HW5 (100 POINTS) FROM FALL 23

These instructions are applicable to all scripts that you will write in python3.

```
Instructions for Python3 Homework submission

#class header
Q&A format

#Q1
var1=124
print(var1)

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

This HW has 3 parts; HW5-part1, HW5-part2, and HW5-part3 and will have 3 separate submissions on Gradescope.

All these parts are due Sunday March 3rd 11:59pm

If you submit:

- **HW5-part1 by Sunday Feb 25th (11:59pm) will get 3 extra points.**
- **HW5-part2 by Wed March 1st (11:59pm) you will get 3 extra points.**
- **HW5-part3 Sat March 2nd (11:59pm) you will get 2 extra points.**

General grading

We do not give points for instructions but take 1 point off for not following them.

We take all points off for syntax errors for that question/exercise where errors occur (including a simple typo).

We take all points off for using out of class material code/concepts.

Policy: Work on the HW problems on your own. It is not allowed to share code in anyway. The use of chatGPT is not allowed. Using internet to find out how to solve problems is not allowed. In the case of cheating, you will be reported to the Ethics Board.

HW5-part1 (27)

1. (15) The Moon orbits Earth. In turn, Earth and the other planets orbit the Sun. The space directly above our atmosphere is filled with artificial satellites in orbit. We examine the simplest of these orbits, the circular orbit, and we determine the period *T* for the International Space Station (ISS) by using this formula:

$$Eq1. \qquad T = 2\pi \sqrt{\frac{R^3}{G\ M_e}}$$

Where $R = R_e + R_s$

G = 6.67 x 10^{-11} N m^2/Kg^2 is the gravitational constant

M_e = 5.96 x 10^{24} Kg is the mass of the Earth

R_e = 6.36 x 10^6 m is the Earth radius

Rs = 4.00 x 10^5 m is the distance of the ISS from the Earth's surface

Write a python script called **ex5-space.py** to calculate the orbital speed v_{orbit} and period T for the ISS.

Curiosity: The ISS is in low Earth orbit (LEO). Nearly all satellites are in LEO, including most weather satellites. GPS satellites, at about 20,000 km, are considered medium Earth orbit. The higher the orbit, the more energy is required to put it there and the more energy is needed to reach it for repairs. Of particular interest are the satellites in geosynchronous orbit. All fixed satellite dishes on the ground pointing toward the sky, such as TV reception dishes, are pointed toward geosynchronous satellites. These satellites are placed at the exact distance, and just above the equator, such that their period of orbit is 1 day. They remain in a fixed position relative to Earth's surface.

Q1. (2) Import the math module. Use **function import** and import only the *sqrt* function and the π constant.

Q2. (3) Make 4 variables called G, Me, Re, and Rs and assign to them the corresponding values reported above. Use scientific notation to write the numbers.

Q3. (1) Make a variable called var1 and assign to it the string *International Space Station*.

Q4. (3) Use variables var1 and Rs to print this formatted output to screen:

```
The International Space Station orbits at 4e+05 m from the Earth.
```

Q5. (1) Make a variable called R, and assign to it the results of this calculation:
$R = R_e + R_s$

Q6. (2) Calculate T using Eq1, and store the result in a variable called T.
Use G, Me and R variables, the *sqrt* function and π constant from the math module.

Q7. (2) Use variable T and print this formatted output. In the print function, you should convert T into minutes by dividing T by 60.

```
The period is 92 minutes.
```

When you run the script, you should obtain:

```
The International Space Station orbits at 4e+05 m from the Earth.
The period is 92 minutes.
```

2. (12) A ball is dropped from a tower of height h. It has initial velocity zero and accelerates downwards under gravity. Write a program called **ex5-ball.py** that asks the user to enter the height in meters of the tower and a time interval t in seconds, then prints on the screen the height of the ball above the ground at time t after it is dropped, ignoring air resistance.

The steps involved are the following:

Q1 (4) Use two input functions, one to get the value of h and one to get the value of t from the user. Both h and t should be real numbers. You should use the text reported below in the print function.

Q2 (3) Calculate how far the ball falls in the given time t, using the standard kinematic formula:

$$s = 1/2\ gt^2$$

where $g = 9.81\ ms^{-2}$ is the acceleration due to gravity. You need to define the variable g before using it.

Q3 (4) Print the height above the ground at time t, which is equal to the total height of the tower minus the distance the ball has fallen. Do not format the output. The calculation of the height above the ground should be within the print function.

Q4 Use this program to calculate the height of a ball dropped from a 100m high tower after 2 seconds. When you run the program, the output should be:

```
Enter the height of the tower: 100
Enter the time interval: 2
The height of the ball is 80.38 meters.
```

Upload the following file to Gradescope HW5-part1:
ex5-space.py
ex5-ball.py

HW5-part2 (23)

1. **(23)** The provided script **ex5-interact.py** contains several variables.

   The variables are reported here, but please edit the provided script.

