Week-5, Activity

Theory

Problem-1

What is the output of the following code-snippet? Explain.

```python
foo()

def foo():
    print("Please move me to the top.")
    print("If you wish to call me.")
```

Answer

```
NameError: name 'foo' is not defined
```

- Python executes commands from the top-bottom.
- Therefore, function definition should precede the function call.

Problem-2

Execute the following snippet of code. What happens? Explain.

```python
def foo(x):
    return x ** 2

def foo(x):
    return x ** 3

print(foo(2))
```

Answer

Output:

```
8
```

- Python executes commands from the top-bottom.
- The second function definition overrides the first.

Problem-3

What is the output of the following code-snippet? Explain.

```
1 def minmax(a, b):
2     if a < b:
3         return a, b
4     return b, a
5 
6 x, y = minmax(2, 3)
7 print(x, y)
```

Answer

Output

```
2 3
```

- Functions can return multiple values.
- Consider the assignment statement on line-6.
- The number of variables on the LHS of the function call must be equal to the number of variables returned inside the function's body.

Problem-4

What is the output of the following code-snippet? Explain.

```python
def first():
    second()
    print('first')

def second():
    third()
    print('second')

def third():
    print('third')

first()
```

Answer

```
third
second
first
```

- When the function `second` is called from inside the function `first`, `first` is suspended.
- When the function `third` is called from inside `second`, `second` is suspended. At this stage, two function calls are in suspended state.
- So, the first string to be printed is `third`.
- Then, the flow of control transfers to the most recent suspended function, which is `second`. Now, the string `second` is printed.
- Finally, the string `first` is printed.

Problem-5

Consider the following code. Why do you think the name of the function is `triangular`?

```python
def triangular(n):
    if n == 1:
        return 1
    return triangular(n - 1) + n

N = int(input())
print(triangular(N))
```

Execute it for the following inputs:

- 10
- 1000

Now, use the following code snippet to run the code. What do you think has changed?

```python
# sys is an important library in Python
import sys
# Search the web to find out what setrecursionlimit does
sys.setrecursionlimit(5000)

def triangular(n):
    if n == 1:
        return 1
    return triangular(n - 1) + n

N = int(input())
print(triangular(N))
```

Answer

- The name of the function is `triangular` because it is computing triangular numbers.

First code block

- There is no problem with `triangular(10)` in the first code block. For `triangular(1000)`, the interpreter throws a `RecursionError`.
- This has to do with the concept of recursion depth, which is nothing but the maximum number of levels in the recursion. In other words, the recursion depth is the number of times a function is allowed to call itself recursively. Note that the adverb "recursively" at the end of the previous sentence is an important qualifier.
- The recursion depth or recursion limit usually default to 1000 in most systems.

Second code block

- `sys` is a library in Python. Among other things, it can be used to modify the recursion depth.
- `sys.setrecursionlimit(5000)` sets the limit to `5000`. Now, if this code is executed with an input of `1000`, the interpreter will not throw any error. Any input in excess of the recursion limit will still be a problem.

Problem-6

Consider the following recursive program:

```python
def power(x, n):
    '''Return x raised to the power n'''
    if n == 0:
        return 1
    if n % 2 == 0:
        root = power(x, n // 2)
        return root * root
    else:
        return x * power(x, n - 1)

power(2, 4)
```

How many times is the function `power` called when the above code is executed? In general, can you edit the above code so that it counts the number of times `power` is called?

Answer

```python
def power(x, n):
    '''Return x raised to the power n'''
    global count    # keeps track of number of function calls
    count += 1
    print(f'Call number {count}: power({x}, {n})')
    if n == 0:
        return 1
    if n % 2 == 0:
        root = power(x, n // 2)
        return root * root
    else:
        return x * power(x, n - 1)

count = 0
power(2, 4)
print(count)
```

Programming

Problem-7

In a throwback to CT days, recall the functions that we defined for lists in CT. Write down the Python equivalents of these functions.

- `first(L)`
- `last(L)`
- `init(L)`
- `rest(L)`
- `member(x, L)`

Do not use list methods.

Answer

```python
def first(L):
    return L[0]

def last(L):
    return L[-1]

def init(L):
    return L[:-1]

def rest(L):
    return L[1:]

def member(x, L):
    return x in L
```

Problem-8

Write a function `insert` that accepts a sorted list `L` of numbers and a number `x` as input. The function should return a sorted list with the element `x` inserted in the input list at the right place. The original list should not be disturbed in the process.

