꿈꾸는 라이언

정보처리기사 실기

기출풀이집 - 프로그래밍 언어편

꿈꾸는 라이언

목 차

안녕하세요! **꿈꾸는라이언**입니다.
먼저 정처기 필기 합격을 진심으로 축하드려요! 🎉🎊

이제 실기를 준비하시는 분들을 위해 **프로그래밍 언어편 요약 노트**를 준비했습니다.

요즘 정처기 실기 시험의 프로그래밍 언어 비중이 매우 높아졌어요. 전체 20문항 중 8~10문제가 프로그래밍 언어 관련 문제라니 절반에 가까운 수준이죠. 😳 따라서 프로그래밍 언어를 제대로 준비하지 않으면 아무리 이론을 잘 알아도 합격이 쉽지 않은게 현실입니다.

실기에서 다루는 프로그래밍 언어는 C언어, 자바, 파이썬인데요, **출제 비중은 C언어(3~4문제) > 자바(2~3문제) > 파이썬(1~2문제) 순**이에요. 그래서 점수의 비중으로는 C언어가 가장 중요하고, 그 다음으로 자바와 파이썬 순이에요!
하지만 이 노트에서는 오히려 정말 유형도 적고 이해하기 쉬운 파이썬부터 시작해서 자바, C언어 순서로 정리했어요.
처음 프로그래밍 언어를 접하시는 분들도 차근차근 따라오실 수 있게요! 😉

1. **프로그래밍 언어에서는 기출 유형이 반복돼요!** 그래서 20년부터 23년도까지의 모든 프로그래밍 언어 기출문제를 분석하여 자주 출제되는 유형을 요약했어요. 단순 암기식 문제부터 응용력을 요구하는 문제까지 다양한 레벨의 문제가 있지만, 결국 같은 유형이 반복되고 있었어요. 이 유형만 확실히 잡으시면 프로그래밍 언어 문제는 충분히 고득점 받으실 수 있을 거예요! 👍

2. **모든 코드를 이해할 필요는 없어요!** 코드 한줄 한줄 이해를 하고 넘어가면 정말 좋지만, 문제 풀이에 있어서는 출제 유형을 빠르게 파악하는 것이 관건입니다. 실제로 많은 문제들이 숫자만 바뀌어서 동일한 유형으로 출제되거든요. 예를 들어 완전수를 찾는 문제가 매번 나오는데, 사실 완전수는 100이하에서는 6과 28뿐이에요. 그런데 문제의 코드를 전부 해석하려 하면 손으로 100번까지 계산해야 할 수도 있어요. 😱 하지만 완전수의 특징만 알고 있다면 30초 안에 답을 찾을 수 있죠. 그러니 모든 코드를 이해하는데 집중하기보다는 먼저 어떤 유형의 문제인지를 캐치하는 것이 효과적인 전략이 될 거예요. 이를 위해 제가 정리한 분석을 통해 반복적으로 출제되는 유형을 정확히 파악하시는 걸 추천드려요!
   추가로 각 기출 문제마다 아래와 같이 난이도를 표시했어요! **낮은 난이도부터 우선해서 학습해주세요!**
   ★☆☆ : 반드시 맞춰야 하는 쉬운 문제
   ★★☆ : 처음에는 다소 어렵지만 합격을 위해서는 도전해야 하는 문제
   ★★★ : 실제 시험에서는 풀지 말아야할 정도로 어려운 문제

   프로그래밍의 실행 순서가 복잡한 문제의 경우, 작은 숫자(①②…)로 코드 진행 순서를 각 코드 앞에 표기해놨어요!

3. **유튜브에 풀이가 있어요!** 모든 문제에 대한 효율적인 풀이 및 코드에 대한 한 줄 한 줄 설명을 적었지만, 이해가 어려운 경우에는 모든 문제에 기출 연도와 회차를 적어두었으니 유튜브에 검색해서 자세한 설명을 보시는 것도 가능해요! 그리고, 프로그래밍 기출 문제는 사람들이 각자 복원한 문제이기에 정리마다 조금씩 코드가 다를 수는 있어요. 하지만, 문제의 유형은 동일하니 걱정하지 않으셔도 됩니다.

