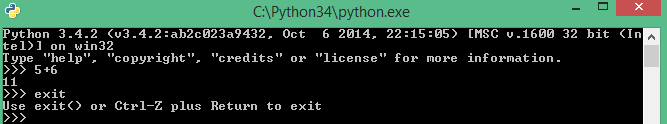
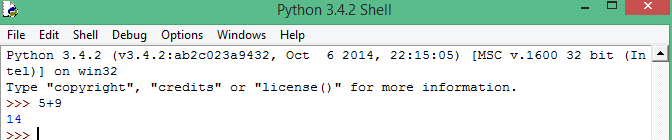
Part - 1.

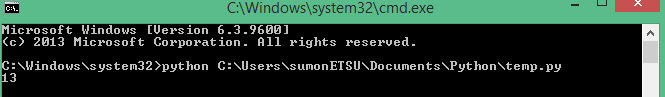
a) Problem 1



Any one of problems 2-4. I did number 2 . It will execute an statement. First, add 1 and 2 then print the result 3.



Problem 5



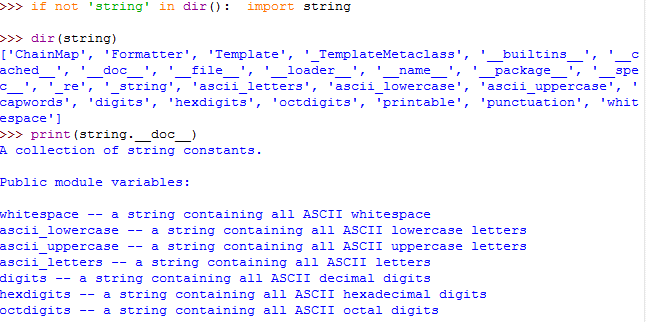
Part - 2 section 3.

a) For each i, it can calculate '2\*\*i' , which is very big number if i is very big number and takes log10 of that big number without any overflow and returns a floating point number then ceil function convert this floating point to the nearest maximum integer.

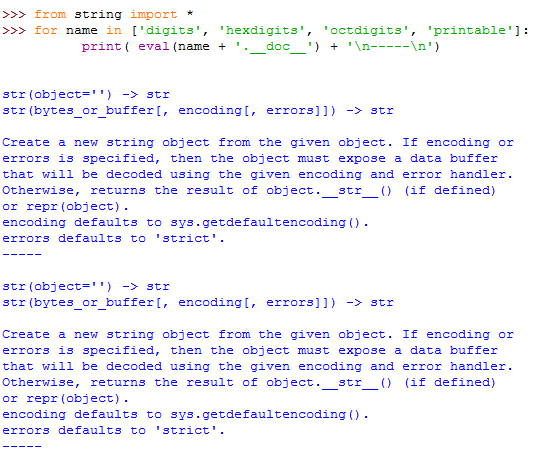
b) if list is empty, python consider it as false. Line 279 returns false. If list is not empty, python consider it true. Line 280 returns true.

c) Adding one with big number such as 'sys.float\_info.max' does not make any change and display almost same number. But when double the 'sys.float\_info.max' number, it goes beyond the range that's why it displays infinity.

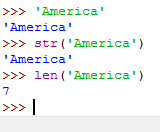
d) In python, int is considered as very big integer. Python can add, multiply big number without any overflow. Big number addition , multiplication are done as string, \_\_str\_\_.

e) 

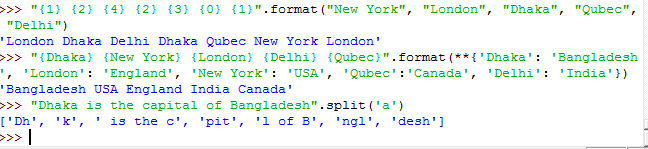
f)



g)



h)



i) First parameter is the starting index, second parameter indicate ends, third parameter is the step. If first parameter is give, it will start from that index otherwise from 0. If second parameter is given, it will end at that point minus one otherwise it will go until end. If step is given, it will escape that number of character. If no parameters are given, it will start from default(0) and will go until end. For the reverse loop, If first parameter is negative index, it will start from that index otherwise from -1, count from opposite site. If second parameter is negative index, it will from 0 and will end in len- index. If negative step is given, it will escape that number of character from opposite direction's.

j) Given the following loop,

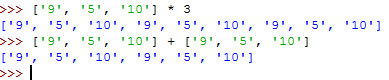
for i in range(…… ): print(i, end=" ")

else: print()

