Python 기초

1. Python 개발환경

* 인터프리터 언어(Interpreter)

소스코드를 컴파일하지 않고, 한 줄씩 소스 코드를 읽어서 바로 실행

컴파일 언어에 비해 느릴 수 있지만 빌드 과정이 없이 바로 실행 가능함

* 객체 지향 프로그래밍(Object Oriented Programming)

파이썬은 모두 객체로 이뤄져 있음

* 동적 타이핑

변수에 별도의 타입 지정이 필요 없음

* 개발환경

대화형 환경

Python 기본 Interpreter

Python Jupyter Notebook

* 인터프리터가 대화형 모드로 동작함

프롬프트에 코드를 작성하면 해당 코드가 실행됨

여러 줄의 코드가 작서오디는 경우 보조 프롬프트가 사용도미

* Python이 설치된 환경에서는 기본적으로 활용 가능하나 디버깅 및 코드 편집, 반복 실행이 어려움
* Jupyter Lab

1. 기초문법

* 코드 스타일 가이드

코드를 ‘어떻게 작성할지’에 대한 가이드 라인 (반드시 읽어보기)

잘못된 습관과 같은 부분을 교정하는게 필요하다

ppt확인 – pep 8 가이드

* 주석(Comment)

한 줄 주석은 #으로 표현한다.

여러 줄의 주석은 한 줄씩 #을 사용하거나, “””또는 ‘’’으로 표현한다.

다만 “””또는 ‘’’으로 표현하는 방법은 docstring을 위해 사용한다.

Docstring : 함수 / 클래스의 설명을 작성

Ex) foo.\_\_doc\_\_ ( 이런식으로 하면 클래스 설명이 보임)

다중 주석처리 : 드래그 후 ctrl + /

* 코드 라인

코드는 1줄에 1문장(statement)이 원칙

문장(statement)은 파이썬이 실행 가능(executable)한 최소한의 코드 단위

기본적으로 파이썬에서는 세미콜론(;)을 작성하지 않음

한 줄로 표기할 때는 세미콜런(;)을 작성하여 표기할 수 있음

1. 변수와 식별자

* 변수 : 변수는 할당 연산자(=)를 통해 값을 할당

Type() : 변수에 할당된 값의 타입 확인

I d() : 변수에 할당된 값(객체)의 고유한 아이덴티티(identity) 값이며, 메모리 주소를 확인

* 할당연산자(=)

같은 값을 동시에 할당할 수 있음

다른 값을 동시에 할당 할 수 있음

* 값 swap

임시 변수 활용 : tmp를 이용해서 값 바꾸기

X, y = y, x

* 식별자(Identifiers)

변수, 함수, 모듈, 클래스 등을 식별하는데 사용하는 이름(name)

규칙

식별자의 이름은 영문 알파벳, 언더스코어(-), 숫자로 구성

첫 글자에 숫자가 올 수 없음

길이 제한이 없고, 대소문자를 구별

다음의 키워드(Keywords)는 예약어(reserved words)로 사용할 수 없음

예약어(Reserved words)는 사용할 수 없음

내장함수나 모듈 등의 이름으로도 만들면 안됨

기존의 이름에 다른 값을 할당하게 되므로 더 이상 동작하지 않음

1. 데이터 타입

* 숫자(Number)

Int(정수, integer)

* Python 3부터 long타입이 없고, 모두 int로 표기
* 매우 큰 수를 나타낼 때 오버플로가 발생하지 않음
* 오버프롤(overflow) : 데이터 타입별로 사용할 수 있는 메모리의 크기를 넘어서는 상황
* Arbitrary precision arithmetic(임의 정밀도 산술)을 통해 고정된 혀애의 메모리가 아닌 가용 메모리들을 활용하여 모든 수 표현에 활용

Float(부동소수점, 실수, floating point number)

* 정수가 아닌 모든 실수는 float 타입
* 부동소수점

실수를 컴퓨터가 표현하는 방법 - 2진수(비트)로 숫자를 표현

이 과정에서 floating point rounding error가 발생하여 예상치 못한 결과가 발생

Floating point rounding error : 부동 소수점에서 실수 연산 과정에서 발생가능

값 비교하는 과정에서 정수가 아닌 실수인 경우 주의할 것

실수를 bit로 표현하면 유한개의 비트로는 표현할 수가 없음

차이가 매우 근소하면 같다라고 하는 것이다.(유한대 표현의 한계)

따라서, 실수는 == (동등연산자로는 힘듬)

