No. date / /

title

혼공
용어 노트

혼자 공부하는 파이썬 개정판

한빛미디어 Hanbit Media, Inc.

혼자 공부하며 함께 만드는

혼공 용어 노트

목차

가나다 순

가비지 컬렉터 garbage collector 24
개발 환경 development environment 07
객체 지향 프로그래밍
OOP: Object Oriented Programming 21
객체 object 23
구문 오류 syntax error 18
논리 연산자 logical operators 13
대화형 셸 interactive shell 07
들여쓰기 indent 13
디코딩 decoding 20
라이브러리 library 20
람다 lambda 17
리스트 내포 list comprehension 15
리터럴 literal 10
리턴 return 15
매개변수 parameter 15
메모화 memoize 16
메소드 method 24
모듈 module 19
문장 statement 07
바이너리 데이터 binary data 21
반복문 loop statement 13
변수 variable 08
부동 소수점 floating point 11
비교 연산자 comparison operators 13
비파괴적 함수 non destructive function 15
상속 inheritance 22
생성자 constructor 23
소멸자 destructor 23
소스 코드 source code 06
스트림 stream 17
스택 stack 18
식별자 identifier 08
엔트리 포인트 entry point 19
연산자 operator 10
예외 exception 18
예외 객체 exception object 19
예외 처리 exception handling 18
오버라이드 override 24
유닉스 타임 unix time 19
이스케이프 문자 escape character 11
이진 숫자 binary digit 06
이터러블 iterable 14
인덱스 index 11

인스턴스 instance 23
인코딩 encoding 20
인터프리터 interpreter 07
자료형 data type 10
재귀 recursion 16
전개 연산자 spread operator 15
제어 역전 IoC: Inversion of Control 20
조건문 conditional statement 12
주석 comment 09
추상화 abstraction 22
캐스트 cast 12
클래스 class 22
키워드 매개변수 keyword parameter 16
키워드 keyword 08
텍스트 데이터 text data 21
텍스트 에디터 text editor 06
통합 개발 환경
IDE: Integrated Development Environment 06
튜플 tuple 17
트리 tree 16
파괴적 함수 destructive function 15
패키지 관리 시스템 pip: Python Package Index 19
팩토리얼 factorial 16
표현식 expression 07
프라이빗 변수 private variable 24
프레임워크 framework 20
프로그래밍 언어 programming language 06
피보나치 수열 fibonacci numbers 16
함수 function 09
힙 heap 18
URL Uniform Resource Locator 20

목차

ABC 순

abstraction 추상화 22
binary data 바이너리 데이터 21
binary digit 이진 숫자 06
cast 자료형 변환 12
class 클래스 22
comment 주석 09
comparison operators 비교 연산자 13
conditional statement 조건문 12
constructor 생성자 23
data type 자료형 10
decoding 디코딩 20
destructive function 파괴적 함수 15
destructor 소멸자 23
development environment 개발 환경 07
encoding 인코딩 20
entry point 엔트리 포인트 19
escape character 이스케이프 문자 11
exception 예외 18
exception handling 예외 처리 18
exception object 예외 객체 19
expression 표현식 07
factorial 팩토리얼 16
fibonacci numbers 피보나치 수열 16
floating point 부동 소수점 11
framework 프레임워크 20
function 함수 09
garbage collector 가비지 컬렉터 24
heap 힙 18
IDE; Integrated Development Environment 통합 개발 환경 06
identifier 식별자 08
indent 들여쓰기 13
index 인덱스 11
inheritance 상속 22
instance 인스턴스 23
interactive shell 대화형 셸 07
interpreter 인터프리터 07
IoC; Inversion of Control 제어 역전 20
iterable 이터러블 14
keyword 키워드 08
keyword parameter 키워드 매개변수 16
lambda 람다 17
library 라이브러리 20
list comprehension 리스트 내포 15

literal 리터럴 10
logical operators 논리 연산자 13
loop statement 반복문 13
memoize 메모화 16
method 메소드 24
module 모듈 19
non destructive function 비파괴적 함수 15
object 객체 23
OOP; Object Oriented Programming
객체 지향 프로그래밍 21
operator 연산자 10
override 오버라이드 24
parameter 매개변수 15
pip; Python Package Index 패키지 관리 시스템 19
private variable 프라이빗 변수 24
programming language 프로그래밍 언어 06
recursion 재귀 16
return 리턴 15
source code 소스 코드 06
spread operator 전개 연산자 15
stack 스택 18
statement 문장 07
stream 스트림 17
syntax error 구문 오류 18
text data 텍스트 데이터 21
text editor 텍스트 에디터 06
tree 트리 16
tuple 튜플 17
unix time 유닉스 타임 19
URL Uniform Resource Locator 20
variable 변수 08

01장 파이썬 시작하기

□ 이진 숫자 — **binary digit** [01장 034쪽]

0과 1로 이루어진 수.

