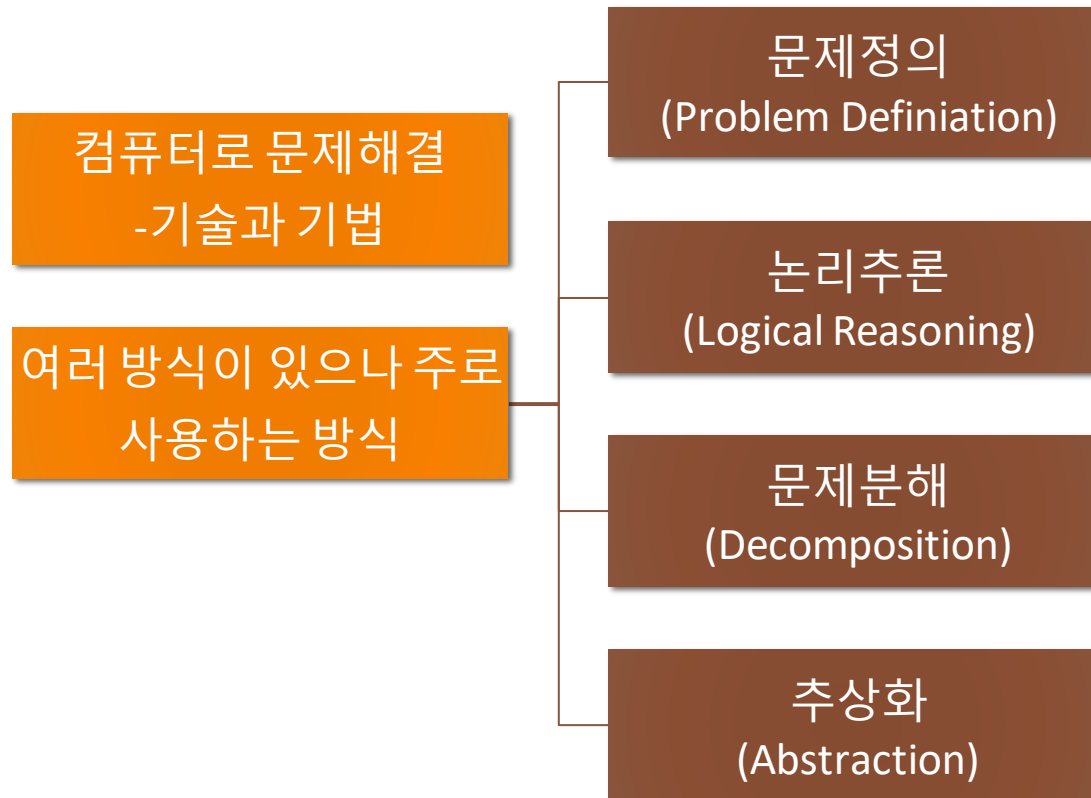


4. 문제정의, 변수와 자료형(숫자와 문자열)

2018.12

일병 김재형

문제해결



문제해결1-문제정의

소프트웨어가 할 작업들을 기술

- 개발 목표가 된다.
- 문제 해결 여부, 동작 여부를 확인할 때 사용

문제정의의 결과물

- 요구사항(requirements)
 - 기능적 요구사항: 소프트웨어가 수행할 작업과 관련된 부분
 - 비기능적 요구사항: 소프트웨어 관련 특성이나 제약조건
소프트웨어 안전성, 신뢰성, 보안, 성능, 고객지원 등

문제해결1-문제정의

문제정의의 결과물

- 잘 작성된 요구사항
 - 분명해야 한다.
 - F implied N과 같은 논리적 명제로 전환 가능해야 한다.
 - 일관성 있어야 한다.
 - 요구사항들 간에 모순이 없어야 한다.
 - 완전해야 한다.
 - 모든 사용 시나리오가 고려되어야 한다.

문제해결1-문제정의

함축(implies)

- 하나가 참이면, 논리적 필연성에 의해 일부 다른 것들도 참이어야 한다
- "P implies Q"라는 명제($p \rightarrow q$)
- P: 차량 배터리가 다 떨어진다.
Q: 시동이 걸리지 않는다.

| P | Q | P implies Q |
|----|----|-------------|
| 거짓 | 거짓 | 참 |
| 거짓 | 참 | 참 |
| 참 | 거짓 | 거짓 |
| 참 | 참 | 참 |

문제해결1-문제정의

자판기(vending_machine.py)

- 자판기를 만들고 싶다고 고객이 요청
- 질문을 통해 고객 요구사항을 정리
 - 자판기의 물품을 보여줌
 - 자판기에 돈을 넣으면 집어넣는 돈이 증가
 - 물품 번호를 선택하면 물품이 나오며 집어넣은 돈이 감소
 - 거스름돈을 누르면 거스름돈이 반환
 - 돈이 부족할 때 물품을 선택하면 물품이 나오지 않고 '돈이 부족합니다.'가 반환
- 종료를 선택하면 프로그램이 종료된다.

문제해결1-문제정의

요구사항의 예-자판기

| 일련번호 | V1 |
|------|--|
| 이름 | 돈 입력 |
| 행동 | 넣을 돈을 입력하면 넣은 돈의 총액이 증가한다. 이 후 초기화면으로 돌아간다. |

| 일련번호 | V2 |
|------|--|
| 이름 | 물품 출력 |
| 행동 | 넣은 돈의 총액이 물품 가격보다 높으면 물품을 출력하고 물품 가격만큼 넣은 돈의 총액을 뺀다. 이후 초기화면으로 돌아간다. |

문제해결1-문제정의

요구사항의 예-자판기

| 일련번호 | V3 |
|------|--|
| 이름 | 거스름돈 반환 |
| 행동 | 거스름돈 반환을 선택하면 거스름돈이 나온다. 그 후 초기화면으로 돌아간다. |

| 일련번호 | V4 |
|------|--|
| 이름 | 돈 부족 |
| 행동 | 넣은 돈이 물품가격보다 적으면 "금액이 부족합니다."를 출력하고 물품이 나오지 않는다. 이후 초기화면으로 돌아간다. |

문제해결1-문제정의

요구사항의 예-자판기

| 일련번호 | V5 |
|------|------------------------|
| 이름 | 종료 |
| 행동 | 거스름돈을 반환하고 프로그램을 종료한다. |

문제해결1-문제정의

요구사항의 완전성 판단

—상태-활동 테이블로 판단

- 첫 열에는 사용자들이 가능한 행동을 열거
- 첫 행에는 프로그램의 가능한 상태를 열거
- 각 셀에는 해당하는 요구사항 번호를 적는다.
- 생길 수 없는 상태는 해당 셀을 회색으로 마킹
- 회색 셀도 아니면서 요구사항 번호도 없는 셀은 요구사항이 추가로 필요함으로 요구사항을 추가로 작성

문제해결1-문제정의

자판기 상태-활동 테이블

| | 프로그램 미실행 | 물품의 가격보다 돈이 부족 | 물품의 가격을 충족 |
|---------------|-------------|-------------------|---------------|
| 프로그램 시작 | | | |
| 돈을 넣음 | | V1 | V1 |
| 물품을 선택 | | V4 | V2 |
| 거스름돈 반환 선택 | | V3 | V3 |
| 종료 | | V5 | V5 |

문제해결1-문제정의

요구사항을 추가한다.

| 일련번호 | V0 |
|------|--|
| 이름 | 초기화면 |
| 행동 | 자판기에서 뽑아 먹을 수 있는 물품과 현재 입력된 돈을 출력하고 돈을 입력 받을지, 거스름돈을 출력할지, 물품을 출력할지, 종료를 할지 입력받는다. |

변수

변수란?

