Exercise - cnsh-121 Python Programming Problems (2024.09.14)

Python Programming Problems

1. 다음 재귀함수에서 괄호 내부를 비워두고 동작하도록 채우세요.

```
def gcd_recursive(a: int, b: int) -> int:
    a, b = abs(a), abs(b)
    if b == 0:
        return a
    return gcd_recursive(____, ____)  # 여기를 채우세요
```

입력예시

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

출력예시

```
6
25
```

해설

```
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)  # 정답: b, a % b
```

2. 다음 EightPuzzle 클래스의 display 메서드에서 빈 라인을 완성하여 3x3 숫자판이 올바르게 출력되도록 하세요.

```
class EightPuzzle:
    def __init__(self, numbers):
        self.numbers = numbers

    def display(self):
        for i in range(3):
            for j in range(3):
                # 이 라인을 완성하세요
            print()
        print()

    def find_zero(self):
        for i in range(3):
            for j in range(3):
                if self.numbers[3*i + j] == 0:
                    return i*3 + j
        print("Zero not found")

    def move_up(self):
        pos = self.find_zero()
        if pos > 2:
            self.numbers[pos], self.numbers[pos-3] = self.numbers[pos-3],
self.numbers[pos]
        else:
            print("Cannot move up")
        self.display()
```

입력예시

```
puzzle = EightPuzzle([1, 2, 3, 4, 0, 5, 6, 7, 8])
puzzle.display()
```

출력예시

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

해설

```
class EightPuzzle:
    def __init__(self, numbers):
        self.numbers = numbers

    def display(self):
        for i in range(3):
            for j in range(3):
                print(self.numbers[3*i + j], end=' ')  # 정답
            print()
        print()

    def find_zero(self):
        for i in range(3):
            for j in range(3):
                if self.numbers[3*i + j] == 0:
                    return i*3 + j
        print("Zero not found")

    def move_up(self):
        pos = self.find_zero()
        if pos > 2:
            self.numbers[pos], self.numbers[pos-3] = self.numbers[pos-3],
self.numbers[pos]
        else:
            print("Cannot move up")
        self.display()
```

3. 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)
```

4. 다음 코드에서 {} 내부를 올바르게 채워서 평균 점수가 소수점 2자리까지 출력되도록 하세요.

```
class Student:
    def __init__(self, name, scores):
        self.name = name
        self.scores = scores

    def avg(self):
        return sum(self.scores) / len(self.scores)

    def grade(self):
        avg = self.avg()
        if avg >= 90:
            return 'A'
        elif avg > 80:
            return 'B'
        elif avg > 70:
            return 'C'
        elif avg > 60:
            return 'D'
        elif avg > 50:
            return 'E'
        else:
            return 'F'
```

입력예시

```
student1 = Student('홍길동', [90, 75, 80])
print(f"{student1.name}의 평균 점수는 {    }이고, 등급은 {student1.grade()}입니다.")
```

출력예시

```
홍길동의 평균 점수는 81.67이고, 등급은 B입니다.
```

해설

```
print(f"{student1.name}의 평균 점수는 {student1.avg():.2f}이고, 등급은
{student1.grade()}입니다.")
# {} 내부에 student1.avg():.2f 를 입력하면 됩니다.
```

5. 출력값이 5 4 3 2 1이 되도록 range() 부분을 완성하세요.

```
import queue

q = queue.Queue()
for i in range(____):  # 여기를 채우세요
    q.put(i)
while not q.empty():
    print(q.get(), sep=' ')
```

출력예시

```
5
4
3
2
1
```

해설

```
import queue

q = queue.Queue()
for i in range(5, 0, -1):  # 정답: 5, 0, -1
    q.put(i)
while not q.empty():
    print(q.get(), sep=' ')
```

6. 다음 재귀 합계 함수에서 n부터 1까지의 합이 나오도록 return 부분을 완성하세요.

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

입력예시

```
print(sum(4))
print(sum(6))
```

출력예시

```
10
21
```

해설

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

7. 다음 짝수합 함수를 한 줄로 바꿔서 구현하세요.

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

# 한 줄로 구현하세요
def even_sum_oneline(n):
    return ____
```

입력예시

```
print(even_sum_oneline(10))
print(even_sum_oneline(6))
```

출력예시

```
30
12
```

해설

```
def even_sum_oneline(n):
    return sum(i for i in range(n+1) if i % 2 == 0)  # 정답
```

8. 다음 재귀 합계 함수와 같은 기능을 하는 while문 기반 함수를 구현하세요.

