EXAM PYTHON (100 POINTS)

- This is an examination. It is to be your work and your work alone.
- No exchange of information with another human entity in any form is acceptable.
- It is ok to read Google documents, however no exchanges via the internet are acceptable.
- It is ok to use class material, notes, your programs, your labs, and any other notes you have written for class.
- It is ok to use a textbook.

These instructions are general for all the scripts that you will write in python3

```
Instructions
Do NOT use python 2, make sure you are using python 3 and above.

Include the class header

Question/Answer pairs

#1.1a
var1=124

#1.1b
var2=3+var1
```

```
Deadline and late penalty
The deadline is 8:30pm with 3 min grace period.
After 8:33pm, the penalty is 2 points for each late minute.
Gradescope Exam2 will close at 8:40m

IF YOU HAVE TECHNICAL ISSUES DURING THE EXAM LET THE PROFESSOR KNOW
```

Policy for cheating: sharing code on the exam is unacceptable and will earn you a 0 and take you straight to the ethics board. Do not share any study documents either!

We give partial points in case of syntax errors.

We **will subtract -0.5 points for missing or incorrect header and/or Q&A for each script.**

After you download the Exam2-S2022.zip from Blackboard, unzip the file if necessary, and a directory called **Exam2-S2022** will be created. Change into Exam2-S2022 and start your work in that directory.

1. **(52)** In thermodynamics and engineering, a heat engine is a system that converts heat to mechanical energy.
The Carnot efficiency describes the maximum thermal efficiency that a heat engine can achieve as permitted by the Second Law of Thermodynamics.

The Carnot efficiency (η_{Carnot}) is a maximum theoretical efficiency of an ideal heat engine operating between the two temperatures (T_{cold} and T_{hot}), given by this formula:

$$\eta_{Carnot} = 1 - \frac{T_{cold}}{T_{hot}} \qquad (equation1)$$

A different measure of an ideal heat-engine efficiency is given by Curzon–Ahlborn formula:

$$\eta_{Curzon} = 1 - \sqrt{\frac{T_{cold}}{T_{hot}}} \quad (equation2)$$

In both formulas the temperatures T_{cold} and T_{hot} are in Kelvin.

1.1 (27+0.5 instructions) The provided module, called **module1.py** contains the following functions:

- Function **covert**, which takes one temperature value in Celsius, and returns that temperature in Kelvin, according to this formula:

$$T_{kelvin} = T_{Celsius} + 273$$

- Function **names,** which tales 0 parameters, and returns two strings, 'Carnot' and 'Curzon'

Edit the module and add the following:

- As a comment include this sentence in the script. **The place where to write the honor sentence is specified in the file.** Keep in mind that your exam **will not be graded,** and you will get a 0 if you do not have this sentence as a comment.

I promise not to communicate with another human being in any way about this exam.

a. (11) A function called **efficiencies,** which takes two temperature values as parameters (T_{cold} and T_{hot}), and returns two values: the Carnot efficiency (equation1, see above) and the Curzon efficiency (equation2).
To evaluate the $\sqrt{\frac{T_{cold}}{T_{hot}}}$ in equation2, you should import the appropriate module, and use a function from this module. If you do not know how to do this, you can use (**0.5) for a loss of 2 points.

b. (16)

b1. (3) Test the functions within the if __name__ statement

b2. (1) First, define two temperature values, which are in Celsius:

```
Tc_c=10
Tc_h=300
```

b3. (2) Use function **covert** to convert the temperatures from Celsius to Kelvin, and store results in two variables, Tk_c and Tk_h. Do not hardcode numbers 10 and 300, but use variables Tc_c and Tc_h.

b4. (2) Call function **efficiencies** on Tk_c (this is the Tcold) and Tk_h (this is the Thot), and store results in variable R.

b5. (8) Print to screen this formatted output, reporting the results of the calculated efficiencies. Do not hardcode *Carnot* and *Curzon* but obtain these strings by calling function **names**. It is OK to use the print function twice, i.e., you do not need a loop.

```
Carnot efficiency is 0.51
Curzon efficiency is 0.30
```

1.2 (24+0.5 instructions) The provided data file **power_data.dat** contains information on the observed efficiency of a few power stations.

The first line of the file is the headerline, and the subsequent lines store the data.

The data are organized in the following fields:

1st power type
2nd cold temperature value in Celsius
3rd hot temperature value in Celsius
4th observed efficiency

Make a new file called **my_efficiencies.py.** In it do the following:

a. (2) Import **module1**. You can use function or generic import.

b. (3) Read in the data file **power_data.dat** and store the content in variable L.

c. (2) Store the data only (without the headerline) in list L1. You should not include the first line. Do not use loop but use slicing.

d. (17) Make a for loop over list L1 and produce the following formatted output, where:
- the first field is the power type.
- the 2nd field and 3rd fields are respectively the Carnot and Curzon efficiency, which you should obtain by calling function **efficiencies**. Before using function efficiencies, you should use function **convert** to convert the temperatures from Celsius to Kelvin. You should use the functions from **module1**.
- the 4th field is the observed efficiency.