- 어떠한 값에 대해 이름을 붙여 사용하는 것
수학에서의 변수와 같다.
- 변수가 있기 전에는 번호를 사용했다.
 - 예를 들어, 사람이 영역을 기억하여 '345번에 들어있는 값을 1 증가시켜'라고 지시를 내림
 - 125.209.222.141를 쓰는 것보다 `www.naver.com`을 쓰는게 편하다.
- 이로 인해 프로그래밍 언어에서도 이름으로 대상을 지정할 수 있다.

변수

Assignment(할당, '=')를 통해 대입한다.

- 수학에서는 '같다'지만, 컴퓨터에서는 할당이다.
- 비교를 할 때는 '=='를 사용한다.

Python은 대소문자를 구분한다.

Python에서 모든 것(자료형, 모듈, 함수등)은 모두 객체로 표현된다.

변수-객체

객체(Object)

- 실생활에서 파악할 수 있는 것으로, 소프트웨어 세계에 구현할 대상
- 실행되는 프로그램에 존재하는 구조화된 데이터 덩어리
- 상태(state)와 행위(behavior)
 - 댐을 제어하는 시스템
 - 수문이 닫히고 열린 상태: 객체의 상태
 - 수문을 여는 행위(수문이 닫힌 상태에서 열린 상태로 변화): 행위

변수-클래스

클래스(Class)

- 데이터를 추상화하는 단위
- 실생활의 사물을 소프트웨어로 구현하기 위해서는 추상화(Abstraction, 단순화하는 과정)가 필요.
- 같은 상태와 행위를 가진 객체는 같은 클래스이다.
- 속성(attribute)와 메서드(method)를 가진다.
 - 속성: 객체에 저장된 자료의 특성과 이름을 정의한 코드
 - 메서드: 객체의 행위를 구현한 함수(프로시저)

변수-인스턴스

인스턴스(Instance)

- 인스턴스와 객체는 같은 의미이다.
- 하지만, 인스턴스는 '어떤 클래스에 속하는 특정 사례'라는 뜻으로 관계를 나타낸다.
- ex) '딸'=관계를 나타내는 단어
'여자아이'=독립된 개념

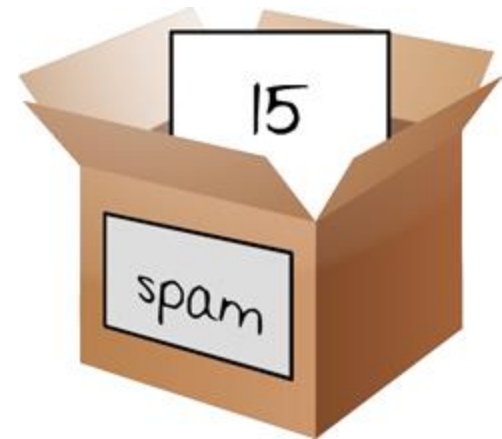
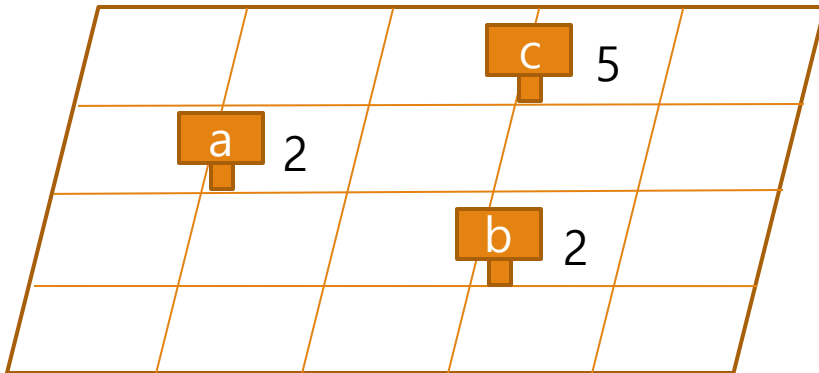
변수의 생성

$a=2$

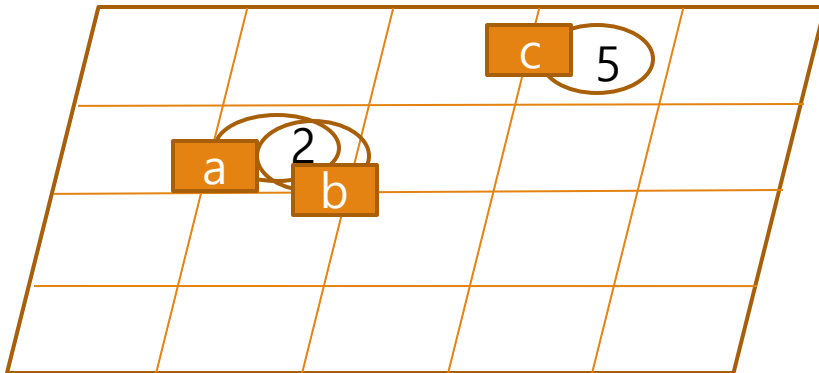
$b=a$

$c=5$

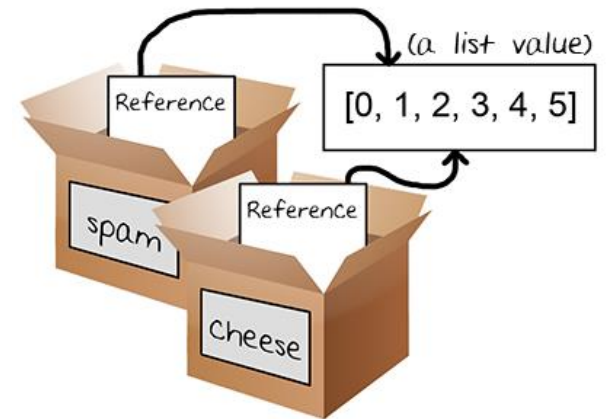
변수의 생성-C언어



변수의 생성-Python



② `cheese = spam`



| 이름 | 객체 |
|----|-----|
| a | → 2 |

변수의 생성-Python

실제로 같은가?

- id(변수): 메모리 주소
- 변수 is 변수: 동일한 객체인지 확인

```
>>> a=2
>>> b=a
>>> id(a)
9330848
>>> id(b)
9330848
>>> a is b
True
```

변수의 생성-interning

CPython에서의 interning(인터닝)

- CPython
 - C로 구현된 Python
 - 일반적으로 사용하는 Python이다.
- 인터닝 대상
 - 문자열: 20자 미만의 공백을 포함하지 않을 때
 - 숫자: -5부터 256사이의 정수
- Cpython의 구현특징으로, 문서화 되어 있는 방법이 아니다.

```
>>> a=2
>>> b=2
>>> a is b
True
>>> id(a)
9330848
>>> id(b)
9330848
```

<http://pythonstudy.xyz/python/article/512-%ED%8C%8C%EC%9D%B4%EC%8D%AC-Object-Interning>