해결책 :

Isclose : 얼마나 가까운지 확인하여 반환

Complex(복소수, complex number)

* 실수부와 허수부로 구성된 복소수는 모두 complex 타입
* 허수부를 j로 표현
* 문자열(string)
* 모든 문자는 str 타입
* 문자열은 작음 따옴표(‘)나 큰 따옴표(“)를 활용하여 표기
* 문자열을 묶을 때 동일한 문장부호를 활용
* 큰 따옴표와 작은 따옴표는 하나만 골라 쓰자
* Pep8에서는 소스코드 내에서 하나의 문장부호를 선택하여 유지하도록 함
* 이스케이프 시퀸스(escape sequence)

문자열 내에서 특정 문자나 조작을 위해서 역슬래시(/)를 활용하여 구분

* 변수의 값을 문자열의 자리 표시자(placeholder)로 대체하는 방법
* 참/거짓(Boolean)
* True/false 값을 가진 타입은 bool
* 비교하기 위한 타입
* None
* 값이 없음을 표현하기 위한 타입은 NoneType
* A = None 형식으로 넣는것도 가능(첫글자 반드시 대문자)
* 타입변환
* 암시적 타입 변환

사용자가 의도하지 않고, 파이썬 내부적으로 타입 변환 하는 경우

Bool, number(int, float, complex)

* 명시적 타입 변환

Str \*(작성된 형식 맞아야함), float - > int

Str \*, int -> float

Int, float, list, tuple, dict -> str

1. 연산자

* 비교 연산자
* 논리 연산자

단축 평가 : 결과가 확실한 경우 두번째 값은 확인하지않음

0 : false, 1 : true 따라서 and와 or 연산자에 따라 값이 바뀜

* 복합 연산자

연산과 대입이 함께 이루어진다

* 특정 변수가 비어있는지 확인하기 위해서는 is 연산자 쓰는 것을 권장

Ex) x in None

* Indexing / Slicing

[]를 통해 값을 접근하고, [:]를 통해 슬라이싱 가능함

* 연산자 우선 순위

()

Slicing

Indexing

\*\*

단항연산자(+,-) : 부호

산술 연산자(\*,/,%)

산술 연산자(+,-)

비교 연산자, in, is

Not

And

or

1. 표현식

표현식은 평가(evaluate)되고, 값으로 변경

하나의 값으로 환원reduce)될 수 있는 문장

식별자, 값, 연산자로 구성

1. 컨테이너

여러 개의 값을 저장할 수 있는 것(객체)

시퀴스형(sequence)형 : 순서가 있는(ordered) 데이터

- 순서가 있다 ! = 정렬되어 있다

- 리스트(list), 튜플(tuple), 레인지(range), 문자형(string), 바이너리(binary)

비 시퀸스형(non-sequence) : 순서가 없는(unordered) 데이터

- 세트(set), 딕셔너리(dictionary)

-시퀸스형 컨테이너

1. 리스트(list)

리스트는 순서가 있는 시퀸스로 인덱스를 통해 접근

대괄호([]) 혹은 list()를 통해 생성

값에 대한 접근은 list[i]

리스트 내부는 데이터 타입이 다른 요소들도 한 리스트 내에 삽입 가능

1. 튜플(tuple)

튜플은 일반적으로 파이썬 내부에서 활용됨

인덱스 통한 접근은 my\_tuple[i]

중괄호를 통해 선언

순서가 있고 값을 수정할 수 없기에

Multiple assignment

추후 함수에서 복수의 값을 반환하는 경우에도 활용

두 변수의 값 swap을 할때도 쓰임 (파이썬 내부에서 쓰인다)

하나의 항목으로 구성된 튜플은 생성시 값 뒤에 쉼표를 붙여야 함

1. 레인지(range)

Range는 숫자의 시퀸스를 나타내기 위해 사용

* 기본형 : range(n)

0부터 n-1까지의 숫자의 시퀸스

* 범위 지정 : range(n,m)

n부터 m-1까지의 숫자의 시퀸스

* 범위 및 스텝 지정 : range(n,m,s)

n부터 m-1까지 s만큼 증가시키며 숫자의 시퀸스

* Range는 형변환을 해야한다. 그대로 사용하면 range type으로 찍힘 '

