2장. 자료형

파이썬 기초 자료형과 구조

Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	associative mapping (aka dictionary)	

Table 1.2: Commonly used built-in classes for Python

그림1. 파이썬 자료형과 그 중 변경 가능한 구조와 그렇지 않은 구조(immutable class)

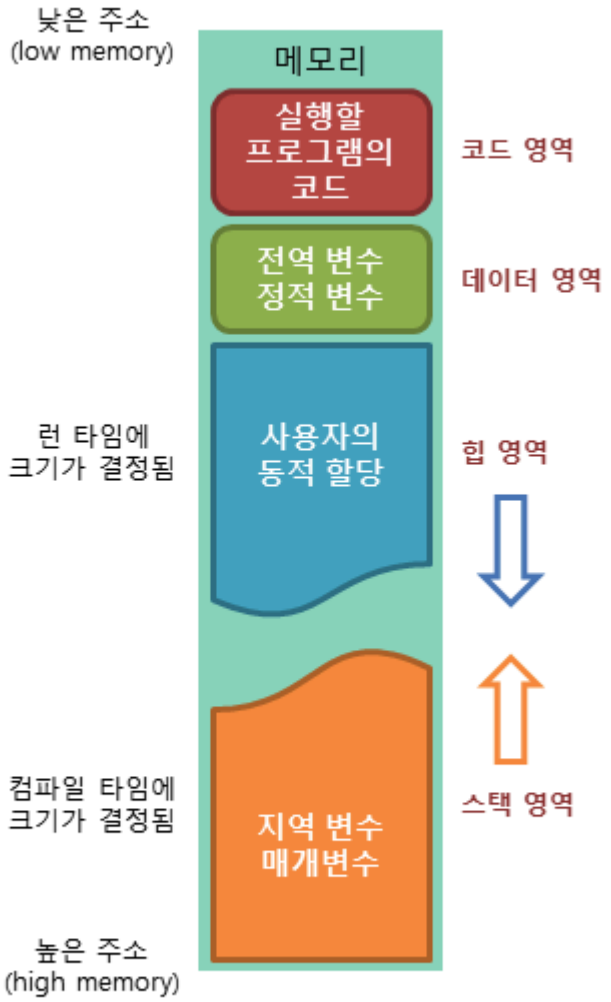
Summary

Collection	Mutable?	Ordered?	Use When…
str	No	Yes	You want to keep track of text
list	Yes	Yes	You want to keep track of an ordered sequence that you want update
tuple	No	Yes	You want to build an ordered sequence that you know won't change or that you want to use as a key in a dictionary or as a value in a set
set	Yes	No	You want to keep track of values, but order doesn't matter, and you don't want duplicates. The values must be immutable.
dictionary	Yes	Yes	You want to keep a mapping of keys to values. The keys must be immutable.

그림2. 순서집합인지와 변경가능한 자료형 요약본(출처: 서울대 데이터사이언스 대학원)

변수: 자료형의 데이터(객체)

객체의 메모리 주소를 같게해서 다른 변수가 동일하게 되게 하는 과정



http://www.tcpschool.com/c/c_memory_structure

- 코드 영역 : 실행할 프로그램의 코드가 저장됨.
- 데이터 영역 : 전역변수와 정적변수가 저장됨
- 힙 영역 : 사용자가 동적으로 할당하는 변수나 메소드가 저장됨
- 스택 영역: 지역변수(local variable)나 매개변수(parameter)가 저장됨

```
class Car:
    def __init__(self,w):
        self.wheels = w

    def getWheels(self):
        return self.wheels
```

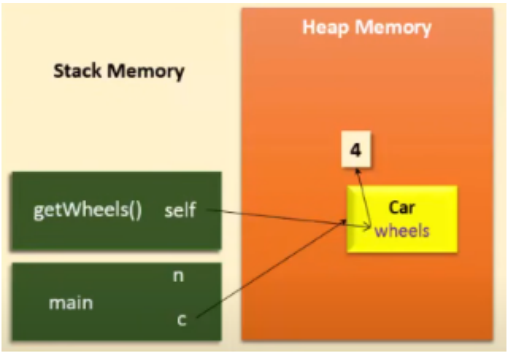


그림 참조

3장. 조건문과 반복문

제어문

- 조건문(if, elif, else): 참과 거짓을 판단하는 문장(논리연산자 필수)

비교연산자	설명
x < y	x가 y보다 작다.
x > y	x가 y보다 크다.
x == y	x와 y가 같다.
x != y	x와 y가 같지 않다.
x >= y	x가 y보다 크거나 같다.
x <= y	x가 y보다 작거나 같다.

