|  |
| --- |
|  |
| **4시간만에 끝내는 파이썬 기초** |
| **“김왼손의 미운코딩새끼” 강좌를 듣고 나서…** |

목차

[1 개요 4](#_Toc516121650)

[1.1 프로젝트 주제 4](#_Toc516121651)

[1.2 프로젝트 추진 배경 및 목표 4](#_Toc516121652)

[2 왜 파이썬인가? 5](#_Toc516121653)

[3 간단한 입력과 출력 6](#_Toc516121654)

[3.1 입력과 출력 6](#_Toc516121655)

[4 프로그래밍의 기초, 자료형 7](#_Toc516121656)

[4.1 변수와 변수명 7](#_Toc516121657)

[4.2 자료형 8](#_Toc516121658)

[4.3 숫자형(Number) 8](#_Toc516121659)

[4.4 문자열 자료형 10](#_Toc516121660)

[4.5 불 자료형 10](#_Toc516121661)

[4.6 리스트 자료형 10](#_Toc516121662)

[4.7 튜플 자료형 10](#_Toc516121663)

[4.8 딕셔너리 자료형 10](#_Toc516121664)

[4.9 자료형 변환 10](#_Toc516121665)

[5 제어문 10](#_Toc516121666)

[5.1 연산자 10](#_Toc516121667)

[5.2 if문 10](#_Toc516121668)

[5.3 for문 10](#_Toc516121669)

[5.4 while문 10](#_Toc516121670)

[5.5 continue와 break문 10](#_Toc516121671)

[6 함수 10](#_Toc516121672)

[7 클래스 11](#_Toc516121673)

[8 모듈 11](#_Toc516121674)

[9 패키지 11](#_Toc516121675)

[10 참고자료 12](#_Toc516121676)

# 개요

## 프로젝트 주제

|  |
| --- |
| 개인 역량 강화 프로젝트 |

표 1- 프로젝트 주제

## 프로젝트 추진 배경 및 목표

|  |
| --- |
| 시큐어가드테크놀러지 입사 전 기초적인 python 문법을 익혀 역량을 강화해보자. |

표 1- 프로젝트 추진 배경 및 목표

# 왜 파이썬인가?

|  |
| --- |
| 문법을 쉽게 배우고 사용할 수 있고 가독성이 뛰어남. |
| 파이썬은 다른 언어로 만든 프로그램을 파이썬 프로그램에 포함시킬 수 있음. |
| 개발 속도가 빠름. |
| 교차 플랫폼 언어이기에 윈도우, 리눅스 등 다른 플랫폼에서도 동일하게 구동 가능. |
| 무료, 오픈소스이기에 자유롭게 파이썬을 이용 가능 |
| 파이썬으로 GUI(Graphical User Interface)를 개발 가능. |

표 2- 파이썬의 장점

|  |
| --- |
| 인터프리터 언어이기에 JAVA나 C++같은 컴파일 언어보다 느림. |
| 동시다발적 멀티스레드를 처리하거나 CPU에 집중된 많은 스레드를 처리하는 데 부적합. |
| 2.X 버전과3.X 버전이 호환되지 않음. |

표 2- 파이썬의 단점

# 간단한 입력과 출력

## 입력과 출력

**print() :** 입력한 자료를 출력한다.



그림 3- print 사용의 간단한 예

print(‘python’ ‘is’ ‘funny’)와 print(‘python’+‘is’+‘funny’)의 결과값은 같게 나온다.



그림 3- print에서 +와 따옴표 연산의 동일함을 보여주는 예

콤마(,)는 문자열의 띄어쓰기 역할을 한다. print(‘python’, ‘is’, ‘funny’)의 결과는 이렇다.



그림 3- print에서 콤마(,)의 역할

한 줄에 결과값을 출력할 수도 있다. for문을 이용해 예제를 만들어보았다.



그림 3- print에서 인수 end를 이용해 한 줄로 연결

(위 3-4 코드는 파이썬 2.7 버전의 경우 print(i, end=’ ‘)가 아닌 print i, 를사용한다.)

**input() :** 사용자에게 입력을 받는다.



그림 3- input() 사용의 간단한 예

사용자에게 입력을 받을 때 안내문구 또는 질문을 출력하고 싶다면 input() 괄호 안에 내용을 입력해주면 내용을 출력한 후 입력을 받는다.