11. containment test

- 시퀸스 포함 여부 확인

- in, not in

- 시퀸스 간의 concatenation(연결/연쇄)가능 [1,2]+[3] 이런 형식으로 더할 수 있음

레인지는 안됨

- 시퀀스 자체를 반복 [0] \* 10 , (1,2) \* 10 가능, 레인지는 안됨

- \* : 에스터리스크

- index를 통해 내부 요소에 접근 가능

- min과 max : 시퀸스에서의 최소/최대값

- count : 시퀸스에서 특정 원소의 개수(시퀸스에 등장하지 않는 경우 0 반환)

12. 세트(set)

- 순서가 없는 자료구조

중괄호({}) 혹은 set()을 통해 생성

빈 세트를 만들기 위해서는 set()을 반드시 활용해야함, {} 이렇게 하면 빈 딕셔너리 선언임

순서가 없어 별도의 값에 접근할 수 없음

- 수학에서의 집합과 동일한 구조를 가짐

집합 연산이 가능

**중복된 값이 존재하지 않음**

- 순서가 없어 인덱스 접근 등 특정 값에 접근할 수 없음

- {1,2,3,1,1,2,3,4} = {1,2,3,4} // 중복값허용이안됨

- 연산자 - | & : 차집합, 합집합, 교집합

- 수학의 집합이라 생각하면 쉬움

13. 딕셔너리(dictionary)

- key와 value가 쌍으로 이뤄진 자료구조

- key는 변경 불가능한 데이터 코드로만 작성가능

- value는 변경가능한 것도 가능, list, dict등

14. 변경이 가능한 데이터

- list, set, dictionary

- 복사의 방법에서 차이가 난다.

- 만약 복사를 통해 가져가게 되면 같은 주소를 바라보기에

- 복사한 인덱스를 바꿔도 원본 역시도 바뀌게 된다.

15. 변경이 불가능한 데이터

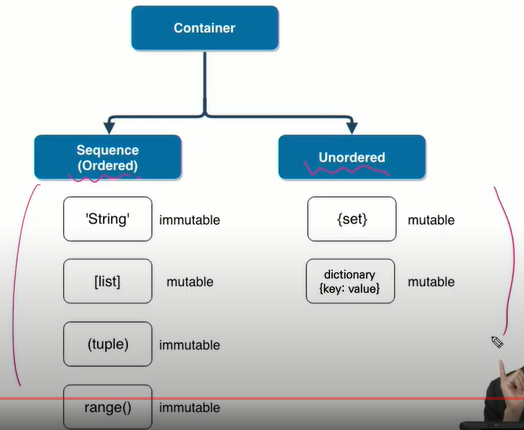
- string, int 등등 dict의 key에 사용할 수 있는 것들은

복사를 해도 주소를 가져오는 것이 아닌 객체를 복사하는 것이기에

재할당의 개념으로 생각해 원본이 바뀌지 않는다

- 변경 시에는 copy() 등과 같이 다른 방법들이 있다.

**\*\*\*\*\*\*\*\*\*\* 컨테이너 간의 형 변환 직접 해보기**

****

16. 제어문(Control Statement)

- 파이썬은 기본적으로 위에서부터아래로 순차적으로 명령을 수행

- 특정 상황에 따라 코드를 선택적으로 실행(분기/조건)하거나 계속하여 실행하는 제어가

필요함

- 제어문은 순서도(flow chart)로 표현가능

- if문 참/거짓을 판단할 수 있는 조건식과 함께 사용

- statement : 문 expression : 표현식

- space != tab (4:1) 원래라면 인식이 되지 않으나 vscode에서는 tab이 4space로

변환시켜줌

17. 반복문(Loop Statement)

- while문

: 종료 조건에 해당하는 코드를 통해 반복문을 종료시켜야 함

- for문

: 반복 가능(iterable)한 객체를 모두 순회하면 종료 (별도의 종료 조건이 필요 없음)

string, tuple, list, range 등과 같은 iterable 객체

- 반복 제어

: break, continue, for-else

18. 리스트 순회하기 - enumerate

내장 함수 enumerate의 경우 (index, value) 형태의 tuple로 구성된 열거 객체를 반환

ex ) for idx, member in enumerate(members) :

print(idx, member)

만약 start = 1 하면 index를 1부터 시작하도록 조정가능

for idx, member in enumerate(members, start = 1):

print(idx, member)

19. for-else

: 끝까지 반복문을 실행한 이후에 else문 실행

break를 통해 중간에 종료되는 경우 else 문은 실행되지 않음

20. pass

: 반복문 아니어도 사용 가능, 특별히 할 일이 없을 때 자리를 채우는 용도로 사용