이 <정처기 실기 요약노트 - 프로그래밍 언어편>으로 공부하시면서 추가 궁금하신 사항이나 의견 있으실 분들을 위해 **꿈꾸는라이언 공식 카페**에 **'정처기 실기 질의응답 게시판'**을 별도 만들었어요! 카페 가입하시고 질문 남겨주시면 본 요약노트 제작에 많은 도움을 준 현업 개발자 분께서 바로는 아니더라도 틈틈이 답변해주실꺼에요! 😃👍
문의주신 모든 분들께 최대한 가능한 선에서 성심껏 빠르게 답변해드린다고 하니 참고해주세요!
📌 꿈꾸는라이언 네이버 카페 : https://cafe.naver.com/dreamingryan

그럼 모두들 정처기 합격을 시작으로 각자 계획하셨던 목표와 소중한 꿈을 이루시길 진심으로 응원하겠습니다.
다들 포기하지 마시고, 끝까지 함께 달려봐요!
합격 미리 축하드립니다!! 💜🎉

이것만 이해해도 문제 풀이에 정말 큰 도움이 돼요

★ 집합(Set)

집합은 순서가 없고 중복을 허용하지 않는 자료형이에요.
아래와 같이 사용할 수 있어요.

 # 집합 생성

```
fruit_basket = {"apple", "banana", "grape"}
```

 # 집합에 요소 추가

```
fruit_basket.add("orange")
print(fruit_basket) ⟶ 출력: {"apple", "banana", "grape", "orange"}
```

 # 집합에서 요소 제거

```
fruit_basket.remove("banana")
print(fruit_basket) ⟶ 출력: {"apple", "grape", "orange"}
```

 # 집합 업데이트

```
more_fruits = ["kiwi", "pear", "apple"]
fruit_basket.update(more_fruits)
print(fruit_basket) ⟶ 출력: {"apple", "grape", "orange", "kiwi", "pear"}
```

★ 비트 연산자(Bitwise Operator)

2진수 형태로 데이터를 다룰 때 사용해요.
비트 연산자를 이해하기 쉬운 예시로는 "전등 스위치"를 들 수 있어요.
전등 스위치는 켜짐(1)과 꺼짐(0), 두 가지 상태만 있어요.
*여기서 켜짐은 True, 꺼짐은 False를 의미해요.

AND 연산(&)은 두 개의 스위치가 모두 켜져 있을 때만 전등이 켜지는 거예요.
OR 연산(|)은 두 개의 스위치 중 하나라도 켜져 있으면 전등이 켜지는 거예요.
XOR 연산(^)은 두 개의 스위치 중 하나만 켜져 있을 때 전등이 켜지는 거예요.
(서로의 결과가 달라야 전등이 켜진다는 의미에요)

아래와 같이 사용할 수 있어요. 내가 연산자를 보고 계산할 수 있으면 돼요.

 # AND 연산

```
result = 5 & 3 ⟶ 101 & 011 = 001
print(result) ⟶ 출력: 1
```

 # OR 연산

```
result = 5 | 3 ⟶ 101 | 011 = 111
print(result) ⟶ 출력: 7
```

 # XOR 연산

```
result = 5 ^ 3 ⟶ 101 ^ 011 = 110
print(result) ⟶ 출력: 6
```

 # 비트 시프트 연산

왼쪽으로 한칸 이동은 맨 오른쪽에 0하나를 추가한다는 의미에요.

```
result = 5 << 1 ⟶ 101 << 1 = 1010
print(result) ⟶ 출력: 10
```

오른쪽으로 한칸 이동은 맨 오른쪽 값을 지운다는 의미에요.
여기서 2진수 010과 10은 같은 2를 뜻해요.

```
result = 5 >> 1 ⟶ 101 >> 1 = 010
print(result) ⟶ 출력: 2
```

★ split()

문자열을 지정된 구분자를 기준으로 분할하여 리스트로 반환합니다.
기본 구분자는 공백입니다.

 # 공백을 기준으로 문자열 분할

```
sentence = "Hello world! How are you?"
words = sentence.split() ⟶ sentence.split(" ") 과 동일한 표현
print(words) ⟶ 출력: ['Hello', 'world!', 'How', 'are', 'you?']
```

★ 슬라이싱(Slicing)

리스트나 문자열에서 일부분을 잘라내는 것을 말해요.
Index의 시작점은 0부터 시작해요.
끝점의 index는 자를 때 포함하지 않아요.
[0:5] 는 첫 번째 문자부터 시작해서 0, 1, 2 ,3, 4까지의 문자를 뜻해요.

