===
Python and GNUPlot Basic
===

Submission Guidelines for assignment
----

1. In the file readme.txt in the team-name directory, which contains the contribution of each team member, roll number and references (cite where you get code/code snippets from).

2. Rename the directory team-name to actual team name instead of
E.g. Coders

3. Compress the directory to <team_name>.tar.gz
e.g. coders.tar.gz

4. Submit one assignment per team. Please.

5. For every python problem, you have to define a function in your script, make sure the name of the function is the same as defined in each problem. Use python3 available in the lab machines.

Problem 1

Generate Fibonacci number series using Generator and yield.

Function Def: fib()

Returns an iterator.

First call to fib(), next() gives 0, Second gives 1, Third gives 1, Fourth gives 2, And so on..

Filename: p1.py

Problem 2

Identify the function and create a 3D plot of the function as shown in figure 2.png. The function is cubic in the input.

What to submit?
In the folder structure.
2.gnuplot, 2.png

Problem 3

You are a given datafile "Plot3Data.csv" with 2 columns: Rollno & Marks, separated by space. Determine the grade of the student using ‘Grade point Calculator’ given below and plot the Rollno Vs Grade point graph with gridlines. Roll number to be shown on X-axis and Grade on the Y-axis.

The marks should be displayed at the top of each grade as shown in the sample output. The output should contain gridlines.

Use the sample input provided in file "Plot3Data.csv".

Grade point Calculator: M denotes marks

M<=39	4
40<=M<=44	5
45<=M<=49	6
50<=M<=54	7
55<=M<=59	8
60<=M<=69	9
70<=M	10

Submit 3.gnuplot and 3.png

Problem 4

Create a plot for the Ackermann recursive function as shown in the figure 4.png

What to submit?
4.gnuplot, 4.png

Problem 5

Your significant other (SO) has a habit of writing a diary every day in a text file. Now, you somehow found the `MyDiary.txt` **(a sample is provided in the folder structure)** from their system.

You are curious to find who they contact or email regularly.

Write a Python program (`p5.py`) to find all the email IDs and phone numbers and their occurrence frequencies from the text file. Dump all the contacts and their frequency (separated by a space in no particular order), including your own contact.

It is clear to you that you should have a serious discussion with your “SO” if you detect more conversations (same email/phone) with someone other than yourself (consider using a list comprehension here!).

First print your own occurrence frequency on a line:

```
      my frequency: <frequency>
```

The program must report all these trespassers (in no particular order) if they exist:

```
      Cheater alert! <trespasser’s contact> <frequency>
      Cheater alert! ...
      ...
```

else state,

```
      It’s all good yo!
```

You have to use regular expressions to find the email IDs and phone numbers.
Description of email IDs (tip: don’t use something off the internet, use regexr.com or regex101.com for practice):-

<email id> = *<local part>* **@** *<domain>*
<local part> = one or more *<alphanumeric>*s separated by *<dot_or_us>*
<alphanumeric> = one or more letters [a-zA-Z] or digits [0-9]
<dot_or_us> = The character **.** or the character _
<domain> = one or more *<alphanumeric>*s separated by *<dot>*. The last *<alphanumeric>* should be a *<alphabetic>*
<alphabetic> = one or more letters.

Sample email id:- fxps_ho.4@anhthu.org

Format of phone number:- 10 consecutive digits

Note:

1. 0123456789 is not a valid phone number as it starts with 0.
2. Also, 98765432100 is not valid as its length is 11.
3. Email and Number will be at a word boundary but may be surrounded or adjacent to punctuations.

Your program will be invoked like:

```
python3 p5.py <path-to-diary> <contact>
# contact could be an email or phone no.
```

Sample output *(for 7758648932)*:

```
my frequency: 7
Cheater alert! emokid@niceguys.com 10
```

Problem 6

You have to create a database of students, which will contain First Name, Last Name, Roll Number, Gender, Mobile, Dept & CGPA. Write a Python program, which in the first run should take input from command line arguments and create a CSV file named **student_database.csv**. Subsequent runs should append the data to the same file.

The argument parser should be used for this task because the order of arguments passed in the command line will be random. If any of the arguments are not present, your program should display an error message as shown in the following example.

Constraints:-

0 ≤ Number of arguments ≤ 7

Every argument will be of type “--<ARG_NAME>=AAA”.

“ARG_NAME” will be one of the 7 mentioned below (no need to worry about spellings) Don’t worry about the type of the “AAA” part, consider it as a string.

Example1:-
```
> python p6.py --first_name=abc --last_name=xyz
--roll_no=17305 --gender=male --mobile=1234567890 --dept=CSE
--CGPA=8.4

> Successfully Added!!
```

Example2:-
```
> python p6.py --CGPA=8.0 --gender=male --mobile=4321567890
--roll_no=17306 --last_name=asd --first_name=zxc --dept=HSS

> Successfully Added!!
```

Example3:-
```
> python p6.py --dept=IE --last_name=qwe --CGPA=9.4
--roll_no=17308 --gender=female --mobile=5684167890
--first_name=rty

> Successfully Added!!
```

Example4:-
```
> python p6.py --dept=IE --last_name=qwe --CGPA=9.4
--roll_no=17308 --mobile=5684167890 --first_name=rty

> the following arguments are required: --gender
```

Example5:-
```
> python p6.py --last_name=qwe --CGPA=9.4 --roll_no=17308
--mobile=5684167890 --first_name=rty

> the following arguments are required: --gender, --dept
```

In the above examples, replace Question5.py with Q5.py
student_database.csv should be generated in the same directory
Even though the order of command arguments changed it should create the CSV file **student_database.csv** containing the three lines that have been successfully added:

```
First Name, Last Name, Roll Number, Gender, Mobile, Dept, CGPA
abc, xyz, 17305, male, 1234567890, CSE, 8.4
zxc, asd, 17306, male, 4321567890, HSS, 8.0
rty, qwe, 17308, female, 5684167890, IE, 9.4
```

In the csv file, first row should contain "First Name, Last Name, ..."

Hint:-
1. import **argparse**, import **csv**
2. Overwrite the parse.error method
If you are using and IDE like PyCharm then you can give the arguments in “run -> run configuration -> parameters”

Problem 7

To keep track of scores during IPL matches, a dictionary is used as follows:

```
{
  "ipl1": { "rohit": 57, "virat": 38 },
  "ipl2": { "smith": 9, "warner": 42 },
  "ipl3": { "rahane": 41, "tare": 63, "russel": 91 }
}
```

Each player and match is represented by a string and the scores by integers.

Input:

Stats of each match will be presented on lines over **`stdin`**:
<number of matches> followed by <number of matches> lines of,
`<match_name>:<player_name>-<runs>,<player_name>-<run>,...`

Output:
Construct and print the dictionary with the structure mentioned above, followed by a list containing tuples `(name, total-score across all matches)` for each player sorted by the total-score in descending order, (and decreasing lexicographic order of player names).

Assume that,
1. match_name will be unique for all matches
2. player_name will be unique in a given match
3. runs will always be greater than equal to 0

For instance:
Input:
```
3
match1:p1-9,p2-38
match2:p3-19,P1-49
m3:p3-1,p4-6,p1-91
```

Output:

```
{'match1':{'p1':9, 'p2':38}, 'match2':{'p3':19, 'P1':49},
'm3':{'p3':1, 'p4':6, 'p1':91}}
[('p1', 100), ('P1', 49), ('p2', 38), ('p3', 20), ('p4', 6)]
```

just print the dictionary and the list (each on a separate line), don't worry about space.

NOTE: Here P1 and p1 are different!

Filename: p7.py