파이썬 정기평가 대비 예상 문제

홍송은

1. 사용자로부터 'True' 또는 'False' (문자열)를 입력받아, 실제 불리언 값으로 변환한 후 조건문을 통해 다음과 같이 출력되도록 빈칸을 채우세요.

   입력: "True" → 출력: 진행합니다

   "False" → 출력: 중단합니다

```
value = input("계속하시겠습니까? (True/False): ")
value = (value == "True") # 문자열을 bool 자료형으로 변환

if value:
    print("진행합니다")
else:
    print("중단합니다")
```

   주의) bool('False')는 True이므로 사용X

2. 다음 딕셔너리에서 이름이 "Alice"인 사람의 나이를 반환하는 함수를 작성하세요.

```
def get_age(info):
    return info['age']

person = {'name': 'Alice', 'age': 25}
print(get_age(person))  # 출력: 25
```

3. 1부터 20까지의 숫자 중 3의 배수만 출력하는 코드를 작성하세요. (단, continue 문을 사용할 것.)

```
for i in range(1, 21):
    if i % 3 != 0:
        continue
    print(i)
```

4. 다음 중 리스트에 요소를 맨 앞에 삽입하는 코드로 올바른 것은?

   a) list.append(0)

   b) list.insert(0, 0)

   c) list.pop(0)

   d) list.remove(0)

5. 다음 코드를 실행했을 때 출력 결과를 작성하라

for i in range(2):
    for j in range(3):
        print(f"{i},{j}")
	결과
	0,0
	0,1
	0,2
	1,0
	1,1
	1,2

6. 다음 코드에서 전역 변수 count가 함수 내부에서 1씩 증가되도록 빈칸을 채우세요.

```
count = 0

def increment():
    global count
    count += 1

increment()
print(count)
```

7. 다음 리스트에서 최대값을 반환하는 find_max 함수를 작성하세요. (단, for 반복문을 사용해야 합니다.)

```
def find_max(numbers):
    # 코드 작성
    max = numbers[0]
    for num in numbers:
        if num > max:
            max = num
    return max
nums = [15, 22, 8, 31, 13]
print(find_max(nums))  # 출력: 31
```

8. 다음 리스트에 대해 선택 정렬 알고리즘을 적용했을 때, 2회전(2번의 정렬 반복)이 끝난 후의 리스트 상태를 출력하세요. (단, 정렬 알고리즘은 직접 구현해야 하며, sort() 또는 sorted() 함수는 사용할 수 없습니다.)

```
data = [40, 10, 25, 18, 35]

# 1. 선택 정렬 코드 작성
def selection_sort_partial(arr):
    n = len(arr)
    for i in range(2):  # 2회전까지만 수행
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr

# 2. 선택 정렬 2회전 이후 리스트 상태를 출력하세요.
result = selection_sort_partial(data)
print(result)
```

```
[10, 18, 25, 40, 35]
```

참고)
data = [40, 10, 25, 18, 35]
1회전: [10, 40, 25, 18, 35]
2회전: [10, 18, 25, 40, 35]
3회전: [10, 18, 25, 35, 40]

9. 다음 중 파이썬의 클래스와 상속에 대해 올바르게 설명한 것을 모두 고르세요.

a) __init__은 객체가 생성될 때 자동으로 호출되는 생성자이다.

b) super()는 자식 클래스에서 부모 클래스의 메서드를 직접 호출할 수 있게 한다.

c) 클래스에서 정의된 모든 변수는 전역 변수이다.

-> 클래스 내에서 정의한 변수: 보통 인스턴스 변수(self.변수명) 또는 클래스 변수.
전역 변수(global variable)가 아니며, 클래스의 범위(scope) 안에 존재.

d) 자식 클래스는 super 키워드를 사용해야만 부모 클래스의 속성에 접근할 수 있다.

-> 자식 클래스에서도 부모 클래스 이름을 통해 직접 접근 가능.
super() 필수X. 편의적 선택!

```
class Parent:
    def greet(self):
        print("Hello from Parent")

class Child(Parent):
    def greet(self):
        # super() 사용하지 않고도 접근 가능
        Parent.greet(self)  # 직접 호출도 가능
```

e) 클래스는 하나의 객체만 생성할 수 있다.

10. 다음 요구사항에 맞는 모듈 math_tools.py를 정의하고, 이를 사용하는 메인 코드를 작성하세요.

<요구사항>

- math_tools.py에는 두 수를 더하는 add(a, b) 함수가 있어야 합니다.
- 메인 스크립트에서 모듈을 호출하여 결과를 출력합니다.

```
#main_tools.py

def add(a, b):
  return a + b
```

```
# main.py
from main_tools import add

result = add(3, 5)
print(result)

또는
import main_tools

result = main_tools.add(3, 5)
print(result)
```

11. 다음 tkinter 코드에서 버튼 클릭 시 "버튼이 클릭되었습니다!"라는 문장을 출력하도록 빈칸을 채우세요.

```
import tkinter as tk

def on_click():
    print("버튼이 클릭되었습니다!")

root = tk.Tk()
btn = tk.Button(root, text="클릭", command=on_click)
btn.pack()
root.mainloop()
```

12. 파일 "error.txt"에서 "ERROR"가 포함된 줄의 개수를 반환하는 함수를 작성하세요.

```
def count_err, or_lines(filepath):
    count = 0
    with open(filepath, 'r') as f:
        for line in f:
            if “ERROR” in line:
                count += 1
    return count
```