처음부터 자르고 싶으면 시작점을 생략할 수 있어요.
[:5]
끝까지 자르고 싶으면 끝점을 생략할 수 있어요.
[8:]

 # 문자열 슬라이싱

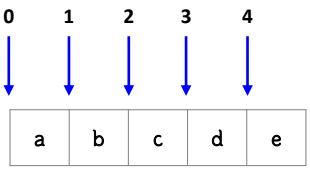
```
message = "Hello, World!"
print(message[0:5]) ⟶ 출력: "Hello"
print(message[:5]) ⟶ 출력: "Hello"
print(message[7:]) ⟶ 출력: "World!"
print(message[7:12]) ⟶ 출력: "World"
print(message[::2]) ⟶ 출력: "Hlo ol!" → 홀수번째 문자열 추출!
```

(2 page 2번 문제 참고)

 # 리스트 슬라이싱

```
numbers = ["a", "b", "c", "d", "e"]
print(numbers[1:4]) ⟶ 출력: ["b", "c", "d"]
print(numbers[:3]) ⟶ 출력: ["a", "b", "c"]
print(numbers[3:]) ⟶ 출력: ["d", "e"]
print(numbers[::2]) ⟶ 출력: ["a", "c", "e"]
```

→ 홀수번째 문자열 추출!
(2 page 2번 문제 참고)



numbers[1:4] 슬라이싱 시,
인덱스 1에서 시작해서
인덱스 4 앞까지 추출
(**인덱스 3**까지만 포함)

★ 문자열 포맷팅 (String Formatting)

문자열 내에 특정 값을 삽입하는 방법이에요.
% 연산자를 사용하여 문자열 포맷팅을 할 수 있어요.

 # % 연산자

```
name = "Alice"
age = 25

print("My name is %s. I'm %d years old." % (name, age))
My name is Alice. I'm 25 years old.
```

★ map()

리스트나 튜플 등의 이터러블 객체의 각 요소에 함수를 적용하여 새로운 이터러블 객체를 반환합니다.
map()은 원본 데이터를 변경하지 않고 새로운 데이터를 생성합니다.

numbers = [1, 2, 3, 4, 5]

 # lambda 함수를 사용하여 각 요소에 10을 더하기

```
numbers = [1, 2, 3, 4, 5]
add_ten = map(lambda x: x + 10, numbers)
print(list(add_ten)) ⟶ 출력: [11, 12, 13, 14, 15]
```

1. [기출 예제] 23년 3회 실기 (난이도: ★☆☆)

다음 Python 프로그램과 그 <실행결과>를 분석하여 괄호에 들어갈 알맞은 예약어를 쓰시오.
첫 번째 라인의 '5 10'은 입력받은 값에 해당한다.
<실행결과>
x, y의 값을 공백으로 구분하여 입력: 5 10
x의 값: 5
y의 값: 10

▶ 정답 split

정답은 split 이에요
split() 메서드는 문자열을 특정 구분자를 기준으로 나누어 리스트로 반환해요.
기본적으로는 공백을 구분자로 사용해요.

split() 메서드는 사용자 입력을 받아 분리할 때 자주 사용돼요.

```
1  x, y = input("x, y의 값을 공백으로 구분하여 입력: ").( 빈 칸 )(" ")
2  print("x의 값: ", x)
3  print("y의 값: ", y)
```

input() 함수는 사용자로부터 입력을 받는 기능을 해요.
"x, y의 값을 공백으로 구분하여 입력: "은
사용자에게 무엇을 입력해야 하는지 알려주는 안내 문구예요.

★ key point
사용자가 5 10 입력한 것을 실행결과에서 알려주고 있어요.
"5 10"이라는 값을 x, y로 공백 기준으로 값을 할당하기 위해서는 공백 기준으로 문자열을 나눠야해요.
문자열을 특정 구분자로 나누는 키워드는 split이에요.
그래서 정답이 split이 돼요.

