

혼자 공부하며 함께 만드는

혼공 용어 노트

목차

가나다순

2진법 binary system	05	소스 코드 source code	05
개발 환경 development environment	06	순서도 flow chart	10
기계어 machine language	05	숫자 데이터 numeric data	08
네이밍 룰 naming rules	09	슬라이싱 slicing	08
네이밍 컨벤션 naming convention	10	연산자 operator	07
데이터 세트 data set	12	운영 체제 Operating System; OS	06
데이터 타입 data type	09	인덱스 index	08
딕셔너리 dictionary	12	인덱싱 indexing	08
라이브러리 library	06	인코딩 encoding	07
리스트 list	12	인터프리터 interpreter	06
머신 코드 machine code	06	정수 integer	08
무한 반복 infinite loop	15	조건식 conditional expression	11
문자 데이터 character data	07	조기 리턴 early return	16
문자 데이터 연결 concatenate	07	주석 comment	09
매개변수 parameter	16	컴파일러 compiler	06
반복문 loop statement	13	코드 블록 code block	15
변수 variable	09	클라우드 서비스 cloud service	07
부동 소수점 수 floating point number	09	키워드 keyword	10
불 연산 boolean operation	12	파이썬 내장 함수 python built-in function	16
불 bool / boolean	11	프로그래머 programmer	05
비교 연산자 comparison operator	11	프로그래밍 언어 programming language	05
사용자 정의 함수 user-defined function	16	프로그래밍 programming	05

프로그램 program	05
하드 코딩 hard coding	15
함수 function	15

ABC 순

binary system 2진법	05
bool / boolean 불	11
boolean operation 불 연산	12
character data 문자 데이터	07
cloud service 클라우드 서비스	07
code block 코드 블록	15
comment 주석	09
comparison operator 비교 연산자	11
compiler 컴파일러	06
concatenate 문자 데이터 연결	07
conditional expression 조건식	11
data set 데이터 세트	12
data type 데이터 타입	09
development environment 개발 환경	06
dictionary 딕셔너리	12
early return 조기 리턴	16
encoding 인코딩	07
floating point number 부동 소수점 수	09
flow chart 순서도	10
function 함수	15
hard coding 하드 코딩	15
HTML; HyperText Markup Language	06



index 인덱스	08	REPL; Read-Eval-Print Loop	07
indexing 인덱싱	08	slicing 슬라이싱	08
infinite loop 무한 반복	15	source code 소스 코드	05
integer 정수	08	user-defined function 사용자 정의 함수	16
interpreter 인터프리터	06	variable 변수	09
keyword 키워드	10		
library 라이브러리	06		
list 리스트	12		
loop statement 반복문	13		
machine code 머신 코드	06		
machine language 기계어	05		
naming convention 네이밍 컨벤션	10		
naming rules 네이밍 룰	09		
numeric data 숫자 데이터	08		
Operating System: OS 운영 체제	06		
operator 연산자	07		
parameter 매개변수	16		
program 프로그램	05		
programmer 프로그래머	05		
programming 프로그래밍	05		
programming language 프로그래밍 언어	05		
python built-in function 파이썬 내장 함수	16		

01 장

□ 프로그램	program	[01장 24쪽]
	특정 작업을 수행하는 일련의 명령어들의 모음. 소프트웨어라고도 부른다.	
□ 프로그래밍	programming	[01장 25쪽]
	프로그램을 만드는 행위, 코딩(coding)과 같은 의미이다.	
□ 프로그래머	programmer	[01장 25쪽]
	프로그램을 만드는 사람.	
□ 기계어	machine language	[01장 28쪽]
	컴퓨터가 사용하는 언어.	
□ 2진법	binary system 참고 용어 불(boolean)	[01장 28쪽]
	0과 1만을 사용하는 수 체계.	
□ 프로그래밍 언어	programming language	[01장 28쪽]
	프로그램을 만드는 데 사용하는 언어. ex) C 언어, 자바(java), 파이썬(Python) 등	
□ 소스 코드	source code	[01장 29쪽]
	인간이 이해할 수 있는 언어로 작성된 프로그램.	