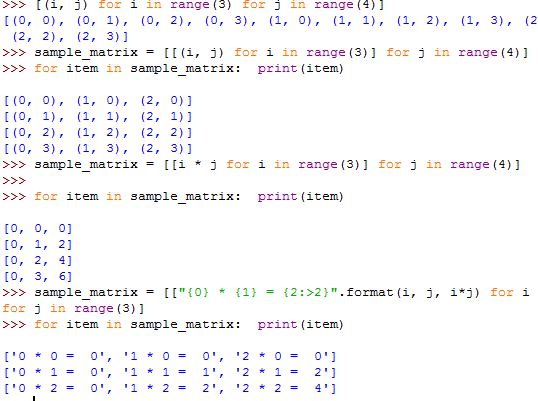
what range expressions can I use to generate the following outputs:

* 0 1 2 3 4 5 6 7 8 9 (ans: (0,10))
* 2 3 4 5 6 7 8 9 (ans: (2,10))
* 2 4 6 8 10 12 (ans:(2,13,2)
* 3 1 -1 -3 -5 -7 (ans:(3,-9,-2)

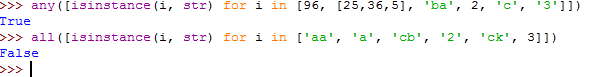
k)



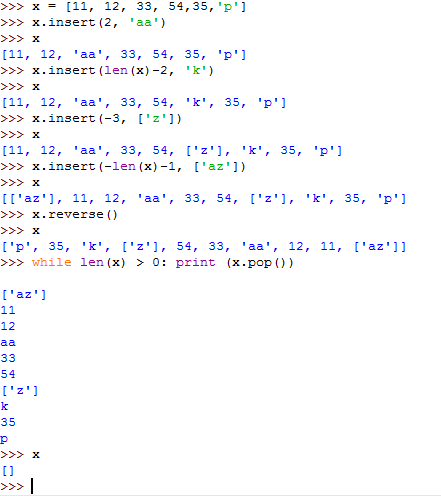
l)



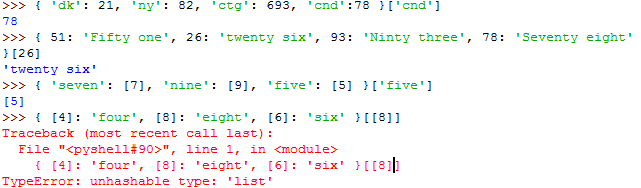
m)



n)



o)



p) 'for c in ('string')' , here for loop iterates through one by one character. Whereas, 'for c in('string',)', here range is a tuple with only one element.

q)

{1: (2,)}

{(1,): 2}

{1: [2]}

{[1]: 2} Ans: List cannot be key of map/dictionary

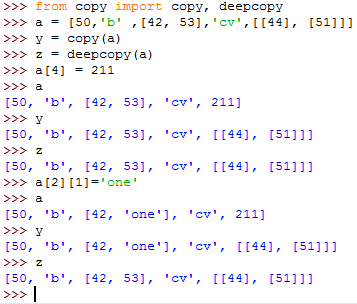
{1: {1:2}}

{{1:2}: 1} Ans: Dictionary/map cannot be key of map/dictionary

{{1}} Ans: Set cannot be key of map/dictionary

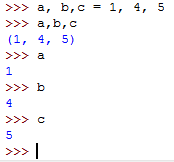
Part - 3 Section 4

a)



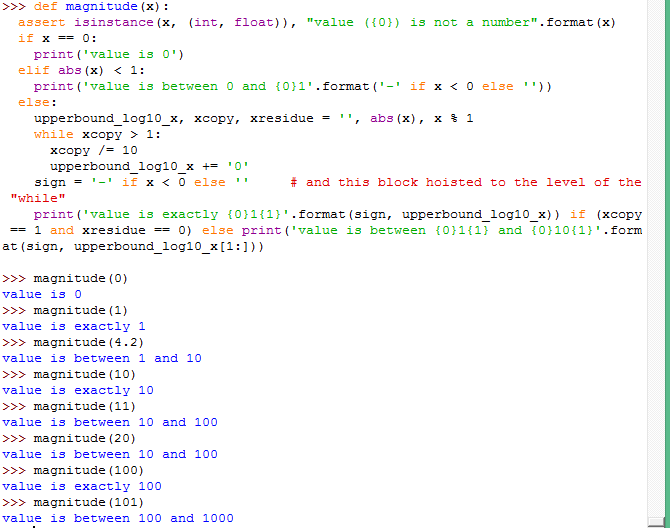
Deepcopy() creates a new copy of original one. If any change happens in original one, it does not affect the copy one. Opposite is true in 'Copy()'. If any change happens in original one, it affects the copy one values. In 'Copy()', both refers to same object.

b)

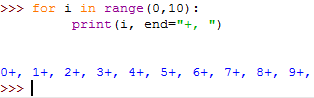


c) No. By del \_\_builtins\_\_ command, it deletes \_\_builtins\_\_ from environment. By using import command, it is not possible to import in environment.

