2024-1
기초 스터디

3. 문자열 (string)

복습 - 조건문

- 조건문
  if 조건식(=bool형 데이터):
  **   반복할 코드 (들여쓰기 유지 필수!)**

- bool형 데이터
  True, False / ==, !=, <, ... / 포함 확인 in, not in

복습 – and, or, not

- 조건식 연결

  and, or

  (or의 경우 True를 만나면 이후 코드는 실행 X)

- 조건식 반전

  not

복습 – elif, else

- **elif, else**

```
if 조건1:
    ~~
elif 조건2:
    ~~
else:
    ~~
```

복습 - 반복문

- **while** 조건식:
  반복할 코드

- **for** 변수 **in** 데이터그룹:
  반복할 코드
- for문에는 주로 **range(횟수)** 이용

복습 - 반복문

- **break** : 반복문을 종료
- **continue** : 현재 반복 건너뛰기

- 빠른 입출력

```
1  import sys
2  input = sys.stdin.readline
3
4  a = input().rstrip()
```

목차

문자열 (string)

- **작은따옴표(‘ ’)** 또는 **큰따옴표(“ ”)** 로 문자/문자열을 감싼 형태의 데이터

- “h”
- ‘hello’
- “hi-arc”

문자열 (string)

- 따옴표를 문자열에 포함하고 싶다면...

- “이 동아리 이름은 ‘하이아크’ 이다.” # “ ” 사이에 ‘ ’ 를 넣으면 된다.
- ‘이 동아리 이름은 “하이아크” 이다.’ # ‘ ’ 사이에 “ ” 를 넣으면 된다.
- “이 동아리 이름은 \”하이아크\” 이다.” # 이스케이프 문자를 사용한다.

여러 줄 표현하기

1. \n 를 사용한다.

“하이아크\n멋있어요” → 하이아크
멋있어요

여러 줄 표현하기

2. 따옴표를 3개씩 사용하여 감싼다. (“““ ”””)

“““하이아크
멋있어요”””

→

하이아크
멋있어요

문자열과 산술 연산

- 두 문자열을 더할 수 있다.

ex) print(“안녕~” + “세상아”) → **안녕~세상아**

문자열과 산술 연산

- 문자열에 **숫자**를 곱할 수 있다.

ex) “안녕” * 3 → **“안녕안녕안녕”**

cf) 지난주 **별 찍기 – 1** 문제도 반복문 하나로 풀 수 있다.

```
1  n = int(input())
2  for i in range(1, n+1):
3      print('*' * i)
```

문자열과 산술 연산

- 덧셈과 곱셈을 섞어서 사용할 수 있다.

ex) “안녕” * 2 + “ 세상아” → **“안녕안녕 세상아”**

변수 값을 문자열로 나타내기

- 자료형 변환 + 문자열 덧셈

```
>>> age = 20
>>> "my age is " + str(age)
'my age is 20'
```

변수 값을 문자열로 나타내기

- **f-string**을 이용하기
- **f“문자열”** 형태로 작성한다.
  변수를 쓸 때는 **{ }** 중괄호 안에 변수명을 쓴다.

```
>>> age = 20
>>> f"my age is {age}"
'my age is 20'
```

문자열 길이

- 문자열의 길이 = **문자열을 구성하는 문자의 개수**

ex) “안녕” → **2**
“hello” → **5**
“안 녕” → **3**
“안 녕\n” → **4**

문자열 속 문자 읽기

- **문자열[문자 위치]**
- 문자 위치는 **문자열 길이 미만의 정수**가 들어간다.
- 첫 번째 문자의 위치는 **0** 이다.

ex) s = “hello”

s[0] → **h** s[2] → **l**

s[4] → **o**

문자열 속 문자 읽기

- **문자열[문자 위치]**
- 문자 위치는 **음수**가 들어갈 수도 있다.
- 음수가 들어가면, 문자열의 맨 뒤에서부터 센다.

ex) s = “hello”

s[-1] → **o** s[-2] → **l**

s[-5] → **h**

문자열 속 문자 읽기

- **문자열[문자 위치]**
- ‘문자 위치’를 가리켜 보통 **‘인덱스’** 라고 부른다.

ex) “hello” 의 인덱스 0의 값 → “h”

문자열 속 문자 읽기 - 표(1)

- 문자열[문자 위치]

H	E	L	L	O
0	1	2	3	4
-5	-4	-3	-2	-1

~~-6~~

~~5~~

문자열 속 문자 읽기 - 표(2)

- 문자열[문자 위치]

H	E	L	L	O	H	E	L	L	O
					0	1	2	3	4
-5	-4	-3	-2	-1					

연습 문제

27866번 문자와 문자열 성공

문제

단어 S와 정수 i가 주어졌을 때, S의 i번째 글자를 출력하는 프로그램을 작성하시오.

