

2장 파이썬

파이썬

→ 용법이 쉬운

→ 코드가 직관적임

→ 인터프리터 방식

→ 스크립트 모드식

식별자 → 어떤 대상을 유연하게 구분할 수 있는 이름

예약어 → 프로그래밍 언어를 다룰 때 필요한 의미가 주어진 이름

리터럴 → 소스코드의 고정된 값을 나타냄

수치	정수 (int)	10 리터럴 -30 0xFFE
	실수 (float)	3.14 -0.45
	복소수 (complex)	Complex(1,2) 1+2j
	불리언 (bool)	True False
시퀀스	문자열 (str)	'game' 'over' "C"
	리스트 (list)	[] [1,2,3] [help, 0.3, 1.4]
	튜플 (tuple)	(0,1,2,3) ('hello', 'world')
매핑	딕셔너리 (dict)	{3.14:"Phi", 4.5:"score"}
집합	집합 (set, frozenset)	{1,2,3} {'one', 'two'}

변수

• 데이터를 담은 공간

• 저장되는 데이터를 마음대로 바꿀 수 있음

• C와는 다르게 미리 변수를 선언하는 부분 없음

• 파이썬에서는 모든 자료가 클래스로 묶여 만들어져 객체

• 변수는 다른 객체를 참조하는 참조형 또는 포인터식

리스트 []

튜플 ()

딕셔너리 {}

가변 길이

변수

a = 1
가변 공간

가변 길이
리터럴

= 1
↓
32

변수 참조 참조형

연산

in 연산자
a in banana
a가 banana에 들어있는가?
True

- 나눗셈 연산자 변경
 * 연산자 / : 실수 나눗셈 (결과가 실수)
 * 연산자 // : 정수 연산 [% => 나머지] $5 \div 2 = 1$
- 아항 연산자: ** 곱승 ($2^{**}5 \rightarrow 2^5$)
- 단항 연산자: ++, -- 제곱 x
 * $x++$ 이 아닌 $x+=1$

9 & 8
1001 1000
1001
1000
1000

- 관계 연산자: >, <, >=, <=, ==, !=
- 논리 연산자: ||, &&, ! => AND, or, not
- bit 연산자 & (AND), | (OR), ^ (XOR), ~ (비트 반전), << (시프트)
- 함수 호출
 예) $y = \text{sum}(a, b)$
 $w = x * y + \text{func}(a, b, c)$
 $\text{print}(\text{"game over"})$

키보드 입력 함수: input() 문자열 or int(input()) -> 입력을 정수로 변환

화면 출력 함수: print()

분기 (제어문)

if, else

if elif else 문

if ~ () 사용하기 tap 잘쓰기

for 변수

반복

in

for in 반복 가능 객체, 시퀀스 :

시퀀스 객체

range() 알아두기 범위 n ~ n-1 까지

반복 가능 객체

S, E, step

start end -1 범위

예) range(2, 10) => 2, 3, 4 ... 9 까지

range(1, 10, 2) => 2, 4, 6, 8 까지

☆ format () 안으로 /
fstring

NO. _____
year month day ()

90년대와 2000년대
가, = -90년대
기양수, 음수 가능

컬렉션 자료형

기본 자료형

리스트 []
배열

- 시퀀스형 [문자열 + 인덱싱, 슬라이싱 str
리스트(list)
튜플

- 매핑형 - 딕셔너리
- 집합형 - 집합

클래스
메소드 리스트엔스
다양성

리스트(list) - [] list = [], big3 = list() / ^{같은 (변하는 행...)} _{이것이 문자열과 다른 것}

- 소변형 배열
- 가장 많이 사용되는 시퀀스형 자료구조
- 언어 외의 다른 프레임에서는 배열을 따로 제공하지 않음
- 리스트를 배열처럼 사용할 수 있음

리스트 연산

- append () 추가 (변형)
- extend () 추가
- count 항목의 개수를 세고 그 개수를 반환
- index (n, [시작], [종료]) 항목 n이 들어 있는 가장 작은 인덱스로 반환
- insert (n, x) n 위치에 항목 x를 추가
- pop 꺼내고 반환
- remove () 제거
- reverse () 순서 뒤집기
- sort ([key], [reverse]) 정렬

튜플 => 변경불가 write 상수이므로 ()로 함 읽기만 가능 read only

- 리스트(list)와 유사하지만 크기나 값으로 변경할 수 없음
- 리스트에서 사용할 수 없는 많은 연산들은 튜플에서도 사용 가능
- 튜플만을 위해 추가된 메소드는 없음
- 메모리는 효율적임

딕셔너리 { } '키' : '값', 없는 키값이 들어오면 추가함, 값은 업데이트(새로고침)

- 키(key)와 값으로 이루어진 항목들의 집합
- 맵(map)을 지칭

★ 집합

- 리스트와 비슷한 개념
- 원소의 중복을 허용하지 않고 원소들의 사기에 순서가 없음
- 선형 자료구조로 보기에 어려움

사용과 정의 함수

- 파이썬 내장 함수

- type(), len(), ord

- 파이썬 사용자 정의 함수 <def>

함수 헤더

def fied_max(A) ← 이름 ← 매개변수

data

Print("max=", fied_max

몸체
이해
정기이다.

```
max = A[0]
for i in
    ...
return max
```

- 함수에서 처리된 결과를 되돌려주기 위해 사용
- 함수에서 여러개의 결과 반환 가능

모듈 패키지

- 함수나 변수 또는 클래스를 모아놓은 파일
- 다른 파이션 프로그램에서 불러와 사용할 수 있게끔 만든 파이션 파일
- 이름공간
- 모든 모듈은 자신의 이름공간을 갖고 있음
- 같은 이름 공간 안에서 모든 식별자를 자유롭게 사용 가능

클래스

- 파이션은 함수가 중심이 되는 순수한 프로그래밍과 객체지향 프로그래밍 기법 모두 지원
- 객체지향 기법은 클래스와 객체가 중심이 되는 프로그래밍 기법
- 클래스는 객체를 찍어내는 틀이며 객체는 변수와 같이 자료로 저장되는 프로그램 구성원

생성자

- 객체가 생성될 때마다 자동으로 호출되는 함수
- 객체에서 사용될 때마다 데이터를 정의하고 초기화함

--init-- (self)
 ↓
 클래스
 ↓
 --del--

- 생성자는 --init-- 으로 정해져 있음.

멤버함수

- Method 라고도 하며, 해당 클래스의 object에서만 호출 가능

1+1=2

연산자 중목

"1"+"1"="11"

- 사용과 같이 클래스는 객체들에게 표준 연산자들을 적용할 수 있도록 연산자 중목 허용 (단항 연산자, 이항 연산자).

재정의

상속 상속된 / 지식 + 부모 클래스 중 하나에 의해 이미 제공된 메소드를 특정 형태로 구현

- 기존의 정해진 클래스로부터 멤버를 추가하여 새로운 클래스를 간단하게 만드는 방법

- 객체지향 언어들에게 클래스의 상속 제공

- 상속 관계는 클래스 → 부모 클래스, 상속 받는 클래스 → 자식 클래스

- 지식 IS-A 관계의 관계

