시험에 나오는 것만 공부한다! 시나공

유형별로 분류한 프로그래밍 – Python 6선

정보처리기사 실기

[Python의 기본]

1. 다음 Python으로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)

```
x, y = 100, 200
print(x==y)
```

답 : False

[해설]

```
❶ x, y = 100, 200
❷ print(x==y)
```

❶ 변수 x, y를 선언하고 각각 100, 200으로 초기화한다.

❷ x의 값 100과 y의 값 200이 같으면 참(True)을, 같지 않으면 거짓(False)을 출력한다.

결과 False

2. 다음 Python으로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)

```
a = "REMEMBER NOVEMBER"
b = a[0:3] + a[12:16]
c = "R AND %s" % "STR"
print(b + c)
```

답 : REMEMBER AND STR

[해설]

```
❶ a = "REMEMBER NOVEMBER"
❷ b = a[0:3] + a[12:16]
❸ c = "R AND %s" % "STR"
❹ print(b + c)
```

❶ 변수 a를 선언하고 “REMEMBER NOVEMBER”로 초기화한다.

❷ a에 저장된 문자열의 0부터 2번째 위치까지의 문자열과 12부터 15번째 위치까지의 문자열을 합쳐 b에 저장한다.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]
a	‘R’	‘E’	‘M’	‘E’	‘M’	‘B’	‘E’	‘R’		‘N’	‘O’	‘V’	‘E’	‘M’	‘B’	‘E’	‘R’

b = REMEMBE

❸ c에 “R AND STR”을 저장한다. %s는 서식 문자열로, % 뒤쪽의 “STR”이 대응된다.

• "R AND %s" % "STR"

❹ b와 c에 저장된 문자열을 합쳐 출력한다.

결과 REMEMBER AND STR

3. 다음 Python으로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)

```
asia = {'한국', '중국', '일본'}
asia.add('베트남')
asia.add('중국')
asia.remove('일본')
asia.update({'한국', '홍콩', '태국'})
print(asia)
```

답 : {'한국', '중국', '베트남', '홍콩', '태국'}

[해설]

세트는 수학에서 배우는 집합(set)과 동일한 역할을 하는 Python의 자료형으로, 중괄호{ }를 이용하여 리스트와 같이 다양한 요소들을 저장할 수 있다. 세트는 순서가 정해져 있지 않으며(unordered), 중복된 요소는 저장되지 않는다는 특징이 있다.

```
❶ asia = {'한국', '중국', '일본'}
❷ asia.add('베트남')
❸ asia.add('중국')
❹ asia.remove('일본')
❺ asia.update({'한국', '홍콩', '태국'})
❻ print(asia)
```

❶ 세트 asia에 '한국', '중국', '일본'의 3개 요소를 저장한다.

asia	'한국'	'중국'	'일본'

❷ 세트 asia에 '베트남'을 추가한다.

asia	'한국'	'중국'	'일본'	'베트남'

❸ 세트 asia에 '중국'을 추가한다. asia에는 이미 '중국' 요소가 있으므로 무시된다.

❹ 세트 asia에서 '일본'을 제거한다.

asia	'한국'	'중국'	'베트남'

❺ 세트 asia에 새로운 세트를 추가하여 확장한다. 새로운 세트 {'한국', '홍콩', '태국'}의 요소 중 '한국'은 이미 asia에 있으므로 무시된다.

asia	'한국'	'중국'	'베트남'	'홍콩'	'태국'

❻ 세트 asia를 출력한다. 세트는 순서가 정해져 있지 않으므로 출력되는 요소들의 순서는 바뀔 수 있다.

결과 {'한국', '중국', '베트남', '홍콩', '태국'}

[Python의 활용]

4. 다음 Python으로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)

```
a = [1, 2, 3, 4, 5]
a = list(map(lambda num : num + 100, a))
print(a)
```

답 : [101, 102, 103, 104, 105]

[해설]

```
❶ a = [1, 2, 3, 4, 5]
❷ a = list(map(lambda num : num + 100, a))
❸ print(a)
```

❶ 5개의 요소를 갖는 리스트 a를 선언한다.

❷ a의 각 요소에 100을 더하는 람다 식을 적용한 후, 100씩 더해진 값들을 다시 리스트로 구성하여 a에 저장한다.

```
a = list(map(lambda num : num + 100, a))
           ㉠
       ㉡
   ㉢
```

- ㉠ lambda num : num + 100 : 인수로 입력된 값에 100을 더하는 람다 식을 정의한다.
- ㉡ map(㉠, a) : 리스트 a의 각 요소를 ㉠에 적용한다.

a	**1**	2	3	4	5	→	lambda 1 : 1 + 100	→ 101 반환
a	1	**2**	3	4	5	→	lambda 2 : 2 + 100	→ 102 반환
a	1	2	**3**	4	5	→	lambda 3 : 3 + 100	→ 103 반환
a	1	2	3	**4**	5	→	lambda 4 : 4 + 100	→ 104 반환
a	1	2	3	4	**5**	→	lambda 5 : 5 + 100	→ 105 반환

- ㉢ a = list(㉡) : ㉡의 실행 결과로 반환되는 값들을 리스트로 구성하여 a에 저장한다.

a	101	102	103	104	105

❸ a를 출력한다. a는 리스트이므로, 리스트를 선언할 때와 같은 형태와 순서로 출력한다.

