CH-2

1. 다음 python 코드의 마지막 print 문의 출력값으로 옳은 것은 무엇인가?

```
X = 10
Y = 5
print( X > Y )
```

1. True ❤
2. False
3. None
4. "10"
5. "5"

2. 다음과 같이 빈 딕셔너리 A를 생성하였다. 보기 중 오류가 발생하는 것은 무엇인가?

```
A = dict()
A
# 출력
# {}
```

1. A['name'] = 'Python'
2. A[('a',)] = 'Python'
3. A1 = 'Python' ❤
4. A[250] = 'Python'
5. A['Python'] = 100

3. n의 다음 코드에서 pass의 역할은 무엇인가?

```
def my_function():
    if condition:
        pass
    else:
        print("Do Something")
```

1. 코드를 실행하지 않고 다음줄로 넘어간다. ❤
2. condition 이 참일 경우만 코드를 실행한다.
3. condition 이 거짓일 경우만 코드를 실행한다.
4. 함수를 정의한다.
5. 변수를 선언한다.

4. 다음과 같이 a, b의 변수에 선언하고, a의 요소값을 변경 후 b를 화면에 출력하는 코드이다. 옳은 출력 결과를 고르시오.

```
a = [1,2,3]
b = a
a[1] = 5
print(b)
```

1. [1,2,3]
2. [5,2,3]
3. []
4. [1,5,3] ❤
5. ValueError

5. 다음 python 코드의 print 로 출력 되는 result 의 값은 무엇인가?

```
numbers = [1, 2, 3, 4, 5]
result = [number * 2 for number in numbers if number % 2 == 0]
print(result)
```

1. [1, 2, 3, 4, 5]
2. [2, 4, 6, 8, 10]
3. [1, 3, 5]
4. [4, 8] ❤
5. []

6. 위 Python 코드에서 딕셔너리 컴프리헨션의 역할로 **옳은** 것은?

```
numbers = [1, 2, 3, 4, 5]
squared_dict = {index: num ** 2 * 2 if num % 2 == 0 else num ** 2 for
index, num in enumerate(numbers)}
print(squared_dict)
```

1. numbers 리스트의 각 요소의 인덱스를 키로, 홀수 요소의 제곱을 값으로 가지는 딕셔너리를 생성.

2. numbers 리스트의 각 요소의 인덱스를 키로, 짝수 요소의 제곱을 값으로 가지는 딕셔너리를 생성.
3. numbers 리스트의 각 요소의 인덱스를 키로, 짝수 요소의 제곱의 두 배를 값으로, 홀수 요소의 제곱을 값으로 가지는 딕셔너리를 생성. ❤
4. numbers 리스트의 각 요소의 값을 키로, 인덱스를 값으로 가지는 딕셔너리를 생성.
5. numbers 리스트의 각 요소의 값을 키로, 짝수 인덱스의 요소의 제곱을 값으로 가지는 딕셔너리를 생성.

7. 다음은 python 에서 Dog class를 선언하는 코드이다. bark() 메서드는 어떤 역할을 수행하는지 옳게 설명한 것은?

```
# Dog class 선언
class Dog:
    def __init__(self, name):
        self.name = name
    def bark(self):
        print(f"{self.name} 왈왈!")

# 객체 생성
dog = Dog("해피")

# 메서드 호출
dog.bark()
```

1. Dog 클래스의 객체를 생성한다.
2. Dog 클래스의 속성 name 을 출력한다.
3. Dog 클래스의 속성 name 을 "해피"로 설정한다.
4. "왈왈!" 문자열을 출력한다.
5. "해피 왈왈!" 문자열을 출력한다. ❤

8. 다음은 pthon 코드는 class를 생성하고 '상속'하여 '다형성'을 만드는 예제이다. 코드를 실행하여 최종 출력되는 내용으로 맞는 것은?

```
class Animal:
    def speak(self):
        pass

class Dog(Animal):
    def speak(self):
        print("왈왈!", end='--')

class Cat(Animal):
    def speak(self):
        print("야옹!", end='--')
```

```
# 동물 객체 생성
animals = [Dog(), Cat(), Dog(), Cat()]

# 모든 동물 객체에게 speak() 메서드 호출
for animal in animals:
    animal.speak()
```

1. 왈왈!--야옹!--
2. 왈왈!--왈왈!--야옹!--야옹!--
3. 왈왈!--야옹!--왈왈!--야옹!-- ❤
4. 야옹!--왈왈!--야옹!--왈왈!--
5. SyntaxError

9. 다음 python 코드에서 `if __name__ == "__main__":` 블록은 어떤 역할을 수행할까?

```
def my_function():
    print("함수 실행")

if __name__ == "__main__":
    my_function()
```

1. `my_function()` 함수를 정의.
2. `my_function()` 함수를 다른 모듈에서 가져오기.
3. `my_function()` 함수를 실행. ❤
4. `my_function()` 함수를 삭제.
5. `my_function()` 함수의 문서를 출력.

