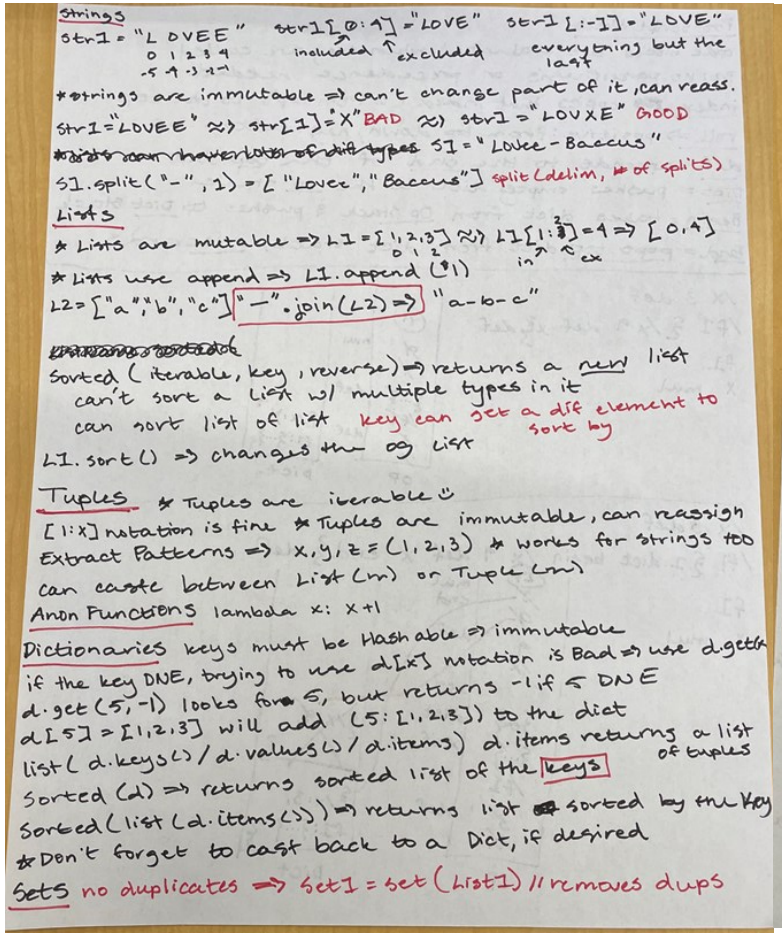

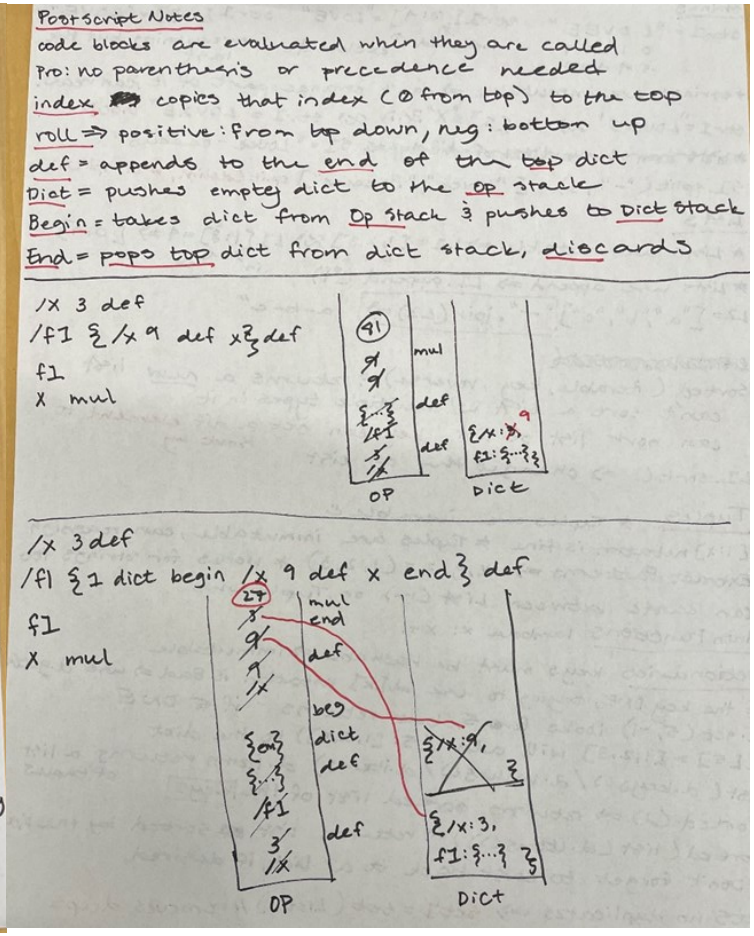


```
## problem 4a) searchDicts(L,k)
def searchDicts(L,k):
    for d in reversed(L):
        dKey = list(d.keys())
        #check if k is one of the keys in the dictionary
        if (dKey[0] == k or d.get(k)):
            return d.get(k)
    return None

## problem 4b) searchDicts2(L,k)
def searchDicts2_helper(tL,k,ind):
    if k in tL[ind][1]:
        return tL[ind][1][k]
    else:
        if ind == 0:
            return None
        else:
            next_ind = tL[ind][0]
            return searchDicts2_helper(tL,k,next_ind)
```

```
#1
log = {'Mar':2706,'Apr':3620,'May':1860,'Jun':2157,'July':5014,'Aug':4327,'Sep':2843}
from functools import reduce
sum_log = lambda log: reduce(lambda x,y: x+y, log.values())

#2
tlog = ('King', {'Mar':2706,'Apr':3620,'May':1860,'Jun':2157,'July':5014,'Aug':4327,'Sep':2843})
('King', 22527)
helper = (lambda tlog: (tlog[0], sum_log(tlog[1])))
map( lambda tlog: (tlog[0], sum_log(tlog[1])), monthlyCases.items() )

#3
alltuples = [('Mar', 4614), ('Apr', 5959), ('May', 3152), ('Jun', 4185), ('July', 11487), ('Aug', 9281), ('Sep', 7403), ('Oct', 278)]
helper = lambda x,y : x if x[1] > y[1] else y
helper(('Mar', 4614), ('Apr', 5959))
reduce(helper, alltuples)
```