변수의 비교

is와 ==

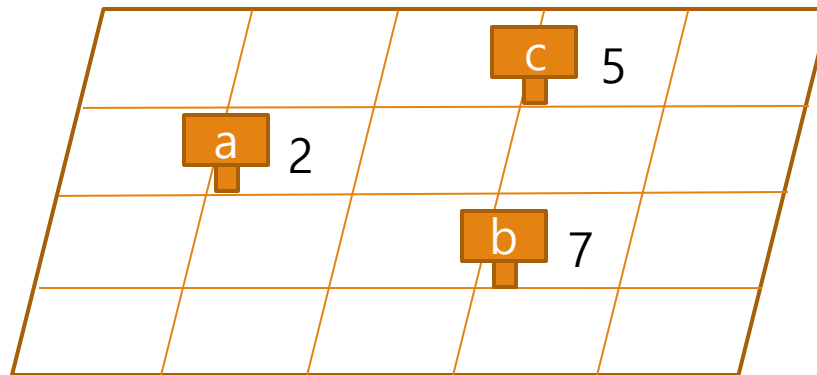
- is
 - 객체의 메모리값을 비교
 - 같은 객체인지 비교(identity)
- ==
 - 객체의 값을 비교(equality)
 - 일반적으로 비교할 때는 ==을 사용

```
>>> num1=1000
>>> num2=1000
>>> id(num1)
139721668920176
>>> id(num2)
139721651954288
>>> num1 is num2
False
>>> num1 == num2
True
```

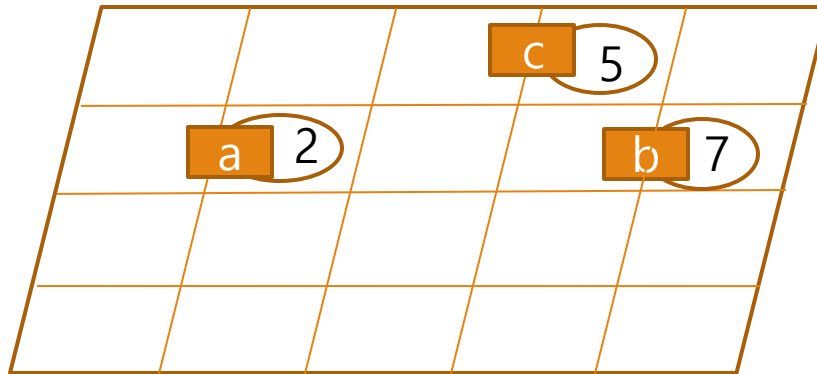
변수의 데이터 변경

$b=a \Rightarrow b=7$

변수의 데이터 변경-C언어



변수의 재정의-Python



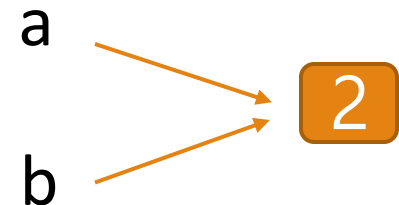
변수의 소멸

변수의 소멸

- del(변수)
- 변수를 제거하는 것

Garbage Collection(GC)

- 사용되지 않는 메모리 영역을 해제
- CPython에서는 레퍼런스 카운트(reference count)를 사용.
- 레퍼런스 카운트(참조 수)가 0이 되면 메모리 영역을 해제



자료형

자료형(Data Type): 데이터 또는 자료의 형식

- 데이터 분류 체계
- 사람이 데이터에 붙인 추가 데이터
- 0과 1로 저장된 데이터를 어떻게 해석할지에 대해 정한 약속

자료형이 없다면?

- 사람에게는 비슷하지만, 컴퓨터는 해석할 수 없다.

부동 소수점 7.0 

정수 7 

자료형

- '어떤 종류의 값인가?'라는 정보를 추가
- 응용이 되며, 사용자가 정의할 수 있는 자료형(형)이 탄생
 - 데이터: struct
 - 메소드도 포함: class

동적/정적 형식 언어

- 정적 형식 언어: 컴파일 시 자료형을 판단
- 동적 형식 언어: 프로그램 실행 시 자료형을 판단
- Python은 동적 형식 언어이다.

자료형-불변형/가변형

불변형(Immutable Type)

- 값을 바꿀 수 없는 자료형
- 생성과정이 간단(내용과 크기가 변경되지 않음)
- 데이터가 변경되지 않아 신뢰 가능
- 숫자, 문자열, 튜플 등

가변형(Mutable Type)

- 값을 바꿀 수 있는 자료형
- 리스트, 집합, 딕셔너리 등
- 데이터를 변경할 수 있으나, 메모리를 많이 사용

자료형

Python에서 자료형 확인 함수

—type()

```
>>> type(123)
<class 'int'>
>>> a = 1234
>>> type(a)
<class 'int'>
```

자료형-숫자

숫자 형태로 이뤄진 자료형-이미 잘 알고 있다!

| 항목 | 예 |
|------|---------------------|
| 정수 | 1, -25, 0 |
| 실수 | 1.25, -12.5, 9.5e10 |
| 복소수 | 2+9j, 2j |
| 2진수 | 0b1010, 0b1110 |
| 8진수 | 0o23, 0o45 |
| 16진수 | 0xFF, 0x2B |

자료형-숫자

정수형(Integer)

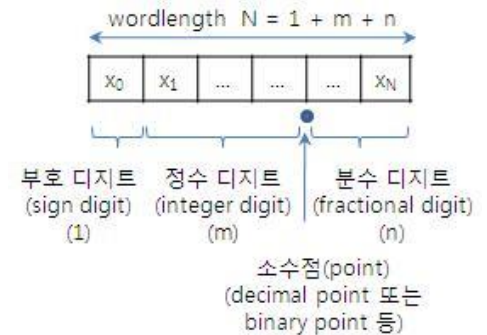
- 정수를 뜻하는 자료형
- 음의 정수, 0, 양의 정수
- Python3에서는 int가 받을 수 있는 숫자의 제한이 없다.
- BigInteger 구현이 필요없어요 >_

```
>>> a = 3
>>> b = 123456789
>>> c = 123456789987654321123456789987654321
>>> a
3
>>> b
123456789
>>> c
123456789987654321123456789987654321
```

자료형-숫자

실수형(float)

- 고정소수점
 - 소수점을 고정시켜 실수를 나타낸다.
 - 나타낼 수 있는 수의 범위가 적다.
- 부동소수점
 - 소수점이 움직인다.
 - 나타낼 수 있는 수의 범위가 크나, 정밀도의 문제가 있다.



※ Python에서는 부동소수점을 사용

자료형-숫자

부동소수점

- 지수와 밑
- 거듭제곱: 주어진 수를 주어진 횟수만큼 곱하는 연산
- 밑: 주어진 수
- 지수: 주어진 횟수
- 밑이 a 이고 지수가 n 인 거듭제곱을 a 의 n 제곱이라고 한다.

$$a^n = a \times a \times a \times \cdots \times a$$

자료형-숫자

부동소수점

— 뜰 부(浮), 움직일 동(動)

123.456



$$1.23456 \times 10^2$$

$$12.3456 \times 10^1$$

$$1234.56 \times 10^{-1}$$

$$12345.6 \times 10^{-2}$$

```
>>> a
12.5
>>> type(a)
<class 'float'>
```

자료형-숫자

부동소수점

| 부호 | 지수 | | | 가수 | | | |
|----|----|--|--|----|--|--|--|
| | | | | | | | |

- 부호: 수가 음수인가 양수인가
- 지수: 지수의 값
- 가수: 가수의 값
- 가수는 2진수 소수로 나타낸다.

