

파이썬 기초 문법

with  python™

01. 파이썬 자료형

◆ 숫자형(Number)

항목	표현 예시
정수 (int)	1, 2, 3, 123, 1345, 0, -123, -1, -2
실수 (float)	12.34, 456.189, 1.2e+5*, 1.2e-5**

☒ Scientific Notation

* $1.2e+5 = 1.2 \times 10^5 = 120000$

** $1.2e-5 = 1.2 \times 10^{-5} = 0.000012$

◆ 문자열(String)

주요 기능	정의
더하기	더하기 순서대로 문자열을 하나로 연결한다는 것을 의미
곱하기	문자열을 반복해서 연결하는 것을 의미
인덱싱	특정 위치에 있는 문자만 지정하여 일부를 추출함. 인덱스번호는 0번째부터 시작함
슬라이싱	특정 위치에 있는 범위를 지정하여 전체 또는 1개 이상의 문자를 추출함

02. 문자열 주요 메서드

◆ 메서드, a는 임의의 문자열이 저장된 객체

주요 메서드	메서드 설명
<code>a.count('p')</code>	특정 문자 ('p')가 몇 개가 있는지 확인
<code>a.find('p')</code>	특정 문자 ('p')가 첫번째로 등장한 위치(인덱스) 번호를 확인
<code>a.upper()</code>	영어 문자를 대문자로 변환
<code>a.lower()</code>	영어 문자를 소문자로 변환
<code>a.lstrip()</code>	문자열 왼쪽에 있는 공백 제거
<code>a.rstrip()</code>	문자열 오른쪽에 있는 공백 제거
<code>a.strip()</code>	문자열 양쪽에 있는 공백 제거
<code>a.replace("x", "y")</code>	문자열에 포함된 x를 y로 변경
<code>b = ', '.join(a)</code>	기존 문자열(a)에 특정 문자(예: ,)를 문자열 사이마다 삽입할 수 있음
<code>a.split(조건)</code>	문자열을 특정 조건에 따라 나눠서 다수의 문자열로 구성된 리스트 자료형으로 변경

