Exercise - cnsh-121 Python Programming Problems (2024.09.14)

Python Programming Problems

1. 다음 재귀함수와 같은 기능을 하는 이중 for문 기반 함수를 구현하세요.

```
def numtriangle(n):
    if n < 1:
        return
    numtriangle(n-1)

    for i in range(1, n+1):
        print(i, end=' ')

    print()

# 이중 for문 버전을 작성하세요
def numtriangle_for(n):
    pass
```

입력예시

```
numtriangle_for(4)
```

출력예시

```
1
1 2
1 2 3
1 2 3 4
```

해설

```
def numtriangle_for(n):
    for row in range(1, n+1):
        for col in range(1, row+1):
            print(col, end=' ')
        print()
```

2. 다음 8퍼즐에서 0 위치가 맨 윗줄이 아니면 위로 올릴 수 있도록 조건식을 완성하세요.

```
s = [0]*9

for i in range(3):
    s[i*3], s[i*3+1], s[i*3+2] = map(int, input().split())

    for j in range(3):
        if s[i*3+j] == 0:
            pos = i*3 + j

if ____:  # 여기를 채우세요
    s[pos], s[pos-3] = s[pos-3], s[pos]

    for i in range(3):
        for j in range(3):
            print(s[i*3+j], end=' ')
        print()
    print()

else:
    print("Impossible")
```

입력예시

```
1 2 3
4 0 5
6 7 8
```

출력예시

```
1 0 3
4 2 5
6 7 8
```

해설

```
if pos > 2:  # 정답: pos > 2
# 이유: 0의 위치가 인덱스 3 이상이어야 맨 윗줄(인덱스 0, 1, 2)이 아님
```

3. 두 수를 비교하여 큰 수를 반환하는 함수를 삼항 연산자를 사용하여 작성하세요.

```
def minmax(a, b):
    return ____  # 여기를 채우세요
```

입력예시

```
print(minmax(5, 3))
print(minmax(2, 8))
print(minmax(10, 10))
```

출력예시

```
5
8
10
```

해설

```
def minmax(a, b):
    return a if a >= b else b  # 정답: a if a >= b else b
```

4. 입력을 4 3 → 0 1 → 2 0 → 3 2 순으로 주었을 때 만들어지는 인접리스트를 구하세요.

```
V, E = map(int, input().split())
adj = [[] for _ in range(V)]

for i in range(E):
    u, v = map(int, input().split())
    adj[u].append(v)
    adj[v].append(u)

print(adj)
```

입력예시

```
4 3
0 1
2 0
3 2
```

출력예시

```
[[1, 2], [0], [0, 3], [2]]
```

해설

```
# 단계별 인접리스트 구성:
# 초기: adj = [[], [], [], []]

# 0 1 입력:
# adj[0].append(1) -> adj[0] = [1]
# adj[1].append(0) -> adj[1] = [0]
# 결과: adj = [[1], [0], [], []]

# 2 0 입력:
# adj[2].append(0) -> adj[2] = [0]
# adj[0].append(2) -> adj[0] = [1, 2]
# 결과: adj = [[1, 2], [0], [0], []]

# 3 2 입력:
# adj[3].append(2) -> adj[3] = [2]
# adj[2].append(3) -> adj[2] = [0, 3]
# 최종 결과: adj = [[1, 2], [0], [0, 3], [2]]
```

5. q.empty()가 True일 때와 False일 때는 각각 어떤 상황인지 서술하세요.

```
import queue

q = queue.Queue()
for i in range(1, 4):
    q.put(i)
while not q.empty():
    print(q.get(), sep=' ')
```

해설

```
# q.empty()가 True일 때:
# - 큐에 아무 데이터도 없는 상태
# - 모든 데이터를 get()으로 꺼내서 큐가 비어있을 때
# - while 루프가 종료되는 조건

# q.empty()가 False일 때:
# - 큐에 하나 이상의 데이터가 있는 상태
# - put()으로 데이터를 넣었거나, 아직 get()으로 모두 꺼내지 않은 상태
# - while 루프가 계속 실행되는 조건
```

6. 다음 8퍼즐에서 0 위치와 바꿀 위치를 완성하고 이유를 설명하세요.

```
s = [0]*9

for i in range(3):
    s[i*3], s[i*3+1], s[i*3+2] = map(int, input().split())

    for j in range(3):
        if s[i*3+j] == 0:
            pos = i*3 + j

if pos > 2:
    s[pos], s[____] = s[____], s[pos]  # 여기를 채우세요

    for i in range(3):
        for j in range(3):
            print(s[i*3+j], end=' ')
        print()
    print()

else:
    print("Impossible")
```

입력예시

```
1 2 3
4 0 5
6 7 8
```

출력예시

```
1 0 3
4 2 5
6 7 8
```

해설

```
s[pos], s[pos-3] = s[pos-3], s[pos]  # 정답: pos-3, pos-3

# 이유:
# - 3x3 퍼즐에서 한 행 위로 이동하려면 인덱스가 3만큼 작아져야 함
# - pos가 현재 0의 위치이고, pos-3이 바로 위 행의 같은 열 위치임
# - 예: pos=4(중앙)이면 pos-3=1(위쪽 중앙)과 교환
```

