Python

ICT Innovation square

김 보 연

Why python?

https://docs.python.org/ko/3/tutorial/index.html
https://docs.python.org/ko/3/tutorial/appetite.html

- 배우기 쉽다.
- 간결한 문법과 풍부한 표준 라이브러리를 가지고 있다.
- 자료 구조들과 객체 지향 프로그래밍에 대해 간단하고도 효과적인 접근법을 제공
- 인터프리터적인 특징들은 대부분 플랫폼과 다양한 문제 영역에서 스크립트 작성과 빠른 응용 프로그램 개발에 이상적인 환경을 제공
- 버전 2.x와 3.x는 일부 호환이 되지 않지만 두 버전 모두 이용자가 많음
- 절차형, 함수형, 객체지향 패러다임 모두를 지원(다양한 방식, 다양한 관점으로 프로그래밍할 수 있으며, 수많은 기법과 디자인 패턴을 접목시킬 수 있음)

자료형(Data Type)

- 값에 대한 type이 중요함
- Data type에 따라 서로 다른 기술적인 체계가 필요
  - 지원하는 연산 및 기능이 다르기 때문
- 컴퓨터에서는 이진수(binary number)를 사용해서 값을 표현하고 관리
  - 수치형 = numeric
    - 산술 연산을 적용할수 있는 값
  - 불리언 = boolean
    - 참/거짓을 뜻하는 대수값. 보통 컴퓨터는 0을 거짓, 0이 아닌 것을 참으로 구분
  - 문자, 문자열
    - 숫자 "1", "a", "A" 와 같이 하나의 낱자를 문자라 하며, 이러한 문자들이 1개 이상있는 단어/문장와 같은 텍스트
  - Compound = Container = Collection
    - 기본적인 데이터 타입을 조합하여, 여러 개의 값을 하나의 단위로 묶어서 다루는 데이터 타입
  - None
    - 존재하지 않음을 표현하는 타입

Numeric

■ 수를 표현하는 기본 리터럴

○정수(integer)

➢0, 100, 123 과 같은 표현

○ 실수(floating-point number)

➢중간에 소수점이 있는 0.1, 4.2, 3.13123 와 같은 식

➢0. 으로 시작하는 실수값에서는 흔히 앞에 시작하는 0 제외 가능. .5 는 0.5를 줄여쓴 표현

○부호를 나타내는 - , +를 앞에 붙일 수 있다. (-1, +2.3 등)

■ 참/거짓을 의미하는 부울대수값.(boolean)

○자체가 키워드로 True/False를 사용하여 표현

■기본적으로 그냥 숫자만 사용하는 경우, 이는 10진법 값으로 해석.

○10진법외에도 2진법, 8진법, 16진법이 존재.

➢2진법 숫자는 0b로 시작(대소문자를 구분하지 않음)

➢8진법 숫자는 0o로 시작(대소문자를 구분하지 않음)

➢16진법숫자는 0x로 시작(대소문자를 구분하지 않음)

■ str

○글자, 글자가 모여서 만드는 단어, 문장, 여러 줄의 단락이나 글 전체

○문자열은 홑따옴표, 쌍따옴표, doc스트링(''' ''')으로 표현

○큰 따옴표와 작은 따옴표를 구분하지 않지만 양쪽 따옴표가 맞아야 함. (문자열을 둘러싸는 따옴표와 다른 따옴표는 문자열 내의 일반 글자로 해석)

➢"apple" , 'apple' 은 모두 문자열 리터럴로 apple이라는 단어를 표현

○따옴표 세 개를 연이어서 쓰는 경우에는 문자열 내에서 줄바꿈 허용됨.

➢흔히 함수나 모듈의 간단한 문서화 텍스트를 표현할 때 쓰임.