3) [20 pts]
(a) [8 pts] Define a Python function, `aroundL`, that takes a list of integers as input and returns a list of pairs containing one more and one less than each number in the list. For example, `aroundL ([1,2,3])` should return `[(0,2),(1,3),(2,4)]` as its answer. Your function may involve a loop or can be recursive.

```
def aroundL(L):
    result = []
    for item in L:
        result.append((item-1, item+1))
    return result
```

(b) [6 pts] Re-write the `aroundL` function from problem 3(a) using list comprehension.

```
def aroundL(L):
    return [(item-1, item+1) for item in L ]
```

(c) [6 pts] Re-write the `aroundL` function from problem 3(a) using high order functions (map, reduce, or filter).

```
def aroundL(L):
    return list(map(lambda x: (x-1,x+1),L))
```

4) [16pts]
(a) [10 pts] Define a Python function `combine` that takes two dictionaries as argument and combines them by merging values of the common keys as lists. The output dictionary includes only the common keys from the two input dictionaries. The order of the elements in the output dictionary can be arbitrary.
Your solution may use loops.
For example:

```
combine({1:'A',2:'B',3:'C',4:'D',5:'E',6:'F'},
        {5:50,1:10,6:60,2:20,7:70,9:90} )
```

returns

```
{1: ['A', 10], 2: ['B', 20], 5: ['E', 50], 6: ['F', 60]}
```

```
def combine(d1,d2):
    d = {}
    for k,v in d1.items():
        if k in d2:
            d[k] = [d1[k],d2[k]]
    return d
```

(b) [10 pts]
Re-write the `combine` function from problem 4(a) using high order functions (map, reduce, or filter).

```
def combine(d1,d2):
    filtered = filter(lambda t: t[0] in d2, d1.items())
    return dict(map(lambda x: ( x[0] , [ x[1], d2[x[0]] ]),filtered))
```

1) [15 pts] Directions: Consider the following Python code.

```
def add(a):
   global x
   x = a+x
   return x

def grow (c):
   return x+c

def outer (f):
   def inner (g, y):
      return f(y)+g(y)
   return inner
```

For each of the following Python expressions, write the valu
that the expressions are evaluated in a new session (i.e., ans
previous lines)

a)

```
x=1
L=[1,2,3,4,5]
list(map(grow,L))
```

```
[2, 3, 4, 5, 6]
```

b)

```
x=1
L=[1,2,3,4,5]
list(map(add,L))
```

```
[2, 4, 7, 11, 16]
```

c)

```
x=1
outer(add)(grow,5)
```

```
17
```

d)

```
x=2
outer(add)(grow,x)
```

```
10
```

e)

```
x=2
outer(grow)(add,x)
```

```
8
```