7. 다음 스택 코드에서 while 조건식을 완성하여 올바르게 동작하도록 하세요.

```
stack = []
goal = [2, 1, 3, 4]

def push(x):
    global stack
    stack.append(x)

def pop():
    global stack
    return stack.pop()

now = 0
for i in range(1, 5):
    push(i)
    while len(stack)>0 and ____:  # 여기를 채우세요
        print(pop())
        now += 1
```

출력예시

```
2
1
3
4
```

해설

```
stack = []
goal = [2, 1, 3, 4]

def push(x):
    global stack
    stack.append(x)

def pop():
    global stack
    return stack.pop()

now = 0
for i in range(1, 5):
    push(i)
    while len(stack)>0 and stack[-1] == goal[now]:  # 정답: stack[-1] == goal[now]
        print(pop())
        now += 1
```

8. 다음 재귀 팩토리얼 함수에서 종료조건을 완성하세요.

```
def factorial(n):
    if ____:  # 여기를 채우세요
        return 1
    return n * factorial(n-1)
```

입력예시

```
print(factorial(5))
print(factorial(1))
```

출력예시

```
120
1
```

해설

```
def factorial(n):
    if n == 1:  # 정답: n == 1
        return 1
    return n * factorial(n-1)
```

9. 다음 최소공배수 재귀함수에서 return 부분을 완성하세요.

```
def lcm(a, b):
    global n, m
    if a == b:
        return a
    if a > b:
        return ____  # 여기를 채우세요
    if a < b:
        return lcm(a+m, b)
```

입력예시

```
n, m = 3, 8
print(lcm(m, n))
```

출력예시

```
24
```

해설

```
def lcm(a, b):
    global n, m
    if a == b:
        return a
    if a > b:
        return lcm(a, b+n)  # 정답: lcm(a, b+n)
    if a < b:
        return lcm(a+m, b)
```

10. 다음 재귀함수와 같은 기능을 하는 while문 기반 함수를 구현하세요.

```
# 참고: 재귀함수 버전
def gcd_recursive(a: int, b: int) -> int:
    a, b = abs(a), abs(b)
    if b == 0:
        return a
    return gcd_recursive(b, a % b)

# 여기에 while문 버전을 작성하세요
def gcd_while(a: int, b: int) -> int:
    pass
```

입력예시

```
print(gcd_while(48, 18))
print(gcd_while(100, 25))
```

출력예시

```
6
25
```

해설

```
def gcd_while(a: int, b: int) -> int:
    a, b = abs(a), abs(b)
    if a == 0: return b
    if b == 0: return a

    while b != 0:
        a, b = b, a % b

    return a
```

11. 다음 재귀함수에서 5번, 6번 라인(stars(n-1)과 print('*'*n))의 순서를 바꾸면 출력이 어떻게 변하는지 설명하세요.

```
def stars(n):
    if n == 1:
        print('*')

    stars(n-1)      # 5번 라인
    print('*'*n)    # 6번 라인
```

입력예시

```
stars(3)
```

해설

```
# 원본 함수 (재귀 호출 후 출력):
# stars(3) -> stars(2) -> stars(1) 순으로 호출
# stars(1)에서 '*' 출력 후, 역순으로 '**', '***' 출력
# 결과: *, **, ***

# 순서를 바꾼 함수 (출력 후 재귀 호출):
def stars_changed(n):
    if n == 1:
        print('*')

    print('*'*n)    # 먼저 출력
    stars_changed(n-1)  # 그 다음 재귀 호출

# stars_changed(3) 호출 시
# '***' 출력 -> stars_changed(2) 호출
# '**' 출력 -> stars_changed(1) 호출
# '*' 출력
# 결과: ***, **, *

# 즉, 재귀 호출 전에 출력하면 역삼각형, 재귀 호출 후에 출력하면 정삼각형이 됩니다.
```

12. 다음 Distance 클래스의 distance() 메서드를 완성하여 두 점 사이의 거리가 계산되도록 하세요.

```
import math

class Distance:
    def __init__(self, x1, y1, x2, y2):
        self.x1 = x1
        self.y1 = y1
        self.x2 = x2
        self.y2 = y2

    def distance(self):
        # 두 점 사이의 거리를 계산하는 공식을 구현하세요
        pass
```

입력예시

```
distance1 = Distance(2, 3, 3, 5)
print(f"두 점 사이의 거리는 {distance1.distance():.2f}")
```

출력예시

```
두 점 사이의 거리는 2.24
```

해설

```
import math

class Distance:
    def __init__(self, x1, y1, x2, y2):
        self.x1 = x1
        self.y1 = y1
        self.x2 = x2
        self.y2 = y2

    def distance(self):
        return math.sqrt((self.x2 - self.x1) ** 2 + (self.y2 - self.y1) ** 2)  # 정답
```

