General Instructions: Do not copy-paste from this file to terminal. If you have doubts, contact the instructors or TAs. And **do not panic**!

- The first two tasks in this worksheet require you to use Python3 shell. You need to copy paste your work (commands and outputs) to a file using gedit or nano.
- The last few problems in this worksheet will require you to write a program.
- You should keep all your files in `CS1101/ws04` folder.
- Use `gedit` or `nano` to type your programs.
- The name of the programs should be `prob-n.py` for n^{th} problem.
- Save the output of your program in a text file `prob-n-output.txt`.
- After you finish, create an archive of the folder `ws04` with name `ws04-idnumber.tgz` and upload in `WeLearn`.

- Open `gedit`.
- Open a terminal and start the python shell.
- Complete the next two tasks given below in the python shell in your terminal.
- You will copy-paste the `python` commands and the corresponding outputs in `gedit` and save the file as `prob-N.txt` where `N` is the number of the task. For each task you need to save one file.
- Task 1: *Exploring conditions*
  1. Type **`python3`** to start a python shell
  2. Let, **`a = 0.0`** and **`b = 3//5`**
  3. Check the output of **`a == b`**.
  4. Check the output of **`a is b`**.
  5. Check the output of **`a == b and a is b`**. Explain your observation.
  6. Type the following in the shell using appropriate indentations

```
a = 0; b = 20; k = 0
while a < b:
    b = b - 3
    k += 1
print(k-1)
```

  7. What is the significance of the final printed value?
  8. Divisibility: check whether a given integer n is an exact multiple (integer division yields an integer and no remainder) of another number m. Try **`n%m==0`** as a test?
  9. Save the `gedit` contents as `prob-1.txt`

- Task 2: *List comprehension and loops and breaks*

  1. Type the following in a python shell
     `[n for n in range(20) if n%2==0]`
  2. The above prints the even numbers in the range 0 to 19.
  3. Generate a list of numbers having values 2^n where **`n`** is from 0 to 20 and is multiple of 3.
  4. Use a **`for`** loop to print numbers from 1 to 20.
  5. Break the loop **`if`** the loop counter is 7.
  6. Try the following in a python shell:

     ```
     a = 0; k = 1
     while 2**k < 20:
         a += 2**k
         k += 1

     print(a, k)
     ```

  7. The above loop tries to evaluate $\sum_k 2^k$ for all k which satisfies $2^k < 20$.
  8. Modify the loop to evaluate $\sum_k 2^k/k$ for all k which satisfies $2^k < 20$.
  9. Save the `gedit` contents as `prob-2.txt`

---

4. Write a program that takes an integer (a) from the user and then finds the largest positive integer k for which $2^k < a$.

5. A prime number is an integer divisible only by one and itself. How do we check whether a number n is prime? To check if n is divisible by another number q, we can check if **`p % q == 0`**. Now notice that for a generally divisible non-prime number (say, 36) we have

$$\begin{aligned} 36 &= 2 \times 18 \\ 36 &= 3 \times 12 \\ 36 &= 4 \times 9 \\ 36 &= 6 \times 6 \\ 36 &= 9 \times 4 \\ 36 &= 12 \times 3 \\ 36 &= 18 \times 2 \end{aligned}$$

   We notice, that after 6 (i.e., $\sqrt{36}$) we do not have a new pair. As such, we need not check for q beyond $\sqrt{n}$. Now, write a program to take an integer from a user and check whether it is prime. Try to use a **`while`** loop to implement it.

6. We have seen that the statement **`a, b = p, q`** assigns the values of p and q to a and b. So, **`a, b = b, a`** will swap (interchange) the values of a and b. Using this trick, write a program to generate all Fibonacci numbers below a given cutoff (taken from the user). Check the previous worksheet for the definitions of Fibonacci numbers.