자료형-숫자

부동소수점

| 부호 | 지수 | | | 가수 | | | |
|----|----|--|--|-----|------|-------|--------|
| | | | | 0.5 | 0.25 | 0.125 | 0.0625 |

- 가수는 2진수 소수로 나타낸다.
- 0.5, 0.25, 0.125 등으로 나타낸 값을 더해서 값을 만든다.
- 1/3과 같은 값은 정확히 나타낼 수 없다.

자료형-숫자

부동소수점의 사용

- 부동소수점은 정밀도(정확한 자리수)의 한계가 있다.
- 이는 부동소수점의 한계이다.

```
>>> a = 43.2-43.1
>>> a
0.1000000000000000142
```

- 이러한 미미한 오차가 있어 부동소수점이 들어간 계산의 비교는 오차범위 이내로 비교해야 한다.

자료형-숫자

컴퓨터적 지수 표현방식

10대신에 E나 e를 사용한다.

$$3.1415 \times 10^2 = 3.1415E2 = 3.1415e2$$

자료형-숫자

복소수(complex)

- 다음과 같은 꼴로 나타내는 수

$$a \pm bi$$

(단, a, b 는 실수, i 는 $i^2 = -1$ 을 만족)

```
>>> a = 5+4j
>>> a
(5+4j)
>>> type(a)
<class 'complex'>
```

자료형-숫자_사칙연산

연산자(Operator)

- 사칙 연산 기호처럼 연산에 사용되는 기호
- 연산자에도 우선순위가 있다.

| 연산 | 기호 |
|----------|----|
| 더하기 | + |
| 빼기 | - |
| 곱하기 | * |
| 제곱 | ** |
| 나누기 | / |
| 나눗셈의 몫 | // |
| 나눗셈의 나머지 | % |

자료형-숫자_사칙연산

연산자(Operator)

```
>>> a=3
>>> b=8
>>> a+b
11
>>> a-b
-5
>>> a*b
24
>>> a**b
6561
>>> a/b
0.375
>>> a//b
0
>>> a%b
3
```

자료형-진법

| 10진수 | 2진수 | 8진수 | 16진수 |
|------|--------|------|------|
| 0 | 0b0000 | 0o00 | 0x0 |
| 1 | 0b0001 | 0o01 | 0x1 |
| 2 | 0b0010 | 0o02 | 0x2 |
| 3 | 0b0011 | 0o03 | 0x3 |
| 4 | 0b0100 | 0o04 | 0x4 |
| 5 | 0b0101 | 0o05 | 0x5 |
| 6 | 0b0110 | 0o06 | 0x6 |
| 7 | 0b0111 | 0o07 | 0x7 |
| 8 | 0b1000 | 0o10 | 0x8 |
| 9 | 0b1001 | 0o11 | 0x9 |
| 10 | 0b1010 | 0o12 | 0xA |
| 11 | 0b1011 | 0o13 | 0xB |
| 12 | 0b1100 | 0o14 | 0xC |
| 13 | 0b1101 | 0o15 | 0xD |
| 14 | 0b1110 | 0o16 | 0xE |
| 15 | 0b1111 | 0o17 | 0xF |

자료형-진법

2진수(Binary number)

- 0b를 접두사로 사용
- bin(숫자)

8진수(Octal number)

- 0o(소문자 O)를 접두사로 사용
- oct(숫자)

16진수(heXadecimal number)

- 0x를 접두사로 사용
- hex(숫자)

```
>>> a=0b1011
>>> a
11
>>> bin(12)
'0b1100'
>>> a=0o25
>>> a
21
>>> oct(32)
'0o40'
>>> a=0x215
>>> a
533
>>> hex(255)
'0xff'
```

자료형-문자열

문자열(string)

- 문자, 단어 등으로 구성된 문자들의 집합
- 컴퓨터는 수를 다루는 기계
- 각 문자마다 번호를 붙이고 이를 표시할 때 문자로 하기로 약속
- ASCII, CP949, Unicode등
- 순서열(Sequence)형식의 한 종류이다.

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|-----|----|-----|-----------------------------|-----|----|-----|-------|-------|-----|----|-----|-------|-----|-----|----|-----|--------|-----|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 | | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | & | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | (| (| 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 |) |) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | : | 91 | 5B | 133 | [| [| 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 |] |] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | | DE |

자료형-문자열 생성

한 줄 문자열 생성

- 큰따옴표(")사용
 - "Hello World"
- 작은따옴표 사용(')
 - 'Hello World'

```
>>> a="Hello World"
>>> type(a)
<class 'str'>
>>> print(a)
Hello World
>>> a='Hello World'
>>> type(a)
<class 'str'>
>>> print(a)
Hello World
```

자료형-문자열 생성

여러 줄 문자열 생성

- 큰따옴표(") 생성
 - `"""Hello World!
Welcome to Python"""`
- 작은따옴표(') 생성
 - `"Hello World!
Welcome to Python"`

```
>>> a="""Hello World!  
... Welcome to Python"""  
>>> type(a)  
<class 'str'>  
>>> a  
'Hello World!\nWelcome to Python'  
>>> print(a)  
Hello World!  
Welcome to Python  
>>> a='''Hello World!  
... Welcome to Python'''  
>>> type(a)  
<class 'str'>  
>>> a  
'Hello World!\nWelcome to Python'  
>>> print(a)  
Hello World!  
Welcome to Python
```


자료형-문자열 생성

"와 ', 두 가지를 사용하는 이유

- 문자열에 작은따옴표나 큰따옴표를 편리하게 집어 넣기 위해서다.

```
>>> a="Python's favorite language"
>>> a
"Python's favorite language"
>>> print(a)
Python's favorite language
>>>
>>> a='"Python is very easy" he says.'
>>> a
'"Python is very easy" he says.'
>>> print(a)
"Python is very easy" he says.
```

```
>>> a='Python's favorite language'
File "<stdin>", line 1
    a='Python's favorite language'
        ^
SyntaxError: invalid syntax
```

자료형-문자열_escape code

이스케이프 코드

(escape code or escape sequence)

- \ (한글 폰트에서는 ₩, 백슬래시)
- 백슬래시 뒤에 문자나 숫자가 오는 조합
- 프로그래밍때 사용할 수 있도록 미리 정의해둔 문자 조합
- Linux에서는 이미 특수기능을 가지고 있는 문자기능을 탈출(escape)하여 일반적인 문자로 사용하기 때문에 escape문자라 한다.

자료형-문자열_escape code

| 코드 | 설명 |
|------|----------------------------------|
| \\ | 문자 \ 를 그대로 표현할 때 사용 |
| \t | 문자열 사이에 탭 간격을 줄 때 사용 |
| \' | 작은따옴표(')를 그대로 표현할 때 사용 |
| \" | 큰따옴표(")를 그대로 표현할 때 사용 |
| \n | 문자열 안에서 줄을 바꿀 때 사용 |
| \r | 캐리지 리턴(줄바꿈 문자, 현재 커서를 가장 앞으로 이동) |
| \f | 폼 피드(줄바꿈 문자, 현재 커서를 다음 줄로 이동) |
| \a | 비프음(출력시 PC 스피커에서 '뽕'소리가 난다. |
| \b | 백 스페이스 |
| \000 | 널 문자 |

자료형-시퀀스 자료형

값이 연속적(sequence)로 이어진 자료형

- 리스트, 튜플, range, 문자열
- 다음의 행동을 시퀀스 자료형에서 사용할 수 있다.
 - 연산
 - 연결
 - 반복
 - 인덱스
 - 슬라이싱

자료형-문자열 연산

문자열 연결하기(Concatenation)

— + 연산을 통해 연결이 가능하다.

```
>>> head="Python"
>>> tail=" is easy!"
>>> head+tail
'Python is easy!'
```

자료형-문자열 연산

문자열 반복하기

—*를 통해 반복이 가능하다.

```
>>> a="abcde"
>>> 2*a
'abcdeabcde'
>>> a*3
'abcdeabcdeabcde'
>>> print("="*15)
=====
```

자료형-문자열 인덱스

인덱스

- 무언가를 '가르킨다'는 의미
- 문자열에서는 각 문자(요소)들의 위치를 표시한 것이다.
- 인덱스를 첨자라고도 함

요소 "I need Python"

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I | | n | e | e | d | | P | y | t | h | o | n |
| 0 | | | | | | | | | | 1 | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 |

인덱스(첨자)

자료형-문자열 인덱스

인덱스

—0부터 시작한다.