그림 3- 내용 출력 후 input()으로 5 입력

# 프로그래밍의 기초, 자료형

## 변수와 변수명

**변수 :** 객체를 가리키는 것이라고도 말할 수 있다. 객체란 자료형과 같은 것을 의미한다. 변수의 상징적인 이름을 데이터의 실제 위치로 치환해야 한다. 변수 값, 형, 위치는 일반적으로 고정된 채 유지되는 반면 위치에 저장되어 있는 데이터는 프로그램 실행 도중 변경될 수 있다

|  |
| --- |
| 영문자(대문자, 소문자), 숫자, 언더바(\_)를 사용할 수 있다. |
| 첫 자리에는 숫자를 사용할 수 없다. |
| 파이썬 키워드는 변수명으로 사용할 수 없다. |

표 4- 변수명의 규칙

파이썬에서 변수를 선언할 때 C, Java 등과 다르게 변수의 자료형과 함께 쓸 필요가 없다. 파이썬은 변수에 저장된 값을 스스로 판단하여 자료형을 알아낸다.

변수가 가리키는 메모리 주소를 id()함수로 알 수 있다. a 변수의 메모리 주소를 알아보자.



그림 4- 변수 a의 메모리 주소

## 자료형

**자료형 :** 프로그래밍을 할 때 쓰이는 숫자, 문자열 등 자료 형태로 사용하는 모든 것을 뜻한다. 프로그램의 기본이자 핵심 단위이다.

|  |  |
| --- | --- |
| 자료형 | 설명 |
| 정수형(Integer, Long) | 양의 정수, 음의 정수, 0을 표현. Long은 데이터가 긴 경우. |
| 실수형(Floating-point) | 소수점이 포함된 숫자. |
| 복소수(Complex number) | 복소수, 실수와 허수부의 합으로 이루어진 수를 표현. |
| 문자열(String) | 문자, 단어 등으로 구성된 문자들의 집합을 의미. |
| 리스트(List) | 순서가 있는 값들의 나열. |
| 튜플(Tuple) | 리스트와 달리 요소의 값을 수정할 수 없음. |
| 딕셔너리(Dictionary) | key와 Value라는 것을 한 쌍으로 갖는 자료형. |
| 집합(Set) | 집합에 관련된 것들을 쉽게 처리하기 위해 만들어진 자료형 |
| 불(Boolean) | 참(True)과 거짓(False)을 나타내는 자료형. |

표 4- 자료형의 종류와 설명

## 숫자형(Number)

숫자형에는 정수형, 실수형이 존재한다.

|  |  |
| --- | --- |
| 항목 | 사용 예시 |
| 정수 | 123, -456, 0 |
| 실수 | 1.23, -45.67, 3.4e10 |

표 4- 자료형의 종류와 설명

만약 정수/정수의 값이 실수의 형태를 가지면 어떻게 출력될까?



그림 4- 3/4의 결과값

파이썬 3.X버전에서는 값이 0.75로 나오지만, 2.X 버전에서는 정수형으로 나눌 경우 정수로만 결과값을 리턴하기 때문에 0이 출력된다.

위에서 /는 실수를 리턴했지만 // 연산자를 사용하면 소수점 아랫자리를 버리고 정수부분만 리턴한다.

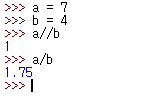


그림 4- // 연산과 /연산의 차이

여기서 주의할 점은 음수를 연산할 시 값이 예상과 다를 수 있다는 점이다. a를 -7로 바꾸어 연산해보자.

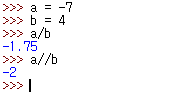


그림 4- 음수일 때 // 연산

연산자는 나눗셈의 결과에서 무조건 소수점을 버리는것이 아니라 나눗셈의 결과값보다 작은 정수 중, 가장 큰 정수를 리턴하기 때문이다. -1.75 > -2 이므로 결과값보다 작은 수 중에서 가장 큰 정수이다.

표 4-3에서 실수형의 예시 중 3.4e10는 무엇일까?

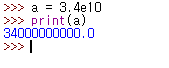


그림 4- e(지수 표현 방식)의 사용 예

3.4e10은 3.4 \* 10^10을 의미한다. 즉, e뒤에 수는 10의 지수를 의미한다.

제곱 연산자도 존재한다. a\*\*b는 a^b를 뜻한다.

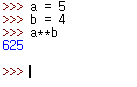


그림 4- \*\*연산자의 사용

a\*\*b의 결과값은 a의 네 제곱인 625가 리턴된다.

파이썬은 복소수를 지원한다. 사용법은 j를 쓰면 된다.

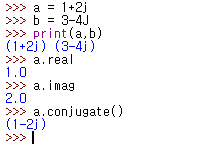


그림 4- a를 여러 방법으로 출력

print(a, b)를 실행하면 a와 b의 켤레복소수를 출력한다. a.conjugate()도 역시 켤레복소수를 리턴한다. a.real은 복소수의 실수 부분을 리턴한다. a.imag은 복소수의 허수부분을 리턴한다.