13. 다음 재귀 팩토리얼 함수와 같은 기능을 하는 while문 기반 함수를 구현하세요.

```
def factorial(n):
    if n == 1:
        return 1
    return n * factorial(n-1)

# while문 버전을 작성하세요
def factorial_while(n):
    pass
```

입력예시

```
print(factorial_while(5))
print(factorial_while(3))
```

출력예시

```
120
6
```

해설

```
def factorial_while(n):
    result = 1
    while n > 0:
        result *= n
        n -= 1
    return result
```

14. factorial(4)를 호출했을 때 재귀함수의 call stack이 어떻게 쌓이고 해제되는지 단계별로 서술하세요.

```
def factorial(n):
    if n == 1:
        return 1
    return n * factorial(n-1)
```

입력예시

```
factorial(4)
```

해설

```
# factorial(4) call stack 과정:
# 1. factorial(4) 호출 -> 4 * factorial(3) 대기
# 2. factorial(3) 호출 -> 3 * factorial(2) 대기
# 3. factorial(2) 호출 -> 2 * factorial(1) 대기
# 4. factorial(1) 호출 -> 1 반환 (종료조건)
# 5. factorial(2) = 2 * 1 = 2 반환
# 6. factorial(3) = 3 * 2 = 6 반환
# 7. factorial(4) = 4 * 6 = 24 반환

# Stack: factorial(4) -> factorial(3) -> factorial(2) -> factorial(1)
# 결과: 24 <- 6 <- 2 <- 1
```

15. permutation(5, 3) 출력값의 계산과정을 단계별로 서술하세요.

```
def permutation(n, r):
    if r == 0:
        return 1
    return n * permutation(n-1, r-1)
```

입력예시

```
permutation(5, 3)
```

해설

```
# permutation(5, 3) 계산과정:

# 1. permutation(5, 3) 호출
#    r ≠ 0이므로 return 5 * permutation(4, 2)

# 2. permutation(4, 2) 호출
#    r ≠ 0이므로 return 4 * permutation(3, 1)

# 3. permutation(3, 1) 호출
#    r ≠ 0이므로 return 3 * permutation(2, 0)

# 4. permutation(2, 0) 호출
#    r == 0이므로 return 1

# 역산 과정:
# permutation(2, 0) = 1
# permutation(3, 1) = 3 * 1 = 3
# permutation(4, 2) = 4 * 3 = 12
# permutation(5, 3) = 5 * 12 = 60

# 결과: 60 (이는 5P3 = 5!/(5-3)! = 5*4*3 = 60과 동일)
```

16. 다음 재귀 합계 함수에서 종료조건을 완성하세요.

```
def sum(n):
    if ____:  # 여기를 채우세요
        return 1
    return n + sum(n-1)
```

입력예시

```
print(sum(5))
print(sum(3))
```

출력예시

```
15
6
```

해설

```
def sum(n):
    if n == 1:  # 정답: n == 1
        return 1
    return n + sum(n-1)
```

17. 다음 전역변수를 사용한 합계 함수에서 while문 조건을 완성하세요.

```
s = 0

def sum(n):
    global s
    k=0
    while ____:  # 여기를 채우세요
        k = k+1
        s = s+k
```

입력예시

```
sum(5)
print(s)
```

출력예시

```
15
```

해설

```
s = 0

def sum(n):
    global s
    k=0
    while k < n:  # 정답: k < n
        k = k+1
        s = s+k
```

18. 짝수합을 재귀함수로 구현하세요.

```
def even_sum(n):
    result = 0
    for i in range(n+1):
        if i % 2 == 0:
            result += i
    return result

# 재귀함수 버전을 작성하세요
def even_sum_recursive(n):
    pass
```

입력예시

```
print(even_sum_recursive(10))
print(even_sum_recursive(4))
```

출력예시

```
30
6
```

해설

```
def even_sum_recursive(n):
    if n < 0:
        return 0
    if n % 2 == 0:
        return n + even_sum_recursive(n-2)
    else:
        return even_sum_recursive(n-1)
```

19. 다음 while문 기반 최대공약수 함수를 재귀함수로 재구현하세요.

```
def gcd(a, b):
    while a != 0:
        a, b = b % a, a
    return b

# 재귀함수 버전을 작성하세요
def gcd_recursive(a, b):
    pass
```

입력예시

```
print(gcd_recursive(48, 18))
print(gcd_recursive(100, 25))
```

출력예시

```
6
25
```

해설

```
def gcd_recursive(a, b):
    if a == 0:
        return b
    return gcd_recursive(b % a, a)
```

20. 다음 재귀함수에서 조건 부분을 완성하세요.

```
def f(n):
    if ____:  # 여기를 채우세요
        print('*')
    else:
        f(n-1)
        print('*'*n)
```

입력예시

```
f(3)
```

출력예시

```
*
**
***
```

해설

```
def f(n):
    if n <= 1:  # 정답: n <= 1
        print('*')
    else:
        f(n-1)
        print('*'*n)
```