자료형과 자료형의 변환

함수명	의미	사용법
type( )	값이나 변수의 자료형을 반환	type(10)
int( )	정수형으로 변환한 값을 반환	int('10')
float( )	실수형으로 변환한 값을 반환	float('1.2')
bool( )	부울형으로 변환한 값을 반환	bool(1)
str( )	문자열로 변환한 값을 반환	str(23)
ord( )	ASCII 코드값 반환	ord('a')
chr( )	ASCII 코드값에 해당하는 글자를 반환	chr(65)
len( )	문자열의 길이 반환	len('abcd')

Operation(연산)

- 산술연산
  - 계산기
- 비교연산
  - 동등 및 대소를 비교. 참고로 '대소'비교는 '전후'비교가 사실은 정확한 표현
  - 비교 연산은 숫자값 뿐만 아니라 문자열 등에 대해서도 적용할 수 있음.
- 비트연산
- 논리연산
  - 비교 연산의 결과는 보통 참/거짓. 이러한 불리언값은 다음의 연산을 적용. 참고로 불리언외의 타입의 값도 논리연산을 적용
- 멤버십 연산
  - 특정한 집합에 어떤 멤버가 속해있는지를 판단하는 것으로 비교연산에 기반을 둠
  - is, is not : 값의 크기가 아닌 값 자체의 정체성(identity)이 완전히 동일한지를 검사
  - in, not in : 멤버십 연산. 어떠한 집합 내에 원소가 포함되는지를 검사 ('a' in 'apple')

연산자 우선 순위

	Operator	Description
lowest precedence	or	Boolean OR
	and	Boolean AND
	not	Boolean NOT
	in, not in	membership
	==, !=, <, <=, >, >=, is, is not	comparisons, identity
	|	bitwise OR
	^	bitwise XOR
	&	bitwise AND
	<<, >>	bit shifts
	+, -	addition, subtraction
	*, /, //, %	multiplication, division, floor division, modulo
highest precedence	+x, -x, ~x	unary positive, unary negation, bitwise negation
	**	exponentiation

- Python에서의 변수
  - 값(객체)을 저장하는 메모리 상의 공간을 가리키는(object reference: 객체 참조) 이름
  - Python은 모든 것이 객체이므로 변수보다는 식별자(identifier)로 언급. 변수로 통용해서 사용하기도 함
  - 변수 식별자의 경우, 선언 및 할당이 동시에 이루어짐
    - 대입문의 좌측에 최초로 나타날 때 변수가 생성됨 --> 선언
    - 대입 연산문에 의해 값이 지정됨 → 할당

list

- 여러 개의 데이터를 하나의 변수에 저장하고자 할 때 매우 유용함
- 리스트에 들어있는 데이터를 아이템(item)이라고 부름
- 리스트는 [ ]로 묶어주고, 리스트의 아이템은 ,(컴마)로 구분함
- 리스트의 아이템에 접근하고자 할 때에는 인덱스를 사용함
- 인덱스는 0부터 시작함!!

List 부분 추출(slicing)

- 리스트 슬라이싱
  - 리스트의 일부를 정한 규칙으로 추출
  - 끝 은 인덱스의 범위를 넘어가도 됨

사용 방법	의미
리스트명[시작: 끝]	[시작, 시작+1, 시작+2, ..., 끝-1] 위치의 원소 추출
리스트명[시작: 끝: 스텝]	[시작, 시작+스텝, 시작+스텝×2, ..., 시작+스텝×k] 위치의 원소 추출 (단, 시작+스텝×k < 끝)
리스트명[시작: 끝: -스텝]	[시작, 시작-스텝, 시작-스텝×2, ..., 시작-스텝×k] 위치의 원소 추출 (단, 시작>끝, 시작-스텝×k > 끝)

- range 함수: 규칙적인 숫자열로 구성된 범위 생성

사용 방법	의미
range(끝)	[0, 1, 2, 3, …, 끝-1]
range(시작, 끝)	[시작, 시작+1, 시작+2, …, 끝-1]
range(시작, 끝, 스텝)	[시작, 시작+스텝, 시작+스텝×2, …, 시작+스텝×k] (단, 시작+스텝×k < 끝)
range(시작, 끝, -스텝)	[시작, 시작-스텝, 시작-스텝×2, …, 시작-스텝×k] (단, 시작+스텝×k > 끝)

