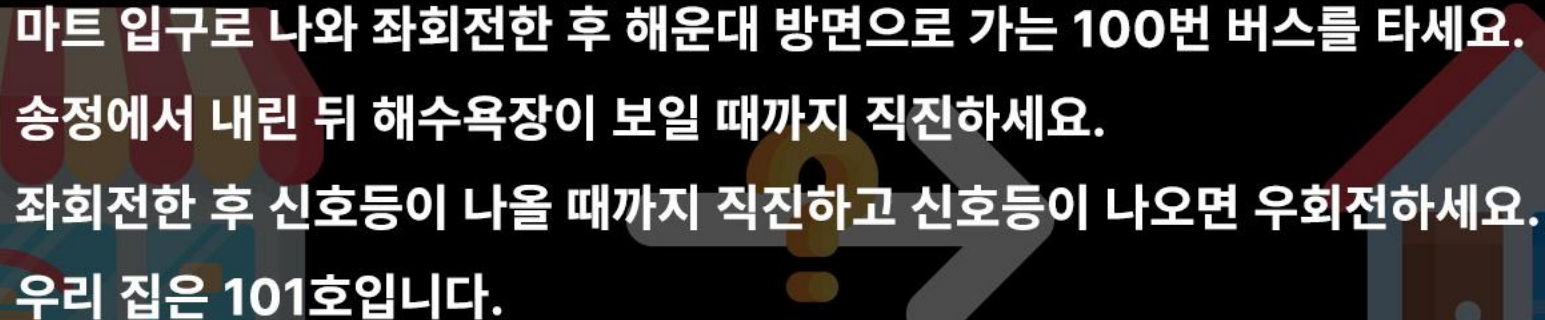


# 선언형 프로그램, 명령형 프로그램



## 명령형



마트 입구로 나와 좌회전한 후 해운대 방향으로 가는 100번 버스를 타세요.  
송정에서 내린 뒤 해수욕장이 보일 때까지 직진하세요.  
좌회전한 후 신호등이 나올 때까지 직진하고 신호등이 나오면 우회전하세요.  
우리 집은 101호입니다.

## 선언형

우리집은 부산 기장군 기장읍 기장해안로 147 입니다.

**HOW**  
(어떻게)

**WHAT**  
(무엇을)

작업을 수행하는 **방법**

수행하는 **작업**

**HOW**  
(어떻게)

작업을 수행하는 **방법**

**WHAT**  
(무엇을)

+ 추상화

수행하는 **작업**

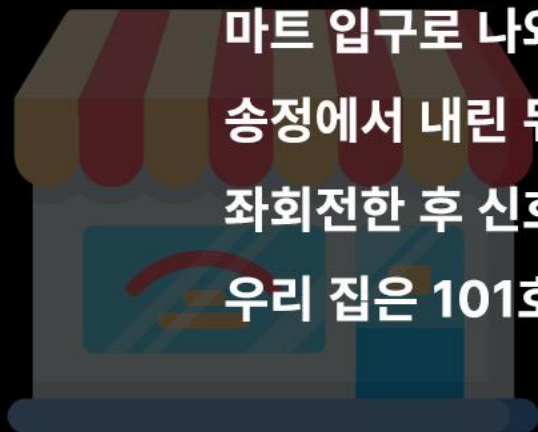
## 명령형

마트 입구로 나와 좌회전한 후 해운대 방향으로 가는 100번 버스를 타세요.

송정에서 내린 뒤 해수욕장이 보일 때까지 직진하세요.

좌회전한 후 신호등이 나올 때까지 직진하고 신호등이 나오면 우회전하세요.

우리 집은 101호입니다.



## 선언형

우리집은 부산 기장군 기장읍 기장해안로 147 입니다.

(+집으로 가는 명령형 단계를 모두 알고 있다고 가정)

