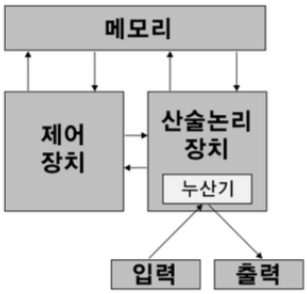
Python 2

Variable

- 가장 기초적인 프로그래밍 문법 개념
- 데이터(값)을 저장하기 위한 메모리 공간의 프로그래밍 상 이름

> **폰 노이만 아키텍처**
>
> 
>
>
> 사용자가 컴퓨터에 값을 입력하거나 프로그램을 실행할 경우, 그 정보를 먼저 메모리에 저장시키고 CPU가 순차적으로 그 정보를 해석하고 계산하여 사용자에게 결과값 전달

- 변수: 프로그램에서 사용하기 위한 특정한 값을 저장하는 공간
- 선언 되는 순간 **메모리 특정영역에 물리적인 공간 할당**
- **변수**: 메모리 주소를 가짐
- **변수에 들어가는 값**: 메모리 주소에 할당
- 변수 이름 작명법
    - 알파벳, 숫자, 언더스코어(_) 로 선언 가능
    - 변수명은 의미 있는 단어로 표기하는 것이 좋음
    - 변수명은 대소문자 구분
    - 특별한 의미가 있는 예약어는 쓰지 않음

기본 자료형 (primitive data types)

- datatype: 파이썬이 처리할 수 있는 데이터 유형

유형			설명	예시	선언 형태
수치자료형	정수형	integer	양/음의 정수	1,2,3,100, -9	data = 1
	실수형	float	소수점이 포함된 실수	10.2, -9.3, 9.0	data = 9.0
문자형(문자형)		string	따옴표 (' / ")에 들어가 있는 문자형	abc, a20abc	data = 'abc' data ='a'
논리/불린 자료형		boolean	참 또는 거짓	True, False	data = True

Dynamic Typing

코드 실행시점에 데이터의 Type을 결정하는 방법

```
# python
f_int = 100
```

```
# java
Integer f_int = 100;
```

연산자(Operator)와 피연산자(operand)

- +,-,*,/같은 기호들을 연산자라고 칭함
- "3+2"에서 3과 2는 피연산자,+는 연산자임
- 수식에서 연산자의 역할은 수학에서 연산자와 동일
- **연산의 순서는 수학에서 연산 순서와 같음**
- 문자간에도 + 연산 가능 => concatenate

> **컴퓨터가 이진수 이유**
>
> 컴퓨터는 실리콘이라는 재료로 만든 반도체로 구성
>
> 반도체는 특정 자극을 줬을 때 전기를 통할 수 있게 하는 물질
>
> 도체와 부도체에 반해 반도체는 전류의 흐름의 제어가 가능
>
> **전류가 흐를 때1, 흐르지 않을 때 0으로만 숫자를 표현할 수 있음**
>
> 이진수 한자리를 bit라 칭하고 8개의 bit는 1byte

list

시퀀스 자료형, 여러 데이터들의 집합

- 인덱싱 indexing

  list에 있는 값들은 주소(offset) 가짐

  주소를 사용해 할당된 값 호출
- 슬라이싱 slicing

  list의 값들을 잘라서 쓰는 것

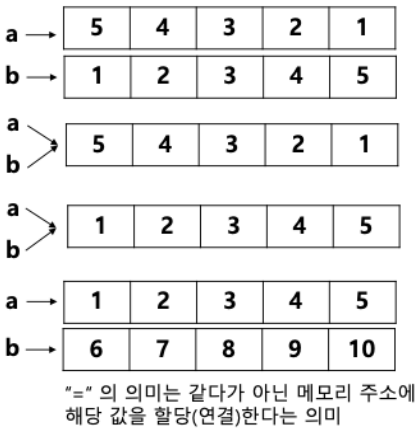
  list의 주소 값을 기반으로 부분 값을 반환
- 리스트연산
- 추가삭제

  append, extend, insert, remove, del 등 활용

```
# 이전 장과 연결 돼서 실행
color.append("white") # 리스트에 “white” 추가
color.extend(["black","purple"]) # 리스트에 새로운 리스트 추가
color.insert(0,"orange") # 0번째 주소에 “orange” 추가
print (color)
# ['orange', 'yellow', 'blue', 'green', 'white', 'black', 'purple']
color.remove("white") # 리스트에 “white” 삭제
del color[0] # 0번째 주소 리스트 객체 삭제
print (color)
# ['yellow', 'blue', 'green', 'black', 'purple']
```

- 메모리 저장 방식

  파이썬은 해당 리스트 변수에는 리스트 주소값이 저장됨



- 패킹과 언패킹
  - 패킹: 한 변수에 여러 개의 데이터를 넣는것
  - 언패킹: 한 변수의 데이터를 각각의 변수로 반환
- 이차원 리스트

  리스트 안에 리스트를 만들어 행렬(Matrix) 생성

Function

- 코드를 논리적인 단위로 분리
- **캡슐화**: 인터페이스만 알면 타인의 코드 사용
- parameter 유무, 반환 값(return value) 유무에 따라 함수의 형태가 다름

	parameter 없음	**parameter 존재**
반환 값 X	함수 내의 수행문만 수행	parameter를 사용, 수행문만 수행
반환 값 O	parameter없이, 수행문 수행 후 결과값 반환	parameter를 사용하여 수행문 수행 후 결과값 반환

print formatting

- 프린트문: 기본적인 출력 외에 출력의 양식, 형식 지정 가능
    1. % string
    2. format 함수
    3. fstring
- padding: 여유 공간을 지정하여 글자배열 + 소수점 자릿수를 맞추기

