math.sqrt(x) => return as a FLOAT
create a list within given range: list(range(0, 100, 2) => [0, 2, 4, …, 98]
create a list within a list: [list(range(0, 100, 2)) => [[0, 2, 4, …, 98]]
List.count(VAL) => count the number of VAL in the List
List.index(1) => Return the first-appear 1 in the List
List.remove(VAL) => remove the first-appear val in the List, AND will cause the latter values move 1 index towards
List.pop(index) => remove the value in the list given its index, AND will cause the latter values move 1 index towards

List.copy() / list(List) / list[:] => assign new memory
List2 = List / List2 = [List] / function calls => share same memory

List slicing:
IE: x = [1, 2, 3, 4]

1. Print(x[0:100]) => go through the last position of the List
2. Print(x[::2]) => start at the START of the list
3. Print(x[::]) => will always slice from the left to right unless is defined
4. If START >= END, return []

convert a string to a list (every character as an element)
str = "1234"
List = list(str) => List = ['1', '2', '3', '4']
Join every element in the List by the given object, then become a String
List1 = [1, 2]
Str = '-'.join(List1) => "1-2"
Str.find(sub, [start, [end]]) => find the substring in the String. If not found, return -1
No return value in str.sort()
List.insert(index, obj) => will move the objects in current and latter indexes for 1 (when index > len, always intert to the last position)
List = s.split(' ') => Breaking String into List
letter = char(ascii) => Convert the ascii code into its corresponding letter
str = str.title() => a string converted all FIRST letters and LETTERS BEHIND "-" become capital
str.rjust(len, char) => return a string filled its left side by CHAR when its length < len
str.ljust(len, char) => (same as above but filled its RIGHT side)
Both will return the original str when the length of str > len
str.center(n) => place the str at the center of "n spaces"
str.count('a') => count the num of 'a' in the str

Common ascii:
Num (0 – 9): 48:0 => 57:9
Capt (A – Z): 65:A => 90:Z
Lower(a – z): 97:a => 122:z

Open static webpage:
Import urllib.request
Word_rul = http://
Word_file = urllib.request.urlopen(word_url) => same as f = open('filename') now

Function execution:
Def one():
    Return True
Def two():
    return False
Print(one() or two()) => will only execute the one(). Because in OR operation, program will stop process once a True is found. i.e. one() is True
Print(one() and two()) => will execute both
Print(two() or one()) => will execute both

File:
[readline() => return a single line, readlines() => return lines as a list, read() => return the whole file as a string]
F = open('filename') => open a file
F = open("filename', 'w') => overwrite the file if previously existed
F = open("filename', 'a') => keep the content of current file and append new content to the file
F.write("hello world") => to actually write the file
Access the file:
For line in open("filename"):
REMEMBER: to close() the file after the program finished.

Set: [sorted(set())] => will convert to a sorted list
Define a set: set = {}
Function:
Set.add(x) => add an element if it is not existed
Set.discard(x) => remove an element form the set, otherwise won't change anything
Set.clear() => empty the set
s1 - s2 [s1.difference(s2)] => create a new set with values in s1 that are not in s2
s1 & s2 [s1.intersection(s2)] => create a new set with values in both s1 and s2
s1 | s2 [s1.union(s2)] => create a new set with values in either s1 or s2 (can also be used to act as .add() by s1 |= s2)
s1 <= s2 [s1.issubset(s2)]=> boolean: whether all values in s1 are also in s2
s1 >= s2 [s1.issuperset(s2)]=> boolean: whether all values in s2 are also in s1
s1 ^ s2 [s1.symmetric_difference(s2)]=> create a new set that in either s1 or set but not both
Dict: [val in dict = val in dict.keys()]
define a dict: dic = dict()
functions:
dict.clear() => clean the dict, no return value
dict.get(key) => return the value in the specific key. Compare to [key], this can return None when not found
dict.pop(key) => return the item from the dict, delete the key & value in Dict given its key
dict.update(dict2) update the keys in Dict correspond to the values in dict2. If keys in dict2 is not found in Dict, create on at the last position of Dict.
dict.keys() => return all the keys in dict
dict.values() => return all the values in dict
obj in Dict => return a boolean whether this KEY is existed in Dictionary, WILL return False if only value exist
obtain key by value:
maxVal = max(list(count.values()))
maxPos = list(count.values()).index(maxVal)
maxKey = list(count.keys())[maxPos]