□ HTML	HyperText Markup Language	[01장 38쪽]
	웹사이트를 표현하는 마크업 언어. 마크업 언어는 데이터를 표현하고 저장하는 방법에 중점을 둔 규칙이다.	
□ 라이브러리	library	[01장 39쪽]
	파이썬이 기본적으로 제공하거나 다른 사람이 만든 프로그램. 라이브러리를 이용하면 필요한 기능을 직접 개발하지 않고 이미 충분히 검증된 프로그램을 쉽고 빠르게 가져다 쓸 수 있다.	
□ 머신 코드	machine code	[01장 42쪽]
	별도의 절차 없이 컴퓨터가 바로 실행할 수 있는 기계어(이진 부호)로 이루어진 프로그램.	
□ 컴파일러	compiler	[01장 43쪽]
	소스 코드를 머신 코드로 번역(컴파일; compile)하는 프로그램.	
□ 인터프리터	interpreter	[01장 43쪽]
	컴파일러와 마찬가지로 소스 코드를 머신 코드로 번역하지만, 소스 코드를 한 줄씩 번역한다.	
	그것이 알고싶다 컴파일러 vs 인터프리터 컴파일러는 소스 코드를 줄 단위가 아닌 덩어리로 번역하고, 인터프리터는 줄 단위로 번역한다.	
□ 운영 체제	Operating System; OS	[01장 44쪽]
	컴퓨터를 켜면 제일 처음 실행되는 핵심 프로그램을 말한다. ex. 윈도우(Windows), 맥오에스(macOS), 리눅스(Linux) 등	
□ 개발 환경	development environment	[01장 45쪽]
	코딩을 위해 준비하는 과정을 ‘개발 환경을 구성한다’라고 표현한다.	

□ 클라우드 서비스

cloud service

[01장 46쪽]

필요한 프로그램(파이썬 인터프리터 등)을 인터넷에 접속해서 실행하는 방식.

□ REPL

Read-Eval-Print Loop

[01장 46쪽]

소스 코드를 읽고(read), 실행(eval)하고, 출력(print)하는 과정을 반복(loop)한다는 의미.

02장

□ 문자 데이터

character data

[02장 59쪽]

세상에 존재하는 것을 기호(한글, 영어 알파벳, 아라비아 숫자 등)로 표기한 것.
컴퓨터가 처리할 수 있는 데이터 중 하나.

□ 인코딩

encoding

[02장 67쪽]

데이터를 다른 형식으로 변환하는 과정. 문자 데이터를 기계어로 번역하는 과정을 문자 인코딩이라고 한다.

□ 연산자

operator

[02장 73쪽]

어떤 기능을 쉽게 할 수 있도록 기호로 표시한 것.

그것이 알고싶다 문자 데이터 연산자의 종류

- + : 문자 데이터 연결 연산자. 문자 데이터를 연결(concatenate)하는 기능을 한다.
- * : 문자 데이터 반복 연결 연산자. 어떤 문자 데이터를 반복해서 연결하는 기능을 한다.

□ 문자 데이터 연결

concatenate

[02장 73쪽]

두 개의 문자 데이터를 합쳐서 한 개의 새로운 문자 데이터를 만든다는 의미.

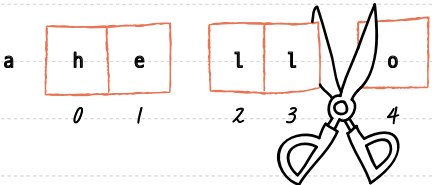
□ 슬라이싱

slicing

[02장 79쪽]

특정 위치에서 문자 간의 연결 고리를 끊어서 새로운 문자를 만드는 것.

`a[2:4]` (=2번 인덱스부터 '4-2'개의 문자를 자르겠다)



이 용어가 땡길 땐, 얇게 잘린 슬라이스(slice) 치즈를 생각하세요.

□ 인덱스

index

[02장 80쪽]

문자 데이터에서 각 문자의 위치를 0부터 시작해서 1씩 더하면서 부여하는데, 이때 각 위치에 부여된 숫자를 인덱스(index)라고 한다.

□ 인덱싱

indexing

[02장 82쪽]

문자 데이터에서 하나의 문자를 고르는 것.

□ 숫자 데이터

numeric data

[02장 90쪽]

사칙 연산과 같이 산술 연산을 할 수 있는 데이터. 일반적으로 코딩할 때 다루게 될 숫자 데이터는 소수점이 없는 것과 있는 것, 이렇게 두 가지 밖에 없다.

