

Python algorithm

심은선

6장. 문자열 조작

Contents

Unit 00 | Intro

Unit 01 | 파이썬 문자열 슬라이싱

Unit 02 | 02. 문자열 뒤집기

Unit 03 | 03. 가장 긴 팰린드롬 부분 문자열

Unit 00 | Intro

파이썬의 타입 명시

- From typing import TypeVar

```
# 파이썬(Python)
from typing import TypeVar

T = TypeVar('T')
U = TypeVar('U')

def are_equal(a: T, b: U) -> bool:
    return a == b

are_equal(10, 10.0)
```

파이썬 구조체

- Dataclass 데코레이터 이용
- From dataclass import dataclass

```
# 파이썬(Python) 3.7+
from dataclasses import dataclass

@dataclass
class Product:
    weight: int = None
    price: float = None

apple = Product()
apple.price = 10
```

Dataclass 데코레이터는 생성자를 자동으로 생성해주므로 클래스 정의에도 유용

Unit 00 | Intro

파이썬 프로그래밍

1. 네이밍 컨벤션

- 스네이크 케이스: 단어를 _로 구분
ex) snake_case

2. 타입 힌트

- 파이썬은 동적언어 (실행시 자료형 결정)
- 따라서 타입 명시로 버그를 줄여야 함

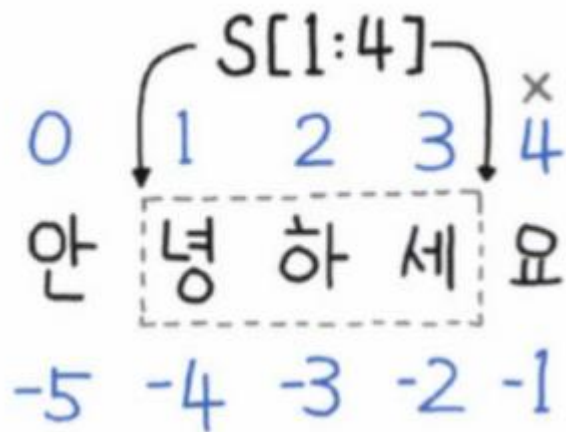
```
def fn(a: int) -> bool:
```

```
...
```

Unit 01 | 파이썬 문자열 슬라이싱

파이썬 문자열 슬라이싱

- 매우 빠른 속도 (내부적으로 포인터 이용)
- 문자열 조작에 슬라이싱을 우선적으로 사용하기



`S[idx]`

`S[start_idx : end_idx]` (`end_idx` 미포함)

`A = S (_ _ 복사)`

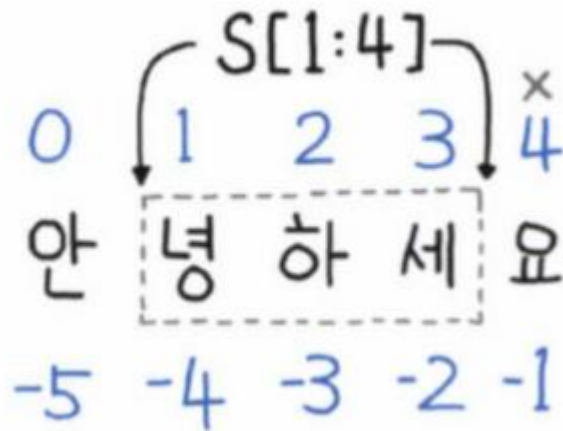
`A = S[:] (_ _ 복사)`

`S[st : end : step]`

Unit 01 | 파이썬 문자열 슬라이싱

파이썬 문자열 슬라이싱

- 매우 빠른 속도 (내부적으로 포인터 이용)
- 문자열 조작에 슬라이싱을 우선적으로 사용하기



`S[idx]`

`S[start_idx : end_idx]` (`end_idx` 미포함)