```
Laz=['***','temperature anomaly AZ',1.60,1.48,1.78,1.80,1.88,1.90,1.50]

Tstate=('A~r`i^z@o>n#a', 'Nevada','Colorado','Colorado')

D={1:'This', 2:'is', 3:'an interactive', 4: 'script.'}

T=("Hello,","no sense", "this is the end.")

Dprot={'prot1':{'code':'GATTACA'},'prot2':['ACATTA',34,56,78],'prot3':('D
NA','RNA')}
```

Edit the provided script, and in it do the following:

Q1. (3) Use slicing on Laz to make a new list, called Laz_clean, which only contains the numbers. Print the list to screen. Do not hardcode the length of the list.

Q2. (3) Access the first element of the tuple Tstate (i.e. string *A~r`i^z*o`n@a*) and use stepwise slicing to obtain the string *Arizona*. Use only indexing and slicing. Print the string to screen. Do not hardcode unnecessary numbers within slice, including the length of the list.

Q3. (5) Create this string out of dictionary D by using type conversion functions/methods and print it to screen. Do it in one line of code. Do not use slicing.

```
This is an interactive script.
```

Hint: Make a list of values, convert the list of values to string.

Q4. (3) Use one input function to ask the user to enter one number and one color, separated by a comma, and store it in the variable called num_col.

Q5. (3) Use type conversion to convert the color entered by the user to a list type. Print the list to screen. If the user enters *10,green* this list should be produced:

```
['g', 'r', 'e', 'e', 'n']
```

Q6. (4) Access the string “RNA” in dictionary Dprot, convert it to a tuple and print it to screen:
`('R', 'N', 'A')`
Do this with one line of code.

Q7. (2) Unpack the tuple T (i.e., assign a variable to each value of the tuple), and use the correct variables in the print function to print to screen:

```
Hello, this is the end.
```

This should be the output of the script:

```
[1.6, 1.48, 1.78, 1.8, 1.88, 1.9, 1.5]
Arizona
This is an interactive script.
Enter a number and a color separated by comma: 10,green
['g', 'r', 'e', 'e', 'n']
('R', 'N', 'A')
Hello, this is the end.
```

2. **(10)** The formula for exponential growth or decay is:

$$A = A_o \ e^{(k\,t)}$$

where:

- A is the final number of a quantity you're dealing with (e.g., money, bacteria growing in a petri dish, radioactive decay of an element)
- A_0 is the number of that same quantity at time 0
- k is the growth or decay rate
- t is time.

A certain type of bacteria, given a favorable growth medium, doubles in population every 6.5 hours, i.e., the rate is **k = 0.10661/h**. How many bacteria will there be in a day and a half, i.e., **t = 36** hours?

To answer this question, make a script called **ex5-bacteria.py** to implement the formula.

The value for A_o will be provided from the command line when you run the script.

Q1 (2) Import the necessary modules and attributes.

Q2 (1) Make the variables k and t for the bacteria described above.

Q3 (4) Make a variable called Ao that will store a value for Ao provided from the command line. Ao should store a numeric type.

Q4 (2) Use variables, k, t, and Ao and calculate the formula. Store the result of the calculation in a variable called Abac.

Q5 (1) Print a formatted statement reporting the result by using the variable *Abac*.

If 100 is provided from the command line, your script should produce this formatted output:

```
The number of bacteria will be 4.6e+03
```

Upload the following file to Gradescope HW5-part2:
ex5-interact.py
ex5-bacteria.py

HW5-part3 (50)

1. **(26)** The provided script **ex5-misc.py** contains different variables.

The variables are reported here, but please use the provided script.

```
Laz=['***','temperature anomaly AZ']

L=['Nevada','Colorado','Colorado']

L1=["New Mexico", 'Utah',"California"]

D={'California':[1.23,1.13,1.34,1.52,
1.46,1.98],'Colorado':[1.28,1.80,1.57,1.45]}

DNA='aca##a#g#atg##c#attgt'

Dchem={'hydrogen': ['H', 3], 'helium': ['He', 2], 'lithium':['Li',
3],'Adamantium':['Ad',22]}
```

Edit the script and do the following:

Q1. (6) Use list Laz, concatenation, repetition, indexing and slicing to print the following to screen:

```
******   temperature anomaly  ******
```

Notice: there should be 3 spaces between the stars and the text you should access nested types achieve this with one line of code

Q2. (3) Access the list of temperature values for California in dictionary D and use a function to calculate the minimum temperature value in California. Store the minimum value in variable mCA. You should achieve this with one line of code.

Q3. (2) Use variable mCA to print the following to screen. You can hardcode California. No need to format, and do not concatenate.

```
In California the minimum recorded temperature is 1.13
```

Q4. (2) Extend list L by L1. Use a method to update the original object. No new objects should be generated. Print the updated list L to screen.