Answer

```python
def insert(x, L):
    '''insert x in L'''
    out_L = [ ]         # store the output in this list
    inserted = False    # initially we haven't inserted x
    for elem in L:
        # check if we have already inserted x
        # if we haven't, then check if the time is right to insert x
        if (not inserted) and (elem > x):
            out_L.append(x)
            inserted = True     # don't forget to update this flag
        # elem should be appended at every iteration
        out_L.append(elem)
    # the following line checks two corner cases
    # empty list; x is greater than all elements in L
    if (not inserted):
        out_L.append(x)
    return out_L
```

Problem-9

Write a function `isort` that accepts an unsorted list `L` of numbers as input and returns a sorted list. The original list should not be disturbed in the process. Use the insertion sort procedure that you studied in week-5 of the CT course. Also, make use of the `insert` function defined in the previous problem.

Answer

```python
def isort(L):
    out_L = [ ]
    for elem in L:
        # keep inserting elem in the right place
        # shows the power of functions
        # very easy on the mind
        out_L = insert(elem, out_L)
    return out_L
```

Problem-10

Write a function that accepts a square matrix A as input and returns `True` if it is an identity matrix and `False` otherwise. An identity matrix has ones on the main diagonal and zeros everywhere else. For example, the 3×3 identity matrix looks like this:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Answer

```python
def is_identity(A):
    dim = len(A)
    for i in range(dim):
        for j in range(dim):
            # this condition is for diagonal elements
            if i == j and A[i][j] != 1:
                return False
            # this condition is for non-diagonal elements
            elif i != j and A[i][j] != 0:
                return False
    return True
##### Input part #####
A = [ ]
dim = int(input())
for i in range(dim):
    A.append([ ])
    for num in input().split():
        A[-1].append(int(num))

print(is_identity(A))
```

Problem-11

Write a function that accepts a square matrix A and a positive integer n as input and returns A^n.

Answer

```python
def zero_matrix(dim):
    Z = [ ]
    for i in range(dim):
        Z.append([ ])
        for j in range(dim):
            Z[-1].append(0)
    return Z

# basic idea: A^n = A^{n - 1} * A
# a simple recursive function
def A_power_n(A, n):
    # base case
    if n == 1:
        return A
    # recursive call to compute A^{n - 1}
    P_one_less = A_power_n(A, n - 1)
    dim = len(A)
    # create a zeor-matrix to store the product
    P = zero_matrix(dim)
    for i in range(dim):
        for j in range(dim):
            for k in range(dim):
                P[i][j] += P_one_less[i][k] * A[k][j]
    return P

##### Input Part ######
dim = int(input())
A = [ ]
for i in range(dim):
    A.append([ ])
    for num in input().split():
        A[-1].append(int(num))
n = int(input())

print(A_power_n(A, n))
```

Problem-12

Consider two sequences of integers — X and Y — of equal length. Let the length of both sequences be L. We say that sequence X is "greater" than sequence Y if $X_i > Y_i$ for every value of i in the range $1 \leq i \leq L$. Write a function that accepts two sequences as input and returns `True` if the first sequence is greater than the second sequence, and `False` otherwise.

Answer

```python
def greater(X, Y):
    L = len(X)
    for i in range(L):
        if X[i] <= Y[i]:
            return False
    return True
```

Problem-13

Write a function named `swap` that accepts the following inputs:

- `M` — matrix
- `i` — integer
- `j` — integer
- `dim` — integer

Consider the following scenarios:

- If `dim` is equal to 0, it should swap rows `i` and `j` in-place and return the matrix `M`.
- If `dim` is equal to 1, it should swap columns `i` and `j` in-place and return the matrix `M`.
- If `dim` is not specified, then it should default to 0.