## 문자열 자료형

문자열을 생성할 때는 4가지 방법을 사용할 수 있다.

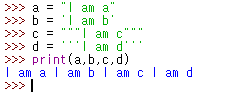


그림 4- 문자열을 생성하는 예

큰따옴표로 감싼 문자열 안에 있는 작은따옴표는 문자열을 나타내기 위한 기호로 인식되지 않는다. 또한 작은따옴표로 감싼 문자열 안에 있는 큰 따옴표도 문자열을 나타내기 위한 기호로 인식되지 않는다.

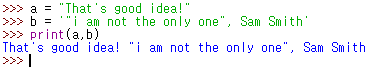


그림 4- 쌍따옴표(“ ”)와 따옴표(‘ ‘)의 사용 예

백슬래시(\)를 이용해 ‘와 “를 문자열에 포함시킬 수 있다.

|  |  |
| --- | --- |
| 코드 | 설명 |
| \n | 개행(줄바꿈) |
| \t | 수평 탭 |
| \\ | 문자 “\” |
| \’ | 작은따옴표 그대로를 출력 |
| \” | 큰따옴표 그대로를 출력 |
| \r | 캐리지 리턴 |
| \f | 폼 피드 |
| \a | 벨 소리 |
| \b | 백 스페이스 |
| \000 | 널문자 |

표 4- 이스케이프 코드의 종류

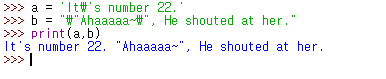


그림 4- \의 사용 예

### 문자열 연산하기

파이썬에서는 문자열을 + 기호로 더하여 연결할 수 있다.

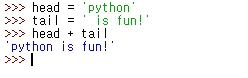


그림 4- +로 문자열 연결하기

또한 \*로 문자열을 곱하여 출력할 수도 있다.

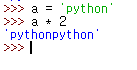


그림 4- \*로 문자열 반복하기

\* 연산을 사용하면 문자열을 그 수만큼 반복한다는 의미이다. 이 기능만 봐서는 사용할 상황이 떠오르지 않을 것이다. 아래와 같이 콘솔을 간단하게 디자인 할 수 있다.

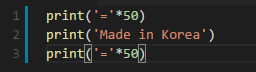


그림 4- 문자열에서 \*연산의 사용 예

이 코드의 실행 결과는 이렇다.

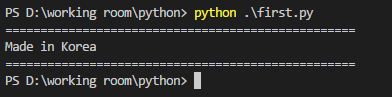


그림 4- [그림 4- 13]의 결과

=를 50개만 출력했을 뿐인데 훨씬 보기 좋아진 기분이다.

### 문자열 인덱싱과 슬라이싱

**인덱싱(Indexing) :** 어떤 것을 가리킨다는 의미이다.

**슬라이싱(Slicing) :** 어떤 것을 잘라낸다는 의미이다.

문자열 인덱싱의 예를 살펴보자.

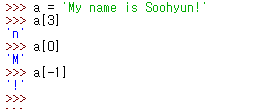


그림 4- 인덱싱의 표현 방식

a[3]은 문자열 a의 맨 앞(a[0])에서부터 3칸 떨어진 n을 리턴한다. a[0]은 맨 앞의 M을 리턴한다. 만약 위치가 a[-1]처럼 음수일 경우에는 뒤에서부터 차례대로 리턴한다는 의미이다. a[-1]은 맨 뒤에서 첫번째인 !가 리턴된다.

위에서는 문자열의 요소 중 하나만을 뽑아내었다. 그렇다면 ‘name’이나 ‘Soohyun’과 같이 단어를 뽑아내는 방법이 바로 슬라이싱이다.

‘name’을 인덱싱과 슬라이싱으로 추출해보자.

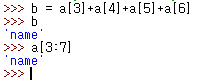


그림 4- 인덱싱과 슬라이싱의 예

인덱싱은 a[3]부터 a[6]까지 더하였다. 그렇다면 슬라이싱은 왜 a[3:7]로 선언했을까? 슬라이싱은 a[x:y]라고 한다면 x<=a<y인 수식으로 슬라이싱을 한다. 3<=a<7을 만족하는 a[]는 a[3] ~ a[6]이다.

슬라이싱 기법을 사용하기에 좋은 예시를 들어보자.

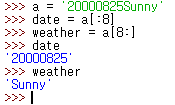


그림 4- 슬라이싱의 좋은 사용 예

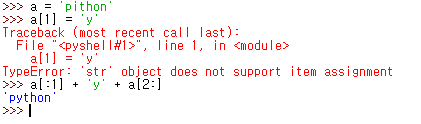


그림 4- 문자열 수정 후에 출력하기

문자열은 요소의 값을 바꿀 수 없다. 그렇기에 a[0]과 a[2] 사이에 ‘y’라는 문자를 추가한 후 이어 붙여서 만든다.