`A = S` (얕은복사) – 주소 공유

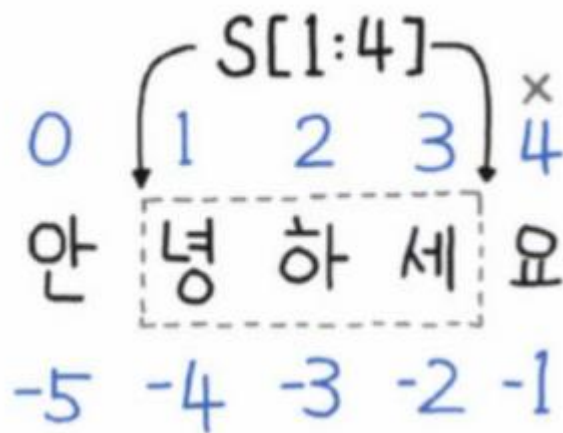
`A = S[:]` (깊은복사) – 별도의 주소

`S[st : end : step]`

Unit 01 | 파이썬 문자열 슬라이싱

파이썬 문자열 슬라이싱

- 매우 빠른 속도 (내부적으로 포인터 이용)
- 문자열 조작에 슬라이싱을 우선적으로 사용하기



Questions!

S[1:100] ==

S[:] ==

S[1:-2] ==

S[::2] ==

S[::1] ==

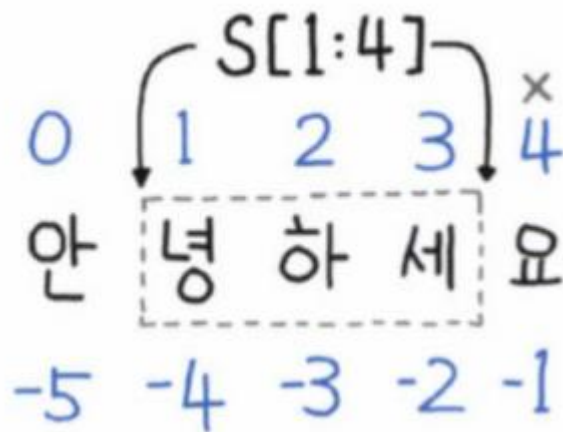
S[::-1] ==

S[-1] ==

Unit 01 | 파이썬 문자열 슬라이싱

파이썬 문자열 슬라이싱

- 매우 빠른 속도 (내부적으로 포인터 이용)
- 문자열 조작에 슬라이싱을 우선적으로 사용하기



Answers!

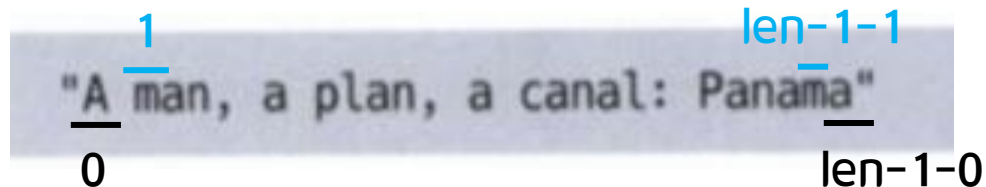
 $S[1:100] == \text{녕하세요}$ $S[:] == \text{안녕하세요}$ $S[1:-2] == \text{녕하}$ $S[::2] == \text{안하요}$ $S[::1] == \text{안녕하세요}$ $S[::-1] == \text{요세하녕안}$ $S[-1] == \text{요}$

Unit 01 | 파이썬 문자열 슬라이싱

01. 유효한 팰린드롬

- '소주 만 병만 주소' 처럼 뒤집어도 똑같은 문장이 되는 것 (팰린드롬)
- 대소문자를 구분하지 않으며, 영문자와 숫자만을 대상으로 함

1. 숫자, 알파벳만 필터링
2. 뒤집어서 똑같은 말이 되는지?



1 len-1-1
"A man, a plan, a canal: Panama"
0 len-1-0

Unit 01 | 파이썬 문자열 슬라이싱

01. 유효한 팰린드롬

- '소주 만 병만 주소' 처럼 뒤집어도 똑같은 문장이 되는 것 (팰린드롬)

```
##### Sol1 #####
def isPalindrome(self, s: str) -> bool:
    #문자와 숫자만 리스트에 넣기
    alpha = []
    for i in s:
        if i.isalpha():
            alpha.append(i.lower())
        elif i.isdigit():
            alpha.append(i)

    #팰린드롬 검사하기
    half = int(len(alpha)/2)
    length = len(alpha)
    for i in range(half):
        if alpha[i] != alpha[length-1-i]:
            return False

    return True
```

```
##### Sol4 #####
# 슬라이싱을 사용한 풀이
def isPalindrome(self, s: str) -> bool:
    s = s.lower()
    # 정규표현식으로 문자, 숫자 제외 필터링
    s = re.sub('[^a-z0-9]', '', s)

    #배열을 뒤집어서 같은지 비교
    return s == s[::-1]
```

배열끼리 == 연산자가 value의 비교다!