□ 프로그래밍 언어 — **programming language** [01장 034쪽]

컴퓨터가 이해할 수 있는 이진 코드로 변환되는 것을 목표로 만들어진, 사람이 쉽게 이해할 수 있는 형태의 언어.

대표적인 프로그래밍 언어로는 파이썬, C, C#, C++, 자바, 루비, 자바스크립트 등

□ 소스 코드 — **source code** [01장 034쪽]

사람들이 쉽게 읽고 이해할 수 있도록 프로그래밍 언어로 작성한 코드.

사람들은 이 코드로 작성하고 읽는 것이 힘들기 때문에 프로그래밍 언어로 소스 코드를 만들고, 이를 컴퓨터가 이해하는 이진 코드로 바꾼다.

□ 텍스트 에디터 — **text editor** [01장 045쪽]

글자를 입력할 수 있는 모든 종류의 프로그램.

메모장도 텍스트 에디터이며, 프로그래밍 작성 시 사용할 수는 있으나 최대한 프로그래밍 언어를 쉽게 작성할 수 있도록 도와주는 텍스트 에디터를 사용하면 좋다.

텍스트 에디터의 종류에는 비주얼 스튜디오 코드(Visual Studio Code) 외에 서브라임 텍스트(Sublime Text), 아톰(Atom) 등이 있다.

□ 통합 개발 환경 — **IDE; Integrated Development Environment** [01장 053쪽]

텍스트 에디터와 코드 실행기, 이 두 가지를 모두 포함하고 있는 프로그램.

프로젝트 생성, 자동 코드 완성, 디버깅 기능을 제공하는 환경을 말한다.

프로그램 내의 코드 오작동을 찾아내는 것

ex) 자바의 이클립스, C언어의 Visual Studio

□ **개발 환경** — development environment [01장 040쪽]

컴퓨터, 텍스트 에디터, 파이썬 인터프리터 등과 같이 프로그래밍을 할 수 있는 환경.

텍스트 에디터를 포함해서 컴파일러 버전과 같은 개발 플랫폼을 말한다.

웹 프로그래밍에선 웹 브라우저도 개발 환경이 된다.

개발 환경이 달라지면 프로그램의 작동 결과가 다를 수 있다.

□ **인터프리터** — interpreter [01장 043쪽]

프로그래밍 소스 코드를 곧바로 실행해 주는 프로그램.

한 번에 코드 한 줄씩 읽어 실행.

파이썬 코드를 실행할 수 있는 도구는 파이썬 인터프리터.

□ **대화형 셸** — interactive shell [01장 044쪽]

컴퓨터와 상호 작용하는 공간이라는 의미에서 대화형 셸이라고 부른다.

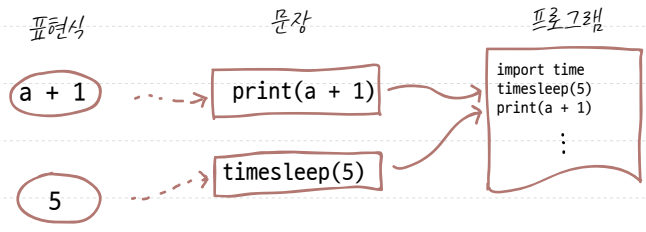
프롬프트라고 불리는 >>>에 코드를 한 줄 한 줄 입력하면 곧바로 실행결과를 볼 수 있다.

□ **표현식** — expression [01장 069쪽]

어떠한 값을 만들어 내는 간단한 코드. 값이란 숫자, 수식, 문자열 등을 의미한다.

□ **문장** — statement [01장 069쪽]

표현식이 하나 이상 모인 것. 파이썬에서는 한 줄이 하나의 문장이 된다.



□ **키워드**

keyword [01장 070쪽]

의미가 부여된 특별한 단어.

언어 내에서 문법적인 용도로 사용되고 있는 단어.

사용자가 지정하는 이름에는 사용 불가.

□ **식별자**

identifier [01장 071쪽]

함수나 변수의 이름을 붙일 때 사용하는 단어.

식별자를 만들 때는 특별한 규칙을 따라야 한다.

- 스네이크 케이스(snake_case): 단어 사이에 _ 기호를 붙여 만든 식별자.

```
phone_number = "010-XXXX-XXXX"
```

- 캐멀 케이스(CamelCase): 단어들의 첫 글자를 대문자로 만든 식별자. 클래스 식별자를 만들 때 사용.

```
class WorldMap:
    pass
```