*구분자를 넘겨주지 않는 split() 은 split(" ") 과 동일해요. 즉, 공백 문자열이 기본값이에요.

2. [기출 예제] 23년 2회 실기 (난이도: ★☆☆)

다음 파이썬 코드에서 알맞는 출력값을 작성하시오.

▶ 정답 engneing

```
1  a = "engineer information processing"
2  b = a[:3]
3  c = a[4:6]
4  d = a[28:]
5  e=b+c+d
6  print(e)
```

변수 a에 문자열 "engineer information processing"를 할당해요.

a 문자열의 처음부터 3번째 문자까지 가져와 b에 저장합니다.
여기서 a[:3]은 a의 인덱스 0부터 인덱스 2까지를 의미해요.
따라서 b에는 "eng"가 저장됩니다.

a 문자열에서 5번째부터 6번째 문자까지 가져와 c에 저장합니다.
(인덱스 4부터 인덱스 5까지)
여기서 a[4:6]은 "ne"을 의미해요.

변수 d에 a의 29번째 문자부터 끝까지의 슬라이스를 할당해요.
슬라이싱에서 두 번째 인덱스를 생략하면 문자열의 끝까지 슬라이스 해요.
a의 29번째 문자부터 끝까지는 "ing" 부분 문자열 이에요.
따라서 d에는 "ing"이 할당돼요.

변수 e에 b, c, d를 연결한 문자열을 할당해요.
b는 "eng", c는 "ne", d는 "ing"이므로,
e에는 " engneing"이 할당돼요.

[6행] 변수 e의 값인 " engneing"을 출력해요.

★ 슬라이싱 (자주 쓰이니 외우면 좋아요)
sequence[start:end:step]의 형태로 사용돼요.

start는 슬라이스가 시작되는 인덱스이고,
end는 슬라이스가 끝나는 인덱스이며,
[해당 인덱스는 문자열에서 제외]
step은 슬라이스가 진행되는 간격이에요.

start와 end는 생략할 수 있고, 생략하면 각각
시퀀스의 시작과 끝이 기본값으로 사용돼요.
step도 생략할 수 있고,
생략하면 기본값 1이 사용돼요.

★ 집합 (자주 쓰이니 외우면 좋아요)
중복되지 않는 요소들의 모음이에요.
같은 값을 여러 번 추가해도 집합에는 한 번만 저장돼요.

순서가 없어요.
집합의 요소들은 특정한 순서를 가지지 않아요.

3. [기출 예제] 23년 1회 실기 (난이도: ★☆☆)

다음 파이썬 코드에서 알맞는 출력값을 작성하시오.

▶ 정답 {'한국', '중국', '베트남', '홍콩', '태국'}

```
1  a = {'한국', '중국', '일본'}
2  a.add('베트남')
3  a.add('중국')
4  a.remove('일본')
5  a.update({'홍콩', '한국', '태국'})
6  print(a)
```

a라는 이름의 집합을 생성하고 초기값으로 '한국', '중국', '일본'을 갖게 해요.

a 집합에 '베트남'을 추가해요.
집합에 새로운 요소를 추가할 때는 add() 메서드를 사용해요.
이 연산 후에 a는 {'한국', '중국', '일본', '베트남'}이 돼요.

a 집합에 '중국'을 추가하려고 해요.
하지만 '중국'은 이미 a에 존재하는 요소예요.
집합은 중복된 요소를 허용하지 않기 때문에
이 연산은 a를 변화시키지 않아요.
이 연산 후에도 a는 여전히 {'한국', '중국', '일본', '베트남'}이에요.

a 집합에서 '일본'을 제거해요.
집합에서 특정 요소를 제거할 때는 remove() 메서드를 사용해요.
이 연산 후에 a는 {'한국', '중국', '베트남'}이 돼요.

최종적인 a 집합을 출력해요.
정답은 {'한국', '중국', '베트남', '홍콩', '태국'}이에요

a 집합에 다른 집합의 요소들을 추가해요.
update() 메서드는 인자로 주어진 집합의 모든 요소를 a에 추가해요.
'한국'은 이미 a에 존재하므로 중복 추가되지 않아요.

4. [기출 예제] 22년 3회 실기 (난이도: ★★☆)

▶ 정답 [101,102,103,104,105]

다음 파이썬 코드에 대한 출력값을 작성하시오.