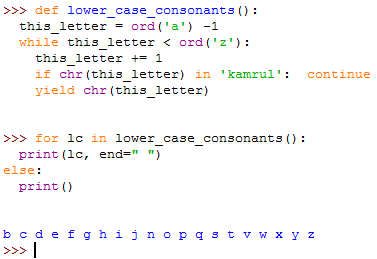
d)



e) Yes.



f)



g) Fib returns memory object reference. List comprehension creates output list by accessing those reference.

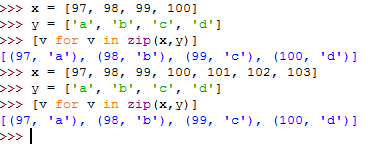
h) Fib() is called with no argument, as a result the implementation, def fib(val\_count = float('inf')), will take infinity as a default value. As a result, output fails.

i) In the range of for loop, len gives initial x length. One by one x elements are removed but len still hold same initial value. As a result, Index go out of range and throw exception.

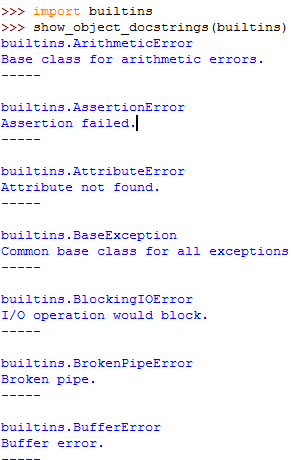
j) All the exceptions would be handled by superclass 'Exception', no subclass(ArithmeticError, FloatingPointError) will catch any exception to handle.

4. Section - 5

a)

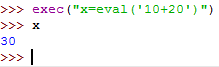


b)

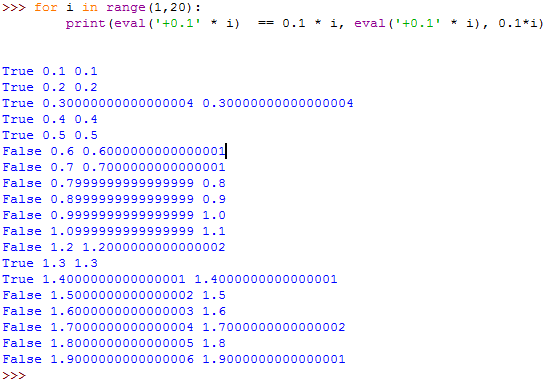


c)

5_c_1.PNG



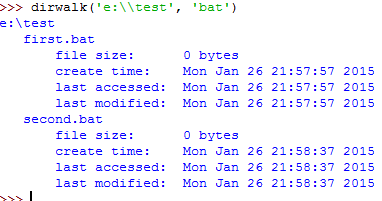
d)



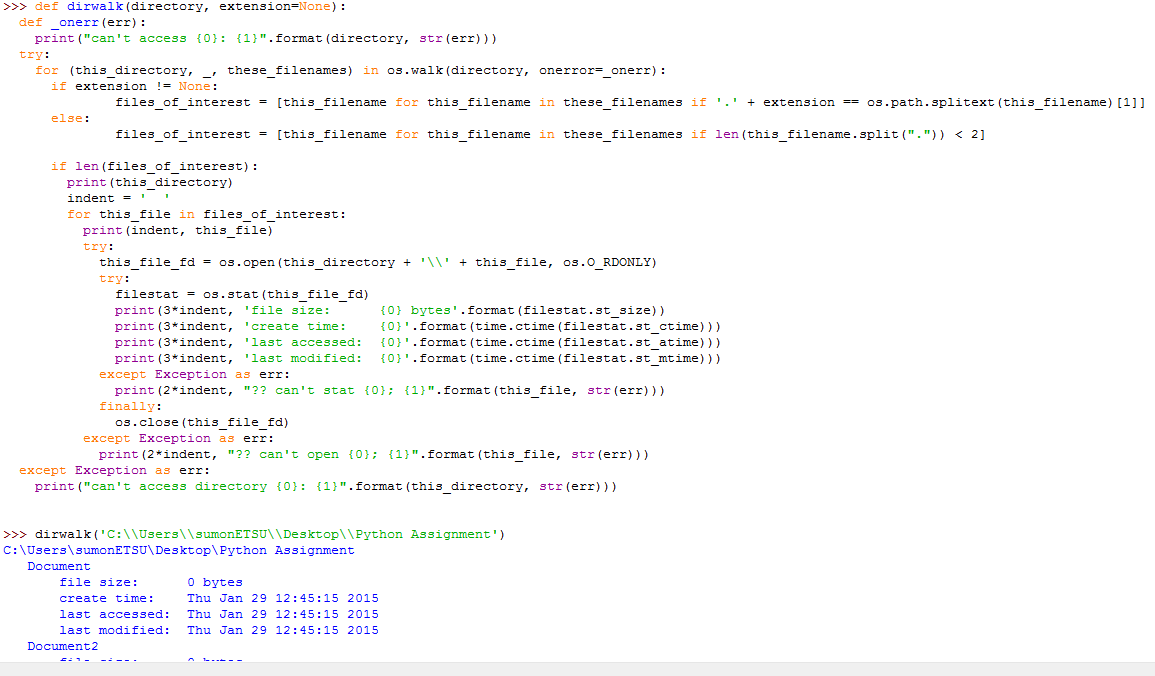
True or false depends on how floating points are represent in binary. When there is floating multiplication, result might be bigger. There is more prone to error comes. When there is floating point multiplication, programmer should be very careful.