02. 문자열 인덱싱 및 슬라이싱

◆ 인덱스는 0번째부터 시작

슬라이싱을 시작할 위치

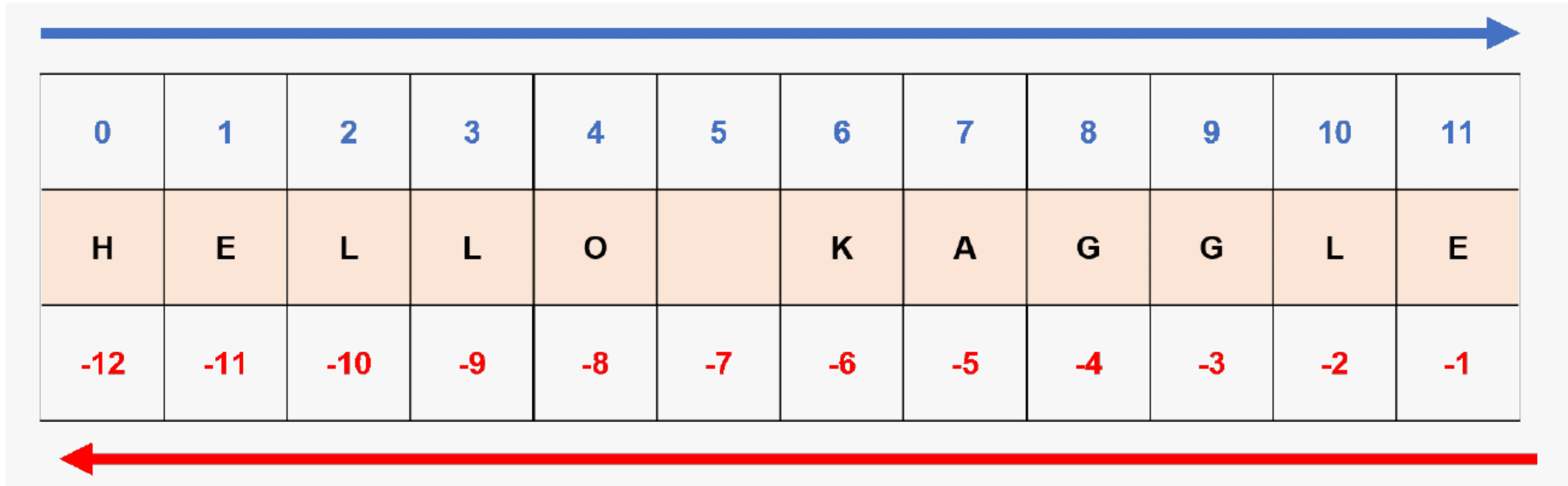
입력한 숫자값 만큼 건너 뛴

`[start_index:end_index:step_num]`

슬라이싱을 끝낼 위치, *end 숫자는 포함되지 않음

02. 문자열 인덱싱 및 슬라이싱

◆ 인덱스는 0번째부터 시작



The diagram illustrates string indexing and slicing for the string "HELLO KAGGLE". It features a 3x12 grid. The top row contains indices from 0 to 11 in blue. The middle row contains the characters of the string: 'H', 'E', 'L', 'L', 'O', an empty space, 'K', 'A', 'G', 'G', 'L', 'E'. The bottom row contains negative indices from -12 to -1 in red. A blue arrow at the top points right, indicating forward indexing. A red arrow at the bottom points left, indicating backward indexing.

0	1	2	3	4	5	6	7	8	9	10	11
H	E	L	L	O		K	A	G	G	L	E
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

03. 리스트

◆ 리스트는 숫자형, 문자열 자료형을 하나의 집합으로 구성할 수 있음

```
1 : []  
2 : [2,4,6,8,10]  
3 : ['빅데이터', '분석', '기사']  
4 : [1, 2, 3, ['빅데이터', '분석', '기사']] # 중첩리스트(Nested List)  
5 : ['빅데이터', '분석', '기사', 1, 2, 3]
```

◆ 문자열과 마찬가지로 더하기, 곱하기 같은 사칙연산을 활용해서 리스트를 합치거나 반복 가능

```
1 : a = [1, 2, 3]  
2 : b = [4, 5, 6]  
3 : c = a + b  
4 : d = c * 3  
5 : print(d)
```

03. 리스트

- ◆ 문자열과 마찬가지로 인덱싱과 슬라이싱 가능
- ◆ 중첩된 리스트에 대해서도 인덱싱과 슬라이싱 가능

```
1 : a = [1, 2, 3, ["a", "b", "c", "d"]]  
2 : print(a)  
3 : print(a[3][0]) # 중첩된 리스트 인덱싱 및 슬라이싱
```

```
[1, 2, 3, ['a', 'b', 'c', 'd']]  
a
```

03. 리스트 - 주요 메서드

◆ a는 임의의 리스트가 저장된 객체

주요 메서드	메서드 설명
a.append(x)	리스트의 맨 마지막 요소 위치에 x라는 요소를 추가한다.
a.sort()	리스트의 요소들을 정렬하는 함수가 있다.
a.index(x)	리스트 내의 x값의 위치에 해당하는 인덱스 값을 반환한다.
a.insert(x, y)	x값은 인덱스 번호, y값은 특정 값으로 x 인덱스 위치에 y 값을 추가한다.
a.remove(x)	x는 특정값을 의미하며, 리스트에서 x 값을 제거한다.
a.pop(x)	x는 인덱스 번호를 의미, 리스트에서 해당 인덱스의 값을 반환하며, 리스트에서 제거한다.
a.count(x)	특정요소 x의 개수를 계산한다.