- 파스칼 케이스: 캐멀 케이스 중에서 첫 번째 글자가 대문자인 것.

```
Class Person:
    pass
```

□ **변수**

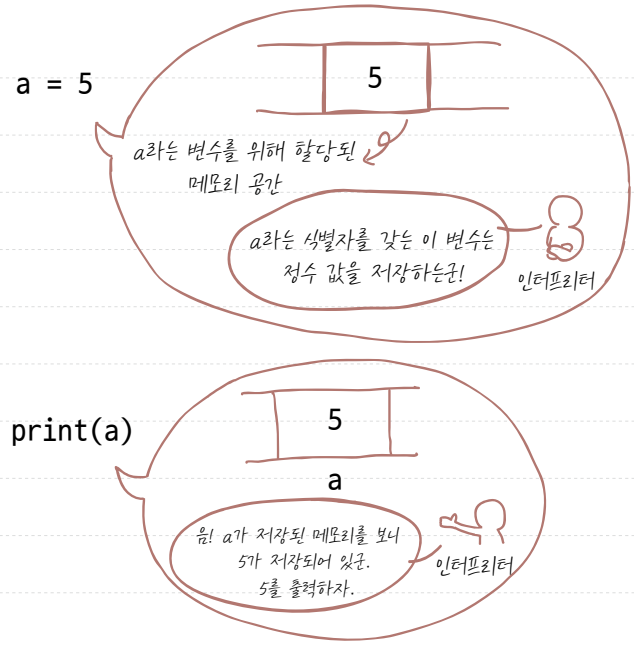
variable [01장 073쪽]

값을 저장할 때 사용하는 식별자.

이름은 '변수'이지만 숫자뿐만 아니라 모든 자료형을 저장할 수 있다.

- 선언: 변수를 사용하려면 식별자는 무엇이고, 어떤 데이터를 가진다라는 것을 알려줘야 하는데, 이는 변수를 '선언한다'라고 한다.
- 할당: 변수에 값을 넣는 것을 '할당한다'라고 한다. 메모리에 변수의 값이 들어갈 공간을 할당해야 하므로
- 참조: 변수에 접근하는 것을 '참조한다'라고 한다.

결국 변수가 저장된 메모리에 접근하는 것인데, 이 메모리의 '주소'를 참조한다고 생각하면 된다!



□ 함수

function 참고 용어 매개변수 [01장 073쪽]

코드의 집합.

식별자 뒤에 괄호가 붙어 있으면 해당 식별자는 함수.

- 함수 호출: 함수를 사용하는 것. 함수를 호출할 때는 괄호 내부에 여러 가지 자료를 넣게 되는데, 이러한 자료를 '매개변수'라 하고, 함수를 호출해서 최종적으로 나오는 결과를 '리턴값'이라고 한다.
- 함수 선언(정의): 함수가 어떤 매개변수를 받아 어떤 동작을 하고 어떤 값을 반환할지 기술하는 것.

```
def func(mylist):
    return sum(mylist)

number = [1, 2, 3]
print(func(number)) #출력: 6
```

□ 주석

comment [01장 075쪽]

프로그램 실행에는 영향이 없으며, 코드에 설명을 붙이기 위해서 사용하는 것.

□ **연산자**

operator [01장 075쪽]

연산에 사용되는 표시나 기호.

- 연산자 우선순위: 파이썬의 연산자 우선순위는 알고 있는 것처럼 곱셈과 나눗셈이 덧셈과 뺄셈보다 우선하고, 우선순위가 같을 때는 왼쪽에서 오른쪽 순서로, 우선순위를 무시하고 무조건 먼저 연산하고 싶은 게 있다면 괄호로 감싸준다.
- 사칙 연산자, // 연산자, 나머지 연산자, 제곱 연산자, 대입 연산자, 복합 대입 연산자

□ **리터럴**

literal [01장 075쪽]

소스 코드 내에서 직접 입력된 값.

자료(data)라고도 한다.

그것이 알고싶다 **상수와 리터럴**

사실 리터럴은 상수와 같은 의미지만, 프로그램에서는 상수를 '값을 한 번 저장하면 변경할 수 없는 변수'로 정의하기 때문에 이와 구분하기 위해 '리터럴'이라는 용어를 사용한다.

02장 자료형

□ **자료형**

data type [02장 085쪽]

자료의 형태.

자료형에 따라 컴퓨터가 처리하는 방법이 달라진다.

파이썬의 기본 자료형

- 숫자: 소수점이 없는 정수형과 소수점이 있는 실수형으로 구분.

  사칙 연산자(+, −, *, /)와 정수 나누기 연산자(//), 나머지 연산자(%), 제곱 연산자(**) 사용
- 문자열: 문자의 나열. 큰따옴표 혹은 작은따옴표로 입력.

문자열 연결 연산자(+), 문자열 반복 연산자(*), 문자열 선택 연산자([ ]), 문자열 범위 선택 연산자([:]) 사용

- 불: True와 False를 나타내는 값. 반드시 첫 글자는 대문자.

□ 이스케이프 문자 **escape character** [02장 089쪽]

\(백슬래시) 기호가 붙은 특수한 문자 리터럴. 문자열 내부에서 특수한 기능을 수행.

□ 인덱스 **index** [02장 095쪽]

리스트, 문자열과 같은 자료형에서 자료가 메모리에 저장된 순서대로 매겨진 번호.