<https://leetcode.com/problems/valid-palindrome>

Unit 02 | 02. 문자열 뒤집기 (reverse-string)

02. 문자열 뒤집기(reverse-string)

- Return 없이 리스트 내부를 직접 조작해서 문자열을 뒤집는 함수 작성

예제 1

• 입력

```
["h","e","l","l","o"]
```

• 출력

```
["o","l","l","e","h"]
```

<https://leetcode.com/problems/reverse-string/>

Unit 02 | 02. 문자열 뒤집기 (reverse-string)

02. 문자열 뒤집기(reverse-string)

- 1. 포인터(배열 인덱스)를 이용한 뒤집기

```
##### Sol1 #####
def reverseString(self, s: List[str]) -> None:
    """
    Do not return anything, modify s in-place instead.
    """
    #리스트 원본에서 뒤집기
    length = len(s)
    half = int(len(s)/2)
    for i in range(half):
        temp = s[i]
        s[i] = s[length-1-i]
        s[length-1-i] = temp
```

```
##### Sol2 #####
# 투포인터를 이용한 swapping
def reverseString(self, s: List[str]) -> None:
    left, right = 0, len(s)-1
    while left < right:
        s[left], s[right] = s[right], s[left]
        left += 1
        right -= 1
```

Unit 02 | 02. 문자열 뒤집기 (reverse-string)

02. 문자열 뒤집기(reverse-string)

- 2. 파이썬 내장함수 이용

```
##### Sol3 #####  
# python method 이용  
def reverseString(self, s: List[str]) -> None:  
    return s.reverse()
```

#내장함수 비교

- a) array.reverse() : return None, 원본 reverse
- b) reversed(array) : return reversed array, 원본 불변

Unit 03 | 03. 가장 긴 팰린드롬 부분 문자열

03. 가장 긴 팰린드롬 부분 문자열(longest-palindromic-substring)

- 주어진 문자열에서 가장 긴 팰린드롬 부분 문자열을 출력하기

• 입력

"babad"

• 출력

"bab"

• 설명

"bab" 외에 "aba"도 정답이다.

<https://leetcode.com/problems/longest-palindromic-substring/>

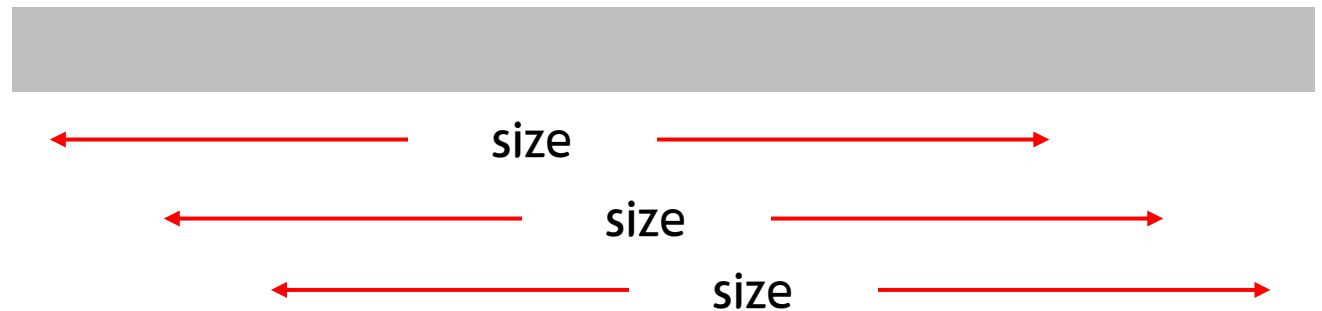
Unit 03 | 03. 가장 긴 팰린드롬 부분 문자열

03. 가장 긴 팰린드롬 부분 문자열(longest-palindromic-substring)