- 튜플(tuple)
  - 순서가 있는 원소들의 묶음.
  - 불변객체
  - 튜플의 원소에 대한 엑세스는 인덱스를 사용하며, [ ]를 사용(리스트와 동일)

- 딕셔너리
  - 사전식 자료 관리
  - key와 value 쌍으로 이루어진 item의 모음
  - key와 value는 : 로 구분
  - 딕셔너리는 중괄호{ }로 묶여 있음
  - key를 인덱스로 사용

Dictionary 관련 함수

Operation	Result
len(d)	Return the number of items in the dictionary *d*.
d[key]	Return the item of *d* with key *key*.
d[key] = value	Set d[key] to *value*.
del d[key]	Remove d[key] from *d*.
key in d	Return True if *d* has a key *key*, else False.
key not in d	Equivalent to not key in d.
clear()	Remove all items from the dictionary.
copy()	Return a shallow copy of the dictionary.

Dictionary 관련 함수

Operation	Result
items()	Return a new view of the dictionary's items ((key, value) pairs).
keys()	Return a new view of the dictionary's keys.
pop(*key*)	If *key* is in the dictionary, remove it and return its value, else return *default*.
get(*key*)	Return the value for *key* if *key* is in the dictionary, else *default*.
popitem()	Remove and return an arbitrary (key, value) pair from the dictionary.
setdefault(*key*)	If *key* is in the dictionary, return its value. If not, insert *key* with a value of *default* and return *default*.
update([*other*])	Update the dictionary with the key/value pairs from *other*, overwriting existing keys. ex. d.update(red=1, blue=2).
values()	Return a new view of the dictionary's values.

구문(statement)

■ 구문(statement) = 문

○ 예약어(**reserved word, keyword**)와 표현식을 결합한 패턴

○ 컴퓨터가 수행해야 하는 하나의 단일 작업(**instruction**)을 명시.

- 할당(대입, **assigning statement**)
  - **python**에서는 보통 '바인딩(**binding**)'이라는 표현을 씀, 어떤 값에 이름을 붙이는 작업.
- 선언(정의, **declaration**)
  - 재사용이 가능한 독립적인 단위를 정의. 별도의 선언 문법과 그 내용을 기술하는 블럭 혹은 블럭들로 구성.
    - » **Ex) Python**에서는 함수나 클래스를 정의
  - 블럭
    - » 여러 구문이 순서대로 나열된 덩어리
    - » 블럭은 여러 줄의 구문으로 구성되며, 블럭 내에서 구문은 위에서 아래로 쓰여진 순서대로 실행.
    - » 블럭은 분기문에서 조건에 따라 수행되어야 할 작업이나, 반복문에서 반복적으로 수행해야 하는 일련의 작업을 나타낼 때 사용하며, 클래스나 함수를 정의할 때에도 쓰임.
- 조건(분기): 조건에 따라 수행할 작업을 나눌 때 사용.
  - **Ex) if** 문
- 반복: 특정한 작업을 반복수행할때 사용.
  - **Ex) for** 문 및 **while** 문
- 예외처리

멤버십연산자

- in : 해당 요소가 있으면 True, 없으면 False를 반환함
- not in : 해당 요소가 없으면 True, 있으면 False를 반환함

논리연산자

연산자	의미	사용 예	의미
and	그리고	x and y	x와 y가 둘 다 True이면 True를 출력하고 둘 중 하나라도 False이면 False를 출력한다.
or	혹은	x or y	x와 y가 둘 다 False이면 False을 출력하고 둘 중 하나라도 True이면 True을 출력한다.
not	~아니다	not x	x가 True면, False를, x가 False이면 True를 출력한다.(반대로 출력)