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|--|---|--|---|--|---|--|---|
| I | | n | | e | | e | | d | | P | | y | | t | | h | | o | | n |
| 0 | | | | | | | | | | | | | | | | 1 | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | | | | | | | | |

—시퀀스객체[인덱스]: 해당 요소에 접근

```
>>> a="I need Python"
>>> a[0]
'I'
>>> a[4]
'e'
>>> a[22]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```


자료형-문자열 인덱스

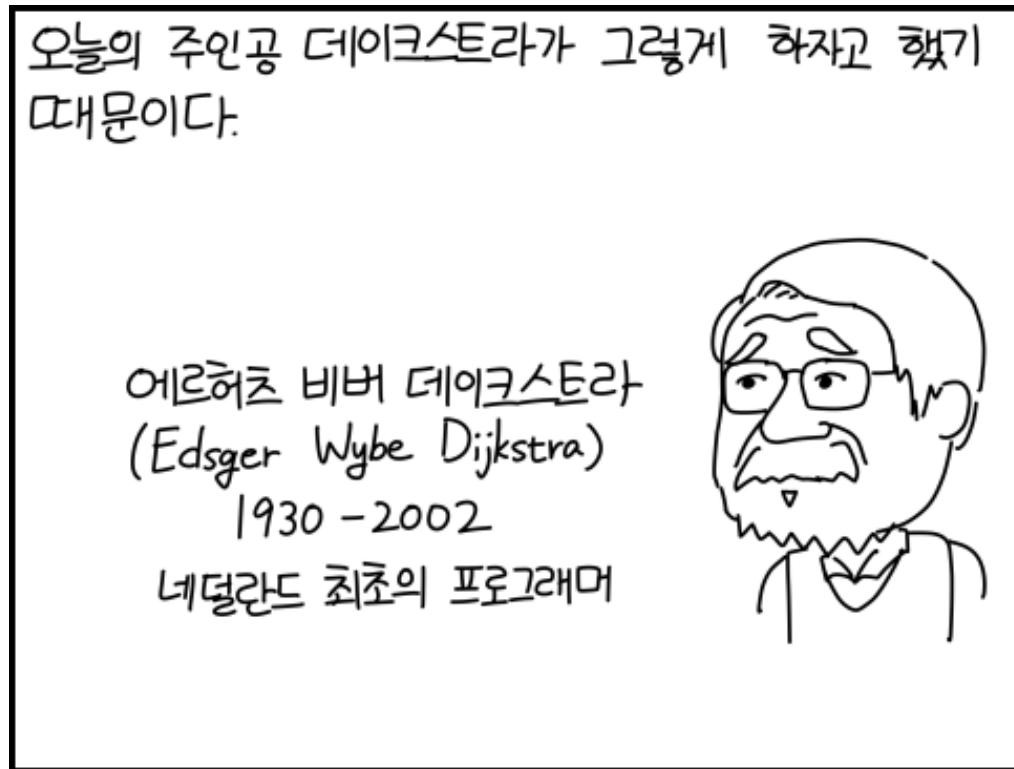
인덱스

--를 붙이면 뒤에서부터 가리킨다.

```
>>> a="I need Python"
>>> a[-1]
'n'
>>> a[-4]
't'
>>> a[-0]
'I'
>>> a[-22]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

자료형-문자열 인덱스

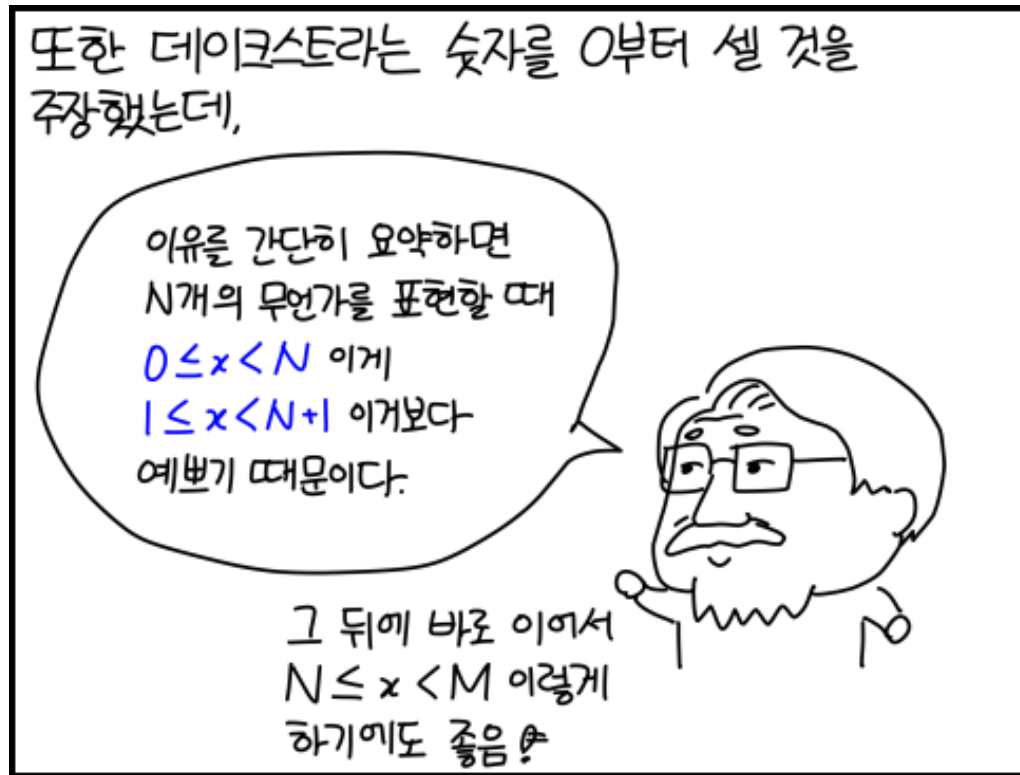
인덱스는 왜 0일까?



<https://www.facebook.com/engineertoon/photospcb.1002506639936191/1002505966602925/?type=3&theater>

자료형-문자열 인덱스

인덱스는 왜 0일까?



자료형-문자열 인덱스

인덱스는 왜 0일까?

— $1 \leq x \leq N$?

공집합을 표현할 때
 $1 \leq x \leq 0$ 으로 표현
해서 비직관적이다.

또한 데이크스트라는 숫자를 0부터 셀 것을
주장했는데,

이유를 간단히 요약하면
N개의 무언가를 표현할 때
 $0 \leq x < N$ 이게
 $1 \leq x < N+1$ 이거보다
예쁘기 때문이다.