결과 [101, 102, 103, 104, 105]

5. 다음 Python로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)

```
def func(num1, num2 = 2):
    print('a =', num1, 'b =', num2)
func(20)
```

답 : a = 20 b = 2

[해설]

```
❷ def func(num1, num2 = 2):
❸     print('a =', num1, 'b =', num2)
❶ func(20)
```

func 메소드를 정의하는 부분의 다음 줄인 3번째 줄부터 실행한다.

❶ 20을 인수로 func( ) 메소드를 호출한다.

❷ func( ) 메소드의 시작점이다. ❶번에서 전달받은 20을 num1이 받는다.

- func( ) 메소드의 매개 변수는 num1, num2 두 개지만 num2는 메소드 정의 시 초기값이 지정되었다.
- 전달된 인수는 매개 변수에 차례로 전달되므로 인수가 하나만 주어지면 num1이 인수를 전달받고, 두 개의 인수가 주어지면 num1과 num2가 차례로 인수를 전달받는다.

❸ a =와 num1의 값 20, b =와 num2의 값 2를 차례대로 출력한다.

결과 a = 20 b = 2

6. 다음 Python으로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)

```
class CharClass:
    a = ['Seoul', 'Kyeongi', 'Inchon', 'Daejeon', 'Daegu', 'Pusan'];
myVar = CharClass()
str01 = ' '
for i in myVar.a:
    str01 = str01 + i[0]
print(str01)
```

답 : SKIDDP

[해설]

```
class CharClass:                          클래스 CharClass를 정의한다.
    a = ['Seoul', 'Kyeongi', 'Inchon', 'Daejeon', 'Daegu', 'Pusan'];
                                          클래스의 속성(변수) a에 6개의 요소를 리스트로 저장한다.
❶ myVar = CharClass( )
❷ str01 = ' '
❸ for i in myVar.a:
❹     str01 = str01 + i[0]
❺ print(str01)
```

❶ CharClass의 객체 변수 myVar를 선언한다.

	myVar.a[0]	myVar.a[1]	myVar.a[2]	myVar.a[3]	myVar.a[4]	myVar.a[5]
myVar.a	'Seoul'	'Kyeongi'	'Inchon'	'Daejeon'	'Daegu'	'Pusan'

❷ 변수 str01을 선언하고, 작은 따옴표 두 개를 이어붙인 빈 문자열을 저장한다.

※ Python은 자료형을 별도로 선언하지 않으므로, 이와 같은 방식으로 해당 변수가 어떤 형식으로 사용될 것인지 지정할 수 있다. 여기서는 ❹번의 연산에서 + 기호로 문자 더하기 연산을 수행하기 위해 지정하였다.

❸ 객체 변수 myVar의 리스트 a의 요소 수만큼 ❹번 문장을 반복 수행한다. 리스트 a는 6개의 요소를 가지므로 각 요소를 i에 할당하면서 ❹번을 6회 수행한다.

❹ str01과 i에 저장된 문자열의 첫 번째 글자(i[0])를 더하여 str01에 저장한다. 즉 str01에 저장된 문자 뒤에 i에 저장된 문자열의 첫 번째 글자가 덧붙여진다.

- **1회** : i에 myVar.a[0]이 저장되고 i의 0번째 글자 S가 str01에 저장된다.

str01		i[0]	i[1]	i[2]	i[3]	i[4]		i
'S'	←	'S'	'e'	'o'	'u'	'l'	←	'Seoul'

- **2회** : i에 myVar.a[1]이 저장되고 i의 0번째 글자 K가 str01에 더해진다.

str01		i[0]	i[1]	i[2]	i[3]	i[4]	i[5]	i[6]		i
'SK'	←	'K'	'y'	'e'	'o'	'n'	'g'	'i'	←	'Kyeongi'

- **3회** : i에 myVar.a[2]가 저장되고 i의 0번째 글자 I가 str01에 더해진다.

str01		i[0]	i[1]	i[2]	i[3]	i[4]	i[5]		i
'SKI'	←	'I'	'n'	'c'	'h'	'o'	'n'	←	'Inchon'

- **4회** : i에 myVar.a[3]이 저장되고 i의 0번째 글자 D가 str01에 더해진다.

str01		i[0]	i[1]	i[2]	i[3]	i[4]	i[5]	i[6]		i
'SKID'	←	'D'	'a'	'e'	'j'	'e'	'o'	'n'	←	'Daejeon'

- **5회** : i에 myVar.a[4]가 저장되고 i의 0번째 글자 D가 str01에 더해진다.

str01		i[0]	i[1]	i[2]	i[3]	i[4]		i
'SKIDD'	←	'D'	'a'	'e'	'g'	'u'	←	'Daegu'

- **6회** : i에 myVar.a[5]가 저장되고 i의 0번째 글자 P가 str01에 더해진다.

str01		i[0]	i[1]	i[2]	i[3]	i[4]		i
'SKIDDP'	←	'P'	'u'	's'	'a'	'n'	←	'Pusan'

결과 SKIDDP