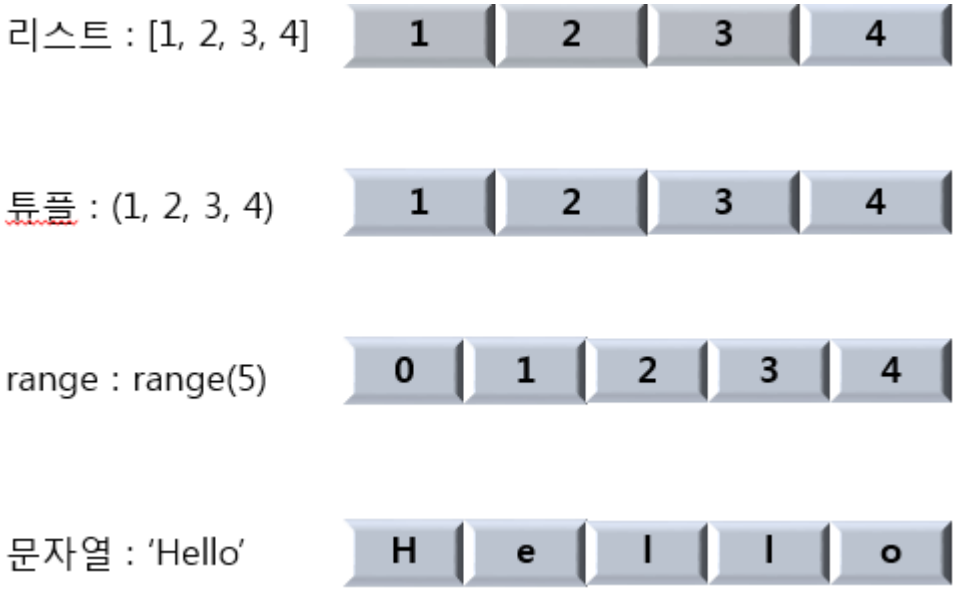
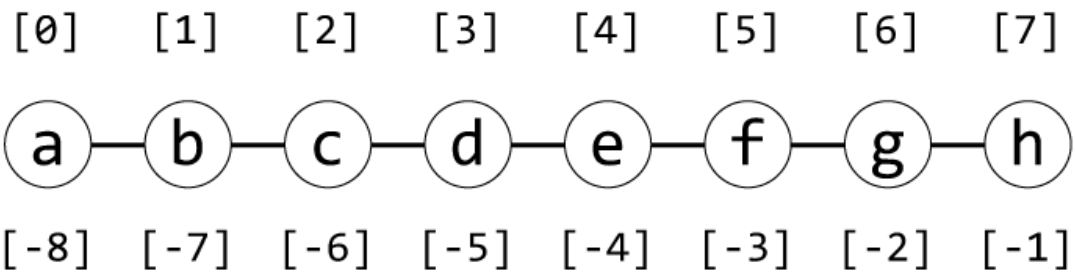
LECTURE 1. 시퀀스(Sequence) 자료

- 시퀀스(sequence)는 위치에 따라 정렬된 항목의 모음
- str, list, tuple, tange, bytearray, bytes



> **시퀀스(Sequence)** : str, list, tuple, tange, bytearray, bytes
> **콜랙션(Collection)** : dict, set, frozenset

1. 첨자 참조(색인) / LAB-1



번호	사용 예	출력(결과)	설 명
1	p = 'abcdefgh'		
	p[0]	'a'	제일 첫번째 항목
2	p = 'abcdefgh'		
	p[7]	'h'	8번째 항목
3	p = 'abcdefgh'		
	p[8]	IndexError: string index out of range	오류 발생
4	p ='abcdefgh'	'h'	

번호	사용 예	출력(결과)	설 명
	p[-1]	'h'	제일 마지막 항목
5	p = 'abcdefgh'		
	p[-8]	'a'	뒤에서 8번째 항목
6	p = 'abcdefgh'		
	for n in range(len(p)) :		
	print(p[n], end=' ')	a b c d e f g h	p[0]~p[7]

LAB-1 첨자 참조(색인)

In [ ]:
```
p = 'abcdefgh'
p[0]
```

In [ ]:
```
p = 'abcdefgh'
p[7]
```

In [ ]:
```
## 오류
p = 'abcdefgh'
p[8]
```

In [ ]:
```
p = 'abcdefgh'
p[-1]
```

In [ ]:
```
p = 'abcdefgh'
p[-8]
```

In [ ]:
```
p = 'abcdefgh'
for n in range(len(p)) : # 0 ~ 8
    print(p[n], end= ' ')
```

(과제)

- 'abcdefgh' 문자열을 거꾸로 출력하시오

In [ ]:
```
p = 'abcdefgh'
for n in range(-1, -len(p)-1, -1) : # -1 ~ -8
    print(p[n])
```