리스트 내부에서 값의 위치를 의미.

- 제로 인덱스: 숫자를 0부터 세는 인덱스. 파이썬은 제로 인덱스 사용.
- 원 인덱스: 숫자를 1부터 세는 인덱스.

```
mylist = [1, 2, 3]
print(mylist[0])      # 출력: 1
```

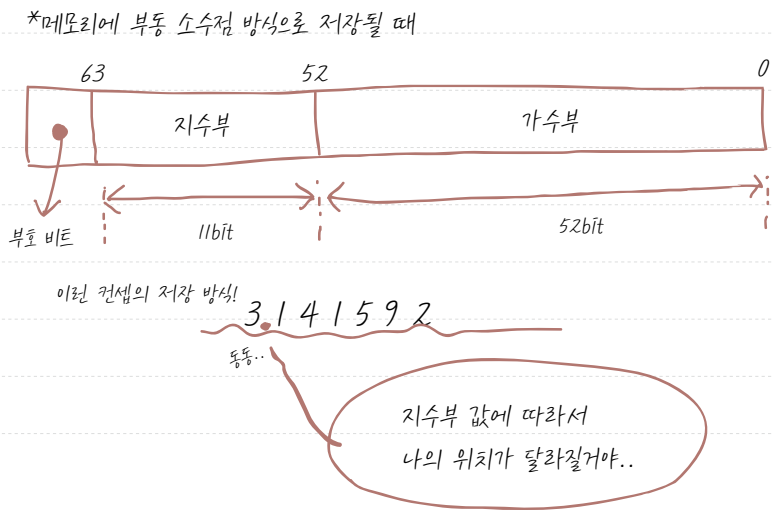
□ 부동 소수점 **floating point** [02장 102쪽]

소수점이 있는 실수 데이터를 저장하는 방식.

'부동'은 '떠다니다'의 의미, 소수점이 떠다닌다는 의미에서 부동 소수점이라고 함.

이 방식에서는 최상위 비트(MSB: Most Significant Bit)를 부호로 결정.

최상위 비트가 0이면 양수, 1이면 음수.



□ **캐스트**

cast [02장 121쪽]

어떤 자료형을 다른 자료형으로 바꾸는 것.

특히 input() 함수의 입력 자료형은 항상 문자열이기 때문에 입력받은 문자열을 숫자로 변환해야 숫자 연산에 활용 가능.

```
# input 값으로 들어오는 문자열 형태의 숫자를 정수형으로 변환
n = int(input())
```

03장 조건문

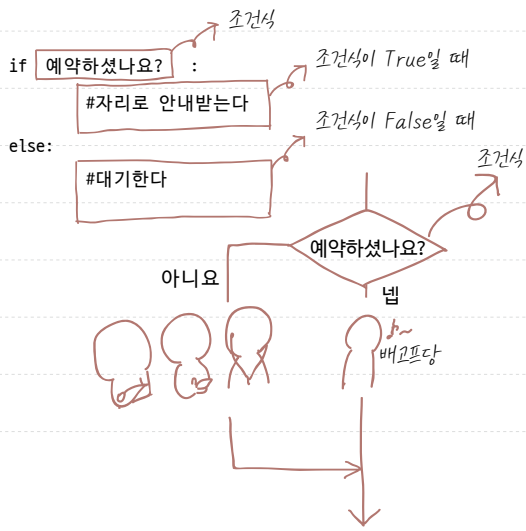
□ **조건문**

conditional statement [03장 163쪽]

조건에 따라 코드를 실행하거나 실행하지 않게 만들고(실행 흐름을 바꾸고) 싶을 때 사용하는 구문.

실행 흐름을 바꾸는 것. 조건 분기

- if문: 괄호 내의 조건식이 참이면 블록 내의 문장을 실행.
- else문: if문의 조건식이 거짓이면 블록 내의 문장을 실행. 필요 없으면 안 써도 됨.
- else if문: if문의 조건식이 거짓일 때 실행시킬 코드에 추가 조건을 걸고 싶은 경우에 사용. 마찬가지로 필요 없으면 else if를 사용하지 않아도 됨.



□ 들여쓰기

indent [03장 164쪽]

코드 앞에 사용하는 '띄어쓰기 4번', '띄어쓰기 2번', '탭'과 같은 것을 의미.

파이썬 개발에서는 일반적으로 '띄어쓰기 4번'을 많이 사용.

그것이 알고싶다 **소프트 탭**

Tab키를 누르면 자동으로 띄어쓰기 네 개를 넣어 주는 기능

□ 비교 연산자

comparison operators [03장 157쪽]

불을 만드는 연산자.

숫자의 크기 비교나 문자열을 비교.

□ 논리 연산자

logical operators [03장 158쪽]

비교 연산자로 만들어진 불끼리 연산할 수 있는 연산자.