예제 입력 1 복사

```
Sprout
3
```

예제 출력 1 복사

```
r
```

연습 문제

- 먼저 풀어봅시다.

- 사람은 1번째, 2번째, ... 순으로 세지만
  컴퓨터는 0번째, 1번째, ... 순으로 셉니다.

연습 문제

- 같이 풀어봅시다.

연습 문제

- 정답 코드

```
1 s = input()
2 i = int(input())
3 print(s[i-1])
```

문자열 수정

- 문자열은 수정할 수 없다.
- 수정해야 한다면 다음 주에 배울 '리스트' 를 활용해야 함.

```
1 s = "hello"
2 s[1] = "a"
```

```
Traceback (most recent call last):
  File "D:\Programming\PS\ps.py", line 2, in <module>
    s[1] = "a"
TypeError: 'str' object does not support item assignment
```

문자열 슬라이싱

- 문자열을 일부 잘라서 (slice) 가져오는 것
- **문자열[시작:끝(:간격)]**

- 시작 : 자르기 시작할 인덱스
  끝 : 자르기를 끝낼 인덱스 **+1**
  간격 : 자를 때 건너 뛸 간격 (생략 가능)

문자열 슬라이싱

- 문자열을 일부 잘라서 (slice) 가져오는 것
- **문자열[시작:끝(:간격)]**

- 시작, 끝, 간격 값 모두 각각 생략할 수 있다.
  이 경우, 값을 생략하고 **:** 만 쓰면 된다.

문자열 슬라이싱

- 문자열[시작:끝]

```
>>> s = "hello"
>>> s[1:4]
'ell'
```

문자열 슬라이싱

- 슬라이싱 할 때는 **인덱스에 제한이 없다.**

```
>>> s = "hello"
>>> s[1:100]
'ello'
```

문자열 슬라이싱

- 문자열[시작:]

```
>>> s = "hello"
>>> s[1:]
'ello'
```

문자열 슬라이싱

- 문자열[:끝]

```
>>> s = "hello"
>>> s[:3]
'hel'
```

문자열 슬라이싱

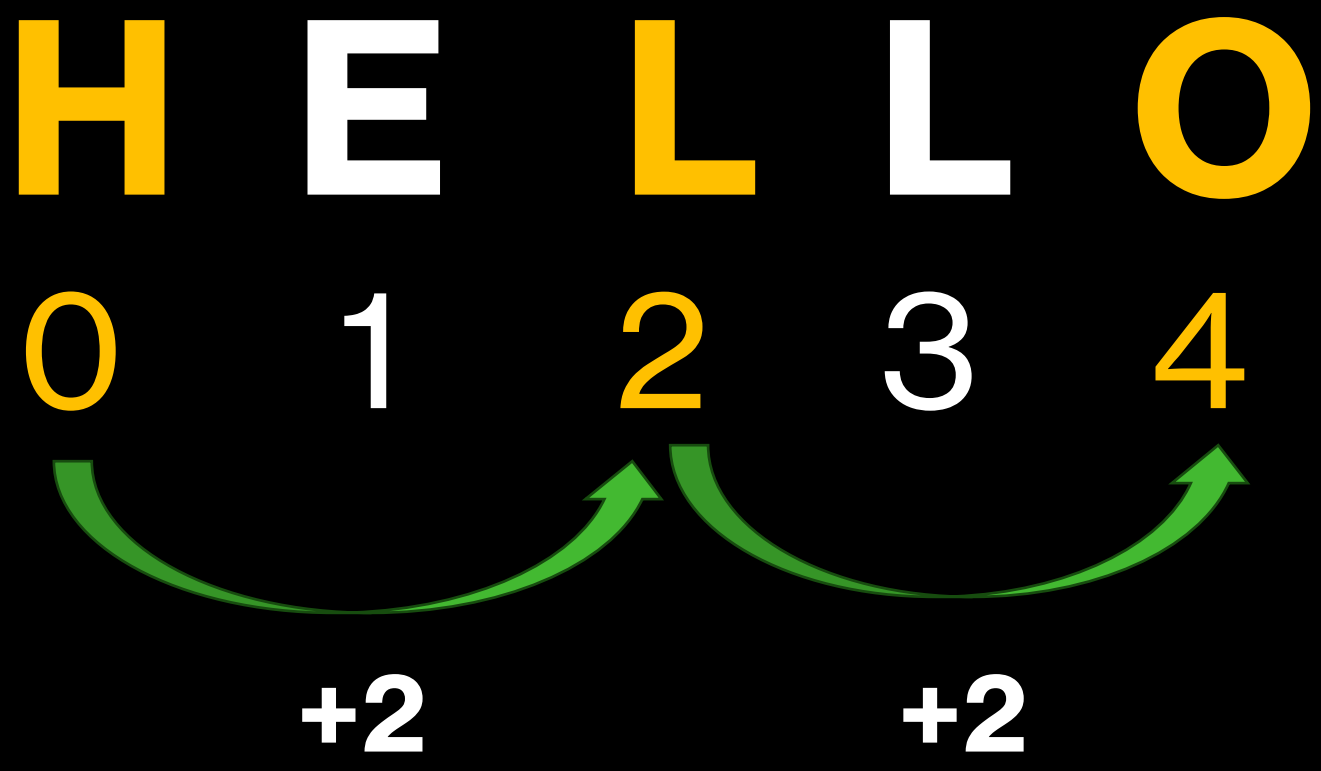
- 문자열[:]

```
>>> s = "hello"
>>> s[:]
'hello'
```