Class:
def …(self, other):
    self … => this will modify the existing object
def __add__() => have to be define in the class
    return Point2d.__add__(…) => this will create a new object
Point2d.__add__(…).x => to get the attribute

random.randint(0, 1)

```
import doctest
doctest.testmod()
'''
>>> func()
True
expected: True
Got sth…
```

magic method: __add__ (self, other)=> + __sub__ (self, other)=> - [Both of them will take the fst parameter as self, and sec as other]

def __init__(self, year=1900, month=1): => default value as the object doesn't define these attribute
    self.year = year => if only N attributes are defined, only the left N attributes will set as define & the rest
    self.month = month remain default

```
def biSearch(x, L):
    low = 0
    high = len(L)
    while low < high:
        mid = (low + high) // 2
        cur = L[mid]
        if x > cur:
            low = mid + 1
        else:
            high = mid
    return low
```

Tkinter:
from tkinter import *
root = Tk()
➔ the objects in the GUI will be formed here
root.mainloop()
⇨ the code will run after the GUI is closed

Parameters in Button()
[when initialization]
- frame_name
- text
- command (process functions)
- padx (padding in left and right)
- pady (padding in top and down)
- width

After initialization as an Object, the button can be changed by Object.configure()

pack() => display the object
may contain: side

```
def insSort(L): => N^2
    for i in range(1, len(L)):
        x = L[i]
        j = i - 1
        while j >= 0 and x[j] > x:
            L[j+1] = L[j]
            j -= 1
        L[j+1] = x
        return L

def merge_sort_interactive(L):  ➔ nlog(N)
        L1 = []
        for item in L:
            L1.append([item])
        while len(L1) > 1:
            L2 = []
            for i in range(0, len(L)-1, 2):
                merged = merge(L1[i], L1[i+1])
                L2.append(merged)
            if len(L1) % 2 == 1:
                L2.append(L1[-1])
            L1 = L2
        return L1[0]

def merge_sort_rec(L): => nlog(N)
        if len(L) == 1:
            return L
        length = len(L)
        mid = length // 2
        left = merge_sort_rec(L[:mid])
        right = merge_sort_rec(L[mid:])
return merge(left, right)
```

list(map(func, list)) => apply FUNC to every object in the list, return as a new list

lambda function: use the RHS equation to calculate a value and return the result
    list(map(lambda x: x ** 2, range(10))) => [0, 1, 4, 9 …]

filter(bool func, list) => apply each object in List to the bool function. If return False, remove it from the List
    list(filter(lamda x: x >0, List) => create a list that removed all non-positive number from the List

key in sorted function:
sorted(pts, key=lambda p:(p[1], p[0]))
as pts is a 2D list, sort it by p[1] then p[0] rather than p[0] then p[1]
key can also be a function that will return a tuple, which compares the [0] position first then [1]
the function has a single parameter e.g list gonna be sorted.

List comprehension:
create a new list:
[(i, j) for I in range(1, 5) for j in range(1, 5) *if i != j*]
(create a list of tuples that keep when i != j)
I: Boolean statement to keep values

create a list from the current list:
[x * x for x in List if x > 0]
create a new list that contain only x in List is > 0 and then multiply itself

all the embedded functions are no need to add "()", just call it by their name

```
Fibonacci:
def fab(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return fab(n-1) + fab(n-2)
```

```
Recursion
normal recursion:
def func():
        return func()
recursion in Class:
class obj(object():
        def func():
            self.func()
```