그림3. 조건문 도식화

그림4. 비교연산자를 통한 논리연산자(참/거짓 bool 형)

◗ 반복문(for, while)

for문

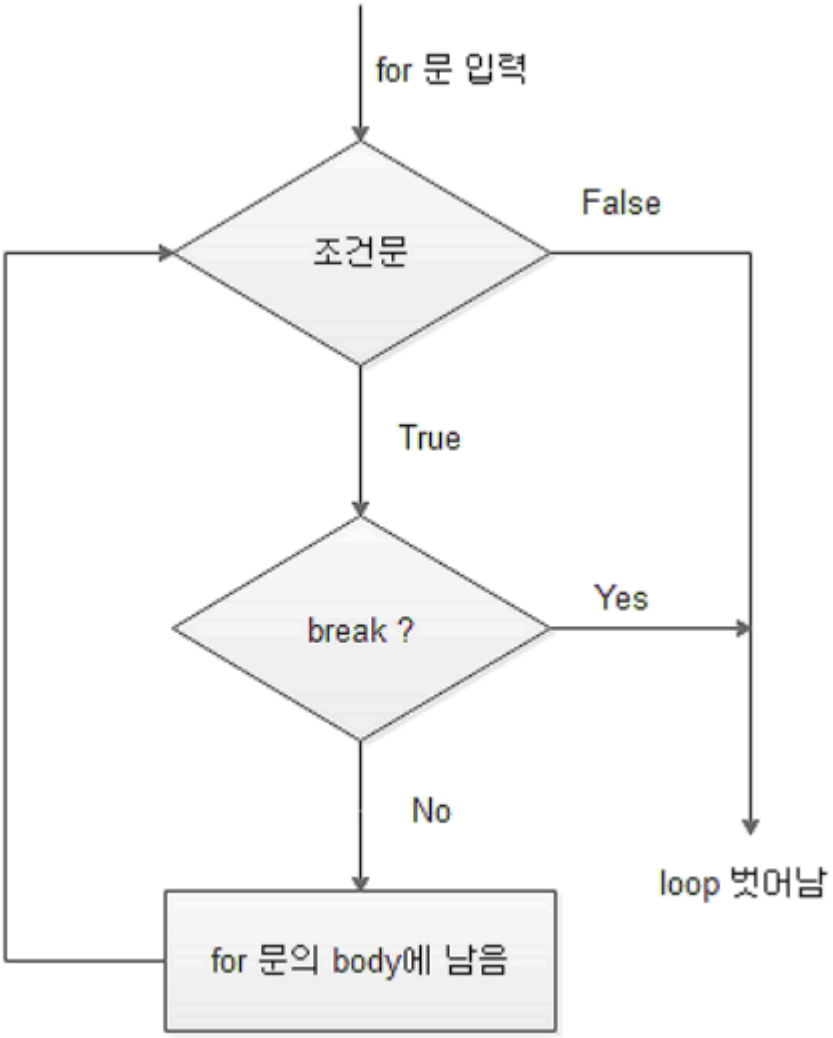


그림5. for문의 전반적인 내용 그림

continue 문: for 문에서 특정 조건에서만 코드를 실행하지 않고 다음으로 이동하고 싶을 때

while문

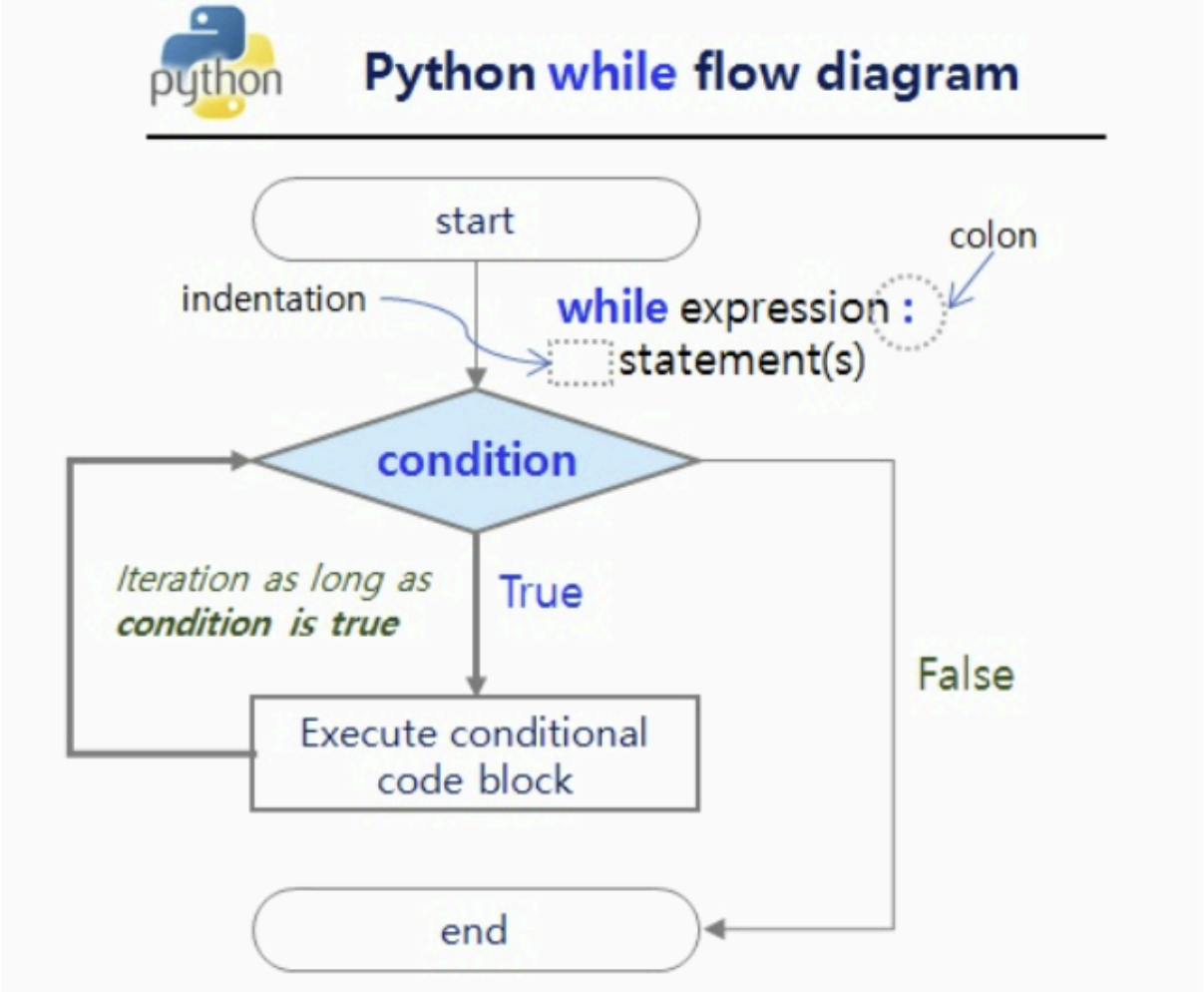


그림6. While문 요약

4장. 함수

함수

정의: 인수(변수, parameter)를 입력 받아 코드를 실행하고 결과물을 출력하도록 하는 입력값과 출력값의 관계

-위치 매개변수: 함수 내 매개변수가 순서로 인해 이뤄지는 것

-키워드 매개변수: 함수 내 매개변수가 변수명으로 인해 이뤄지는 것

-*args: 임의의 개수의 위치인자를 전달(전달된 변수는 tuple 형태로 저장됨)

-**kwargs: 임의의 개수의 키워드인자를 전달(전달된 변수는 딕셔너리 형태로 저장됨)

-local variable: 함수 내부에서만 쓸 수 있는 변수

-global variable: 함수 내부 외에도 사용가능한 변수

→함수를 정의할 땐 대부분 local variable을 쓴다. Global variable을 쓰면 내부 변수랑 충돌할 수 있기 때문에

고차함수

-map: 함수는 이터레이터(iterator)를 반환함 (for문에 쓰이기 적합!)

-filter: 참인 값만 반환(데이터 전처리에 유용)

-reduce: 반복가능한 객체를 인자로 얻음(모듈에서 불러와야함)

-lambda: 함수 정의 코드를 한줄로 간단히 요약하는 함수문법

