파이썬 기본 문법 및 규칙

- 들어쓰기
- 연산
- 변수
- 식별자
- 출력 및 입력
- 주석달기

들여쓰기(indentation)

- 가독성을 높이고 프로그램의 코드 블럭을 구분하기 위한 방법 (다른 프로그램 언어에서는 주로 {}을 사용)
- 파이썬에서 들여쓰기는 문법적인 강제사항
- 방법은 한칸, 두칸, 탭 등 여러가지 방식
- 한 코드 안에서는 반드시 한 종류의 들여쓰기를 사용해야지 혼용하면 안된다.
  - 탭(tab)으로 들여쓰면 계속 탭으로 들여써야 한다.
  - 원칙으로는 공백(빈칸, space) 사용을 권장한다. (PEP 8 권장 사항)

In [ ]:

```
a = int(input('a의 값을 입력하세요'))
b = int(input('a의 값을 입력하세요'))

if a > b:
print('입력값 a가 b 보다 큽니다.')
elif:
print('입력값 b가 a 보다 큽니다.')
else:
print('두 값이 같습니다.')
```

In [ ]:

```
if True:
    print('4 spaces')  # space 4칸 사용
   print('3 spaces')   # space 3칸 사용
```

In [ ]:

```
x = 1
 y = 3
```

연산

숫자 연산

- 사칙연산

문자열 연산

- 덧셈과 곱셈만 가능

In [ ]:

```
5 + 7
```

In [ ]:

```
5 - 7
```

In [ ]:

```
7 * 2
```

In [ ]:

```
'안녕' + '파이썬'
```

In [ ]:

```
'파이썬' * 3
```

In [ ]:

```
'안녕' - '파이썬'
```

In [ ]:

```
'안녕' / '파이썬'
```

변수

변수(variable)란?

- 컴퓨터 메모리 어딘가에 위치해(저장되어) 있는 **객체를 참조하기 위해 사용하는 이름**이다.
  - 객체(object) : 숫자, 문자, 클래스 등 값을 가지고 있는 모든 것이다.
  - 파이썬은 객체지향 언어이며 숫자와 문자를 포함해 모든 것이 '객체'(object)로 구현되어 있다.
- 동일 변수에 다른 객체를 언제든 할당할 수 있기 때문에, 즉 참조하는 객체가 바뀔 수 있기 때문에 '변수'라고 부른다.

In [ ]:

```
x = 1
y = 'Hello'
z = 'Good Bye~~~!'

print(x)
print(y)
print(z)
```

어떤 데이터에는 따옴표가 있고 어떤 데이터에는 따옴표가 없나요?

In [ ]:

```
x = 1
y = '1'

print(x)
print(y)
```

위의 코드를 보면 두 변수 ***x***와 ***y***는 같은 값을 갖고 있는 것처럼 보인다.

하지만 변수의 종류를 알려주는 함수인 **type()**을 이용하면 두 변수가 다른 종류인 것을 알 수 있다.

In [ ]:

```
print(type(x))
print(type(y))
```

다음 코드를 실행해보자.

In [ ]:

```
x = hello
```

- 숫자는 따옴표로 묶지 않기로 약속했지만
- 따옴표로 묶지 않은 문자(문장)는 컴퓨터가 변수로 인식한다.

In [ ]:

```
hello = 'Good Bye~~~!'

print(hello)
print('hello')
```

두 개의 비슷한 **print()** 문이 전혀 다른 값을 출력하는 것을 알 수 있다.

파이썬에서는 따옴표로 묶은 문자는 **문자열**로 인식하고, 따옴표로 묶지 않은 문자는 **변수의 이름**으로 인식한다.

- **print(hello)** : 변수 *hello*가 담고 있는 'Good Bye'라는 문장을 출력
- **print('hello')** : 'hello'라는 문장을 출력

식별자

식별자(identifier)란?

파이썬 '객체'(데이터)를 식별하기 위해 붙이는 이름이다.

즉, 컴퓨터의 메모리 어딘가에 저장(위치)한 객체(데이터)를 컴퓨터와 프로그래머 모두가 쉽게 관리할 수 있도록 **변수**, **함수**, **클래스** 등에 붙이는 이름을 말한다.

식별자 구성 규칙

하나 또는 그 이상의 **공백이 없는 문자와 숫자의 조합**으로 **길이는 제한이 없다**.

하지만 **숫자로 시작할 수는 없다**.

