EXAM PYTHON (90 + 3EC 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.
- Reading Google documents to find out how to solve a problem is not acceptable, including using chatGPT.
- It is ok to use class material, notes, your programs, and any other notes you have written for class.
- You can use the online python documentation pages listed at the end of the Python Notes.

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

```
Instructions
#class header

#Q1
var1=124
print(var1)

#Q2
var2=3+var1
print(var2)
```

```
Deadline and late penalty
The deadline is 8:00pm with a 3 min grace period.
After 8:03pm, the penalty is 2 points for each late minute.
Gradescope Exam2 will close at 8:10m.

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

Policy for cheating: sharing code on the exam or using chatGPT or using google to find out how to solve a problem, 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-S2024.zip** from Canvas, unzip the file if necessary, and a directory called **Exam2-S2024** will be created. Change into **Exam2-S2024** and start your work in that directory.

We deduct points for using Python concepts or modules that are beyond the scope of the class materials.

1. **(20)** Edit the provided script **ex1-s24-while.py** and do the following:

Q0. As a comment, include this sentence as Q0 for this script.
I promise not to communicate with another human being in any way about this exam.

Q1 (15) Make **ONE while loop** to construct a dictionary of **5 key:value** pairs.

- A key is a color randomly selected from the provided list:

```
L=['purple','orange','white','black','yellow','red','purple',
'orange','white','black','yellow', 'red']
```

- A value is a random integer number in the range [1,10].

Name the dictionary as you like. Import the necessary module before the while loop.

Requirements. Within the while loop:

- Generate only one random integer number and one random color in each iteration.
- Add a randomly generated key:value pair to the dictionary only once. This means, only unique key:value pairs should be added; avoid overwriting values during dictionary construction.
- Keep track of the number of iterations required to build the dictionary.

Q2 (5) Once the dictionary is constructed within the while loop, use **a single print function** to display the dictionary on one line, followed in the next line by a sentence reporting the number of iterations, as shown below:

```
{'purple': 8, 'white': 4, 'red': 10, 'yellow': 6, 'orange': 1}
has been constructed in 19 iterations.
```

Note: Ensure the dictionary and the sentence are printed on separate lines as specified.
Your dictionary of 5 key:value pairs and the number of iterations might differ.
Do not format, and do not concatenate. Use variables previously defined to store the dictionary and the iteration count.

2. (30) Suppose we have a point specified in cylindrical coordinates **r, θ, z,** and we want to know the distance **d** between the point and the origin.

The simplest way to do the calculation is to convert r and θ to Cartesian coordinates first (x and y), according to these formulas:

Eq.1
$$x = r\,cos(\theta)$$
$$y = r\,sin(\theta)$$

and then apply Pythagoras' Theorem to calculate d, according to this formula:

Eq.2 $$d = \sqrt{x^2 + y^2 + z^2}$$

In a script called **ex2-s24-distance.py**:

Q1 (4) Use **function import** to import all the necessary functions, including **sin** and **cos**, as well as the constant π from the math module. If you do not know how to do function import, use generic import for a loss of 2 points on this part.

Q2 (12) Make a function called **distance**, which takes 3 parameters, **r, θ, z.** The function should calculate the distance d using the provided equations.
Firstly, it should convert r and θ to Cartesian coordinates (x and y) according to Eq.1, and then calculate d according to Eq.2. Finally, the function should **return** the calculated distance, **d**.

Q3 (14) Different points have been recorded. One point is of this form: **P= (r, θ, z),** with θ in radians (in terms of π).

P1=(4, 2π/3, -2)
P2=(2, π /3, -1)
P3=(-3, π/2, -5)
P4=(3, π , 1)

Your task is to print to screen the point name (P1, P2, P3 or P4) that have the maximum distance d from the origin.

For this do the following:

- (1) Make these variables (hardcode them or copy and paste them).

