String, 문자열

문자열 생성

1. 큰 따옴표(`"`)나 작은 따옴표(`'`)로 양쪽을 둘러싸서 생성
2. 큰 따옴표나 작은 따옴표 3개를 양쪽으로 둘러싸서 생성

위와 같은 방도들은 출력문에서 사용해봤던 요소들입니다.

문자열에서 띄어쓰기를 하고 싶거나, 개행('Enter')을 하고 싶을 때와 같은 경우

이스케이프 코드를 사용할 수 있습니다.

이스케이프 코드

코드	설명
`\n`	문자열 안에서 줄 바꿈
`\t`	문자열 안에서 탭(띄어쓰기) 사용
`\'`	문자 `'`를 쓰고자할 때 사용
`\"`	문자 `"`를 쓰고자할 때 사용
`\r`	캐리지 리턴(줄 바꿈 문자, 현재 커서를 가장 앞으로 이동)
`\f`	폼 피드(줄 바꿈 문자, 현재 커서를 다음 줄로 이동)
`\b`	백 스페이스

주로 사용하는 이스케이프 코드는 `\n`과 `\t`, `\'`, `\"` 입니다.

문자열 연산

1. 문자열 + 연결

파이썬에서는 문자열을 서로 더하여 연결할 수 있습니다.

```
greeting = "Hello, "
name = 'Python'

print(greeting + name) # Hello, Python
```

2. 문자열 곱하기

작성한 문장을 여러번 반복하고 싶을 때 사용할 수 있습니다.

```
value = "반복구 "
print(value * 3) # 반복구 반복구 반복구
```

문장을 작성한 후에 곱하기(`*`) 연산자를 통해 해당 숫자만큼 반복하여 줄 수 있습니다.

3. 문자열 길이

내장함수 len()을 통하여 문자열의 길이를 구할 수 있습니다.

```
print(len("문자열 길이 구하기")) # 10
```

문자열 인덱싱[1]과 슬라이싱[2]

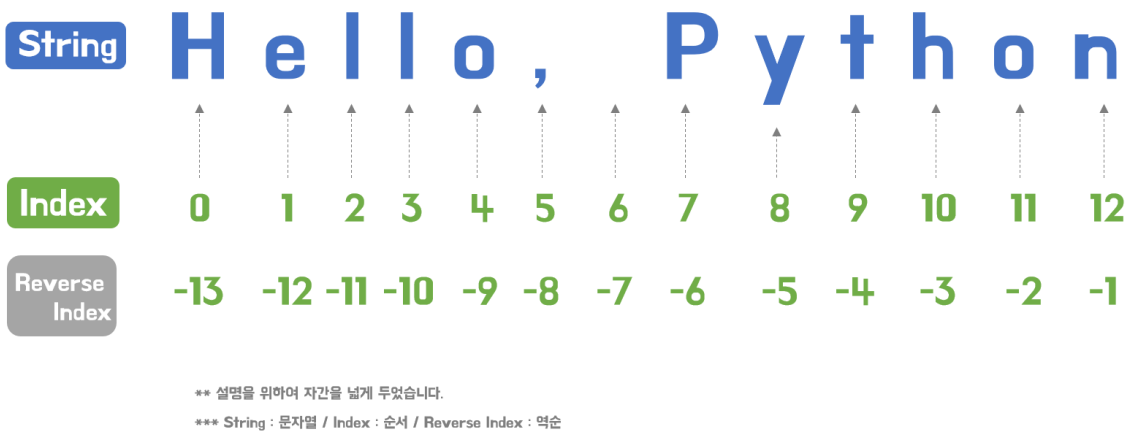
1. 문자열 인덱싱

문자열이란 각 문자의 연결의 결과물입니다.

문자열에서 문자들은 각각의 번호를 가지고 있습니다.

index의 번호는 0번 부터 시작입니다.