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

TestList라는 이름의 리스트를 생성하고 초기값으로 [1,2,3,4,5]를 할당해요.

결과적으로, TestList의 각 요소에 100을 더한 새로운 리스트를 생성해요.

map() 함수는 첫 번째 인자로 함수를, 두 번째 인자로 이터러블(리스트 등)을 받아요.
map() 함수는 **이터러블의 각 요소에 함수를 적용한 결과를 반환**해요.

여기 코드에서는 람다 표현식 lambda num : num + 100이 map() 함수의 첫 번째 인자로 사용되었어요.
이 람다 표현식은 인자 num을 받아 num + 100을 반환하는 함수예요.
map()의 두 번째 인자로는 TestList가 전달되었어요.

따라서, map()은 TestList의 각 요소에 람다 함수를 적용해요.
1 + 100 = 101
2 + 100 = 102
3 + 100 = 103
4 + 100 = 104
5 + 100 = 105

map()의 결과는 맵 객체(map object)로 리스트로 변환하기 위해 list() 함수를 사용했어요.
최종적으로 TestList에는 [101, 102, 103, 104, 105]가 할당돼요.

정답은 [101, 102, 103, 104, 105] 이 출력돼요.

5. [기출 예제] 22년 2회 실기 (난이도: ★★☆)

▶ 정답 REMEMBER AND STR

다음 파이썬 코드에 대한 출력값을 작성하시오.

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

a라는 변수에 문자열 "REMEMBER NOVEMBER"를 할당해요.

문자열 슬라이싱을 사용하여 새로운 문자열을 만들어 b에 할당하는 부분이에요.

a[:3]은 a의 처음부터 3번째 문자까지의 슬라이스로, "REM"이에요.
(인덱스 0부터 인덱스 2까지)
a[12:16]은 a의 13번째 문자부터 16번째 문자까지의 슬라이스로, "EMBE"예요.
(인덱스 12부터 인덱스 15까지)

두 슬라이스를 **+ 연산자로 연결하면 "REMEMBE"**가 돼요.
따라서 b에는 "REMEMBE"가 할당돼요.

문자열 포맷팅을 사용하여 새로운 문자열을 만들어 c에 할당하는 부분이에요.
"R AND %s"는 포맷 문자열로, %s 부분에 다른 문자열이 들어갈 자리예요.
% 연산자 뒤에 오는 "STR"이 %s 자리에 들어가요.
따라서 c에는 "R AND STR"이 할당돼요.

b와 c를 연결한 문자열을 출력하는 부분이에요.
b는 "REMEMBE", c는 "R AND STR"이에요.
두 문자열을 + 연산자로 연결하면 "REMEMBER AND STR"이 돼요.

정답은 REMEMBER AND STR 이에요.

6. [기출 예제] 22년 1회 실기 (난이도: ★☆☆)

▶ 정답 a= 20 b= 2

다음 파이썬 코드에 대한 출력값을 작성하시오.

```
1  def exam(num1, num2=2):
2    print('a=', num1, 'b=', num2)
3  exam(20)
```

이 코드의 특징은 함수의 매개변수에 기본값을 지정하는 방법을 보여줘요.

함수 정의에서 **num2=2와 같이 매개변수에 =로 값을 할당**하면, **그 값이 해당 매개변수의 기본값**이 돼요.
함수를 호출할 때 해당 매개변수에 값을 전달하지 않으면, 기본값이 사용돼요.

여기서는, exam(20)은 num1에 20을 전달하고 num2에는 아무것도 전달하지 않았어요
따라서, num1은 20, num2는 기본값 2가 사용돼요.

그러므로 이 코드의 출력 결과는 a= 20 b= 2 에요.

7. [기출 예제] 21년 3회 실기 (난이도: ★☆☆)

다음 파이썬 코드에 대한 출력값을 작성하시오.

▶ 정답 False

```
1  a,b = 100, 200
2  print(a==b)
```

a에 100을, b에 200을 할당하는 거예요.
이렇게 하면 a와 b에 각각 다른 값이 저장돼요.

a와 b가 같은지 확인하는 거예요.