```
r1=[4,2,-3,3]
z1=[-2,-1,-5,1]
P=['P1','P2','P3','P4']
```

r1 stores the r values, z1 the z values and P the points' names.

- (1) Make a list called **theta1** containing the numeric values of θ, which are 2π/3, π/3, π/2, π. Hardcode the values. Use π from the math module.

  *If you do not know how to write the θ values in terms of π from the math module, make and use the variable theta1, provided in the file **back-s24.py** for a loss of points on this part.*

- (5) Generate a list of distances, called **Ldistance,** using the function **distance** within a list comprehension, and variables you previously made.

  *If you do not know how to do this part, make, and use the list **Ldistance** provided in the **back-s24.py** for a loss of points on this part.*

- (5) Calculate the maximum distance and find the corresponding point name (P1, P2 P3 or P4). For this part use a built-in function, a method and indexing.
  Note: No loops, no comprehension. Do not hardcode, and do not sort.

  If you do not know how to do this part, make this variable name, and use it for the next question.
  ```
  name="P3"
  ```

- (2) Print to screen the point name with maximum distance from the origin. Do not hardcode the point name:

  ```
  The point with maximum distance is P3.
  ```

3. **(40)** The provided file **planets-all.csv** contains data on our planets.

   The fields are:
   1st: parameters
   2nd: values of those parameters for Mercury
   3rd: values of those parameters for Venus
   and so on for the other planets

   Look at the contents of the file and find out what the field separator is.

   In a script called **ex3-s24-planets.py** do the following:

Q1 (5) Read in data and store it in a list called L. Print L to screen.

*If you do not know to do it, use list L provided in the file **back-s24.py**.*

Q2 (8) The **first line of the file is a headerline** reporting the names of the fields, which includes the planets' names.
Your task is to use the headerline stored in list L to create a list called **Lplanets** containing all the planets' names.

The list **Lplanets** should be:

```
['MERCURY', 'VENUS', 'EARTH', 'MOON', 'MARS', 'JUPITER', 'SATURN',
'URANUS', 'NEPTUNE', 'PLUTO']
```

Note: Do not use loops or comprehension.

If you do not make this part, hardcode the list Lplanets for a loss of points on this part.

Q3 (10) Use a for loop to select the line containing in the 1st field the string *Number of Moons* and store the number of moons values in a list of integers called **Lmoon**. The number of moon values are from the 2nd field to the end.
Print the list **Lmoon** to screen.

```
[0, 0, 1, 0, 2, 92, 83, 28, 16, 5]
```

Note: Do not use comprehension, and do not hardcode the last field.

If you cannot make the list of numbers, make instead a list of strings, for a loss of 3 points.

```
['0', '0', '1', '0', '2', '92', '83', '28', '16', '5\n']
```

If you do make this part, hardcode the list of integer numbers Lmoon for a loss of points on this part.

Q4 (8) Use **one input function** to prompt the user to enter an imaginary planet name, and its number of moons, separated by a space.
Add the planet name as the second item in list **Lplanets**, and the number of moons as second item in the list **Lmoon.** The final Lmoon should be a homogenous list. Print to screen the updated lists.

Example, if the user enters:
Tatooine 10

The lists should be:

```
['MERCURY', 'TATOOINE', 'VENUS', 'EARTH', 'MOON', 'MARS', 'JUPITER',
'SATURN', 'URANUS', 'NEPTUNE', 'PLUTO']

[0, 10, 0, 1, 0, 2, 92, 83, 28, 16, 5]
```

Note: Do not use loops or comprehension. Use a list method to add the item. You can hardcode the index of the second item.

Q5 (9) Use a for loop, **Lmoon** and **Lplanets** to write into a file called **moons.dat** this formatted output. No additional empty lines should appear in the file. Align the two fields according to the format below. The number of spaces between the two fields can vary.

```
MERCURY    0
TATOOINE   10
VENUS      0
EARTH      1
MOON       0
MARS       2
JUPITER    92
SATURN     83
URANUS     28
NEPTUNE    16
PLUTO      5
```

Note: If you did not complete Q4, use the lists without the additional planet, and the line depicted in blue should not appear in the file moons.dat.

EC (3) Continue in **ex3-s24-planets.py.** Find the planet with the largest recorded *Surface Pressure(bars)*. Note that the surface pressure remains *Unknown* for some planets, and some pressure values should be cleaned, because they contain extra symbols * (asterisks). You can use variables made in the previous points.

Assume you do not know which values are *Unknown* or contain asterisks and how many asterisks there are. Additionally, assume you do not know which planet has the largest value.

You can use any class material, like flow control, comprehension, methods, etc., and make as many variables as you like. **Only use class material.**

Print this to screen.

```
The planet with the largest surface pressure of 92.0 bars is VENUS.
The surface pressure is still unknown for the following planets:
JUPITER
SATURN
URANUS
NEPTUNE
```

```
Submission to Gradescope Exam2 the following:
ex1-s24-while.py
ex2-s24-distance.py
ex3-s24-planets.py (with or without the EC)
DO NOT SUBMIT A ZIP FILE.
```