그것이 알고싶다 문자 데이터와 숫자 데이터를 구분해서 처리하는 이유

로그래밍 내부에서 문자 데이터와 숫자 데이터를 다르게 처리하기 때문이다. 가령, 0과 1을 더하는 연산을 한다고 생각해 보자. 0이 숫자 데이터라면 덧셈을 문제 없이 할 수 있겠지만, 문자 데이터라면 1과 덧셈을 할 수 없다.

```
print('0')
print(0)      # '0'과 출력값이 같다
'0' + 1       # 에러!
0 + 1         # 문제 없음
```

□ 정수

integer

[02장 93쪽]

소수점이 없는 숫자 데이터.

□ 부동 소수점 수

floating point number

[02장 94쪽]

소수점이 있는 숫자 데이터. '부동 소수점'이라고도 한다.

```
# 파이썬의 숫자 데이터

1      # 정수
0.5    # 부동 소수점 수
1.0    # 부동 소수점 수
```

□ 데이터 타입

data type

[02장 94쪽]

데이터 종류 또는 데이터 형식이라고도 한다. 컴퓨터가 인식하는 데이터의 유형을 말한다.

□ 주석

comment

[02장 97쪽]

소스 코드에는 프로그램의 실행과 관련 없는 문장을 남길 수 있는데, 이것을 주석이라고 한다. 주석은 여러 가지 용도로 사용될 수 있는데, 주로 소스 코드에 대한 설명을 작성하는 데 사용된다. 파이썬에서는 주석 앞에 '#' 기호를 사용하여 작성한다.

□ 변수

variable

[02장 109쪽]

데이터에 붙이는 이름. 데이터를 변수에 '저장한다, 할당한다, 대입한다'라는 표현을 사용하거나, 변수가 데이터를 '가리킨다' 라고 표현한다.

```
a = 1      # a라는 변수에 1을 저장한다.
print(a)   # a라는 변수에 담긴 값(여기에서는 1)을 출력한다.
```

□ 네이밍 룰

naming rules

[02장 112쪽]

변수에 이름을 짓는 규칙. 각 언어마다 네이밍 룰이 조금씩 다를 수 있다.

□ 키워드

keyword

[02장 113쪽]

말 그대로 핵심 단어를 의미한다. 다시 말해서, 파이썬이 '이 이름은 프로그래밍 언어의 문법을 구성하는 요소로 사용되기 때문에 변수 이름으로 사용할 수 없습니다'라고 미리 선점한 이름이다.

□ 네이밍 컨벤션

naming convention

[02장 114쪽]

변수 이름을 짓는 관례.

그것이 알고싶다 네이밍 룰 vs 네이밍 컨벤션

파이썬 언어에서 강제하는 규칙은 네이밍 룰, 지키지 않아도 실행되지만 개발자들 사이에서 관례적으로 지키는 규칙은 네이밍 컨벤션!

03장

□ 조건식

conditional expression

[03장 124쪽]

예/아니오로 대답할 수 있도록 판단의 기준을 질문 형태로 표현한 것, 불식 (boolean expression)이라고도 한다. 비교 연산자를 사용한 표현.

```
weather = "맑음"
print("비가 내리는가?", weather == "비")
if weather == "비": "비가 내리는가?"에 대한 조건식
    print("우산을 가져간다")
else:
    print("우산을 가져가지 않는다")
```

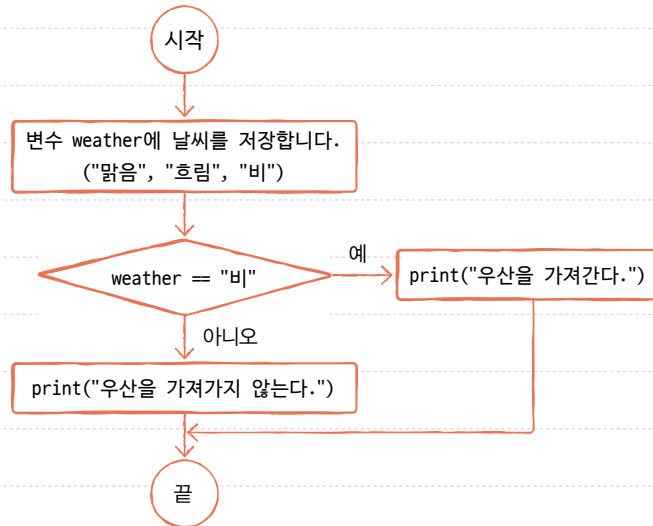
- if~else 구조는 2가지 선택 가능한 대안 중 하나를 선택하는 구조입니다.
- if~elif~else 구조는 3가지 이상의 선택 가능한 대안 중 하나를 선택하는 구조입니다.