```
['Nevada', 'Colorado', 'Colorado', 'New Mexico', 'Utah', 'California']
```

Q5. (3) Remove duplicate *Colorado* from list L. For this do the following:

- Store the index of *Colorado* in variable ind.
- Use ind to remove *Colorado* from list L and print the removed *Colorado* to screen using one method.
- Print the updated list to screen.

Q6. (4) Use a sequence of string methods on variable DNA to remove the pound, substitute character *t* with character *u*, and store the new string uppercase in a variable called RNA. The new string RNA should be:

```
ACAAGAUGCAUUGU
```

Do this in one line of code.
Print the value of RNA to screen.

Q7. (2) Print to screen how many times character *A* occurs in string RNA. Use RNA and a method to count within one print function.

Dictionary Dchem contains information about some chemical elements.
The data in the dictionary are organized in this way:

```
{name of the element : [symbol, atomic number]}
```

Q8. (2) Delete the key *Adamantium* and corresponding value, since this is a fictional metal appearing in Marvel Comics. Print the updated dictionary Dchem to screen.

Q9. (2) In Dchem the atomic number of the hydrogen is incorrect. Replace the 3 with 1. You should only replace the number in the list, and not the entire list. **You should access nested types.**
Print the updated dictionary Dchem to screen.

Run the script. You should obtain this output:

```
******   temperature anomaly   ******
In California the minimum recorded temperature is 1.13
['Nevada', 'Colorado', 'Colorado', 'New Mexico', 'Utah', 'California']
Colorado
['Nevada', 'Colorado', 'New Mexico', 'Utah', 'California']
ACAAGAUGCAUUGU
A occurs 5 times
{'hydrogen': ['H', 3], 'helium': ['He', 2], 'lithium': ['Li', 3]}
{'hydrogen': ['H', 1], 'helium': ['He', 2], 'lithium': ['Li', 3]}
```

2. (24) Our planets orbit around the sun at very high velocity, and each planet has a different average temperature. In the provided script **ex5-speed-planets.py**, the dictionary Dplanets is defined, and it contains information about the orbital velocity of a planet around the sun in km/h and the average temperature in Celsius.
In this dictionary, the key is the planet name (In Upper Case letters), and the corresponding value is a list; the first item of the list is the velocity in km/h and the second item the temperature in Celsius.

```
Dplanets: {"planet name": ["orbital velocity", temperature]}
```

This variable is also defined in the script:

```
L=['s', 'p', 'a', 'c', 'e']
```

Edit **ex5-speed-planets.py** and do the following:

Q1. (4) Use a string method to convert list L into this string:

```
SPACE
```

and print the string to screen.

Q2. (5) In the provided dictionary Dplanets, the temperature value for the Earth is missing. The temperature of the Earth is 15 Celsius.
Do the following:

- Append the number 15 to the list associated with value 'EARTH' in dictionary Dplanets.
- Print to screen the value associated with key ‘EARTH'.
  The output should be:
  `['107280', 15]`

Q3. (4) Use dictionary Dplanets and type conversion to make a list of the planet names, in alphabetic order, and store the sorted list in variable Lplanets.

Q4. (3) Use variable Lplanets, and print to screen the list of planets and the text reported below:

```
The planets of our solar system are:
['EARTH', 'JUPITER', 'MARS', 'MERCURY', 'NEPTUNE', 'PLUTO', 'SATURN', 'URANUS', 'VENUS']
```

Use only one print function. Notice that there are two lines.

Q5. (3) **Use the input function** to ask the user to enter the name of a planet.
Store the user's input in a variable called name. Your input function should contain text.
Use a method to be sure that the name variable will store the letters in uppercase.

Q6. (2) Use variable name to access the associated orbital velocity in Dplanets and store the value converted in km/s in variable vel.
To convert the orbital velocity from km/h to km/s, divide the orbital velocity by 3600.
Variable vel should store a float type.
Notice that the orbital velocity is stored as a string type in Dplanets.

Q7. (3) **Print a formatted statement** – see example below - reporting the name of the planet entered by the user, the corresponding orbital velocity around the sun in km/s and the temperature value.

If the user enters Earth, the output should be:

```
EARTH orbital velocity around the sun is 29.8 km/s and its temperature
is 15.0 C
```

Do not hardcode the planet name, the orbital velocity, or temperature, but use variables name, vel, and Dplanets.

Run the script. You should obtain this output:
Notice the last two lines will differ based on the planet entered after "Enter a planet name:"

```
SPACE
['107280', 15]
The planets of our solar system are:
 ['EARTH', 'JUPITER', 'MARS', 'MERCURY', 'NEPTUNE', 'PLUTO', 'SATURN',
'URANUS', 'VENUS']
Enter a planet name: earth
EARTH orbital velocity around the sun is 29.8 km/s and its temperature
is 15.0 C
```

Upload the following file to Gradescope HW5-part3:
ex5-misc.py
ex5-speed-planets.py