문자열 슬라이싱

- 문자열[::간격]

```
>>> s = "hello"
>>> s[::2]
'hlo'
```



문자열 슬라이싱

- 문자열[::간격]

```
>>> s = "hello"
>>> s[::-1]
'olleh'
```

연습문제

10988번 팰린드롬인지 확인하기 성공

문제

알파벳 소문자로만 이루어진 단어가 주어진다. 이때, 이 단어가 팰린드롬인지 아닌지 확인하는 프로그램을 작성하시오.

팰린드롬이란 앞으로 읽을 때와 거꾸로 읽을 때 똑같은 단어를 말한다.

level, noon은 팰린드롬이고, baekjoon, online, judge는 팰린드롬이 아니다.

예제 입력 1 복사	예제 출력 1 복사	예제 입력 2 복사	예제 출력 2 복사
level	1	baekjoon	0

연습문제

- 먼저 풀어봅시다.
- 방금 배운 문자열 슬라이싱을 활용하면 쉽습니다.

연습문제

- 같이 풀어봅시다.

연습문제

- 정답 코드

```python
s = input()
if s == s[::-1]:
    print(1)
else:
    print(0)
```

```python
s = input()
print(1 if s == s[::-1] else 0)
```

```python
"bool 형을 int형으로 변환"
s = input()
print(int(s == s[::-1]))
```

쉬어가는 시간

- 컴퓨터공학과의 진로 분야

- 프로그램 개발자 (프론트엔드, **백엔드**, 모바일, 게임, PC)
- 보안 (해커, 백신 개발자)
- 임베디드 개발자 (냉장고, 세탁기, 자동차...)
- 데브옵스 (서버 컴퓨터 관리)
- 데이터분석 (파이썬을 쓰기 좋은 곳)
- 인공지능 (대학원 거의 필수....)

백엔드

- 화면 **뒤에서 데이터의 처리**를 담당하는 역할

- Ex) 사용자가 로그인 버튼을 누르면, 프론트엔드는
  사용자가 입력한 ID, PW를 백엔드에 전달합니다.
  백엔드는 전달받은 ID, PW가 DB에 들어있는지 확인하고,
  그 결과를 프론트엔드에 다시 알려줍니다.

백엔드

- 프론트엔드 공부를 하다보면 정해진 형식의 데이터가 반복적으로 필요한 경우가 생깁니다.

- 백엔드는 위와 같이 프론트가 화면을 그릴 때 필요한 데이터를 보내줍니다.

백엔드

- **프론트**는 백엔드에 필요한 **데이터를 요구**하고, **백엔드**는 프론트가 요구한 **데이터를 보냅니다**.

- 이 기본 구조만 알면 다음부터는 어떻게 백엔드를 만들지만 선택하면 돼요!

어떻게 만드나요?

- **자바 + 스프링**
- 파이썬 + 장고
- 자바스크립트 + Express.js
- 파이썬 + 플라스크 (처음엔 비추)
- ..등등

어디서 공부하나요?

- 백엔드는 혼자 검색만으로 공부하기 힘드니 책이나 강의를 보는 걸 추천드려요!

- 유튜브 (nodejs express 백엔드 강의 추천)
- 인프런 (김영한 Spring 입문 무료강의 추천)
- 책

어떻게 공부하나요?

- 제가 공부한 순서는 아래와 같습니다!