04. 튜플(Tuple) 자료형

◆ 튜플과 리스트는 비슷한 역할을 수행함. 대괄호가 아닌 소괄호를 사용해야 함.

```
1 : (1, 2, 3)
2 : (1, )
3 : 1, 2, 3
4 : (1, 2, 3, ('빅데이터', '분석', '기사')) # 중첩튜플(Nested Tuple)
5 : ('빅데이터', '분석', '기사', 1, 2, 3)
```

◆ 리스트와 마찬가지로 더하기, 곱하기 같은 사칙연산을 활용해서 튜플을 합치거나 반복 가능

◆ 인덱싱과 슬라이싱 가능

◆ 리스트와의 가장 큰 차이점

- 리스트 : 요소의 생성, 삭제, 수정 가능
- 튜플 : **요소 변경 불가능**

05. 딕셔너리(Dictionary 자료형)

◆ 키(Key)와 값(Value)으로 이루어진 자료형

```
1 : a = {'name' : 'evan', 'age' : 30, 'birth' : [4, 30]}
```

◆ Value 값을 구하기 위해서는 Key를 활용해야 함.

```
1 : a = {'name' : 'evan', 'age' : 30, 'birth' : [4, 30]}
2 : print(a['name'])
3 : print(a['birth'])
```

```
evan
[4, 30]
```

05. 딕셔너리(Dictionary 자료형) - 메서드

◆ 주요 메서드는 다음과 같음, a는 임의의 딕셔너리로 저장된 객체

주요 메서드	메서드 설명
a.keys()	딕셔너리의 키(Key)의 리스트를 추출할 수 있음
a.values()	딕셔너리의 값(Value)의 리스트를 추출할 수 있음
a.items()	딕셔너리의 Key, Value를 튜플 구조로 묶고, 리스트로 추출할 수 있음
a.get(x, y)	딕셔너리의 x의 키가 존재하지 않을 때 y값 반환하도록 처리

06. 파이썬 제어문

◆ If문 : 조건문1을 테스트하여 참이면 if문, 아니면 elif 조건문2를 테스트하여 참이면 코드

```
1  :  if 조건문1:
2  :      코드 실행 1 # 들여쓰기는 공백(Spacebar) 또는 탭(Tab) 사용
3  :      코드 실행 2
4  :  elif 조건문2:
5  :      코드 실행 1
5  :      코드 실행 2
6  :  else:
7  :      코드 실행 1
8  :      코드 실행 2
9  :  A = [1, 2, 3]
10 :  ...
```

06. 파이썬 제어문

◆ 조건문 작성 방법

조건문 표현 방법	조건문 의미
$x < y$	x가 y보다 작다면
$x > y$	x가 y보다 크다면
$x == y$	x와 y가 같다면
$x != y$	x와 y가 같지 않다면
$x >= y$	x가 y보다 크거나 같다면
$x <= y$	x가 y보다 작거나 같다면

06. 파이썬 제어문

◆ 조건문 비교 연산자 (AND)

X	Y	Result
True	True	True
True	False	False
False	True	False
False	False	False

◆ not x : x가 거짓이면 참이다

◆ 조건문 비교 연산자 (OR)

X	Y	Result
True	True	True
True	False	True
False	True	True
False	False	False

06. 파이썬 제어문

◆ while 반복문 : 조건문이 참인 동안에 while 아래의 문장을 반복해서 수행함

```
1  : a = 0
2  :
3  : while a < 3:
4  :     print(f'현재 a값은 {a} 입니다.')
5  :     a = a + 1
6  :
7  : print("종료")
```

```
현재 a값은 0 입니다.
현재 a값은 1 입니다.
현재 a값은 2 입니다.
종료
```

06. 파이썬 제어문

◆ for 반복문 : 리스트, 문자열, 튜플 등 0번째 인덱스부터 마지막 인덱스까지 차례로 변수에 대입되어 명령문이 실행

```
1 : numbers = [100, 200, 300]
2 :
3 : for num in numbers:
4 :     print(num)
5 :
6 : print("종료")
```

```
100
200
300
종료
```


07. 사용자 정의 함수

◆ def문 : 함수 이름을 임의로 만들고, 이름 뒤 괄호 안의 매개변수는 입력으로 전달되는 값을 받는 변수

```
1  :  def 함수이름(매개변수1, 매개변수2, ...)  
2  :      코드 1  
3  :      코드 2  
4  :      ...  
5  :      코드 N  
6  :      return 결과값
```