그 뒤에 바로 이어서
 $N \leq x < M$ 이렇게
하기에도 좋음!



자료형-문자열 인덱스

인덱스는 왜 0일까?

- 그 외에 초기값이 주어지고, 일정하게 메모리 연산을 할 때도 인덱스를 0으로 계산하는 것이 편하다.



자료형-문자열 슬라이싱

슬라이싱

- 무언가를 '자른다'
- 시퀀스객체[시작인덱스:끝인덱스]
- 시작인덱스에서 시작
- 끝인덱스-1 에서 끝
- 초과해도 문제가 없음

```
>>> a="I need Python"
>>> a[0:4]
'I ne'
>>> a[4:6]
'ed'
>>> a[4:22]
'ed Python'
```

자료형-문자열 슬라이싱

슬라이싱

- 시퀀스객체[시작인덱스:끝인덱스]
- 시작과 끝 인덱스에 -를 넣어도 가능하다.

자료형-문자열 슬라이싱

슬라이싱

- 시퀀스객체[시작인덱스:끝인덱스]
- 시작인덱스가 없으면 맨 처음부터 시작한다.
- 끝인덱스가 없으면 맨 끝까지 표시한다.
- 둘 다 없으면 전체를 반환한다.

```
>>> a="I need Python"
>>> a[:6]
'I need'
>>> a[6:]
' Python'
>>> a[:]
'I need Python'
```


자료형-문자열 슬라이싱

슬라이싱

- 시퀀스객체[시작인덱스:끝인덱스:인덱스증가폭]
- 인덱스 증가폭만큼 뛰어서 문자열을 연결

```
>>> a="I need Python"
>>> a[5:11:2]
'dPt'
>>> a[::2]
'Ine yhn'
```

자료형-문자열 변경

"Pithon"을 "Python"으로 바꾸기

—불변형임으로 다음은 불가능

```
>>> a="Pithon"
>>> a[1]="y"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

—다음과 같이 한다.

```
>>> a="Pithon"
>>> a[:1]+'y'+a[2:]
'Python'
```

자료형-문자열 포매팅

내부의 숫자나 값만 다른 것을 출력할 때 사용

—Ex)

"현재 온도는 18도입니다."

"현재 온도는 20도입니다."

※C언어의 서식지정자와 유사

자료형-문자열 포매팅

문자열 포매팅

- 숫자 바로 대입

```
>>> "I ate %d apples." %3  
'I ate 3 apples.'
```

- 문자열 바로 대입

```
>>> " I ate %s apples." %"three"  
' I ate three apples.'
```

- 변수로 대입
- 2개 이상의 값 대입

자료형-문자열 포매팅

문자열 포매팅

—변수로 대입

```
>>> num = 3
>>> "I ate %d apples." %num
'I ate 3 apples.'
```

—2개 이상의 값 대입

```
>>> num = 3
>>> day = "three"
>>> "I ate %d apples for %s days" %(num, day)
'I ate 3 apples for three days'
```

자료형-문자열 포매팅

문자열 포맷코드

| 코드 | 설명 |
|----|------------------------|
| %s | 문자열(string) |
| %c | 문자 1개(character) |
| %d | 정수(integer) |
| %f | 부동 소수점(floating-point) |
| %o | 8진수 |
| %x | 16진수 |
| %% | 문자 % ('%' 그 자체) |

- ※ 문자열 포매팅을 사용하면서 %기호를 표시하고 싶을 때 %%를 사용

자료형-문자열 포매팅

문자열 포맷코드

- %s는 어떤 형태의 값이든 변환해 넣을 수 있다.
- 모르겠으면 %s쓰면 된다.

```
>>> "pi is %s" % 3.141592
'pi is 3.141592'
>>> "I ate %s apples" % 3
'I ate 3 apples'
```

자료형-문자열 포매팅

문자열 정렬

—오른쪽 정렬



```
>>> "%10s" %"hi"  
'          hi'
```

—왼쪽 정렬



```
>>> "%-10s" %"hi"  
'hi          '
```


자료형-문자열 포매팅

소수점 표현

—자릿수 제한

```
>>> "%0.4f" % 3.15215464
'3.1522'
```

—자릿수 제한과 정렬

| | | | | | | | | | |
|--|--|--|--|---|---|---|---|---|---|
| | | | | 3 | . | 1 | 5 | 2 | 2 |
|--|--|--|--|---|---|---|---|---|---|

```
>>> "%10.4f" % 3.15215464
'      3.1522'
```

자료형-문자열 메서드

문자열이 기본적으로 가진 메서드(함수)

문자 개수 세기(count)

```
>>> a="Hello"  
>>> a.count('l')  
2
```

자료형-문자열 메서드

문자 존재 위치 확인하기(find)

```
>>> a="Python is easy"
>>> a.find('s')
8
>>> a.find('q')
-1
```

—뒤쪽부터 확인하려면 rfind()를 사용한다.

문자 존재 위치 확인하기(index)

```
>>> a.find('q')
-1
>>>
>>> a="Python is easy"
>>> a.index('s')
8
>>> a.index('q')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
```

자료형-문자열 메서드

시작 문자열 확인하기(startswith)

```
>>> a="Hello"  
>>> a.startswith('He')  
True  
>>> a.startswith('lo')  
False
```

끝 문자열 확인하기(endswith)

```
>>> a="Hello"  
>>> a.endswith('He')  
False  
>>> a.endswith('lo')  
True
```

자료형-문자열 메서드

공백제거

—왼쪽 공백 제거(lstrip)

```
>>> ' Strip '.lstrip()  
'Strip '
```

—오른쪽 공백 제거(rstrip)

```
>>> ' Strip '.rstrip()  
' Strip'
```

—양쪽 공백 제거(strip)

```
>>> ' Strip '.strip()  
'Strip'
```

자료형-문자열 메서드

소문자-대문자 바꾸기(upper)

```
>>> "hello".upper()  
'HELLO'
```

대문자-소문자 바꾸기(lower)

```
>>> "HELLO".lower()  
'hello'
```

문자열 바꾸기(replace)

```
>>> a="Java is too easy"  
>>> a.replace("Java", "Python")  
'Python is too easy'
```

자료형-문자열 메서드

문자열 평가

- 숫자와 기호를 제외한 알파벳(영문, 한글)(isalpha)

```
>>> "ABC안녕하세요".isalpha()
True
>>> "ABC안녕 123".isalpha()
False
```

- 수만 존재(isnumeric)

```
>>> "ABC안녕 123".isnumeric()
False
>>> "123".isnumeric()
True
```

자료형-문자열 메서드

문자열 평가

- 알파벳과 수로만 이뤄져 있는지 평가(isalnum)

```
>>> '1234안녕'.isalnum()
True
>>> '1234'.isalnum()
True
>>> 'ABC'.isalnum()
True
>>> 'ABC 1234'.isalnum()
False
```


자료형-문자열 메서드

문자열 삽입(join)

- join에 들어온 문자열 사이에 원본 문자열을 넣는다.

```
>>> a="123"  
>>> a.join('abc')  
'a123b123c'
```

문자열 나누기(split)

- split에 입력된 구분자(delimiter)를 기준으로 문자열을 나눈다. 기본값은 화이트스페이스(스페이스, 탭, 엔터)이다. 반환값은 리스트이다.