- not: 불을 반대로 전환. True는 False로, False는 True로.
- and: 피연산자 두 개가 모두 참일 때 True를 출력하며, 나머지의 경우는 모두 False를 출력.
- or: 피연산자 두 개 중에 하나만 참이라도 True를 출력하며, 두 개가 모두 거짓일 때만 False를 출력.

04장 반복문

□ 반복문

loop statement [04장 207쪽]

조건문과 같이 프로그램의 진행을 바꿀 때 사용하는 것.

매우 많은 횟수 또는 무한하게 반복 작업을 하고 싶을 때 사용하는 문법.

- for문: 반복 횟수가 정해졌거나 변수가 이터러블한 경우에 주로 사용.

  이 경우 range( ) 함수 사용

- while문: 반복 횟수를 모르거나 무한 루프를 만들 때 주로 사용.

• break문: 반복문, 조건문 블럭을 빠져 나오게 하는 키워드.

• continue문: 반복문 내에서 continue문을 만나면 아래에 있는 코드를 실행하지 않고 위로 돌아가 반복문 조건을 검사한 후 반복을 할지 말지를 결정. 주로 조건문 안에 넣어 사용.

```
# 리스트와 for 반복문
for n in mylist:                    # 208쪽
    print(n)                        # 리스트의 요소 출력

# 딕셔너리와 for 반복문
for key in mydict:                  # 226쪽  # key는 딕셔너리의 키!
    print(key)
for value in mydict.values():       # value는 딕셔너리의 값!
    print(value)
for key, value in mydict.items():
    print('key: {}'.format(key))
    print('value: {}'.format(value))

# range()를 사용한 반복문
for i in range(5):                  # 233쪽  # 5회 반복
    print(i)

# 리스트, 범위, for 반복문 조합하기
for i in range(len(array)):         # 234쪽
print("{}번째 반복: { }".format(i, array[i]))

# while 반복문
i = 0                               # 240쪽
while i < 10:
print("{}번째 반복입니다.".format(i))
i += 1
```

반복자

그것이 알고싶다 **리스트와 딕셔너리의 차이점**

리스트는 각 요소를 인덱스로 접근하는 반면, 딕셔너리는 인덱스 대신 키를 사용해 요소에 접근한다.

□ **이터러블**

iterable [04장 264쪽]

반복을 적용할 수 있는 성질.

내부에 있는 요소들을 차례차례 꺼낼 수 있는 객체.

- 문자열
- 리스트
- 딕셔너리
- 범위(range() 함수)

□ **비파괴적 함수** — non destructive function [04장 201쪽]

원본을 변화시키지 않는 함수 → 종류: str 자료형의 lower(), upper(), split() 등

□ **파괴적 함수** — destructive function [04장 201쪽]

원본을 변화시키는 함수 → 종류: list 자료형의 append(), remove(), pop() 등

□ **리스트 내포** — list comprehension [04장 257쪽]

반복문의 요소를 삽입할 때, 한 줄로 작성할 수 있도록 제공하는 문법.

```
# 리스트를 선언합니다.
array = [i * i for i in range(0, 20, 2)]
# 출력합니다.
print(array)
```

□ **전개 연산자** — spread operator [04장 211쪽]

리스트 요소를 나열해 주는 연산자. 리스트 앞에 * 기호를 사용한다.

05장 함수

□ **리턴** — return [05장 285쪽]

함수를 실행했던 위치로 돌아가게 하는 것.

리턴값을 가지는 함수는 반드시 리턴할 때 반환하는 값이 있어야 한다.

함수의 실행 결괏값.

- 조기 리턴: 함수 내의 필요한 위치에서 return 키워드를 사용하는 것.

□ **매개변수** — parameter [05장 275쪽]

함수를 호출할 때 필요한 데이터를 외부로부터 받기 위해 사용하는 것.

- 가변 매개변수: 매개변수를 원하는 만큼 받을 수 있는 함수.
- 기본 매개변수: 매개변수를 입력하지 않았을 경우 들어가는 기본값.

그것이 알고싶다 **전달인자(argument)와 매개변수(parameter)**

```
int mySum(arr):
    return sum(arr)

...
myVar = mySum(myArray)
...
```

매개변수

전달인자

□ **키워드 매개변수** **keyword parameter** [05장 280쪽]

매개변수의 이름을 지정해서 값을 입력하는 매개변수.

```
print('you are not alone!!', end='')
```

키워드 매개변수

□ **재귀** **recursion** [05장 293쪽]

함수 내부에서 자기 자신을 호출하는 것.

□ **메모화** **memoize** [05장 300쪽]

재귀 호출에서 한 번 연산한 값을 중복해서 연산하지 않기 위해 새로운 값을 연산할 때마다 그 결과에 따른 값을 저장하는 것.

저장된 값은 중복 호출될 때마다 가져와서 사용하게 된다!