- 주어진 문자열에서 가장 긴 팰린드롬 부분 문자열을 출력하기

```
##### Sol1 #####
# 주어진 길이의 모든 sub문자열을 확인한다
def longestPalindrome(self, s: str) -> str:
    #최대 길이부터 확인한다
    for size in range(len(s), 0, -1):
        #sublist를 확인한다
        start = 0
        end = start + size
        #size에 대해 한 칸씩 옮겨가면서 확인
        while end <= len(s):
            sublist = s[start:end]
            if sublist == sublist[::-1]:
                return sublist
            start += 1
            end += 1
    return s[0]
```

Solution1



고정 길이 size에 대해 substring을 옮겨가면서 팰린드롬 확인하기

<https://leetcode.com/problems/longest-palindromic-substring/>

Unit 03 | 03. 가장 긴 팰린드롬 부분 문자열

03. 가장 긴 팰린드롬 부분 문자열(longest-palindromic-substring)

- 주어진 문자열에서 가장 긴 팰린드롬 부분 문자열을 출력하기

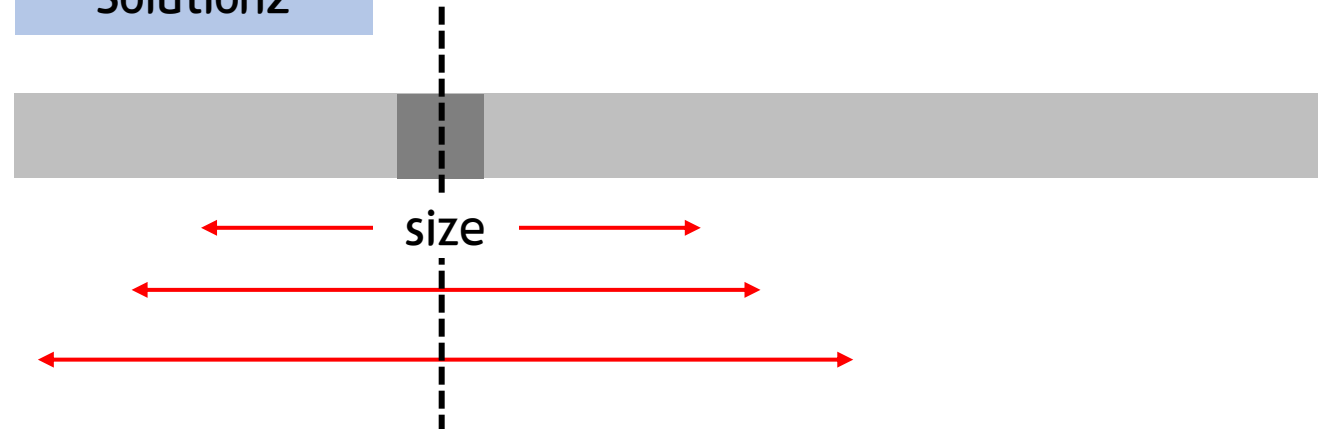
```
##### Sol2 #####
# 슬라이딩 윈도우로 투포인터를 이용해 팰린드롬 확인
def longestPalindrome(self, s: str) -> str:
    # 팰린드롬 판별 및 투포인터 확장
    def expand(left: int, right: int) -> str:
        # [left, right)를 같을 때까지 확장한다
        while left >= 0 and right <= len(s) and s[left] == s[right-1]:
            left -= 1
            right += 1
        return s[left+1:right-1]

    # 해당 사항이 없을 때 빠르게 return
    if len(s) < 2 or s == s[::-1]:
        return s

    result = ''
    # 슬라이딩 윈도우 우측으로 이동
    for i in range(len(s) - 1):
        result = max(result, expand(i, i+1), expand(i, i+2), key=len)

    return result
```

Solution2



각 위치에서 시작해 홀수, 짝수 개수씩 문자열을 확장하면서
팰린드롬 확인하기

<https://leetcode.com/problems/longest-palindromic-substring/>

Q & A

들어주셔서 감사합니다.