---

condition

비교연산자	비교상태	설명
x < y	~보다 작음	
x > y	~ 보다 큼	
x == y	같음	x와 y과 같은지 검사
x is y	같음	(값과 메모리 주소)
x != y	같지 않음	x와 y과 다른지 검사
x is not y	같지 않음	(값과 메모리 주소)
x >= y	크거나 같음	
x <= y	작거나 같음	

- 논리 키워드 사용: and, or, not
- 삼항 연산자(Ternary operators)

```
value = 12
is_even = True if value % 2 == 0 else False
print(is_even)
# True
```

loop

- 반복문 변수명

  임시적인 반복 변수는 대부분 i, j, k로 정함

  수학에서 변수를 x, y, z로 정하는 것과 유사한 관례
- 0부터 시작하는 반복문

  반복문은 대부분 0부터 반복을 시작

  이것도 일종의 관례로 1부터 시작하는 언어도 존재

  2진수가 0부터 시작하기 때문에 0부터 시작하는 걸 권장
- 무한 loop

  반복 명령이 끝나지 않는 프로그램 오류

CPU와 메모리 등 컴퓨터의 리소스를 과다하게 점유

debugging

- **문법적 에러**를 찾기 위한 에러 메시지 분석
- **논리적 에러**를 찾기 위한 테스트도 중요

String

- 시퀀스 자료형, 문자형 data를 메모리 저장
- 영문자 한 글자: 1byte의 메모리공간 사용

함수명	기능
len(a)	문자열의 문자 개수를 반환
a.upper()	대문자로 변환
a.lower()	소문자로 변환
a.capitalize()	첫 문자를 대문자로 변환
a.titile()	제목형태로 변환 띄워쓰기 후 첫 글자만 대문자
a.count('abc')	문자열 a에 'abc'가 들어간 횟수 반환
a.find('abc') a.rfind('abc')	문자열 a에 'abc'가 들어간 위치(오프셋) 반환
a.startswith('ab'c)	문자열 a는 'abc'로 시작하는 문자열여부 반환
a.endswith('abc')	문자열 a는 'abc'로 끝나는 문자열여부 반환

함수명	기능
a.strip()	좌우 공백을 없앰
a.rstrip()	오른쪽 공백을 없앰
a.lstrip()	왼쪽 공백을 없앰
a.split()	공백을 기준으로 나눠 리스트로 반환
a.split('abc')	abc를 기준으로 나눠 리스트로 반환
a.isdigit()	문자열이 숫자인지 여부 반환
a.islower()	문자열이 소문자인지 여부 반환
a.isupper()	문자열이 대문자인지 여부 반환

문자	설명	문자	설명
₩ [Enter]	다음 줄과 연속임을 표현	₩n	줄 바꾸기
₩₩	₩ 문자 자체	₩t	TAB 키
₩`	` 문자	₩e	ESC 키
₩"	" 문자		
₩b	백 스페이스		

function

- 함수에서 parameter를 전달하는 방식
  1. 값에 의한 호출(Call by Value)

     함수에 인자를 넘길 때 값만 넘김

     함수 내에 인자 값 변경 시, 호출자에게 영향을 주지 않음
  2. 참조의 의한 호출(Call by Reference)

     함수에 인자를 넘길 때 메모리주소 넘김

     함수 내에 인자 값 변경 시, 호출자의 값도 변경
  3. 객체 참조에 의한 호출(Call by Object Reference)
- 파이썬은 **객체의 주소가 함수로 전달되는 방식**
- 전달된 객체를 참조하여 변경 시 호출자에게 영향을 주나, 새로운 객체를 만들 경우 호출자에게 영향을 주지 않음

recursive function

- 자기자신을 호출하는 함수
- 점화식과 같은 재귀적 수학 모형을 표현할 때 사용
- 재귀 종료 조건 존재, 종료 조건까지 함수호출 반복

fuction type hints

- 사용자에게 인터페이스를 명확히 알려줄 수 있다.
- 함수의 문서화시 parameter에 대한 정보를 명확히 알 수 있다.
- mypy 또는 IDE, linter 등을 통해 코드의 발생 가능한 오류를 사전에 확인
- 시스템 전체적인 안정성을 확보할 수 있다.

docstring

- 파이썬 함수에 대한 상세스펙을 사전에 작성 => 함수 사용자의 이행도 UP
- 세개의 따옴표로 docstring 영역 표시(함수명 아래)

함수 작성 가이드 라인

- 함수는 가능하면 짧게 작성할 것 (줄 수를 줄일것)
- 함수 이름에 함수의 역할, 의도가 명확히 들어낼 것

- 하나의 함수에는 유사한 역할을 하는 코드만 포함
- 인자로 받은 값 자체를 바꾸진 말 것 (임시 변수 선언)

파이썬 코딩 컨벤션의 기준

PEP8

- flaske8
- black