□ **팩토리얼** **factorial** [05장 293쪽]

예: 5! = 5 X 4 X 3 X 2 X 1 = 120

n!으로 표현.

1부터 n까지의 숫자를 곱하는 연산.

□ **피보나치 수열** **fibonacci numbers** [05장 295쪽]

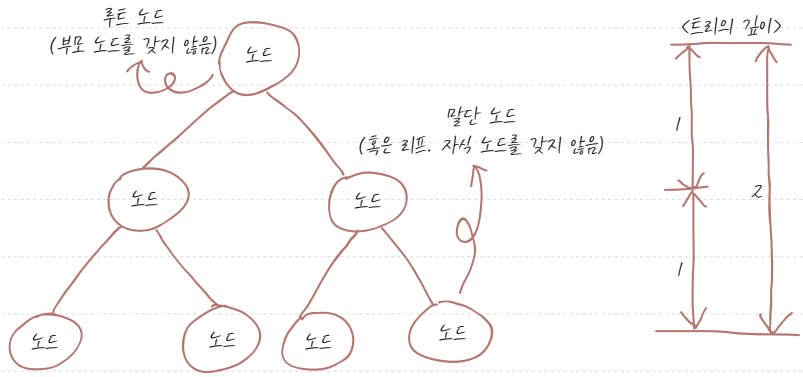
첫 번째와 두 번째 항이 주어지며, 세 번째 항부터 앞의 두 항을 더한 값을 갖는 수열.

□ **트리** **tree** [05장 298쪽]

비선형 자료 구조의 일종.

노드끼리 연결된 모습이 나무 모양의 구조인 데에서 이름이 유래.

- 노드: 트리의 각 항목
- 리프: 자식 노드가 없는 노드
- 부모 노드: 자신의 바로 위에 연결된 노드
- 자식 노드: 자신의 바로 아래에 연결된 노드
- 루트: 해당 트리의 최상위 노드



□ **스트림** **stream** *stream의 의미가 '시냇물'인 것을 떠올리면 기억하기 쉽다!* [05장 329쪽]

프로그램이 외부 파일, 외부 네트워크 등과 통신할 때 데이터가 흐르는 길.

이때 외부는 같은 컴퓨터 내부라도 프로그램의 바깥에 별도로 존재하는 것을 말한다.

□ **람다** **lambda** [05장 322쪽]

간단한 함수를 쉽게 구현할 수 있는 기능.

□ **튜플** **tuple** [05장 317쪽]

리스트와 비슷한 자료형이나 리스트와 다르게 한번 결정된 요소를 바꿀 수 없다.

```
#괄호가 없는 튜플
tuple_test = 10, 20, 30, 40    # tuple_test 변수는 튜플 자료형!

#튜플을 사용한 할당
(a, b) = (10, 20)
a, b = 10, 20
```

□ **스택**

stack [05장 342쪽]

기본 자료형이 차곡차곡 정리된 공간.

숫자, 문자열, 불

□ **힙**

heap [06장 343쪽]

객체 자료형이 저장되어 있는 공간.

숫자, 문자열, 불을 제외한 자료형

06장 예외 처리

□ **구문 오류**

syntax error [06장 361쪽]

프로그램 실행 전에 발생하는 오류.

구문 오류가 있으면 프로그램 자체가 실행이 되지 않는다.

□ **예외**

exception [06장 362쪽]

런타임 오류(runtime error).

프로그램 실행 중에 발생하는 오류를 의미.

try-except 구문으로 처리 가능.

```
try:
    number_input_a = int(input("정수 입력> "))
except:
    # 정수로 변환 불가능한 자료가 입력될 경우 실행됩니다.
    print("무언가 잘못되었습니다.")
```

□ **예외 처리**

exception handling [06장 363쪽]

예외가 발생하지 않게 미리 예외가 발생할 수 있는 부분에 안전 장치를 설치하는 것.

예외를 처리하는 방법

• 조건문 사용: 조건문 등을 사용해 예외를 처리하는 기본적인 방법.

• try 구문 사용: 예외 처리에 특화된 구문.

□ **예외 객체** **exception object** [06장 383쪽]

예외가 발생했을 때 예외에 관련된 정보가 저장되는 객체.

07장 모듈

□ **유닉스 타임** **unix time** [07장 414쪽]

세계 표준시(UTC)로 1970년 1월 1일 0시 0분 0초를 기준으로 몇 초가 지났는지를 정수로 나타낸 것.

□ **모듈** **module** [07장 400쪽]

코드를 분리하고 공유할 수 있도록 만들어 주는 문법.

• 표준 모듈: 파이썬에 기본적으로 내장되어 있는 모듈.

• 외부 모듈: 다른 사람이 만들어서 공개한 모듈.

□ **패키지 관리 시스템** **Package Management System** [07장 423쪽]

파이썬의 외부 모듈을 설치할 때 사용하는 프로그램.

pip(Python Package Index)를 일컬음.