인덱스로 접근하는 방법은 `변수명[숫자]`를 이용하여 해당 값을 가져올 수 있습니다.




```
value = "Hello, Python"
print(len(value)) # 13 : 문자열의 길이
print(value[0]) # H
print(value[1]) # e
print(value[2]) # l
print(value[3]) # l
print(value[4]) # o
print(value[5]) # ,
print(value[6]) #
print(value[7]) # P
print(value[8]) # y
print(value[9]) # t
print(value[10]) # h
```

```
print(value[11]) # o
print(value[12]) # n
print(value[13]) # IndexError: string index out of range
```

해당 값이 가지고 있는 Index를 넘어가게 되면 `IndexError: string index out of range`가 발생하게 됩니다.

index는 길이나 사이즈의 값보다 -1까지를 가질 수 있습니다. (0부터 시작함이기에)

2. 문자열 슬라이싱

문자열 슬라이싱은 인덱싱되어 있는 값들에서 하나의 문자만 가져오는 것이 아니라,

가져오고 싶은 만큼 가져오는데 사용됩니다.

`변수[시작Index : 끝index]`로 해당 문자만큼 가지고 올 수 있습니다.

(⚠단, 끝index는 포함하지 않습니다.)

```
value = "문자열 학습을 하고 있습니다."

# '문자열 학습'까지만 값을 가져오도록 해보겠습니다.

"""
문 자 열    학 습 을    하 고    있  습 니 다  .
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
"""
print(value[0:6]) # 문자열 학습

# '하고 있습니다.'까지만 값을 가져오도록 해보겠습니다.
print(value[8:16]) # 하고 있습니다.
print(value[8:len(value)]) # 하고 있습니다.
print(value[8:]) # 하고 있습니다.

# 모든 문자열 가져오기
print(value) # 문자열 학습을 하고 있습니다.
print(value[:]) # 문자열 학습을 하고 있습니다.
```

`변수명[시작index : 끝index : step]`으로서 step은 몇 칸씩 뛰어서 사용할지를 정할 수 있습니다.

시작index와 끝index만 적었을 경우에 뒤에 오는 step의 초기값은 1입니다.

"홀짝수수는는 123456.....으으로로 진진행행됩됩니니다다.." 라는 문자가 있다고 하였을 때,

```
value = "홀짝수수는는  123456.....으으로로  진진행행됩됩니니다다.."

#홀수는 136...으로 진행됩니다.
print(value[0::2]) # 홀수는 135...으로 진행됩니다.

#짝수는 246...으로 진행됩니다.
print(value[1::2]) # 짝수는 246...으로 진행됩니다.
```

심화

> 문자열(str) 타입은 값 변경이 불가능(**불변객체, immutable**)합니다.

```
value = "Pithon"
print(value[1]) # i
value[1] = 'y' # TypeError: 'str' object does not support item assignment
print(value)
```

```
value = "Pithon"
print(value) # Pithon

value = "Python"
print(value) # Python
```

Q. 위의 경우에는값을 변경할 수 있는데, 왜 불가능인가요?

A. 위의 경우에는 변수에 접근하는 메모리값이 변경된 것입니다.

> 쉽게 보자면, 초기에는 Pithon이라는 값을 value에 넣어줬다가
>
> 아래의 `value = "Python"`과정에서 이전의 값을 무시하고 덮어씌웠다고 보시면 됩니다.

3. 문자열 포매팅

문자열 안에 어떤 값을 삽입할 때 사용하는 방법입니다.

여러 방법이 있지만, python 3.6버전 이상을 사용한다고 가정하고 **f-string**을 다루도록 하겠습니다.

f-string문법은 파이썬 3.6 버전에서 추가된 문법이며, 문자열 포매팅을 너무도 쉽게 할 수 있습니다.

사용법은 `f"{변수명}"`으로 사용할 수 있습니다.

아래의 예시를 참고하여주세요.

```
data = "가변될수 있는 변수"

print(f"포매팅을 확인하기 위해 {data}를 넣어봅니다.") # 포매팅을 확인하기 위해 가변될수 있는 변수를 넣어봅니다.
```

위의 내용만 봐서는 그냥 문자열로만 사용하면 될 것을 왜 포매팅이 필요한가 생각하실 수도 있습니다.

이는 **제어문**을 배우면 알수 있습니다.

한 예로, 변수에 값을 바꿔주면서 다른 문구는 똑같게 출력해야할 때 사용할 수 있습니다.

4. 문자열 관련 함수들

---

1. 인덱싱(Indexing) : 가리킨다 ↩
2. 슬라이싱(Slicing) : 잘라낸다 ↩