Pre lab

2. For the following `mystery1` algorithm,

```
Algorithm mystery1(list)
Input : a list of integers, list
Output : ?

    IF length of list is 1 THEN
        RETURN first element in the list        → base case
    ELSE
        a ← first element in list
        b ← mystery1(rest of the list)          → recursive
        IF a > b THEN
            RETURN a
        ELSE
            RETURN b
```

2.1. What are base case and recursive case of `mystery1`
2.2. Trace function `mystery1` for a list of 5 integers. Show the call stack.
2.3. What does `mystery1` do ?
2.4. Write an iterative version for `mystery1` (using pseudocode)

2.2

mystery 1 (l), len(l) = 1
mystery 1 (l), len(l) = 2
mystery 1 (l), len(l) = 3
mystery 1 (l), len(l) = 4
mystery 1 (l), len(l) = 5

2.3 find maximum number in list

2.4

```
input : list of integer
max = 0
for i in list
    if i > max
        then max = i
    end if
return max
```

3. Write recursive algorithms (using pseudocode) to
   3.1. find out whether a given String is a palindrome.
   3.2. calculate `a*b` where `a` and `b` are positive integers. Note that you are not allowed to use the " * " operator.

code . prclab3

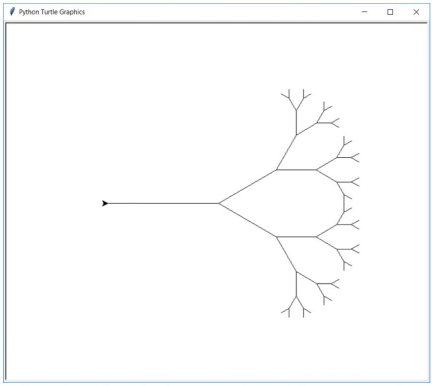
3.1
```
input  x , y
    product (x, y)
      if x = 1
       then  return y
      else
       return  y + product (x-1 , y)
```

3.2
```
input   s (string)
ispalidrome (s)
if length of s is less than 1
  then return true
else
 if first letter equal to last letter
  return  ispalidrome ( s[1:-1])
else                          ↓
  return false          remove first and
end if                    last letter

      a = get input
      if  ispalidrome (a) equal to true
        then print " Palindrome"
        else
          print " not a Palindrome"
```

In lab

1. For the following tree like figure, do the following



1.1. Write a recursive algorithm (using pseudocode) to create this tree.

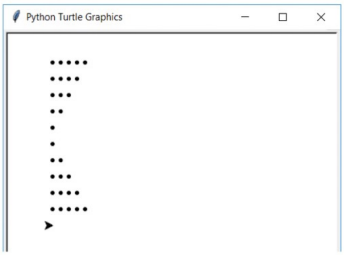
1.1.1. What is base case ?
1.1.2. What is recursive case ?
1.2. (optional) Write a program to draw this tree using `turtle`. ⟶ code . turtree

1.1

```
input size
tree (size)
if size less than 10
  then return  → base case
end if
forward (size)
left (30)
tree (size × 0.5)  → recursive case
right (60)
tree (size × 0.5)  → recursive case
left (30)
backward (size)
```

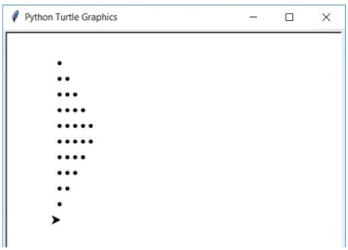
2. Write a recursive function `stars1(n)` to generate the following star pattern. Figure below is the output when `n` is 5. Print stars on Python console and use `turtle` to draw star pattern.

```
*****
****
***
**
*
*
**
***
****
*****
```



3. Write a recursive function `stars2(n)` to generate the following star pattern. Figure below is the output when `n` is 5. Print stars on Python console and use `turtle` to draw star pattern.

```
*
**
***
****
*****
*****
****
***
**
*
```



code . patty 1

patty 2

Post lab

A sequence of Fibonacci numbers is as follows

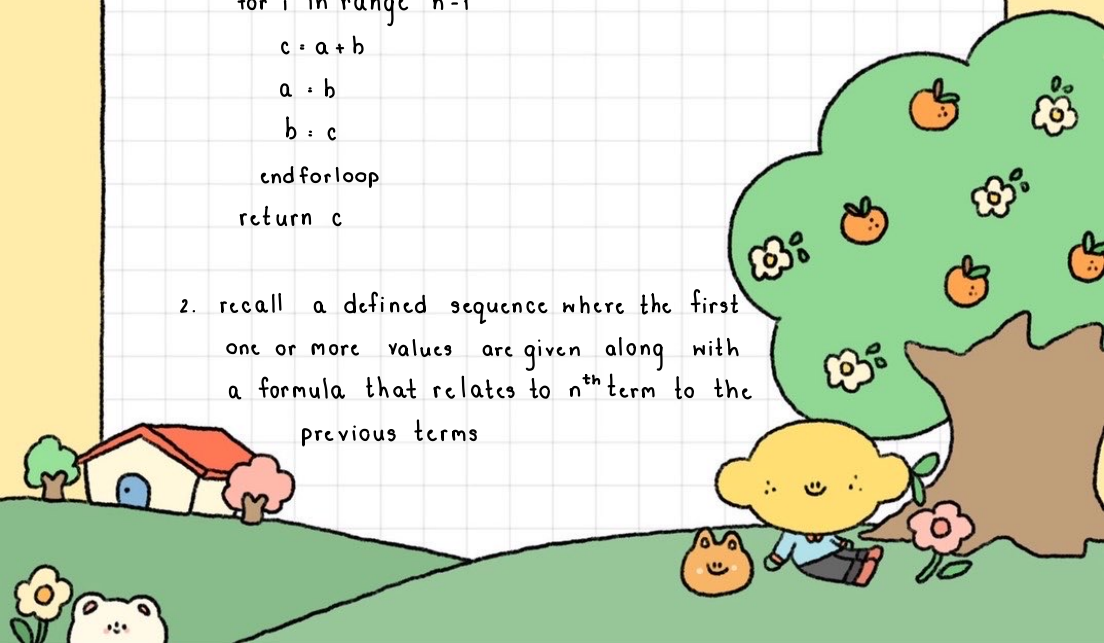
1, 1, 2, 3, 5, 8, 13, 21, 34, …

fib(n) returns Fibonacci number at the n^{th} position; e.g, fib(1) = 1 and fib(4) = 3.

1. write an iterative algorithm (using pseudocode) of fib(n)
2. write recursive definition of fib(n)
3. write a recursive algorithm (using pseudocode) of fib(n)
4. implement algorithms in 1 and 3 in Python
5. run your programs against a number of n's. record running time and draw graph (plot running time of both Fibonacci versions on the same graph)
6. explain what you found

1.
```
input n
a = 0
b = 1
if n = 1
  then return 1
endif
for i in range n-1
    c = a + b
    a = b
    b = c
  endforloop
return c
```

2. recall a defined sequence where the first one or more values are given along with a formula that relates to n^{th} term to the previous terms



3. input n

```
fib(n)
 if n less than or equal to 1
    then return n
 else
   return fib(n-1) + fib(n-2)
```

4, 5. code. fibograph.ipynb

6. recursive take more time to run the program than the iterative program.