10. 다음 Python 코드에서 클로저를 활용한 특징으로 **옳지 않은** 것은 무엇입니까?

```
def make_counter(x):
    count = 0
    local_var = x
    def inner_function(y):
        nonlocal count
        print( f"{x}에 대해서 {y}를 더합니다. 이것은 {count + 1}번째 시행입니다.")
        count += 1
    return print(f"더한 결과는 {local_var + y} 입니다.")
return inner_function

closure_instance = make_counter(10)
result = closure_instance(5)
result = closure_instance(15)
```

1. `inner_function`은 `outer_function`의 지역 변수 `local_var`에 접근하여 값을 유지.
2. `inner_function`은 `outer_function`의 외부 변수 `count`를 `nonlocal` 키워드를 통해 수정.
3. `closure_instance`는 `outer_function`이 종료된 후에도 `local_var`와 `count`의 상태를 기억.
4. `outer_function`의 인자 `x`는 `inner_function`에서 마치 전역 변수처럼 사용.
5. `inner_function`은 `outer_function`의 지역 변수 `y`에 접근할 수 없음. ❤

11. Python 코드에서 데코레이터 `my_decorator`의 역할로 **옳은** 것은?

```
def my_decorator(func):
    def wrapper():
        print("함수 실행 전")
        func()
        print("함수 실행 후")
    return wrapper

@my_decorator
def say_hello():
    print("안녕하세요!")

say_hello()

# 출력결과
'''
함수 실행 전
안녕하세요!
함수 실행 후
'''
```

1. `say_hello` 함수의 실행 결과를 변경.
2. `say_hello` 함수의 실행 시간을 측정.
3. `say_hello` 함수의 인자를 검증.
4. `say_hello` 함수의 실행 전후에 추가적인 작업을 수행. ❤
5. `say_hello` 함수를 여러 번 반복 실행.

12. Python 패키지에서 `__init__.py` 파일의 역할로 **옳지 않은** 것은 무엇인가?

1. 해당 디렉토리를 Python 패키지로 인식하게 함.
2. 패키지 초기화 코드를 실행하는 데 사용.
3. 패키지 내의 모듈을 네임스페이스로 관리하는 데 도움을 줌.
4. 패키지에 포함된 모든 모듈의 코드를 실행하는 역할. ❤

5. 페키지에서 `from package import *` 와 같은 와일드카드 임포트를 사용할 때 임포트할 모듈을 정의.

13. Python의 `map()` 내장 함수에 대한 설명으로 **옳지 않은** 것은?

1. `map()` 함수는 주어진 함수를 순회 가능한(iterable) 객체의 각 요소에 적용하고, 그 결과를 `map` 객체로 반환.
2. `map()` 함수의 첫 번째 인자는 적용할 함수이고, 두 번째 인자는 순회 가능한 객체.
3. `map()` 함수는 리스트, 튜플, 집합 등 다양한 순회 가능한 객체에 사용할 수 있음.
4. `map()` 함수는 항상 새로운 리스트를 반환.❤
5. `map()` 함수는 람다(lambda) 함수와 함께 사용하여 간단한 연산을 수행할 수 있음.

14 Python 클래스에서 `super()` 함수의 역할로 **옳지 않은** 것은?

1. 자식 클래스에서 부모 클래스의 메서드를 호출하는 데 사용.
2. 다중 상속 환경에서 메서드 결정 순서(MRO)를 따름.
3. 자식 클래스에서 부모 클래스의 속성을 초기화하는 데 사용.
4. 자식 클래스에서 부모 클래스의 모든 비공개(private) 메서드를 호출할 수 있음. ❤
5. 코드의 재사용성을 높이고 유지보수를 용이하게 함.

15. 다음 중 Python 표준 라이브러리 모듈의 용도에 대한 설명으로 **옳지 않은** 것은?

1. `os` 모듈은 운영체제와 상호작용하기 위한 다양한 기능을 제공.
2. `math` 모듈은 수학적인 연산과 관련된 함수들을 제공.
3. `random` 모듈은 난수 생성과 관련된 함수들을 제공.
4. `requests` 모듈은 웹 서버에 HTTP 요청을 보내고 응답을 처리하는 기능을 제공. ❤ (해당 기능을 수행하는 표준 라이브러리는 `urllib`)
5. `datetime` 모듈은 날짜와 시간 데이터를 처리하는 기능을 제공.