1. 간단한 프론트 앱 만들기 (todo 리스트도 좋아요)
2. 프론트 앱이 사용할 백엔드를 **express.js** 로 만들기
3. 백엔드의 역할을 이해한 뒤 **장고/스프링 중 하나를 골라 공부**해보기(개인적으로는 스프링 추천)

문자열과 반복문

- 문자열 문제를 풀 때,
  문자열 속 모든 문자를 읽는 경우가 많다.

- 이때, 반복문을 이용해 모든 문자를 읽을 수 있다.

문자열과 반복문

- for 반복문 + 문자열 속 문자 읽기

```
1  s = "hello"
2  for i in range(5):
3      print(s[i])
```

→

```
>>> %Run ps.py
 h
 e
 l
 l
 o
```

문자열 관련 함수

1. len(문자열) : 문자열의 길이를 알려준다.
2. ord(문자) : 문자의 아스키코드를 알려준다.
3. chr(아스키코드) : 아스키코드에 해당하는 문자를 알려준다.

Len() 함수 활용

- 문자열 속 모든 문자를 읽을 때,
  반복 횟수를 명시하는 용도로 사용할 수 있다.

```
1  s = "hello"
2  for i in range(len(s)):
3      print(s[i])
```

→

```
>>> %Run ps.py
 h
 e
 l
 l
 o
```

아스키코드

- 컴퓨터는 알파벳, 한글같은 '문자'를 이해하지 못함.
- 컴퓨터는 언제나 **'숫자'만 이해**함.

- **아스키코드**는 **문자마다 숫자를 1:1로 매칭**해서
  컴퓨터가 문자를 이해할 수 있도록 정해둔 규칙

아스키코드

10진	16진	문자	10진	16진	문자
64	0x40	@	96	0x60	`
65	0x41	A	97	0x61	a
66	0x42	B	98	0x62	b
67	0x43	C	99	0x63	c
68	0x44	D	100	0x64	d
69	0x45	E	101	0x65	e
70	0x46	F	102	0x66	f
71	0x47	G	103	0x67	g
72	0x48	H	104	0x68	h
73	0x49	I	105	0x69	i
74	0x4A	J	106	0x6A	j
75	0x4B	K	107	0x6B	k
76	0x4C	L	108	0x6C	l
77	0x4D	M	109	0x6D	m
78	0x4E	N	110	0x6E	n
79	0x4F	O	111	0x6F	o
80	0x50	P	112	0x70	p
81	0x51	Q	113	0x71	q
82	0x52	R	114	0x72	r
83	0x53	S	115	0x73	s
84	0x54	T	116	0x74	t
85	0x55	U	117	0x75	u
86	0x56	V	118	0x76	v
87	0x57	W	119	0x77	w
88	0x58	X	120	0x78	x
89	0x59	Y	121	0x79	y
90	0x5A	Z	122	0x7A	z

- 아스키코드 표를 외울 필요는 없다.
- 이 표에서 중요한 점은 **연속된 알파벳은 연속된 숫자로 표현된다**는 것이다.

연습 문제

11654번 아스키 코드 성공

문제

알파벳 소문자, 대문자, 숫자 0-9중 하나가 주어졌을 때, 주어진 글자의 아스키 코드값을 출력하는 프로그램을 작성하시오.

연습 문제

- 먼저 풀어봅시다.

연습 문제

- 같이 풀어봅시다.

연습 문제

- 정답 코드

```
1 c = input()
2 print(ord(c))
```

연습 문제

10987번 모음의 개수 성공 다국어

문제

알파벳 소문자로만 이루어진 단어가 주어진다. 이때, 모음(a, e, i, o, u)의 개수를 출력하는 프로그램을 작성하시오.

예제 입력 1 복사	예제 출력 1 복사
baekjoon	4

연습 문제

- 같이 풀어봅시다.

연습 문제

- 정답 코드

```python
s = input()

count = 0
for i in range(len(s)):
    if s[i] == 'a':
        count += 1
    elif s[i] == 'e':
        count += 1
    elif s[i] == 'i':
        count += 1
    elif s[i] == 'o':
        count += 1
    elif s[i] == 'u':
        count += 1

print(count)
```

연습 문제

- 문자열도 **'문자 데이터의 그룹'** 이다.
- 따라서 **in** 연산자를 사용할 수 있다.

```
• 데이터 in [데이터 그룹]      # 포함 되면 True
• 데이터 not in [데이터 그룹]  # 포함 안되면 True
```

연습 문제

- 이렇게 푸는 것도 가능합니다.

```
1  s = input()
2
3  count = 0
4  for i in range(len(s)):
5      if s[i] in 'aeiou':
6          count += 1
7
8  print(count)
```