□ **엔트리 포인트** **entry point** [07장 442쪽]

메인(main)이라고도 하며, 프로그래밍 언어에서 프로그램의 진입점을 이르는 말.

□ **URL**

Uniform Resource Locator [07장 415쪽]

네트워크의 자원이 어디에 위치하는지 확인할 때 사용하는 것.

□ **인코딩**

encoding [07장 452쪽]

특정한 방식을 기반으로 어떤 형식을 다른 형식으로 대응시키는 것.

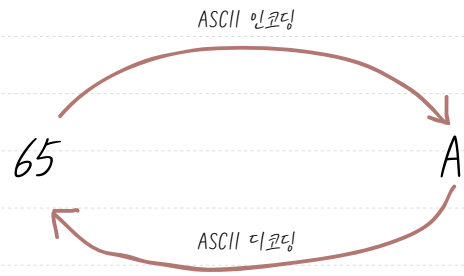
그것이 알고싶다 **ASCII CODE**

파이썬은 기본 인코딩 방식으로 아스키를 사용한다. 아스키(ASCII: American Standard Code for Information Interchange, 미국 정보 교환 표준 부호)는 7비트로 표현되는 영문자 기반 인코딩이며, 33개의 제어 문자들과 95개의 출력 가능한 문자들로 이루어져 있다.

□ **디코딩**

decoding [07장 452쪽]

인코딩된 데이터를 반대로 돌리는 것.



□ **라이브러리**

library [07장 433쪽]

개발자가 모듈의 함수를 실행하여 사용하는 것.

정상적인 제어를 하는 모듈.

□ **프레임워크**

framework [07장 433쪽]

모듈이 개발자의 함수를 실행하여 사용하는 것. 제어 역전이 발생하는 모듈

□ **제어 역전**

IoC; Inversion of Control [07장 433쪽]

개발자가 모듈의 함수를 호출하는 것이 일반적인 제어 흐름인데, 이와 반대로 개발자가 만든 함수를 모듈이 실행하는 것.

그것이 알고싶다 **라이브러리와 프레임워크를 구분하는 법**

제어 역전이 발생하는 모듈이 프레임워크, 정상적인 제어를 하는 모듈이 라이브러리다. 즉, 흐름(flow)을 주도하는 쪽이 누구인가에 따라 라이브러리, 프레임워크로 구분한다.

□ **텍스트 데이터** **text data** [07장 449쪽]

내부적으로는 앞에서 언급한 이진수로 저장되지만, 우리가 쉽게 읽을 수 있는 텍스트로 변환될 수 있는 이진수 데이터.

□ **바이너리 데이터** **binary data** [07장 449쪽]

텍스트 에디터로 열었을 때 의미를 이해할 수 없으며, 전용 에디터를 사용해서만 편집할 수 있는 데이터. → 이미지, 동영상

08장 클래스

□ **객체 지향 프로그래밍** **OOP; Object Oriented Programming** [08장 460쪽]

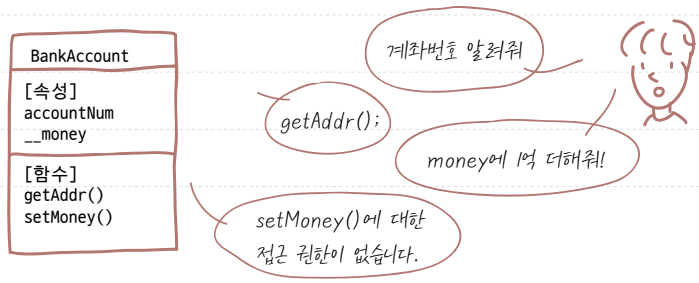
프로그램을 개발하는 기법으로 부품에 해당하는 객체들을 먼저 만들고, 이것들을 하나씩 조립 및 연결해서 전체 프로그램을 완성하는 기법.

객체 지향 프로그래밍의 특징

- 상속
- 다형성
- 캡슐화

그것이 알고싶다 **캡슐화(encapsulation)**

캡슐화는 객체 지향 프로그래밍의 특징 중 하나로, 객체의 필드, 메소드 클래스 외부에서 실행할 수 없게 감추는 것을 말한다. 외부의 잘못된 사용으로 인해 객체가 손상되지 않도록 필드와 메소드를 캡슐화하여 보호하는 것이다.



□ **상속**

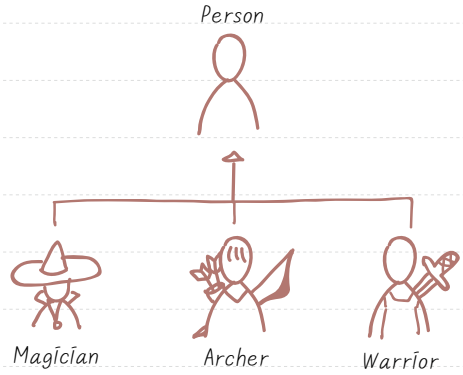
inheritance [08장 474쪽]

상위(부모) 객체를 기반으로 하위(자식) 객체를 생성하는 관계를 말한다.

