Instructions

This assignment is composed of 10 “questions”.

There are two types of questions:

- Type 1 questions for which you have to write a script.
- Type 2 questions for which you have to comment or modify a given script.

For both types of questions, your answer will be a python file:

- Created by you for type 1
- Modified by you for type 2

There are 5 type 1 questions and 5 type 2 questions.

The type 2 questions are all about the same script.

You are expected to use the default packages coming with python 3. It should not be necessary to install an additional package to run your script.

Criteria:

Type 1:

- Questions will describe what the script should do
- Questions will have specific requirements on how to write the script.

You should write a script that behaves as expected and fulfills the requirements.
In addition, you should comment your script to explain your design choice.
(Point allocation: 50% behavior 30% requirements 20% comments)

Type 2:

- The script modification parts will be evaluated like type 1
  (Point allocation: 50% behavior 30% requirements 20% comments)

- The questions about script understanding will be evaluated according to how well you understood/explained the problem.

All the questions (Type 1 and Type 2) are graded on 10 points (10x10 = 100 points in all).

Some type 1 questions will have specific requirements on how to write the script. However, if you cannot write it according to the requirement but still produce a script that gives the expected output and explain it, you will get some points.

For example: Type 1 question

Write a script that displays the number from 0 to 5.
Requirements: use a for loop.

```
# I didn't know how to use a for loop
# I used print
# I know this will not be good for larger numbers
print(“0")
print(“1")
print(“2")
print(“3")
print(“4")
print(“5")
```

This script does not fulfill the requirement but it creates the desired output and some comments explain the situation. This will get up to 50% of the points (depending how much of the behavior is implemented).

If there are some parts you do not know how to do, you can leave part of the script unimplemented. This way you can still implement the other parts. Be sure to add some comments.

For example:

This part of script shows a dummy function in place of a function that was not implemented.

```
def compute_cosine(x):
    # I did not know how to implement this function
    # This is just a placeholder to be able to run the script
    return 0
```

If part of your script is not running, you can comment it out and add an explanation to show that you understand it is not correct.

```
# I get a SyntaxError from the next statement
# a{5} = 3
# I don’t know how to fix it so I commented it out.
```

How to submit your work:

You should group all your files together in a zip archive and submit it through PandA.

- Your are expected to submit at least one python file per question.
  The file name should be: id_<your student id>_q_<question number>.py

- You can submit a solution to a single question using multiple files. For example a main script that import functions from another script. If you choose to do so, you should document it in the main script. Your main script should have the expected name and the other scripts use that name as prefix:

  id_<your student id>_q_<question number>_<given name>.py

  For example:
  If your student id is 123456 then for question 5, the file name is: id_123456_q_5.py
  If you have another file it may be called: id_123456_q_5_tools.py

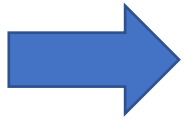
- For type 2 questions (part 2), the script id_none_part2.py contains the script to correct/modify.
  You should use this script as base and modify it to answer the questions in part 2.
  For each question, you should write a script and give it the name: id_<your student id>_q_<question number>.py
  The type 2 questions are relatively independent.

  For example:
  If your student id is 123456 then for question 7, the file name is: id_123456_q_7.py

Some Notations used in this document:

Grey boxes show code:

```
print("####################")
print("# Monday    # 18 #")
print("# Tuesday   # 19 #")
print("# Wednesday # 20 #")
print("# Thursday  # 21 #")
print("# Friday    # 22 #")
print("# Saturday  # 23 #")
print("# Sunday    # 24 #")
print("####################")
```



This script produces this output

White on black boxes show console outputs:

```
####################
# Monday    # 18 #
# Tuesday   # 19 #
# Wednesday # 20 #
# Thursday  # 21 #
# Friday    # 22 #
# Saturday  # 23 #
# Sunday    # 24 #
####################
```

Part I: 5 type 1 questions

Question #1

Write a python script that displays the following output:

```
####################
# Monday    # 18 #
# Tuesday   # 19 #
# Wednesday # 20 #
# Thursday  # 21 #
# Friday    # 22 #
# Saturday  # 23 #
# Sunday    # 24 #
####################
```

The alignment is important.
There is a space on each side of the longest word (Wednesday)
There is a space on each side of the numbers

Your script should use the following variables to create the display:

```
days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

start = 18
```

It should be possible to replace “#” by another symbol easily.
You should use string operations.
Your script should be able to work with a list of months and start = 1.