함수 정리

- 함수는 어떻게 만들지?
  - 함수 정의 파트와 함수 호출 파트로 이루어진다.
  - 함수 정의 파트가 함수 호출 파트보다 먼저 작성되어 있어야 한다.
- 전달인자, 매개 변수, 반환
  - 함수 호출 파트(외부)에서 함수 정의 파트(내부)로 전달되는 값을 전달인자라 한다.
  - 함수 정의 파트에서는 값을 받을 때 매개변수로 받는다.(매개변수는 지역변수)
    - 전달인자나 매개변수를 파라미터라고도 부름
  - 함수 정의 파트에서 함수 호출 파트로 값을 내 보낼 때에는 반환(return)을 사용한다.
  - 전달인자, 매개변수, 반환이 없는 함수도 있다.
- 지역 변수, 전역 변수
  - 함수 안에서 선언된 지역변수와 함수 밖에서 선언된 전역변수가 있다.

The `print()` function prints the specified message to the screen, or other standard output device.

The message can be a string, or any other object, the object will be converted into a string before written to the screen.

Syntax

```
print(object(s), sep=separator, end=end, file=file, flush=flush)
```

Parameter Values

Parameter	Description
object(s)	Any object, and as many as you like. Will be converted to string before printed
sep='*separator*'	Optional. Specify how to separate the objects, if there is more than one. Default is ' '
end='*end*'	Optional. Specify what to print at the end. Default is '\n' (line feed)
file	Optional. An object with a write method. Default is sys.stdout
flush	Optional. A Boolean, specifying if the output is flushed (True) or buffered (False). Default is False

- input()
  - 프로그램 실행 중에 키보드로부터 데이터를 입력 받고자 할 때 사용하는 함수
  - 입력 받은 값은 문자열(string)타입!!!

If the *prompt* argument is present, it is written to standard output without a trailing newline. The function then reads a line from input, converts it to a string (stripping a trailing newline), and returns that. When EOF is read, `EOFError` is raised.

디폴트 인자 (default argument)

- 디폴트 인자(default argument, = 기본 인자) : 함수의 매개변수는 기본값을 가질 수 있다.
- 인자가 부족하면 기본값을 채워준다.

키워드 인자(keyword argument)

- **매개 변수 이름을 지정해서 값을 전달하는 방식**
- 키워드 인수의 순서는 상관없음.

가변 인자

- 몇 개의 인자를 받을지 정해지지 않은 인자
- 함수로 전달되는 값의 개수와 상관없이, 한 개의 매개변수로 받을 수 있다.

키워드 가변 인자

- 키워드 가변 인자: 키워드 + 가변
- 함수에서 인자를 받을 때, 매개변수 앞에 ** 을 붙인다.

lambda 함수

- 한 줄짜리 함수
- 메모리의 효율적 사용하기 위한 방법
- 한번 사용하고 다음 줄로 넘어가면 메모리(힙(heap) 영역)에서 사라짐(보통 map, reduce 함수에서 같이 사용)

enumerate 함수

- 요소를 추출할 때 번호를 붙여서 추출

zip 함수

- list/tuple 에서 같은 위치에 있는 값을 병렬적으로 추출함

```
alist = ['a1', 'a2', 'a3']
blist = ['b1', 'b2', 'b3']
```

```
list( zip(alist, blist) )
```

```
[('a1', 'b1'), ('a2', 'b2'), ('a3', 'b3')]
```

List Comprehension

- 리스트 조건 제시법
  - for 반복을 사용하여 자동으로 리스트를 생성하는 방법
- 문법

  [생성연산식 for 변수 in 컬렉션]

➔ 조건제시법: 리스트, 딕셔너리, 집합

generator

- Generator
  - Iterable 자료를 생성하는 함수
  - yield 문이 포함된 함수
  - next() 함수 사용해서 자료에 접근

generator expression

- generator expression : iterable 자료를 생성하는 식

```
( 연산 for 변수 in 컬렉션)
```

- generator 생성식을 객체화화여 next() 함수에 의해 값을 추출