상위 객체는 자기가 가지고 있는 변수와 함수를 하위 객체에게 물려주어 하위 객체가 사용할 수 있도록 한다.

- 부모(parent class, super class, 상위 클래스): 해당 클래스에 선언된 변수와 함수를 물려주는 클래스.
- 자식(child class, sub class, 하위 클래스): 다른 클래스로부터 물려받고자 하는 클래스

게임 캐릭터를 만들 때, 캐릭터 이름과 같은 속성은 모든 캐릭터가 같지만 각각의 직업은 다르기 때문에 다른 스킬을 사용한다. 이 때 공통된 속성을 갖는 클래스를 상속받아 스킬만 다르게 구현할 수 있다.



□ **클래스**

class [08장 460쪽]

객체에 포함할 변수와 함수를 미리 정의한 것.

객체의 설계도에 해당.

□ **추상화**

abstraction [08장 461쪽]

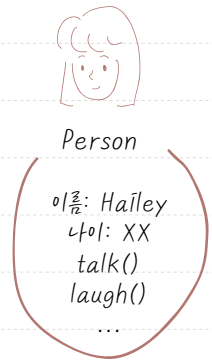
프로그램에서 필요한 요소만을 사용해서 객체를 표현하는 것.

복잡한 자료, 모듈, 시스템 등으로부터 핵심적인 개념 또는 기능을 간추려 내는 것을 의미.

□ **객체**

object [08장 461쪽]

여러 가지 속성을 가질 수 있는 대상.



□ **인스턴스**

instance [08장 466쪽]

클래스 기반으로 만들어진 객체.

> 그것이 알고싶다 **클래스, 객체, 인스턴스**
>
> 속성을 가진 대상인 객체는 클래스를 이용하여 가공하기 쉬운 인스턴스로 구현할 수 있다. 클래스는 객체가 가지는 기능을 메소드와 속성으로 정의하고, 필요할 때마다 생성자 호출로 같은 기능과 속성을 가지는 인스턴스를 만들 수 있도록 한다. 때문에 흔히 객체와 인스턴스라는 용어는 혼용되어 사용된다. 인스턴스는 객체를 프로그래밍으로 구현한 개념이기 때문이다.

□ **생성자**

클래스 이름으로 호출한다

constructor [08장 467쪽]

클래스 이름과 같은 함수.

클래스 내부에 __init__이라는 함수를 만들면 객체를 생성할 때 처리를 작성할 수 있다.

```
class Student:
   def __init__(self, name):
      self.name = name

Student("윤인성")       # name 속성에 "윤인성" 문자열을 저장
```

□ **소멸자**

destructor [08장 468쪽]

인스턴스가 소멸될 때 호출되는 함수.

□ **메소드**

method [08장 469쪽]

클래스가 가지고 있는 함수

= 멤버 함수, 인스턴스 함수

□ **프라이빗 변수**

private variable [08장 491쪽]

캡슐화를 위해 만들어진 문법.

파이썬은 클래스 내부의 변수가 외부에서 사용하는 것을 막고 싶을 때, 인스턴스 변수 이름을 __〈변수 이름〉 형태로 선언한다.

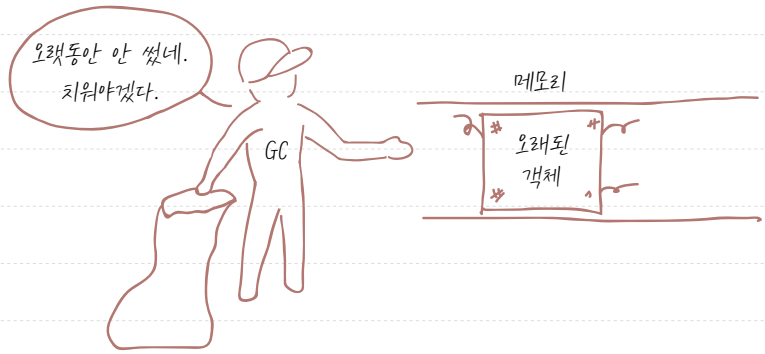
그것이 알고싶다 **게터(getter)와 세터(setter)**

외부에서 마음대로 읽고 변경할 수 없도록 제어하는 메소드. 만약 외부에서 필드값을 읽을 수만 있고 변경하지 못하도록 하려면(읽기 전용) getter 메소드만 선언하면 된다.

□ **가비지 컬렉터**

garbage collector [08장 489쪽]

더 이상 사용할 가능성이 없는 데이터를 메모리에서 제거하는 프로그램.



□ **오버라이드**

override [08장 499쪽]

부모에게서 상속받은 메소드를 자식 클래스에서 다시 정의하는 것.

자르는 선

혼자
공부하는
사람들을 위한
용어 노트