문자 : 유니코드 문자(Unicode characters) 중 **영어 알파벳**(a ... z, A ... Z), **밑줄**('_'), **한글** 등

- 변수 이름이 길거나 두 개 이상의 단어 조합인 경우 단어 사이에 밑줄('_')을 사용하면 가독성이 높아진다.
  - e.g., *firstname*, *lastname*보다는 *first_name*, *last_name*이 좋다.
- 한글도 사용 가능하지만 플랫폼에 따라 인코딩(encoding) 문제가 발생할 수 있기 때문에 가급적이면 영어를 사용하는 것이 좋다.

숫자 : 0, 1,..., 9

- 첫 글자로 숫자가 올 수 없지만, 첫 글자 이 외에는 사용이 가능하다.
  - *x1* (okay)
  - *1x* (wrong)

대소문자를 구분(case sensitive)한다.

- *southkorea*, *Southkorea*, *SouthKorea*, *SouthKOREA*, *SOUTHKOREA*은 모두 다른 식별자다.

퀴즈

다음 중 식별자로 적절한 것과 아닌 것을 구별하시오.

tax4

답 : OK

In [ ]:

```
tax4 = 0.05
print(tax4)
```

4tax

답 : NO

In [ ]:

```
4tax = 0.07
print(4tax)
```

interest-rate

답 : NO

In [ ]:

```
interest-rate = 0.02
print(interest-rate)
```

interest_rate

답 : OK

In [ ]:

```
interest_rate = 0.02
print(interest_rate)
```

korea's

답 : NO

In [ ]:

```
korea's = '태극기'
print(korea's)
```

str

답 : OK but **not recommended!**

In [ ]:

```
str = 'interest-rate'
print(str)
```

예약어(Reserved Words)와 키워드(Keywords)

예약어(reserved words)란 특별한 용도로 사용하기 위해 미리 예약한 **식별자**(reserved identifiers)

- 예약어는 주로 컴퓨터 프로그래밍 언어에서 사용하면 명령어들로 구성되어 있지만 '예약'이란 단어의 의미에서 알 수 있듯이 때로는 아직 구현하지 않은 명령어를 미리 예약어로 지정하기도 한다.
- 따라서 예약어는 변수나 함수 이름 등 **사용자가 정의하는 식별자로는 사용할 수 없는 단어(이름)들**이다.
- 흔히 예약어는 **키워드(keywords)**와 같은 혼용해서 사용한다.

파이썬 예약어/식별자는 어떻게 확인할 수 있을까?

In [ ]:

```
help('keywords')
```

In [ ]:

```
help('or')
```

In [ ]:

```
import keyword
print(keyword.kwlist)
```

출력

출력은 특히 초보 프로그래머들에게 필수인 기능(함수)이다.

파이썬에서는 **print()** 함수를 이용해 다양한 것들을 출력할 수 있다.

In [ ]:

```
print('Hello, Python!')
```

출력 방식과 출력 형식

print() 함수를 사용하지 않고 출력

- 대화형 모드에서만 가능하다.
- 객체의 자료형을 알 수 있는 **대표 형식**으로 출력한다.

In [ ]:

```
'안녕 파이썬!!!'
```

In [ ]:

```
11 + 22
```

print() 함수를 사용해서 출력

- 대화형 모드와 인터프리터 모드 둘 다에서 사용할 수 있다.
- 객체를 사람이 보기 편한 **텍스트 형식**으로 출력한다.

In [ ]:

```
print('안녕 파이썬!!!')
```

In [ ]:

```
print(11 + 22)
```

고급 : print() 함수 탐구하기

작성 방법

print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False) (https://docs.python.org/3/library/functions.html#print)

- 다수의 위치 전달인자(*objects)와 4개의 키워드 전달인자를 받을 수 있다.
  - 각 키워드 전달인자는 기본 값이 있음
- ***sep*** 매개변수의 기본 값은 공백(' ')
  - 두 개 이상의 위치 전달인자가 주어질 경우 각각의 전달인자는 **sep** 값으로 나뉘어 출력한다.
  - 위치 전달인자가 하나만 주어졌을 경우 **sep** 매개변수는 아무런 역할도 하지 않는다.
- ***end*** 매개변수의 기본 값은 새줄바꿈(newline)이다.
  - 위치 전달인자들을 출력하고 난 후 마지막에 ***end*** 매개변수의 값을 출력한다.
- ***file*** 매개변수의 기본 값은 표준 출력 스트림이다.
  - 표준 출력 스트림(standard output stream)은 주로 콘솔(console)을 사용한다.
- ***flush*** 매개변수는 파이썬 3.3에 추가한 기능이다.
  - 값이 **True**이면 ***file*** 스트림을 강제적으로 내보낸다(flush).