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

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

입력예시

```
print(sum_while(5))
print(sum_while(3))
```

출력예시

```
15
6
```

해설

```
def sum_while(n):
    result = 0
    i = 1
    while i <= n:
        result += i
        i += 1
    return result
```

9. 다음 코드의 print() 문을 완성하여 "P의 평균 점수는 Q(소수점2자리까지)이고, 등급은 R입니다." 형식으로 출력되도록 하세요.

```
class Student:
    def __init__(self, name, scores):
        self.name = name
        self.scores = scores

    def avg(self):
        return sum(self.scores) / len(self.scores)

    def grade(self):
        avg = self.avg()
        if avg >= 90:
            return 'A'
        elif avg > 80:
            return 'B'
        elif avg > 70:
            return 'C'
        elif avg > 60:
            return 'D'
        elif avg > 50:
            return 'E'
        else:
            return 'F'
```

입력예시

```
student1 = Student('홍길동', [90, 75, 80])
# 다음 print 문을 완성하세요
print()
```

출력예시

```
홍길동의 평균 점수는 81.67이고, 등급은 B입니다.
```

해설

```
print(f"{student1.name}의 평균 점수는 {student1.avg():.2f}이고, 등급은
{student1.grade()}입니다.")
```

10. 다음 순열(permutation) 재귀함수에서 return 부분을 완성하세요.

```
def permutation(n, r):
    if r == 0:
        return 1
    return ____  # 여기를 채우세요
```

입력예시

```
print(permutation(5, 3))
print(permutation(4, 2))
```

출력예시

```
60
12
```

해설

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

11. 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)
```

12. 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
```

13. 다음 카운트다운 함수에서 range() 파라메터를 완성하여 카운트다운이 되도록 하세요.

```
def countdown(n):
    for i in range(____, ____, ____):  # 여기를 채우세요
        print(i)
```

입력예시

```
countdown(5)
```

출력예시

```
5
4
3
2
1
```

해설

```
def countdown(n):
    for i in range(n, 0, -1):  # 정답: n, 0, -1
        print(i)
```

14. 다음 팩토리얼 함수에서 range()의 세 파라메터를 바르게 채우세요.

```
def factorial(n):
    result = 1
    for i in range(____, ____, ____):  # 여기를 채우세요
        result *= i

    return result
```

입력예시

```
print(factorial(5))
print(factorial(4))
```

출력예시

```
120
24
```

해설

```
def factorial(n):
    result = 1
    for i in range(n, 0, -1):  # 정답: n, 0, -1
        result *= i

    return result
```

15. 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
```

16. 다음 Movie 클래스를 분석하고, 주어진 코드를 실행했을 때의 출력 결과를 예측하세요.

```
class Movie:
    def __init__(self, name, seat):
        self.name = name
        self.total_seats = seat

    def reserve(self):
        if self.total_seats > 0:
            self.total_seats -= 1
            return True
        else:
            return False
```

입력예시

```
seat1 = Movie('ET', 5)
result = seat1.reserve()
print('예약 성공' if result else '예약 실패')

# 추가 예약 시도
for i in range(6):
    result = seat1.reserve()
    print(f"{i+2}번째 예약: {'성공' if result else '실패'}")
```

출력예시

```
예약 성공
2번째 예약: 성공
3번째 예약: 성공
4번째 예약: 성공
5번째 예약: 성공
6번째 예약: 실패
7번째 예약: 실패
```

해설

```
class Movie:
    def __init__(self, name, seat):
        self.name = name
        self.total_seats = seat

    def reserve(self):
        if self.total_seats > 0:
            self.total_seats -= 1
            return True
        else:
            return False

# 처음 total_seats가 5이므로 5번까지는 예약이 가능하고,
# 6번째부터는 좌석이 없어서 예약 실패가 됩니다.
```

17. 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
```

18. 다음 코드의 출력값이 어떻게 되는지 서술하세요.

```
import queue

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

해설

```
# 출력값: 1, 2, 3 (각각 새 줄에)
# 이유: Queue는 FIFO(First In First Out) 구조이므로
# for문에서 1, 2, 3 순으로 put() 했기 때문에
# while문에서도 1, 2, 3 순으로 get()됨
```

19. 다음 Student 클래스의 grade() 메서드를 완성하여 평균 점수에 따른 학점이 계산되도록 하세요.

```
class Student:
    def __init__(self, name, scores):
        self.name = name
        self.scores = scores

    def avg(self):
        return sum(self.scores) / len(self.scores)

    def grade(self):
        avg = self.avg()
        # 학점 계산 로직을 구현하세요
        # 90 이상: A, 80 초과: B, 70 초과: C, 60 초과: D, 50 초과: E, 그 외: F
        pass
```

입력예시

```
student1 = Student('홍길동', [90, 75, 80])
student2 = Student('김철수', [95, 88, 92])
student3 = Student('이영희', [60, 65, 70])
print(f"{student1.name}: {student1.grade()}")
print(f"{student2.name}: {student2.grade()}")
print(f"{student3.name}: {student3.grade()}")
```

출력예시

```
홍길동: B
김철수: A
이영희: D
```

해설

```
def grade(self):
    avg = self.avg()
    if avg >= 90:
        return 'A'
    elif avg > 80:
        return 'B'
    elif avg > 70:
        return 'C'
    elif avg > 60:
        return 'D'
    elif avg > 50:
        return 'E'
    else:
        return 'F'
```

20. 다음 큐 코드가 올바르게 동작하도록 import 문을 완성하세요.

```
import ____  # 여기를 채우세요

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

출력예시

```
1
2
3
```

해설

```
import queue  # 정답: queue

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