The fields must be aligned as below. Number of spaces can vary.

```
coal-fired power     0.64            0.40            0.36
nuclear power        0.48            0.28            0.30
geothermal power     0.33            0.18            0.16
```

2. (18.5=18+0.5 instructions) Edit the provided file **riddle.py**, which contains some variables. Your goal is to produce the following output on screen:

```
What is always in front of you but cannot be seen ?
The future
```

You can use two print statements for this, one for each line (sentence).

To generate the two sentences, do the following:

a. (11) For the first sentence, generate one list of strings, containing all the words as separate items. The list should look like this:

```
['What', 'is', 'always', 'in', 'front', 'of', 'you', 'but',
'cannot', 'be', 'seen', '?']
```

To make the list, you should use various data type manipulation/conversion methods as well as list concatenation. It is OK to generate intermediate variables.

b. (2.5) Use a type conversion method to convert the list generated in part a. to a string before printing the sentence to screen.

c. (4.5) For the second sentence, simply access values of dictionary d2 and print to screen.

3. **(**29.5=29+0.5 instructions) Edit the provided file **worldle.py**. In this exercise you will complete a script to guess a five-character word. Some code is already provided in the script. Do the following:

a. (14) First, use a while loop to check that the user's input is a five-character world. You should incorporate into the while loop the input function given in the script. The location of the while loop in the script should be where the input function is.
For this, make a while loop to check that the word entered by the user is 5 characters long. If the word entered is not 5 characters long, the user should be informed by a statement letting her/him know that the word entered is so and so characters long.

For example, if a user enters *Hi*, the statement should print:
`You entered a 2 character word. Try again.`

Then the user should be prompted again to enter a five-character word.

If you were unable to do part a., continue with part b. by using the input function as given to you. You can comment out your unsuccessful attempt at a. for partial credit.

b. (15) In this part will use the lists of characters, **target, user,** and **guess**. The goal is to replace values in list **guess** based on a test that you will perform on characters of list **target** and list **use**r.
For this do the following:
Make a for loop over integer numbers 0,1,2,3,4. Use the range function for this.
You will use these indices inside the loop to access values in lists **target**, **user** and **guess.**

Inside the for loop, do the test by using **an if-elif-else statement**.

- If a character from list **user** matches a character from list **target** in the same position, replace the value in list **guess** with the uppercase of that character. Note that the user can enter words in either uppercase or lowercase
- If a character from list **user** matches a character from list **target** but it is not in the same position, replace the value in list **guess** with the lowercase of that character. If you entered a word with a repeated character, it is OK that character appears multiple times.
- Otherwise, if there is no match, append the character to the list **notinword**.

c. Use a type conversion method to convert the final list **guess** into a string and final list **notinword** into a string, and print the two strings to screen, together with some additional text. For example, if the user entered the word *early*, the output for this part should be:

```
Your word: E a - - -
Not in word: r l y
```

EC. (3) This problem is all or nothing. Only fully correct code will be accepted.

In a file **worldle_EC.py** make a worldle game that will allow the user to guess a 5-character word. No need to check the user's input. Assume, the user will enter a 5-character world.

The player has a maximum of 6 guesses. If a player guesses the word (e.g., *exams*) the game should stop, and the user should be informed of the correct guess with a statement 'You win in round x' (where x is the round of the game when user won). You should keep track of the number of attempts.
If the player does not guess the word, then you should make two lists, as in exercise 3. List **guess** and list **notinword,** but with some modifications:

- The list **notinword** should contain only unique characters, i.e., the same character should not appear multiple times.

- If the user enters a word with multiple same characters, the repeated characters that do not appear in the guessed word should not be displayed in the **guess.**
  For example, if you enter the word *bliss,* and the word being guessed is *exams*

  it should not print:
  ```
  Your word: - - - s S
  ```

  but it should print:
  ```
  Your word: - - - - S
  ```

For each user's input the script should produce an output like in exercise 3. For example, if the user enters *bliss* in first iteration, and the word being guessed is *exams,* this should be printed to screen:

```
Please, input 5-character word: bliss
Your word: - - - - S
Not in word: b l i
```

Upload the following files to Gradescope Exam2:
module1.py
my_efficiencies.py
riddle.py
worldle.py
worldle_EC.py
DO NOT SUBMIT ANY ADDITIONAL FILES