```
>>> a="Python is easy"  
>>> a.split()  
['Python', 'is', 'easy']  
>>> a="a:b:c:d"  
>>> a.split(":")  
['a', 'b', 'c', 'd']
```

자료형-고급 문자열 포매팅

문자열의 format 메서드를 사용한 포매팅
좀 더 다양한 스타일로 문자열 포맷이 가능

—인덱스사용-직접 대입

```
>>> "I eat {0} apples".format(3)
'I eat 3 apples'
>>> "I eat {0} apples".format("three")
'I eat three apples'
```

자료형-고급 문자열 포매팅

—인덱스사용-변수

```
>>> number = 3
>>> "I eat {0} apples".format(number)
'I eat 3 apples'
>>> number = 3
>>> day = "three"
>>> "I eat {0} apples for {1} days".format(number, day)
'I eat 3 apples for three days'
```

자료형-고급 문자열 포매팅

—이름(키)으로 넣기

※ name=value 형태로 넣는다.

```
>>> num = 3
>>> day = "three"
>>> "I ate {num} apples for {day} days".format(num=num, day=day)
'I ate 3 apples for three days'
```

—인덱스와 이름 혼용

```
>>> "I ate {0} apples for {day} days".format(3, day="three")
'I ate 3 apples for three days'
```

자료형-고급 문자열 포매팅

정렬

—왼쪽 정렬

```
>>> "{0:<10}".format("hi")  
'hi          '
```

—가운데 정렬

```
>>> "{0:^10}".format("hi")  
'      hi      '
```

—오른쪽 정렬

```
>>> "{0:>10}".format("hi")  
'          hi '
```

자료형-고급 문자열 포매팅

공백 채우기

※ 정렬문자 앞에 채울 문자를 넣음

```
>>> "{0:@^10}".format("hi")  
'@@@@hi@  
>>> "{0:=>10}".format("hi")  
'=====hi'
```

자료형-고급 문자열 포매팅

소수점 표현하기

```
>>> a=4.34984514
>>> "{0:0.4f}".format(a)
'4.3498'
>>> "{0:10.4f}".format(a)
'      4.3498'
```

'{'또는 '}' 표현하기

```
>>> "{or}".format()
'{or}'
```

자료형-자료형 변환

자료형 변환

- 모든 자료형은 객체임으로 생성자 존재
- 이를 통해 자료형을 변환 가능
- class에서 다시 설명
- 함수가 있어 이를 처리한다 봐도 무방하다
 - 정수: `int(변환 객체(ex 문자열))`
 - 실수: `float(변환 객체)`
 - 복소수: `complex(변환 객체)`
 - 문자열: `str(변환 객체)`

자료형-자료형 변환

예시

```
>>> a="123"  
>>> a  
'123'  
>>> int(a)  
123  
>>> float(a)  
123.0  
>>> a=123  
>>> str(a)  
'123'
```

입력

사용자가 입력한 값을 변수에 할당
함수 input을 사용한다.

※ 함수는 뒤에서 다시 다룬다.

```
>>> a=input()  
Hello, World!  
>>> a  
'Hello, World!'
```

input은 입력되는 값을 모두 문자열 취급한다.

입력

프롬프트를 띄워서 사용자 입력받기

—input("프롬프트 내용")

※ C처럼 print이후 scanf를 할 필요가 없다.

```
>>> num = input("숫자를 입력하세요: ")
숫자를 입력하세요: 23
>>> num
'23'
```

출력

print(출력할 객체)

- print함수를 통해 출력이 가능
- 기본적으로 사용되는 자료형은 출력이 가능
- 관련 내용은 "더블 언더바 변수"에서 더 자세히 다룬다.

출력-심화

문자열을 그대로 쓰면 + 연산과 동일

```
>>> print('Python'+'is'+'easy')  
Pythoniseasy
```

값을 여러 개 출력할 때는 ,(coma)로 구분

```
>>> print(1, 2, 3)  
1 2 3  
>>> print('Python', 'is', 'easy')  
Python is easy
```

출력-심화

값 사이에 문자 넣기(sep)

- 공백이 아닌 다른 문자를 넣기.
- separator(구분자)에서 옴

```
>>> print(1, 2, 3, sep='x')
1x2x3
>>> print(4, 5, 6, sep='\n')
4
5
6
```

출력-심화

한 줄에 결과값 출력하기(end)

- print()는 자동으로 줄바꿈이 된다. 키워드 매개 변수 end="를 통해 줄바꿈을 없앤다.
※ 키워드 매개 변수는 함수에서 배운다.

```
>>> for i in range(3):  
...     print(i)  
...  
0  
1  
2
```

```
>>> for i in range(3):  
...     print(i, end=' ')  
...  
0 1 2 >>>  
>>> for i in range(3):  
...     print(i, end='&')  
...  
0&1&2&>>>  
>>>  
>>> for i in range(3):  
...     print(i, end='')  
...  
012>>>
```

Coding Convention?

Coding convention(코딩 규약)

- 특정 프로그램 언어에서 사용하는 가이드라인
- 프로그래밍의 스타일, 실행 방법 등을 추천한다.
- 또한 들여쓰기, 주석, 이름짓기 등이 기재되어 있다.
- 이를 통해 소스코드를 읽는 것과 프로그램을 유지 보수하는 것을 쉽게 한다.

PEP8

-Python 코드 스타일 가이드

PEP(Python Enhance Proposal)

- 파이썬을 개선하기 위한 개선 제안서
- PEP8은 언어의 코딩 컨벤션을 나타낸 것
- <https://www.python.org/dev/peps/pep-0008/>

Python은 다른 사용자가 코드를 만들어도 비슷한 코드로 수렴하도록 디자인되어 있다.

PEP8

-Python 코드 스타일 가이드

코드 레이아웃

- 들여쓰기는 공백 4칸
 - 탭을 사용할 수도 있으나, 공백 4칸을 권장하며 둘을 혼용해서는 안된다.
 - IDE에서 탭을 공백 4칸으로 바꿔준다.
 - C와는 다르게 Python에서는 블록을 들여쓰기로 나눈다.
- 한 줄은 최대 79자
- 최상위 함수와 클래스 정의는 2줄씩 띄어쓴다.
- 클래스 내의 메소드 정의는 1줄씩 띄어쓴다.

PEP8

-Python 코드 스타일 가이드

Whitespace

- 불필요한 공백은 피한다.
 - 대괄호와 소괄호 안
(ex. `spam(ham[1]) => spam(ham[1])`)
 - ,(comma)와 ;(semicolon), :(colon) 앞
(ex. `print x , y ; => print x, y;`)
- 키워드 인자와 인자의 기본값의 =은 붙여쓴다.

PEP8

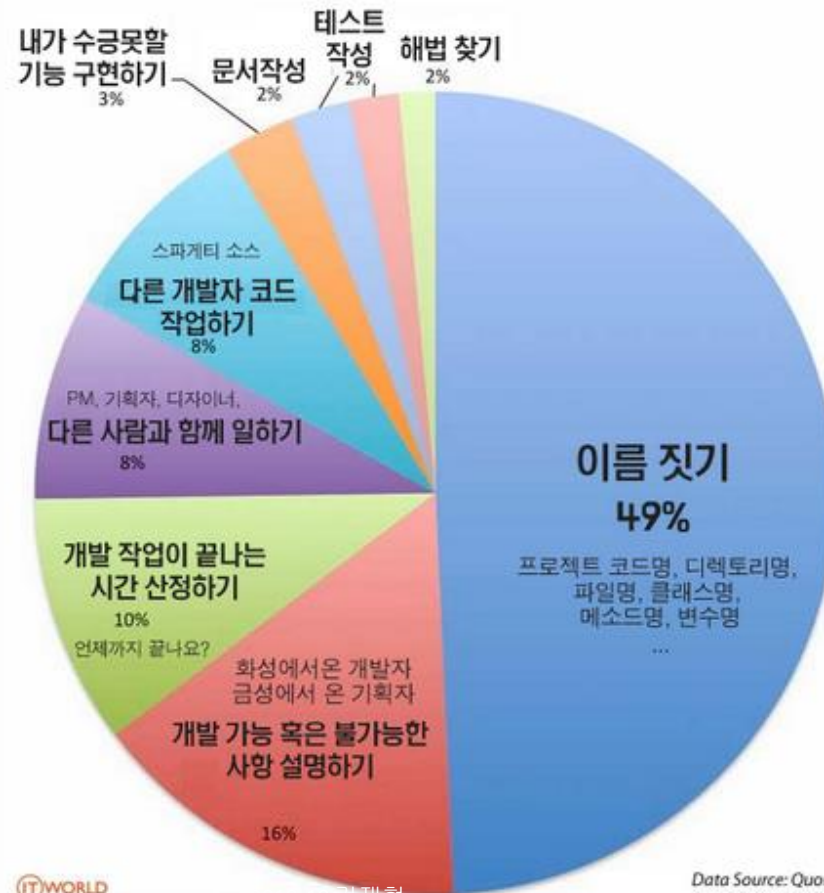
-Python 코드 스타일 가이드

Naming(이름 짓기)

- 소문자 l, 대문자 O, 대문자 i는 변수명으로 사용하지 않는다.
- 폰트에 따라 가독성이 좋지 않다.
- 함수, 변수, 속성: lowercase_underscore
- 상수: ALL_CAPS
 - 단, 상수는 모듈수준에서 사용.
- 클래스와 예외: CapitalizeWord(카멜 케이스)
- 모듈: 짧은 소문자로 구성, 필요하면 밑줄로 나눈다.

Naming(이름 짓기)

프로그래머가 가장 힘들어하는 일은?



Naming(이름짓기)