In True case, the values of evaluation of expression and multiplication gives same result. In False case, the values of evaluation of expression and multiplication gives not same result.

e)



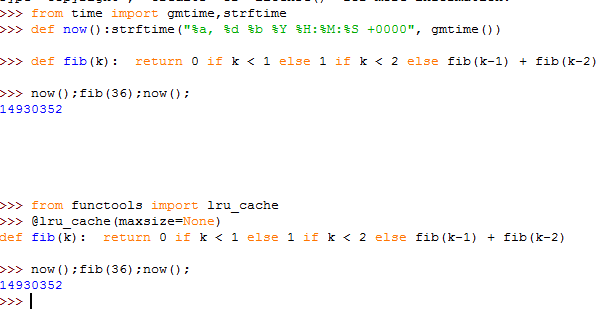
f)



g) f() output nothing because it returns nothing(empty) but print(f()) outputs 'None'.

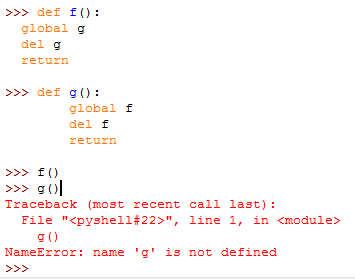
h) k\_pairs first parameter is function 'pair', second parameter is calling 'pair' function with two argument that return tuple, third parameter is number. First, pair(1,2) is called that returns tuple (1,2). Then k\_pair(pair,(1,2), 1) is called with three arguments that returns a list, [(< function pair at 0x04bABG>, (1,2))]. Then it gets function name 'pair' from list with index [0][0] then it call that function with two parameters like pair(7,8).

i)

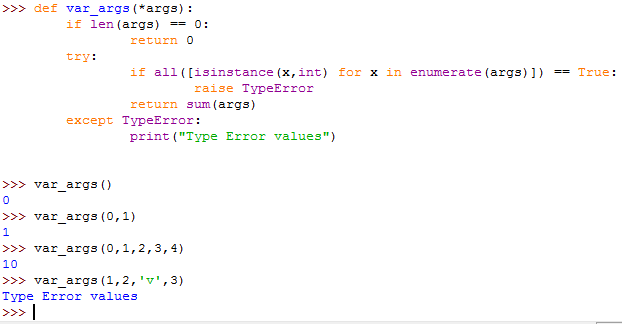


Second one takes less time. It's may be because of cache.

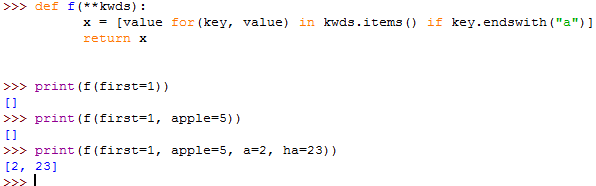
j)



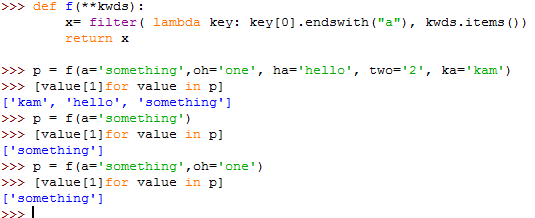
k)



l)



m)



Part 5 - Section 6

a)

5_a.PNG

b)