In [ ]:
```
p = 'abcdefgh'
p[-1::-1]
```

2. 슬라이싱 참조 / LAB-2

번호	사용 예	출력(결과)	설 명
1	'python'[1:5]	'ytho'	p[1] ~ p[4]
2	'python'[0:6]	'python'	p[0] ~ p[5]

번호	사용 예	출력(결과)	설 명
3	'python'[0:len('python')]	'python'	p[0] ~ p[5]
4	'python'[1:]	'ython'	p[1] ~ p[5]
5	'python'[:5]	'pytho'	p[0] ~ p[4]
6	'python'[1:2]	'y'	p[1] ~ p[1]
7	'python'[-5:-1]	'y'	p[-5] ~ p[-2]
8	'python'[-len('python'): -1]	'pytho'	p[-6] ~ p[-2]
9	'python'[: -1]	'pytho'	p[-6] ~ p[-2]
10	'python'[-5:]	'ython'	p[-5] ~ p[-1]
11	'python'[:]	'python'	p[0] ~ p[5]
12	'python'[::2]	'pto'	p[0] p[2] p[4]
13	'python'[1::2]	'yhn'	p[1] p[3] p[5]
14	'python'[5:0:-1]	'nohty'	p[5] ~ p[1]
15	'python'[-1::-1]	'nohtyp'	p[-1] ~ p[-6]
16	'python'[::-1]	'nohtyp'	p[-1] ~ p[-6]

LAB-2 슬라이싱 참조

In [ ]:
```
'python'[1:5]
```

In [ ]:
```
'python'[0:6]
```

In [ ]:
```
'python'[0:len('python')]
```

In [ ]:
```
'python'[1:]
```

In [ ]:
```
 'python'[:5]
```

In [ ]:
```
'python'[1:2]
```

In [ ]:
```
'python'[-5:-1]
```

In [ ]:
```
'python'[-len('python'): -1]
```

In [ ]:
```
'python'[: -1]
```

In [ ]:
```
'python'[-5:]
```

In [ ]:
```
 'python'[:]
```

In [ ]:
```
'python'[::2]
```

In [ ]:
```
'python'[1::2]
```

In [ ]:
```
'python'[5:0:-1]
```

In [ ]:
```
'python'[-1::-1]
```

In [ ]:
```
'python'[ ::-1]
```

3. 리스트의 슬라이스로 일부분 수정 / LAB-3

- 리스트 수정 (튜플은 수정 불가)
- 슬라이스 부분을 수정할 때는 리스트로 수정해야함 (특정 항목을 수정할 때는 값을 주면 됨)

> list[start:end] = [item_0, item_1, ... , item_n]
> list[index] = value

번호	사용 예	출력(결과)	설 명
1	sports = ['축구', '족구', '비치사카', '야구', '농구', '배구']		
	sports[0:3] = ['축구']		sports[0]~sports[2] 부분을 '축구'로 대체
	print(sports)	['축구', '야구', '농구', '배구']	
2	t = [1, 2, 3, 4, 5, 6]		
	t[1:3] = [30, 40]		t[1]~t[2] 부분을 [30, 40]으로 대체'
	print(t)	[1, 30, 40, 4, 5, 6]	
3	t = [1, 2, 3, 4, 5, 6]		**(주의)** 특정항목이 리스트 값으로 수정됨
	t[3] = [40]		t[3] 값을 리스트 [40]으로 대체'
	print(t)	[1, 2, 3, [40], 5, 6]	
4	sports = ['축구', '족구', '비치사카', '야구', '농구', '배구']		
	sports[0:2] = '탁구'		문자열 '탁구'가 리스트로 변환된 후, 대체
	print(sports)	['탁', '구', '야구', '농구', '배구']	

LAB-3 리스트의 슬라이스로 일부분 수정

In [ ]:
```
sports = ['축구', '족구', '비치사카', '야구', '농구', '배구']
sports[0:3] = ['축구']
print(sports)
```

In [ ]:
```
t = [1, 2, 3, 4, 5, 6]
t[1:3] = [30, 40]
print(t)
```

In [ ]:
```
t = [1, 2, 3, 4, 5, 6]
t[3] = [40]
print(t)
```

In [ ]:
```
sports = ['축구', '족구', '비치사카', '야구', '농구', '배구']
sports[0:2] = '탁구'
print(sports)
```

4. 시퀀스 연산자 +와 *, 소속 연산자 in / LAB-4

- 수식 연산자 +, *를 시퀀스에도 사용

> 'java' + 'python' : 'javapython'
> 3*'java' : 'javajavajava'
> 'java' in ('c++', 'c#', 'java', 'python') : True

번호	사용 예	출력(결과)	설 명
1	colors = ['red', 'orange']		
	others = ['yellow', 'green']		list와 list를 합친 list 반환
	colors + others	['red', 'orange', 'yellow', 'green']	
2	name = 'dongyang mirae'		
	title = 'university'		str과 str을 합친 strt 반환
	name + '/' + title	'dongyang mirae/university'	
3	'py' * 3	'pypypy'	
4	season = ('봄', '여름') * 2		
	print(season)	('봄', '여름', '봄', '여름')	
5	maze=[0] * 10		list 초기화
	print(maze)	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	
6	'java' in ('c++', 'c#', 'java', 'python')	True	항목에 java 있나요?