*파이썬 3.6버전 이후부터 print(f“blah : {something}, …”)으로 매개변수 입력 간단해짐

5장. 클래스 및 객체 지향 언어

객체

객체: 실제 복잡한 문제를 추상화한 상태와 행동을 갖는 독립적 단위(self라는 것은 바로 그 클래스의 객체를 가리킴)

OOP(Object-Oriented Programming, 객체 지향 언어**(**프로그래밍**))**

OOP 이전의 컴퓨팅 역사: 초기에는 명령어의 순서와 절차에만 신경 썼기 때문에 코드의 재사용이 어려움

OOP(Python, Java, C++) 개발 이후: 객체들간의 상호작용을 통해 데이터 관리 및 문제해결

객체지향 프로그래밍의 구조 및 핵심용어 정리

- 생성자(__init__): 일관되게 클래스 실행. 객체 생성 시 초기상태를 설정할 수 있음
- 소멸자(__del__): 객체를 소멸시키는 문구(안전한 종료)
- 클래스: 객체를 만들기 위한 설계도(일반적)
- 객체: 클래스를 기반으로 실체화된 인스턴스(구체적)
- 인스턴스: 클래스에서 생성**(**실체화**)**된 객체
- 메서드: 인스턴스의 상태를 변경**(**클래스 내부의 함수**)**
- 오버로딩: 하나의 클래스 내에서 동일한 이름의 메서드나 생성자를 여러 개 정의하는 것
- 메서드 오버로딩: 같은 이름의 메서드 여러 번 정의, but 매개변수 유형이나 수를 다르게하는 것
- 생성자 오버로딩: 생성자도 메서드 일종. 동일한 이름의 생성자 여러개 정의
- 상속: 기존의 클래스를 새로운 클래스에 전달하는 것
- 오버라이드: 상위 클래스의 메소드를 하위 클래스에서 재정의하는 것