□ 순서도

flow chart

[03장 125쪽]

문제 처리 과정을 그림으로 표현한 것.

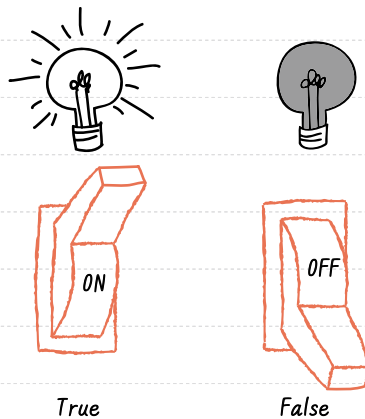


□ 불

bool / boolean 참고 용어 2진법

[03장 133쪽]

예/아니오를 나타내는 데이터. 보통 '예'를 나타내는 데이터는 이진수의 1, '아니오'를 나타내는 데이터는 이진수의 0으로 나타낸다.



□ 비교 연산자

comparison operator

[03장 134쪽]

어떤 두 데이터를 비교(comparison)한 뒤, 그 결과를 True 또는 False로 알려주는 연산자를 의미한다.

두 개의 조건식을 하나로 연결하거나, 조건식의 결과를 반대로 만드는 연산.
논리 연산이라고 부르기도 한다.

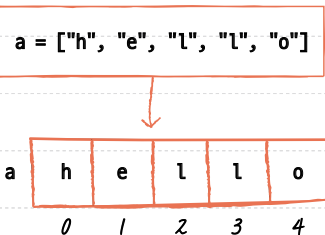
04장

여러 개의 데이터를 하나의 세트로 관리하는 것.

데이터 세트의 종류

- 리스트: 여러 개의 데이터를 대괄호를 사용해서 하나의 데이터 세트로 표현한 것.
- 딕셔너리: 여러 개의 데이터를 하나의 데이터 세트로 표현할 때, 데이터에 이름표를 붙이고, 중괄호를 사용해 표현한 것.
- 레인지: `range()` 함수를 사용해서 얻는 데이터 세트. 주로 정수가 담긴 데이터 세트를 반환한다. 예를 들어, `range(5)`라면 0, 1, 2, 3, 4가 담긴 데이터 세트를 반환한다. 0부터 5개의 정수가 담긴 데이터 세트를 반환한다고 생각하면 쉽다.

데이터 세트의 한 종류로, 여러 데이터를 하나의 세트로 만든 것. 파이썬의 리스트와 같은 방식으로 데이터 세트를 만든 것을 일반적으로 배열(array)이라고 부른다.



데이터 세트의 한 종류로, 여러 개의 데이터에 각각 이름표를 붙여서 하나의 데이터 세트로 만드는 것. 프로그래밍 언어에 따라 오브젝트(object), 해시 맵(hash

map) 등 다양하게 불린다.

- 요소(element): 딕셔너리에서 꺼낸 각각의 데이터
- 키(key): 이름표
- 값(value): 데이터
- 키-값 쌍(key-value pair): 딕셔너리의 데이터

```
book_dict = {  
    "입문자용": "혼자 공부하는 첫 프로그래밍",  
    "초보자용": "혼자 공부하는 파이썬",  
}
```

key(키)

value(값)

"입문자용"

"혼자 공부하는 첫 프로그래밍"

"초보자용"

"혼자 공부하는 파이썬"

TIP: 딕셔너리의 개념이 잘 안 외워진다면, 사전을 생각해 보자. 키(key)는 사전에서 찾을 단어, 값(value)은 사전에 나오는 의미라고 생각하자! book_dict이라는 사전에서 "입문자용"이라는 단어를 찾으면 "혼자 공부하는 첫 프로그래밍"이라는 의미가 나오는 것이다.

05장 ✓

□ 반복문

loop statement

[05장 195쪽]

코드의 반복 처리를 위해 만들어진 문법.

반복문의 종류

- while 반복문: 주어진 조건식을 만족하는 동안 코드를 무한 반복 처리하는 방법. 반복되는 횟수를 알 수 없을 때 의도적으로 무한 루프를 만들고, 특정 조건을 만족하면 멈추는 방법으로 종종 사용된다.

```
count = 1
while count < 4:
    print(str(count) + "!")
    count = count + 1
```

- for 반복문: 필요한 반복 횟수만큼의 데이터를 준비하고, 그것을 모두 소비하는 방식으로 반복 처리하는 방법. 보통 반복되는 횟수를 알 때 사용한다.