명령형

- C
- JAVA
- ASSEMBLY

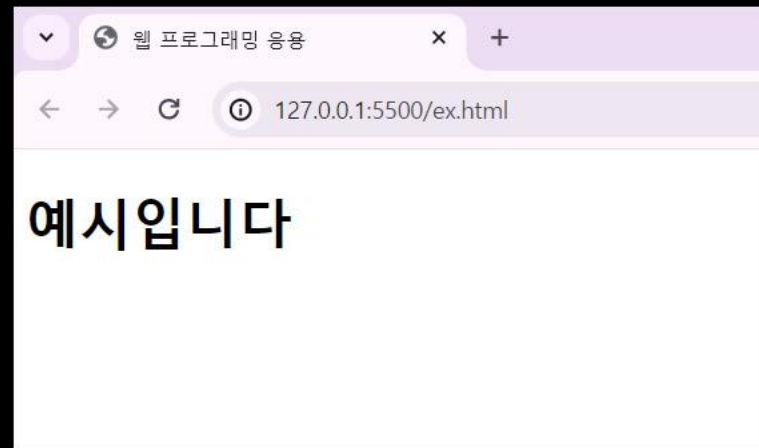
둘 다

- JAVASCRIPT
- PYTHON
- C#

선언형

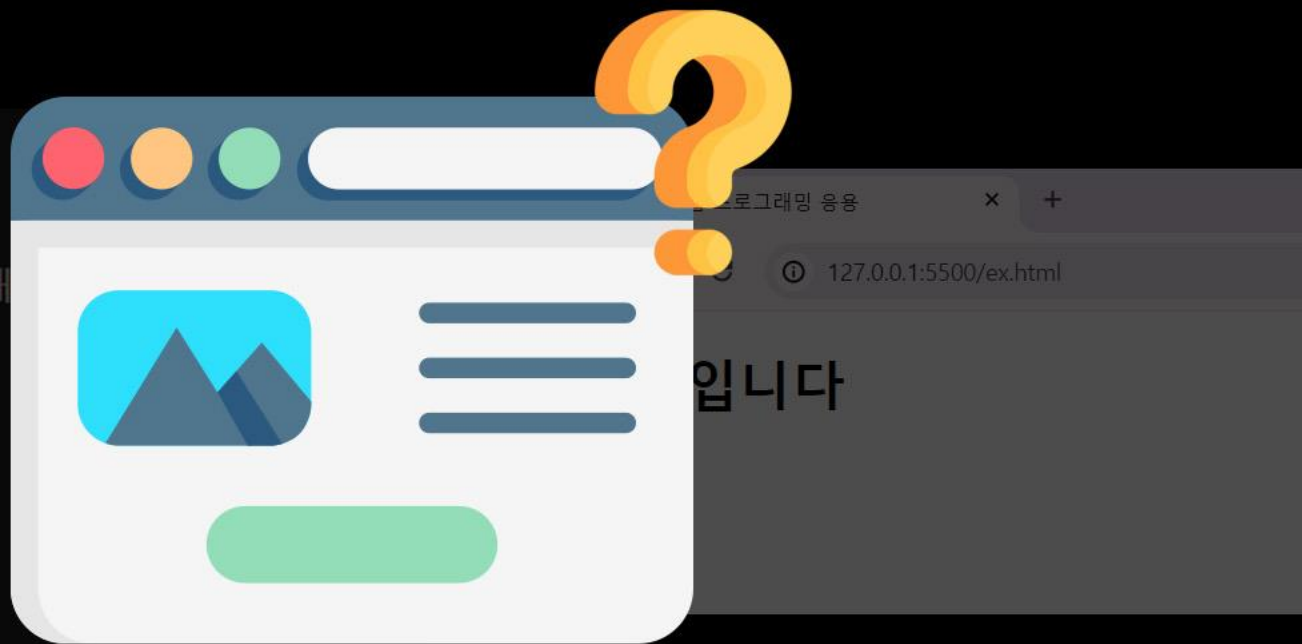
- HTML
- SQL
- XSLT

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>웹 프로그래밍 응용</title>
5    </head>
6    <body>
7      <h1>
8        예시입니다
9      </h1>
10   </body>
11 </html>
```

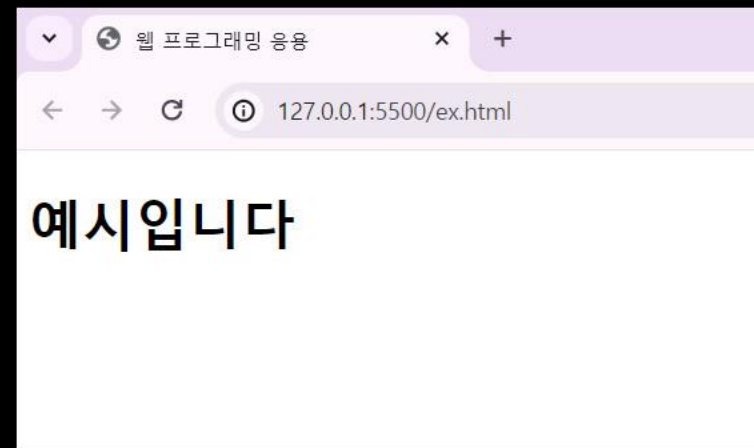




```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>웹 프로그래
5   </head>
6   <body>
7     <h1>
8       예시입니다
9     </h1>
10  </body>
11 </html>
```



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>웹 프로그래밍 응용</title>
5   </head>
6   <body>
7     <h1>
8       예시입니다
9     </h1>
10  </body>
11 </html>
```



원하는 **방식**보다는 **무엇**을 하고 싶은지에 관심

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int a[5];
6
7      for(int i=0; i<5; ++i) {
8          scanf("%d", &a[i]);
9      }
10
11     for(int i=0; i<5; ++i) {
12         a[i] = a[i]*2;
13     }
14
15     return a;
16 }
```

```
      27 |      return a;
        |      ^
1 2 3 4 5
2 4 6 8 10
```

```
1 #include <stdio.h>  
2
```

**a[5]**  
=  
**{1, 2, 3, 4, 5}**

**for문**

**X2**



5  
10

**a[5]**  
=  
**{2, 4, 6, 8, 10}**

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int a[5];
6
7      for(int i=0; i<5; ++i) {
8          scanf("%d", &a[i]);
9      }
10
11     for(int i=0; i<5; ++i) {
12         a[i] = a[i]*2;
13     }
14
15     return a;
16 }
```

```
27 | return a;
   | ^
1 2 3 4 5
2 4 6 8 10
```

기능을 수행하는 방법에 대한 단계를 설명

```
function double (arr) {  
  let results = []  
  for(let i =0; i<arr.length; ++i) {  
    results.push(arr[i]*2)  
  }  
  return results;  
}
```

명령형

```
function double (arr) {  
  return arr.map((item) => item * 2)  
}
```

선언형

## 명령형

**장점** : 모듈 분리 없이 연속적인 계산 과정으로 이루어져 있어 실행 속도가 빠르다

**단점** : 가독성이 떨어지며 복잡할수록 관리가 어렵다

## 선언형

**장점** : 가독성과 코드에 대한 이해도가 높아지며 추상화된 함수를 재사용함으로써 선언적인 코드를 사용할 수 있다

**단점** : 함수 호출 방식으로 인해 메모리 할당량이 증가한다



**Q&A**