```
import random, pygame, sys
from pygame.locals import *
def hhh():
    global a, b
    pygame.init()
    a = pygame.time.Clock()
    b = pygame.display.set_mode((640, 480))
    j = 0
    k = 0
    pygame.display.set_caption('Memory Game')
    i = c()
    hh = d(False)
    h = None
    b.fill((60, 60, 100))
    g(i)
    while True:
        e = False
        b.fill((60, 60, 100))
        f(i, hh)
        for eee in pygame.event.get():
            if eee.type == QUIT or (eee.type == KEYUP and eee.key == K_ESCAPE):
                pygame.quit()
                sys.exit()
```

Naming(이름짓기)

변수나 함수가 표현하고 있는 것을 정확하게 설명

- 일반적으로 최적의 변수 길이는 10-16문자
- sum, total, average, max, min 등
- 숫자를 받는다고 num이라고만 쓰지 말고, 어떤 숫자인지도 같이 적어준다.
- 임시변수도 a나 temp를 자주 사용하지만, 유지되는 기간이 길면 무엇을 하는지 알려주는 것이 좋다.

Naming(이름짓기)



PEP8

-Python 코드 스타일 가이드

Naming(이름짓기)

- 변수명에서 _(밑줄, under_score)를 앞과 뒤에 붙이는 것은 각 의미가 있다.
- `_single_leading_underscore`: 내부적으로 사용되는 변수
- `single_trailing_underscore`: 파이썬 기본 키워드와의 충돌을 피한다.
- `__double_leading_underscore`: 클래스 속성으로 사용되면 그 이름을 변경
(ex. FooBar에 정의된 `__boo`는 `_FooBar__boo`가 된다.)
- `__double_leading_and_trailing_underscore__`: 마술(magic)을 부리는 용도로 사용되거나 사용자가 조정할 수 있는 네임스페이스의 속성. 즉, 파이썬 내에서 중요하게 사용되니 만들지 말것.

PEP8

-Python 코드 스타일 가이드

Comments(주석)

- 코드에 따라 주석은 갱신되어야 한다.
- 불필요한 주석은 달지 말기
- 한 줄 주석은 신중히
- Docsting

기본과제-자판기1

자판기(vending_machine.py)

- 본 과제는 수업 전반에서 수정 및 작성할 파일이다.
- 본 과제를 통해 컴퓨터적 사고를 통한 문제해결방식 및 파이썬 기초를 학습하도록 한다.
- 수업 초반에 배운 문제정의에서 요구사항을 확인했었다.
- 수업 후반에 배울 문제분해 방법 중 "연속 프로토타이핑"으로 제작

기본과제-자판기1

vending_machine.py

- 고객이 프로토타입으로 자판기의 외형을 만들기 원한다.
- 프로그램을 시작하면 다음을 입출력하고 종료 돈을 넣으세요: (사용자로부터 입력받기)
 1. 블랙커피(100원)
 2. 밀크커피(150원)
 3. 고급커피(200원)
 4. 거스름돈넣은 돈: (입력 받은 출력)원

기본과제-자판기1

vending_machine.py

- 아스키코드를 넣어 외형을 꾸며도 상관없다
- 본인의 디렉터리 아래에 vending_machine 디렉터리를 만들어 저장한다.
- 단, 프로그램에 입력받는 값은 정수이다.
(에러처리를 하지 않는다.)

기본과제-자판기1

vending_machine.py

—예시

```
돈을 넣으세요 : 100
1. 블랙커피 (100원)
2. 밀크커피 (150원)
3. 고급커피 (200원)
4. 거스름돈
넣은 돈 : 100원
root@goorm: /workspace/PythonSeminar18(master) #
```

기본과제-화씨, 섭씨 변환

convert_fahrenheit_celsius.py

- 화씨 온도($^{\circ}\text{F}$, Fahrenheit)는 미국에서 많이 사용하는 온도 체계로 영하 17.7°C 에서 37.7°C 를 백등분한 것이다.
- 섭씨 온도($^{\circ}\text{C}$, Celsius)는 전 세계에서 널리 쓰이는 온도 단위로 물의 어는점을 0도, 끓는 점을 100도로 정해 백등분 한 것이다.

기본과제-화씨, 섭씨 변환

convert_fahrenheit_celsius.py

—화씨온도를 섭씨로 변환하는 프로그램을 제작한다.

1. 변환할 화씨온도를 입력받는다.

2. 화씨온도에서 32를 뺀 후, 5/9를 곱하면 섭씨온도가 된다.

—단, 입력받는 값은 정수나 부동소수점이다.

—단, 소수점 둘째자리까지 출력한다.

※ r+한자를 통해 °C, °F를 입력할 수 있다.

기본과제-화씨, 섭씨 변환

예시

```
변환할 화씨 온도 (°F)를 입력하십시오 : 25
25.00°F는 -3.89°C입니다 .
root@goorm:/workspace/PythonSeminar18/Exercise/Lecturer/conv
elsisus(master)# python3 convert_fahrenheit_celsius.py
변환할 화씨 온도 (°F)를 입력하십시오 : 0
0.00°F는 -17.78°C입니다 .
root@goorm:/workspace/PythonSeminar18/Exercise/Lecturer/conv
elsisus(master)# python3 convert_fahrenheit_celsius.py
변환할 화씨 온도 (°F)를 입력하십시오 : 100
100.00°F는 37.78°C입니다 .
```