x = 2$

Interpretation of the visualization:

- This is function $-x^2 + 2x$ that defines `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?
- You are required to implement a function `find_max(n)` to find the max value of `f1(x)`?

  One possible solution you may 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 Question 2 (Version-II)

Hill climb on linear regression

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 in Figure 3:

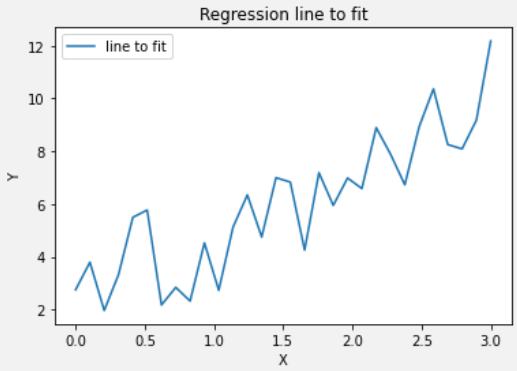

Figure 3: Regression line to fit

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 in 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()
```