번호	사용 예	출력(결과)	설 명
7	'java' in 'C++, C#, java는 프로그래밍 언어'	True	문자열에 java 있나요?
8	'java' not in 'C++, C#, java는 프로그래밍 언어'	False	문자열에 java 나요?

In [ ]:
```
colors = ['red', 'orange']
others = ['yellow', 'green']
colors + others
```

In [ ]:
```
name = 'dongyang mirae'
title = 'university'
name + '/' + title
```

In [ ]:
```
'py' * 3
```

In [ ]:
```
season = ('봄', '여름') * 2
print(season)
```

In [ ]:
```
maze=[0] * 10
print(maze)
```

In [ ]:
```
'java' in ('c++', 'c#', 'java', 'python')
```

In [ ]:
```
'java' in 'C++, C#, java는 프로그래밍 언어'
```

In [ ]:
```
'java' not in 'C++, C#, java는 프로그래밍 언어'
```

LAB-5 (수행과제)

1. 문제 : 웹 서버의 로그 파일에서 각 요청의 클라이언트 IP 주소를 추출
2. 로그 데이터 형식

- **192.168.0.1 - - [21/Jan/2025:10:00:00 +0000] "GET /index.html HTTP/1.1" 200 1024**
- **203.0.113.45 - - [21/Jan/2025:10:01:00 +0000] "POST /login HTTP/1.1" 200 2048**

3. 결과

- IPs: ['192.168.0.1', '203.0.113.45']

> str.strip(), str.split("₩n"), list = []

In [ ]:
```
# 로그 데이터
log_data = """
192.168.0.1 - - [21/Jan/2025:10:00:00 +0000] "GET /index.html HTTP/1.1" 200 1024
203.0.113.45 - - [21/Jan/2025:10:01:00 +0000] "POST /login HTTP/1.1" 200 2048
198.51.100.23 - - [21/Jan/2025:10:02:00 +0000] "GET /dashboard HTTP/1.1" 403 512
"""
```

```
# 로그 데이터를 한 줄씩 나누기
# 코딩하시오

# 각 줄에서 IP 주소 추출 (첫 번째 컬럼이 IP 주소)
# 코딩하시오

# 결과 출력
# 코딩하시오
```

LAB-6 (수행과제)

1. 문제 : HTML 파일에서 '<title>' 태그에 있는 내용을 추출
2. HTML 데이터 형식

```
<html>
      <head>
          <title>
              title is hear
          </title>
      </head>
      <body>
          ...
      </body>
  </html>
```

3. 결과

- Title: title is hear

> str.find(), str.strip, len(str), list[start:end]

In [ ]:
```
# 샘플 HTML 데이터
html_data = """
<html>
<head>
    <title>Welcome to My Website</title>
</head>
<body>
    <h1>Hello, World!</h1>
</body>
</html>
"""

# <title> 태그의 시작과 끝 인덱스 찾기
start_index = html_data.find("<title>") + len("<title>")
end_index = html_data.find("</title>")

# 슬라이싱으로 <title> 내용 추출
# 코딩하시오

# 결과 출력
# 코딩하시오
```

LAB-7 (수행과제)

1. 문제 : 특정 규칙에 따라 암호화된 문자열에서 슬라이싱을 활용하여 원래의 메시지를 복원
2. 암호화 규칙

- 암호화된 문자열은 원래 메시지의 문자들이 역순으로 저장되어 있음
- 각 문자는 두 번씩 반복

3. 입출력 예

- (입력) encrypted_message = "oollllleeHH!! ddllrrooww"
- (출력) Decrypted Message: Hello World!

> str[::-1], str[::2]

In [ ]:
```
# 암호화된 메시지
encrypted_message = "oollllleeHH!!  ddllrrooww"

# 1. 문자열을 역순으로 뒤집기
reversed_message = encrypted_message[::-1]

# 2. 두 번씩 반복된 문자 제거 (짝수 인덱스만 선택)
#
# 코딩하시오.
#

# 결과 출력
print("Decrypted Message:", decrypted_message)
```

In [ ]:
```

```