# PART 2A (4 pts)

In one line of code, change the value of numbers using a Python method, such that its value is now \((3,4), (1,2,3,4), (7,5)\).

```python
numbers = ([3, 4], [1, 2, 3], [7, 5])
result = numbers[1].append(4)  
OR
result = numbers[-2].append(4)
```

# PART 2B (4 pts)

Write the command that uses indexing of \(z\) and concatenation to create the string 'pa':

```python
z = ('blue', 'tan', 'sip', 'rod')
```

There are several solutions in the format:

\[
\text{result} = z[a][b] + z[c][d]
\]

ACCEPTABLE VALUES FOR:

- \(a\): -2 or 2
- \(b\): -1, 2
- \(c\): -3, 1
- \(d\): -2, 1

Examples:

\[
\begin{align*}
\text{result} &= z[2][2] + z[1][1] \\
\text{result} &= z[2][2] + z[1][1] \\
\text{result} &= z[2][-1] + z[1][1] \\
\text{result} &= z[-2][2] + z[1][1] \\
\text{result} &= z[-2][-1] + z[1][1] \\
\text{result} &= z[2][2] + z[1][-2] \\
\text{result} &= z[2][2] + z[-3][1] \\
\text{result} &= z[2][2] + z[-3][-2] \\
\text{result} &= z[2][2] + z[1][1]
\end{align*}
\]

# PART 2C (4 pts)

Write the command that uses set operations to determine what \(s1, s2, \text{ and } s3\) have in common (that is, \(\{20\}\)):

```python
s1 = {5, 10, 15, 20} 
s2 = {10, 20, 30, 40} 
s3 = {20, 40, 60, 80}

SOLUTION:

s1&s2&s3
```

# PART 2D (4 pts)

Write the command that adds 3 to each element from \(lst\) and stores those numbers to create
the list named result that contains [3,4,5,8,10]:

lst = [5,2,7,1,0]

SOLUTION:

sorted(num+3 for num in lst)
OR
sorted([{num+3 for num in lst}])

NOTE: “num” may be anything they specify

# PART 2E (4 pts)
# Given the tuple a, write the command that uses indexing of a to create the list named result that contains [[3, 4], [7, 5]].

a = ((3,4), [1,2,3], [7,5])

SOLUTION:

result=[a[0], a[2]]

NOTE: a[-3] ACCEPTABLE IN PLACE OF a[0]
NOTE: a[-1] ACCEPTABLE IN PLACE OF a[2]

# PART 2F (4 pts)
# Given the dictionary d, use Python methods to create the list named result as a sorted list of the dictionary’s keys. In this case, result would be ['b', 'd', 'l', 'm', 't'].

d = {'t':13, "b":4, "l":21, "m":1, "d":3}

SOLUTION:

result=sorted(d.keys())

Problem 3 (8 points)

Given a list of integers, create the function called sortNumbers that takes one parameter: values is a list of integers. There must be at least one integer stored in values. The function sortNumbers determines the integer remainder of each element divided by 4 and returns a list of those remainders in descending order.

This function must use a list comprehension in solving the problem.

def sortnumbers(values):
# Calculate lambda function then use it in list comprehension
inc=lambda x: x%4
lst3=sorted([inc(num) for num in values], reverse=True)
return lst3

OR

def sortnumbers(values):
    # Calculate lambda function then use it in list comprehension
    lst3=sorted(num%4 for num in values], reverse=True)
    return lst3

Problem 4A (6 points)

#Part A (2 points)

What is the value of lst4 when the program completes?

SOLUTION:


#Part B (2 points)

The value of lst4 was supposed to be the following:


What line(s) of code is/are producing this error?

SOLUTION: Lines 2 and 3 create the issue, because they should be reversed.

#Part C (2 points)

How would you modify the line(s) of code identified in part b) to make this provide the correct output?

SOLUTION: Change the index of x in line 2 to x[1] and line 3 to x[0].

ALTERNATE SOLUTION: Delete lines 2-4 and replace with the following
result2 = sorted(sorted(sorted(lst, key=lambda x: x[1]), key=lambda x: x[0]), key=lambda x: x[2], reverse=True)
Problem 4 PART B (6 points)

a) (2 points)

What lines of code create this error?

SOLUTION: Lines 2 and 3

b) (2 points)

What is the reason for the error(s)?

SOLUTION: The two lines are trying to change the values inside the tuples and tuples are immutable.

c) (2 points)

Assuming you delete the lines of code you identified in part b, what one line of code can replace them to correctly create a new entry for ('BIO/CS', 100) in the list?

SOLUTION: majors.append(('BIO/CS', 100))

Problem 5 Part A (14 points)

We are interested in identifying the state (or DC) with the most historically Black colleges and universities (HBCUs). Create the function countSchools that has one parameter hbcus, which is a list of strings, where each string is school information representing the name of the school and the state the school is located, in the format 'School:State'.

Assume the list hbcus contains at least one school. The function returns the state with the most HBCUs. Assume (for simplicity), that there will only be one state with the highest number of HBCUs.

def countSchools(hbcus):
    states={}
    for entry in hbcus:
        school=entry.split(':')
        if school[1] in states.keys():
            states[school[1]]+=1
        else:
            states[school[1]]=1
    max_state=max(states.keys(), key=lambda x: states[x])
    return(max_state)
Krispy Kreme has a fundraiser competition to see which university in Durham and Chapel Hill (Duke, NCCU, and UNC) sells the most boxes. Each university provides weekly updates to Krispy Kreme on the total number of boxes sold. At the end of the program, the university that sold the most boxes is the winner.

Create the function topseller that takes two parameters. The parameter orders is a list of tuples of all weekly totals for all schools, where each tuple is a string and an integer in the format ('School', # boxes). The parameter topNum is an integer that is the minimum number of boxes to sell to be considered a top-selling school.

The function then determines and returns a sorted list of all schools that sold at least topNum boxes.

Constraint: Assume the list orders has at least one school with at least one box sold.

def topSeller(orders, topNum):
    totals={}
    for entry in orders:
        school=entry[0]
        boxes=entry[1]
        if school in totals.keys():
            totals[school]+=boxes
        else:
            totals[school]=boxes

    #Create list to store schools with at least topNum sold
    lst=[]
    for key, value in totals.items():
        if value>=topNum:
            lst.append(key)
    return lst