문자열

```
string = "혼공족"
for char in string:
    print(char)
```

리스트 데이터 세트

```
book_list = ["혼자 공부하는 첫 프로그래밍", "혼자 공부하는 파이썬"]
for book in book_list:
    print(book)
```

레인지 데이터 세트

```
name_set = ["드링킹 요구르트", "딸기 우유", "홈런공"]
for index in range(3):
    print(name_set[index])
```

딕셔너리 데이터 세트

```
book_dict = {
    "입문자용": "혼자 공부하는 첫 프로그래밍",
    "초보자용": "혼자 공부하는 파이썬",
}
```

```
for book in book_dict:
    print(book)
```

→ for 반복문에 딕셔너리를 사용하면 book 변수에 딕셔너리 요소의 이름표가 저장된다!

그것이 알고싶다 변수명 i

참고로 어떤 프로그래밍 언어에서든지 for 반복문의 변수명으로 i를 사용하는 경우가 많은데, integer의 앞글자인 i를 따온 것이다.

□ 코드 블록

code block

[05장 196쪽]

반복 처리할 코드의 모음. 코드 시작 지점으로부터 4칸 들여쓰기를 한다.

□ 무한 반복

infinite loop

[05장 198쪽]

반복 처리 코드가 종료되지 않고 계속 실행되는 상황을 말한다. 때에 따라 특정한 상황에서 종료되도록 하기 위해서 의도적으로 무한 반복을 만들기도 한다.

□ 하드 코딩

hard coding

[05장 234쪽]

변수에 저장되는 값을 직접 코드에 명시하는 코딩. 일반적으로 하드 코딩이 많아질 수록 데이터가 늘어났을 때 수정해야 할 코드의 양이 많아지므로 유지 보수 비용이 늘어나게 된다. 따라서 하드 코딩을 배제하는 쪽으로 코드를 작성하는 것이 좋다.

```
a = 3 # 하드 코딩
```

```
b = a # a값에 따라 b의 값이 결정되기에 하드 코딩이 아니다.
```

06장 ✓

□ 함수

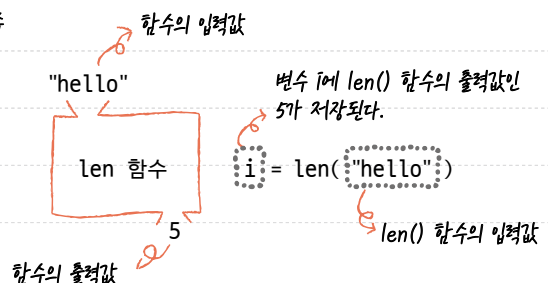
function 참고 용어 네이밍 룰

[06장 256쪽]

어떤 기능을 하도록 작성된 작은 프로그램.

- 함수의 입력값: 함수의 기능을 수행할 때 입력값을 받을 수 있다.
- 함수의 출력값: 함수의 기능을 수행한 뒤, 그 결과값을 전달할 수 있다.

ex. 문자열의 길이를 구하는 함수



그것이 알고싶다 함수 이름 짓는 방법

함수명을 보고 그 함수의 기능을 유추할 수 있어야 한다. 또한 함수도 변수처럼 파이썬의 네이밍 룰을 따른다.

함수의 기본 형태

- 함수 헤더(header): 함수 이름을 지어 주고, 괄호(...)와 콜론(:)을 입력, 이렇게 표현한 명령문을 통틀어 말하는 것.
- 함수 보디(body): 헤더 정의 완료 후 수행할 명령문을 4칸 들여쓰기하여 표현한 명령문을 통틀어 말하는 것.

□ 파이썬 내장 함수

python built-in function

[06장 257쪽]

파이썬이 기본적으로 제공하는 함수.

□ 사용자 정의 함수

user-defined function

[06장 263쪽]

사용자가 필요에 의해 직접 작성한 함수.

② 호출된 함수의 보디를 실행한다.

```
def print_names():  
    print("커피 장인")  
    print("여의도 본점")  
print_names()
```

① 함수를 호출하면 (사용하면)

□ 매개변수

parameter

[06장 271쪽]

헤더의 괄호 안에 함수 입력값을 저장할 변수.

```
def 이름(매개변수1, 매개변수2, ...):  
    return 결과값
```

□ 조기 리턴

early return

[06장 281쪽]

함수 보디 중간에 return 명령어를 사용해서 함수를 종료하는 것을 말한다.