Question #2

Write a script that displays the characters of the string: “This is a test string” one at a time and stops when it reaches r.

Expected output

```
T
h
i
s

i
s

a

t
e
s
t

s
t
Found 'r' stops
```

You have to use a while loop whose condition is the string to parse.

```
string_to_parse = "This is a test string"

while string_to_parse:
        <statements>
            ⋮
```

It should be possible to easily modify the script to stop for a different letter than r.
If the selected letter is not found in the string, the script should display a message
(for example: Did not find ‘z’ in ‘This is a test string’)

Question #3

Write a script that check which attributes of a list are not available with a tuple.

The attributes starting and ending with "__" like "__doc__" are not considered.

Expected output:

```
tuples do not have 'append'
tuples do not have 'clear'
tuples do not have 'copy'
tuples do not have 'extend'
tuples do not have 'insert'
tuples do not have 'pop'
tuples do not have 'remove'
tuples do not have 'reverse'
tuples do not have 'sort'
```

(any order is fine)

You have to provide two methods for checking and a flag to select the method to use.

Method 1:

- You must use code introspection functions dir and hasattr.
- You cannot use sets

Method 2:

- You must use code introspection function dir.
- You must use sets.

Both methods:

- You must use string methods to remove attributes starting and ending with "__".

Question #4

Each row in the file data_q4.txt (given) contains three values denoted by: x, y and r
On each row, these values should be such that: r = cos(x) + y
But, there are some errors. In some rows r ≠ cos(x) + y.

Write a script that:

1. loads the data file
2. parses it to find the rows with error
3. writes a new data file without error (call it corrected_data_q4.txt)

Your script should display the indices of the rows with an error along with the squared error.
The squared error is the quantity $e = (\cos(x) + y - r)^2$.
If for a given row, $e > 0.001$ there is an error on that row.

Your script should give the total number of detected errors.

Your script should replace the row with error by a row without error in the new data file.

Additional requirements:
Use a list comprehension to split the lines read from the file in 3 floats

Question #5

Create a simple dice game:

First, the player select a dice by giving the number of faces of the dice: F
Then, the player select a number between 1 and F (included) and roll the dice.

The game shows the dice value.

If the dice gives the selected number the player scores one point.
At each success, the game reports the ratio points/number of rolls.

The game checks that the player selects a valid number (an integer between 1 and F).
A wrong selection should not stop the game and should not be counted as a trial.

To quit the game use the keyboard interrupt Ctrl+C. The game gives a greeting and stops.

Player's input are obtained using the "input" function.
You should implement this game using objects.

- A Dice class
- A Player class
- A Game class

Example output:

```
Number of faces: 4
Select a number between 1 and 4: 2
Dice roll: 4
Select a number between 1 and 4: 1
Dice roll: 1
Good guess 1/2
Select a number between 1 and 4: 0
Select a number between 1 and 4: 9
Select a number between 1 and 4: 1
Dice roll: 4
Select a number between 1 and 4: 4
Dice roll: 1
Select a number between 1 and 4: 4
Dice roll: 2
Select a number between 1 and 4: 1
Dice roll: 3
Select a number between 1 and 4: 4
Dice roll: 1
Select a number between 1 and 4: 1
Dice roll: 1
Good guess 2/8
Select a number between 1 and 4:
Thanks for playing
```

Part II: 5 type 2 questions

Question #6

The file id_none_part2.py contains a python script for simulating a simple social network.
In this simulation:

- Users are randomly created
- Users randomly post messages
- Users randomly read messages
- Users randomly like messages

Users may unregistered if they are not satisfied. Their satisfaction depends of their reading.

Try this script and you will notice that no new user are created and nothing happens.

Now the script is not running correctly.

Correct the script so that new users are created and messages can be posted.

(do not forget to repeat this correction for the script in questions 7 to 10)

Question #7

Modify the script so that a user is losing one point of satisfaction when reading her/his own message.

Question #8

Modify the script so that when a user unregister, her/his messages are also deleted.

Question #9

Messages are given some score. These scores are updated when a user likes a message.

Modify the script so that during the periodic update it displays:

- Information about the message with the highest score.
  (For example: message id, score, and user id of the author of the message)
- Information about the users satisfaction
  (user id and satisfaction)

Question #10

Modify the script so that:

- messages start with a non null score of 5
- the score of a message is downgraded of one point when the message is read but not liked
- messages disappear when their score reaches 0