==은 두 값이 같은지 비교하는 연산자예요.
만약 a와 b가 같다면 True를,
다르다면 False를 결과로 돌려줘요.

a는 100, b는 200이니까
당연히 둘은 같지 않아요.
그래서 a==b의 결과는 False가 되는 거죠.

★ 연산자 종류

관계	==	같다	>	크다	<	작다
	!=	같지 않다	>=	크거나 같다	<=	작거나 같다
비트	&	and	모든 비트가 1이면 1			
	^	xor	모든 비트가 같으면 0, 하나라도 다르면 1			
	|	or	모든 비트 중 하나라도 1이면 1			
	~	not	각 비트의 부정 (0이면 1 산출)			
	<<	-	비트를 왼쪽으로 이동			
	>>	-	비트를 오른쪽으로 이동			
논리	!	not	&&	and	||	or

8. [기출 예제] 21년 2회 실기 (난이도: ★★★)

다음 파이썬 코드에 대한 출력값을 작성하시오.

▶ 정답 26

```
1  a = 100
2  result = 0
3  for i in range(1,3):
4     result = a >> i
5     result = result + 1
6  print(result)
```

a에 100을 할당하고,
result에 0을 할당했어요.
이렇게 하면 a는 100, result는 0으로 초기화되는 거예요.

for 루프를 사용해서 i를 1부터 2까지 반복해요.
range(시작 숫자, 끝 숫자) 형태를 사용하는데, 이때 끝 숫자는 포함되지 않습니다.
슬라이스와 비슷하게 인덱스 시작 숫자 이상 인덱스 끝 숫자 – 1 이하 까지를 뜻해요.

루프 안에서는 아래 로직을 수행해요.

result = a >> i는 a를 오른쪽으로 i번 시프트한 결과를 result에 할당해요.
>>는 비트 오른쪽 시프트 연산자예요.
이 연산자는 a의 비트를 오른쪽으로 i번 이동시켜요.

조금 어려울 수 있어요.
첫 번째 반복(i=1)에서 a >> 1은 a의 비트를 오른쪽으로 1번 이동시키는 거예요.
2진수로 표현하면 1100100(기존 100)에서 0110010(= 110010)(비트 이동)이 되죠.
이것은 10진수로 50이에요.
여기에 1을 더하면 51이 돼요.

두 번째 반복(i=2)에서는 a >> 2로 a의 비트를 오른쪽으로 2번 이동시켜요.
2진수로 1100100(50)에서 0011001(= 11001)이 되는데, 이것은 10진수로 25예요.
여기에 1을 더하면 26이 돼요.

루프가 끝나면 print(result)로 최종 result의 값을 출력해요.
정답은 26이에요.

★ key point
비트 연산자가 나오면, 2진수로 변환 후
왼쪽 혹은 오른쪽으로 이동 후 다시
10진수로 변환한다고 생각하면 돼요.

9. [기출 예제] 21년 1회 실기 (난이도: ★☆☆)

다음 파이썬 코드에 대한 출력값을 작성하시오.

▶ 정답 skiddp

```
1  class good :
2      li = ["seoul", "kyeonggi","inchon","daejeon","daegu","pusan"]
3
4  g = good()
5  str01 = ''
6  for i in g.li:
7      str01 = str01 + i[0]
8
9  print(str01)
```

good이라는 클래스를 정의했어요.

good 클래스의 인스턴스를 생성해서 g라는 변수에 할당하는 거예요.
이렇게 하면 g를 통해 good 클래스의 속성과 메서드에 접근할 수 있게 돼요.

빈 문자열을 str01이라는 변수에 할당하는 거예요.
이 변수는 앞으로 도시 이름의 첫 글자를 모으는 데 사용될 거예요.

for 루프를 사용해서 g.li의 각 요소를 i라는 변수에 하나씩 할당하면서 반복해요.
여기서는 순서대로, seoul, kyeonggi … 이 값들이 i에 할당돼요.

str01 = str01 + i[0]은 현재 str01의 값에 i의 첫 번째 문자(i[0])를 더하는 거예요.
문자열에서 배열처럼 index로 문자 하나씩 가져올 수 있어요.