MRO(Method Resolution Order): 메소드 결정 순서

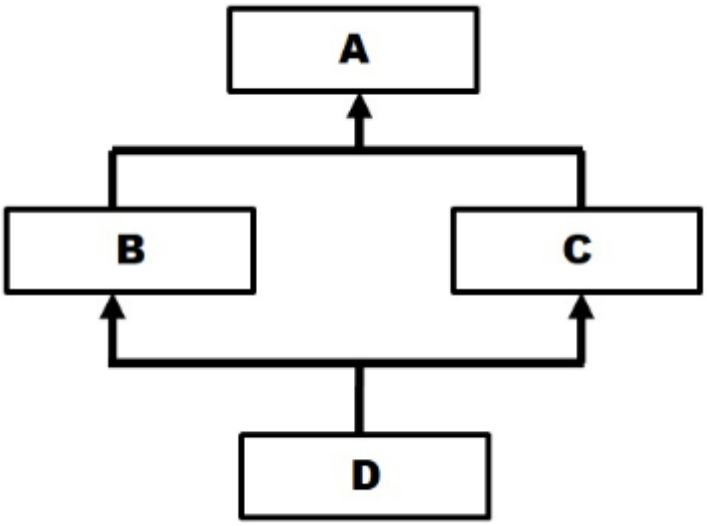


그림7. MRO 중 다중상속이 발생할 수 있으므로 상속 시, 상위(부모) 클래스를 상속하는 순서 필수

SOLID 원칙**:** 효율적인 코드짜기

두문자	약어	개념
S	SRP	단일 책임 원칙 (Single responsibility principle) : 한 클래스는 하나의 책임만 가져야 한다.
O	OCP	개방-폐쇄 원칙 (Open/closed principle) : "소프트웨어 요소는...확장에는 열려 있으나 변경에는 닫혀 있어야 한다."
L	LSP	리스코프 치환 원칙 (Liskov substitution principle) : "프로그램의 객체는 프로그램의 정확성을 깨뜨리지 않으면서 하위 타입의 인스턴스로 바꿀 수 있어야 한다." 계약에 의한 설계를 참고하라.
I	ISP	인터페이스 분리 원칙 (Interface segregation principle) : "특정 클라이언트를 위한 인터페이스 여러 개가 범용 인터페이스 하나보다 낫다."
D	DIP	의존관계 역전 원칙 (Dependency inversion principle) : 프로그래머는 "추상화에 의존해야지, 구체화에 의존하면 안 된다." 의존성 주입은 이 원칙을 따르는 방법 중 하나다.

〈표 1〉 SOLID 원칙의 세부 내용

그림8. SOLID 원칙

-연관관계: 한 객체가 다른 객체와의 관계를 통해 특정작업 수행

-컴포지션: 객체 간의 강력한 연결

-접근 제한자(self.__enter): 특정 변수명이나 메서드를 private으로 만드는 코드(앞에 언더바 2개)

객체지향 프로그래밍의 기본원칙 **4**가지

1. 캡슐화: 데이터와 함수를 하나로 묶는 과정

장점: 코드가 깔끔해지고 관리 용이, 데이터의 무결성 유지, 변경의 유연성, 객체의 독립성 보장

2. 상속: 기존의 클래스를 새로운 클래스에 전달하는 것

3. 다형성: 같은 이름의 메서드나 연산자를 다른 클래스에서 다르게 구현하는 것

4. 추상화: 추상 클래스는 하나 이상의 추상 메소드를 가진 클래스

→추상 메소드는 선언만 있고 구현이 없는 메소드

모듈

모듈: 함수 변수 클래스를 모아놓은 파일

패키지 → __init__(생성자 함수) → 모듈로 저장

모듈의 네임스페이스: 변수, 함수, 클래스 등의 이름이 저장되는 공간

__name__==“__main__”(모듈 실행)

__name__==“모듈 이름”(다른 모듈)

sys.path로 모듈 경로 수정(리스트 형)

독립적(import: 호출)

스크립트: 모듈과 비슷(차이점: 호출이 목적이 아니라 활용이 목적)

패키지

패키지: 모듈들을 하나로 묶은 것

기타

컴프리헨션(Comprehension): 파이썬 내에서 리스트, 딕셔너리, 집합과 같은 자료구조를 생성, 추출, 검색할때 간결하고 좀 더 이해하기 쉽게 만드는 방법