만약, error, Error, eRrOr등 대소문자를 무시해서 찾아내려면?

```
if "error" in line.lower():
```

또는 정규표현식 사용

```
import re
if re.search(r'error', line, re.IGNORECASE):
```

13. 1부터 10까지의 수 중 짝수의 제곱만 생성하는 제너레이터 even_squares()를 구현하고, 이를 이용해 값을 출력하세요.

```
# 제너레이터 정의 및 출력
def even_squares():
    # 코드 작성
   for i in range(1, 11):
        if i % 2 == 0:
            yield i * I    # 반드시 yield 사용

for value in even_squares():
    print(value)
```

출력 결과

```
4
16
36
64
100
```

14. 다음 중 파이썬의 예외 처리 및 고급 함수 관련 설명으로 올바른 것을 모두 고르세요.

a) raise 문은 프로그래머가 직접 예외를 강제로 발생시킬 때 사용된다.

b) map() 함수는 반복 가능한 객체의 각 요소에 함수를 적용해 새로운 리스트를 반환한다.

c) lambda 함수는 반드시 변수에 저장하여 사용해야 한다.

d) try 블록 내부 코드에서 예외가 발생하지 않으면 except 블록은 실행되지 않는다.

e) map()은 두 개 이상의 리스트에는 사용할 수 없다.

-> 두 개 이상의 리스트에도 사용 가능하다!

```
list1 = [1, 2, 3]
list2 = [10, 20, 30]
def add(x, y):
    return x + y
result = list(map(add, list1, list2))
print(result)                # [11, 22, 33]
```

15. 다음 텍스트에서 확장자가 .py, .txt, .csv인 파일 경로만 추출하세요. 디렉터리 경로를 포함할 수 있으며, 파일 이름에는 공백이 없다고 가정합니다.

```
import re

text = """
경로1: ./src/main.py
경로2: ./data/result.csv
경로3: /home/user/readme.txt
무시: ./log/error.log
경로4: script.PY
"""
pattern = r'[₩w./]+?₩.(py|txt|csv)'
print(re.findall(pattern, text))
```

출력 결과

['./src/main.py', './data/result.csv', '/home/user/readme.txt', 'script.PY']

r'' -> ₩를 escape하지 않고 그대로 처리

1) [₩w./]+?

₩w -> 알파벳 대소문자, 숫자, _ -> [a-zA-Z0-9_]

. -> 문자 그대로 점을 의미

/ -> 문자 그대로 슬래시 의미

+? -> + 앞 문자가 1개 이상 반복. ?를 붙임으로써 가능한 짧게 매칭

즉 최소 길이로 반복 매칭

ex) 'file1.py somefile.txt another.csv'의 경우

r'.+₩.txt' 는 'file1.py somefile.txt another.csv'

r'.+?₩.txt'는 'file1.py somefile.txt' .txt를 만나면 멈춤!

2) ₩.

-> ₩를 붙였으므로 진짜 점으로 의미. 특수문자라서 이스케이프 처리

3) (py|txt|csv)

-> |는 '또는'을 의미

주의)

- (py|txt|csv) → 세 가지 **문자열 중 하나**를 고름 (파일 확장자처럼!)
- [py|txt|csv] → "p", "y", "|", "t"... 중 **하나의 문자만** 고름

참고) [₩w./]의 .와 ₩. 차이점?

표현	의미	예시	
.	모든 문자 1개	a, 1, @ 등	정규표현식에서 특수문자
₩.	진짜 점(.) 문자	"main.py"의 .	이스케이프해야만 파일 확장자를 인식 가능
[₩w./]	점(.)도 포함된 문자셋	main.py, ./src/	문자 클래스 안의 점이므로 이스케이프할 필요 없이 마침표 자체 의미

16. 다음은 파이썬으로 작성한 스택 클래스입니다. pop() 메서드가 스택이 비어 있을 경우 -1을 반환하도록 구현되도록 빈칸을 채우세요.

```
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if len(self.items) == 0:
            return -1
        return self.items.pop()
```

17. 다음은 큐(Queue) 클래스입니다. 큐에서 데이터를 하나 제거하고 그 값을 반환하도록 함수 dequeue_and_return()을 완성하세요. 큐가 비었으면 None을 반환합니다.

```
from collections import deque

def dequeue_and_return(q):
    if len(q) == 0:
        return None
    return q.popleft()
```

18. 아래 그래프에서 DFS를 수행한 결과 순서를 출력하세요. 시작 노드는 A입니다.

```
graph = {
    'A': ['B', 'C'],
    'B': ['D'],
    'C': ['E'],
    'D': [],
    'E': []
}

def dfs(graph, node, visited):
    if node not in visited:
        visited.append(node)
        for n in graph[node]:
            dfs(graph, n, visited)
    return visited

visited = []
result = dfs(graph, 'A', visited)
print(result)
```

결과

['A', 'B', 'D', 'C', 'E']

19. 다음 중 Pandas 또는 Numpy의 동작 및 개념에 대한 설명으로 옳은 것을 모두 고르세요.

a) DataFrame은 1차원 배열 데이터를 라벨과 함께 관리하는 객체이다.

-> DataFrame은 2차원 데이터 구조

Series가 1차원 데이터 구조

b) Series는 인덱스를 가지는 1차원 데이터 구조이다.

c) numpy.dot()은 두 배열의 행렬 곱을 계산할 때 사용된다.

d) df.isnull().sum()은 결측치가 있는 열을 삭제하는 함수이다.

e) arr[arr > 3]는 넘파이 배열에서 3보다 큰 값을 조건 필터링하여 새 배열로 반환한다.

20. scipy를 사용하여 다음 연립방정식을 해결하는 프로그램을 작성하세요.

4x + 2y = 10 3x + 5y = 20

```
from scipy.linalg import solve
import numpy as np

A = np.array([[4, 2], [3, 5]])   # 계수 행렬
b = np.array([10, 20])   # 우변 벡터
x = solve(A, b)
print(x)
```