첫 번째 반복에서는 "seoul"의 첫 글자 "s"가 str01에 더해져요.
두 번째 반복에서는 "kyeonggi"의 첫 글자 "k"가 str01에 더해져요.
이런 식으로 계속 반복되면서 str01에는 각 도시 이름의 첫 글자가 모이게 돼요.

루프가 끝나면 print(str01)로 최종 str01의 값을 출력해요.
정답은 skiddp 이에요.

▶ 정답
[1, 2, 3]
7
123
45
6789

10. [기출 예제] 20년 4회 실기 (난이도: ★★★)

다음 파이썬 코드에 대한 출력값을 작성하시오.

```
1  lol = [[1,2,3],[4,5],[6,7,8,9]]
2  print(lol[0])
3  print(lol[2][1])
4  for sub in lol:
5      for item in sub:
6          print(item, end = '')
7      print()
```

리스트들을 요소로 가지고 있는 2차원 리스트예요.

lol의 첫 번째 요소를 출력하는 거예요. lol[0]은 [1,2,3]이므로, 이 코드는 [1,2,3]을 출력해요.

세 번째 요소의 두 번째 요소를 출력하는 거예요.
lol[2]는 [6,7,8,9]이고, lol[2][1]은 7이므로, 이 코드는 7을 출력해요.

중첩 for 루프를 사용하고 있어요.
바깥쪽 루프에서는 lol의 각 요소(리스트)를 sub라는 변수에 하나씩 할당하면서 반복해요.

안쪽 루프에서는 각 sub의 요소를 item이라는 변수에 하나씩 할당하면서 반복해요.

현재 item의 값을 출력하는데,
end = ''는 출력 후 줄바꿈을 하지 않고 바로 다음 출력을 이어서 하라는 의미예요.
이렇게 하면 각 리스트의 요소들이 한 줄에 모두 출력돼요.

안쪽 루프가 끝나면 print()를 사용해서 줄바꿈을 해요.
이렇게 하면 각 리스트의 요소들이 한 줄씩 출력되는 거죠.
item 기준으로는 한줄로 출력하고,
Sub 기준으로는 한줄씩 내려서 출력돼요.

item에 이라는 파이썬 키워드가 있는 건 아니에요.
lol, sub과 동일한 변수명이에요!

첫번째 배열인 [1,2,3] 는 123
두번째 배열인 [4,5] 는 45
세번째 배열인 [6,7,8,9] 는 6789
이렇게 출력돼요.

11. [기출 예제] 20년 2회 실기 (난이도: ★☆☆)

다음 파이썬 코드에 대한 출력값을 작성하시오.

▶ 정답 {'중국', '한국', '베트남', '홍콩', '태국'}

```
1  a={'일본','중국','한국'}
2  a.add('베트남')
3  a.add('중국')
4  a.remove('일본')
5  a.update(['홍콩','한국','태국'])
6  print(a)
```

집합(set) 자료형을 다루고 있어요. 집합은 순서가 없고, 중복된 요소를 허용하지 않는 자료형이에요.
자주 나오고 있어요. 중복되지 않는다는 사실 알고 있으면 좋아요.
중괄호 {}를 사용하여 생성할 수 있고, 빈 집합을 생성할 때는 set()을 사용해요.

a 집합에 '베트남'을 추가하는 거예요. 이 연산 후에 a는 {'일본', '중국', '한국', '베트남'}이 돼요.

a 집합에 '중국'을 추가하려고 하는 거예요. 하지만 '중국'은 이미 a에 존재하는 요소예요.
집합은 중복된 요소를 허용하지 않기 때문에, 이 연산은 a를 변화시키지 않아요.

a 집합에서 '일본'을 제거하는 거예요. 이 연산 후에 a는 {'중국', '한국', '베트남'}이 돼요.

a 집합에 '홍콩', '한국', '태국'을 추가하는 거예요.
update() 메서드는 다른 집합이나 리스트 등의 이터러블 객체를 인자로 받아서,
그 안의 모든 요소를 집합에 추가해요. 이 때, '한국'은 이미 a에 존재하므로 중복 추가되지 않아요.
이 연산 후에 a는 {'중국', '한국', '베트남', '홍콩', '태국'}이 돼요.

최종적인 a 집합을 출력하는 거예요.
{'중국', '한국', '베트남', '홍콩', '태국'}이 출력돼요.