You must not create a new matrix within your function at any point of time.

```python
# useful to write a function like this
# can be written once, called multiple times
def print_matrix(M):
    dim = len(M)
    for i in range(dim):
        for j in range(dim):
            if j != dim - 1:
                print(M[i][j], end = ' ')
            else:
                print(M[i][j])

def swap(M, i, j, dim = 0):
    # swap rows
    if dim == 0:
        M[i], M[j] = M[j], M[i]
    # swap columns
    else:
        for k in range(len(M)):
            M[k][i], M[k][j] = M[k][j], M[k][i]
    return M

M = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Modeling

Problem-14

n tennis players participate in a round-robin tournament in which each player plays exactly one game against every other player. How would you represent this data in Python? Using this representation, write the following functions:

- `winner(results)`: accepts results of the tournament as input and declares the winner; the player who has the maximum number of wins is the winner
- `rank(results)`: accepts results of the tournament as input and ranks the players; the winner occupies the topmost rank

What are some other questions that we can ask in this scenario? You must be able to answer these questions using Python.

Answer

We can represent this as a matrix M. $M[i][j] = 1$ if player i beats player j and is 0 otherwise.

```python
import random

# A useful function to visualize a matrix
def print_matrix(M):
    dim = len(M)
    for i in range(dim):
        for j in range(dim):
            if j != dim- 1:
                print(M[i][j], end = ' ')
            else:
                print(M[i][j])

# A function to initialise a zero matrix
def zero_matrix(n):
    A = [ ]
    for i in range(n):
        A.append([ ])
        for j in range(n):
            A[-1].append(0)
    return A

# Simulate the tournament between the players
# Post on the forum if this is not clear
def init(A):
    dim = len(A)
    for i in range(dim):
        for j in range(i + 1, dim):
            A[i][j] = random.randint(0, 1)
            if A[i][j] == 0:
                A[j][i] = 1
    return A

# Returns one of the winners
# There could be multiple winners
def winner(A):
    dim = len(A)
    max_wins, player = -1, -1
    for i in range(dim):
        win_count = 0
        for j in range(dim):
            win_count += A[i][j]
        if win_count > max_wins:
            max_wins = win_count
            player = i
    return player
```

Problem-15

$l \cdot b \cdot h$ cubes of unit volume are arranged in the form of a cuboid such that the whole setup is h units high and it occupies $l \cdot b$ square units of floor space. Each cube contains some number of coins. How would you represent this data in Python? Note that you need to know which box contains how many coins. Using this representation, write the following functions:

- `maxbox_at_height(data, z)`: accepts data as input and returns the co-ordinates of the box which has maximum number of coins at height z.
- `maxbox(data)`: accepts data as input and returns the co-ordinates of the box which has maximum number of coins across all heights; make use of `maxbox_at_height(data, z)` while writing this function.
- `height_of_maxboxes(data)`: accepts data as input and returns the height at which the cumulative count of coins at that height is maximum

What are some other questions that we can ask in this scenario? You must be able to answer these questions using Python.

Answer

This can be represented as a 3D matrix.

```
boxes[i][j][k] -> contains the number of boxes at coordinate (i, j, k)
```

Challenge

Source: Codeforces, an online judge for programming contests and a repository of some fascinating programming challenges.

Problem-16

You are in front of a balance with two pans and an assorted collection of weights. You randomly distribute the weights onto the two pans and find that the left-pan is heavier than the right-pan. At this stage, a question strikes you:

> What is the minimum number of weights that must be placed on the left-pan so that it becomes heavier than the right-pan?

The input is a sequence of the weights in kilograms. The output is the minimum number of weights that must be placed on the left-pan.

Test Cases

No.	Input	Output
1	`4 4`	2
2	`2 4 4`	2
3	`1 2 3 4 5 6 7 8`	3
4	`1 1 1 1 1 1`	4
5	`1 8 1 2 8 1 9 1 8 1 3 1 8 1 1 4 1 2 1 1 2 1 1`	5

Explanation

- In test case 1, moving a single weight onto the left-pan will not suffice, as that would make the pans of equal weight.
- In test case 2, at least two weights have to be moved. `4 + 2`

Answer

Open to discussion on the forum.

Problem-17

You have x candles made of wax. Each candle burns for an hour. You also possess a candle-making machine that can instantly convert the wax left over from y burnt up candles into a new candle. For a brief minute, suspend the laws of physics. Find the number of hours that the candles can light up your room.

Test Cases

Input	Output
`4 2`	`7`
`6 3`	`8`
`10 7`	`11`
`3 8`	`3`
`4 4`	`5`
`8 3`	`11`

Explanation

In the first test case, you have four candles to begin with. These will burn for four hours. Using four burnt up candles, you can make two new candles, which will burn for another two hours. Finally, with the two burnt up candles you can make one more candle.

Answer

Open to discussion on the forum.