문자열 관련 메서드

- **(r/l)strip('문자')**: 문자열 양 끝에서 문자를 제거한 새로운 문자열을 만들어준다.
- 문자를 명시하지 않으면 '공백문자' 를 제거한다.
  (공백문자 = ' ', '\n', '\t' 등. 편하게 모든 빈칸이라고 생각하면 돼요)

```
1  a = "*12345*"
2  print(a.strip('*'))
3  print(a.rstrip('*'))
4  print(a.lstrip('*'))
```

```
>>> %Run -c
  12345
  *12345
  12345*
```

문자열 관련 메서드

- **split('문자')**: '문자'를 기준으로 문자열을 쪼갠다.
- 문자를 명시하지 않으면 '공백문자' 를 기준으로 쪼갠다.
  (공백문자 = ' ', '\n', '\t' 등. 편하게 모든 빈칸이라고 생각하면 돼요)

```
>>> s = "BTS,에스파,아이브,르세라핌"
>>> s.split(',')
['BTS', '에스파', '아이브', '르세라핌']
```

```
>>> s = "hello world! nice to meet you!"
>>> s.split()
['hello', 'world!', 'nice', 'to', 'meet', 'you!']
```

문자열 관련 메서드

- **lower()** : 주어진 문자열을 소문자로 바꾸기
- **upper()** : 주어진 문자열을 대문자로 바꾸기
- **count('문자')** : 문자열에서 '문자' 의 개수 세기

```
>>> s = "Hello"
>>> s.lower()
'hello'
>>> s.upper()
'HELLO'
```

```
>>> s = "hi everyone"
>>> s.count('e')
3
```

연습 문제

10808번 알파벳 개수 성공

문제

알파벳 소문자로만 이루어진 단어 S가 주어진다. 각 알파벳이 단어에 몇 개가 포함되어 있는지 구하는 프로그램을 작성하시오.

예제 입력 1 복사

```
baekjoon
```

예제 출력 1 복사

```
1 1 0 0 1 0 0 0 0 1 1 0 0 1 2 0 0 0 0 0 0 0 0 0 0 0
```

연습 문제

- 알파벳 a부터 z까지 각 알파벳의 개수를 세야한다.
- 앞에서 배운 count 메서드를 이용하면 되긴 하는데..

```
1  s = input()
2
3  print(s.count('a'))
4  print(s.count('b'))
5  ....
```

- 그런데 **똑같은 코드가 반복**되고 있다!!

연습 문제

- ord(), chr() 함수와 반복문을 이용하여 풀어봅시다.

- **print(데이터, end=‘ ’)**
- 줄을 넘기는 대신 공백으로 출력할 수 있습니다.
  (end=‘ ’ 으로 공백이 한 칸 있어요!)

연습 문제

- 정답 코드

```
s = input()
for i in range(26):
    now_ascii_code = ord('a') + i
    now_alphabet   = chr(now_ascii_code)

    print(s.count(now_alphabet), end=' ')
```

정리

- 문자열 생성
- “ ” ‘ ’ “““ ”””
  ’ ’
- f-string : **f“문자열 {변수이름}”**

정리

- 문자열 연산
- **“hi~” + “everyone”　　= “hi~everyone”**
- **“hello” * 3　　= “hellohellohello”**

정리

- 문자열 속 문자에는 **[ ]** 연산자로 접근한다.
- 문자열 인덱스는 0부터 시작한다.

s = “hello”
s[0] = “h”
s[4] = “o”

정리

- 문자열 슬라이싱

  문자열[시작:끝:간격]

정리

- 문자열 관련 함수
  len(), ord(), chr(), ...

- 문자열 관련 메소드
  strip(), split(), lower(), upper(), count(), ...

이번주 연습 문제

- 9086 문자열 (문자열 인덱스, 음수 인덱스)
- 2675 문자열 반복 (문자열과 반복문, 문자열 곱셈)
- 11720 숫자의 합 (문자열과 반복문, 자료형 변환)
- 2908 상수 (문자열 슬라이싱, 자료형 변환)
- 5622 다이얼 (문자열과 반복문, in 연산자)
- 26068 치킨댄스를 추는... (문자열 슬라이싱, 자료형 변환)

혹시 연습 문제가 쉽다면..

- 2941 크로아티아 알파벳
- 1316 그룹 단어 체커
- 25206 너의 평점은

- 처음 문자열을 접한다면 꽤 어려울 수 있는 문제입니다.
  백준 연습에도 함께 넣어두었으나 이 문제들은
  출석 & 우수 스터디원 선정 조건에 **안** 들어갑니다!
  시간 날 때 도전해보세요!