### 문자열 포매팅

**문자열 포매팅 :** 문자열 내에 어떠한 값을 삽입하는 것을 말한다.

|  |  |
| --- | --- |
| 코드 | 설명 |
| %s | 문자열(String) |
| %c | 문자(Character) |
| %d | 정수(Integer) |
| %f | 부동소수(floating-point) |
| %o | 8진수 |
| %x | 16진수 |
| %% | 리터럴 % (문자 % 그대로) |

표 4- 문자열 포맷 코드

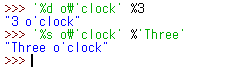


그림 4- 문자열 포매팅의 예

정수나 실수, 문자열, 변수 등을 포함시킬 수 있다. 아래처럼 한번에 여러 개의 값을 삽입할 수 있다. 또한 연산도 할 수 있다.



그림 4- 한 문자열에 여러 개의 포매팅

format 함수를 이용해 좀 더 발전된 스타일로 문자열 포맷을 지정할 수 있다.

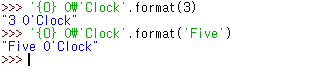


그림 4- format()함수의 사용 예

{0} 다음은 {1}, {2}… 순서로 format()함수를 이용하여 입력도 가능하고 이름을 지어서 대입하는 방법으로도 입력할 수 있다.

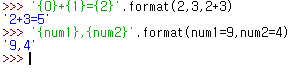


그림 4- format()함수의 다른 입력 방법

파이썬 3.6 버전 이상에서는 f 문자열 포매팅을 사용할 수 있다. 사용법은 문자열 앞에 f 접두사를 붙인다. 이 기능을 이용해 왼쪽이나 오른쪽, 가운데로 정렬하는 것 또한 가능하다. 정렬하여 생기는 공백을 채울 수도 있다.

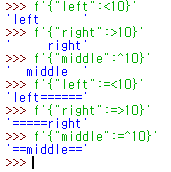


그림 4- 좌, 우, 가운데 정렬 및 공백 채우기

### 문자열 관련 함수

|  |  |  |  |
| --- | --- | --- | --- |
| 함수명 | 기능 | 함수명 | 기능 |
| count() | 문자의 개수 반환 | lstrip() | 왼쪽 공백 지우기 |
| find() | 위치 알려줌 | rstrip() | 오른쪽 공백 지우기 |
| index() | 위치 알려줌 | strip | 양쪽 공백 지우기 |
| join() | 문자열 삽입 | replace | 문자열 바꾸기 |
| upper() | 소문자 => 대문자 | split | 문자열 나누기 |
| lower() | 대문자 => 소문자 |  |  |

표 4- 문자열과 관련된 함수 종류

표의 왼쪽 위의 함수부터 실습해보자.



그림 4- count()함수 사용의 예

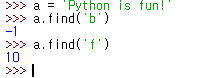


그림 4- find()함수의 사용 예. -1은 찾는 값이 존재하지 않음을 의미

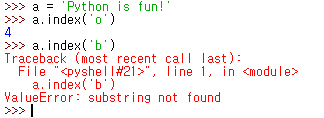


그림 4- index()함수의 사용 예. 오류 메시지는 찾는 값이 존재하지 않음을 의미



그림 4- join()함수의 사용 예



그림 4- upper()함수의 사용 예



그림 4- lower()함수의 사용 예

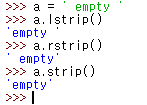


그림 4- lstrip(), rstrip(), strip()함수의 사용 예



그림 4- replace()함수의 사용 예

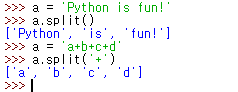


그림 4- split()함수의 사용 예

## 불 자료형

## 리스트 자료형

## 튜플 자료형

## 딕셔너리 자료형

## 자료형 변환

# 제어문

## 연산자

## if문

## for문

## while문

## continue와 break문

# 함수

# 클래스

# 모듈

# 패키지

# 참고자료

김왼손의 미운코딩새끼: 4시간만에 끝내는 파이썬 기초 (<https://www.inflearn.com/course/%ED%8C%8C%EC%9D%B4%EC%8D%AC-%EA%B8%B0%EC%B4%88-%EA%B0%95%EC%A2%8C/>)

점프 투 파이썬 (<https://wikidocs.net/book/1>)

점프 투 파이썬 (이지스퍼블리싱 출판, 박응용 지음)