따라하기 : print() 함수 잘 활용하기

In [ ]:

```
print('Python', 3)
```

In [ ]:

```
print('Python', 3, sep='')
```

In [ ]:

```
print('Python', 3, sep=' version ')
```

In [ ]:

```
print(1, 2, 3, 4, 5, sep=' + ')
```

print() 함수와 문자열 결합

In [ ]:

```
name = '파이썬'

# 아래 코드의 차이가 뭔지 확인하세요.
print('Hi!', name)
print('Hi! ' + name)  # 문자열 결합

print('Hi!', name, sep='!!!')
print('Hi! '+name, sep='!!!')
```

입력

파이썬에서 입력을 받는 대표적인 함수로는 **input()**이 있다.

input([prompt]) 함수

- 사용자로부터 값을 입력 받을 수 있는 내장함수다. 실행하면 즉시 커서가 나타나 입력을 기다린다.
- 대괄호(**[ ]**) 부분에 문자열 값을 넣으면 사용자로부터 입력받을 때 해당 문자열을 출력할 수 있다.
  - 예) **input()** : _ (화면에는 왼쪽과 같이 표시된다)
    - 아무 것도 출력되지 않은 상태에서 사용자의 입력을 기다린다.
  - 예) **input('당신의 이름은 무엇입니까?')** : **당신의 이름은 무엇입니까?**_ (화면에는 왼쪽과 같이 표시된다)
    - 해당 문자열을 출력하고 사용자의 입력을 기다린다.
- 사용자가 값을 입력을 한 후 Enter 키를 누르면 입력을 종료한다.
- 반환 값
  - 사용자가 입력한 값은 항상 **문자열** 형태로 반환한다.
  - 사용자가 아무런 값도 입력하지 않고 또는 키를 누르면 빈 문자열을 반환한다.

In [ ]:

```
input('좋아하는 숫자를 입력해 주세요: ')
```

In [ ]:

```
x = input('좋아하는 숫자를 입력해 주세요: ')
print('당신이 좋아하는 숫자는', x)
```

In [ ]:

```
x + 5
```

In [ ]:

```
print(type(x))
```

input() 함수가 반환하는 값은 모두 **문자열**이다.

따라서 이 값으로 사용하여 계산을 하려면 문자열 자료형을 숫자 자료형으로 변환해야 한다.

형변환

형변환(casting)이란?

어떤 자료형(data type)을 다른 자료형으로 변환시키는 과정

형변환의 필요성

파이썬은 변수를 선언할 때 자료형을 지정할 필요가 없다.

In [ ]:

```
x = 1
y = 2
print(x + y)
```

- 변수 ***x***와 ***y***를 숫자라고 선언해 주지 않았음에도 불구하고, 파이썬은 두 변수가 정수(integer)임을 알아채고 ***x + y***의 결과로 3을 반환한다.
- 명시적으로 변수의 자료형을 지정하지 않았음에도 불구하고 변수의 자료형이 존재한다는 것을 알 수 있다.(viz. 동적 언어)

이번에는 숫자 대신에 문자열을 사용해보자.

In [ ]:

```
x = '1'
y = '2'
print(x + y)
```

이번에는 변수 ***x***와 ***y***를 선언할 때 숫자를 작은 따옴표('')로 둘러싸서 문자열 변수라고 선언해 주자 파이썬은 이를 알아채고 결과로 '12'를 반환한다.

그렇다면, 만약 호환될 수 없는 서로 다른 자료형 변수를 가지고 연산을 한다면 어떤 결과가 나올까?

In [ ]:

```
a = 1
b = '2'
a + b
```

위와 같이 정수와 문자열 변수 간에는 **+** 연산을 할 수 없다면서 **TypeError**가 발생한다.

즉, 두 변수의 자료형이 맞지 않아 에러가 난 것으로, 이런 경우에 연산을 수행하고 싶다면 **형변환(casting)**을 통해 같은 자료형(또는 연산이 가능한 유사 자료형)으로 바꿔 줘야 한다.

In [ ]:

```
x = 1
y = '2'
print(x + int(y))
```

변수 ***y***는 문자형으로 선언되었지만 **int()** 클래스를 통해 정수로 변환했기 때문에 1과 합 연산이 가능하다.

자주 사용하는 형변환 클래스

- **str(객체)** : 객체를 문자형으로 변환
- **int(객체)** : 객체를 정수형으로 변환
- **float(객체)** : 객체를 실수형으로 변환

형변환이 가능한 자료형

형변환은 데이터 분석시 매우 유용하고 강력한 기능이지만

변수의 자료형이 서로 변환 가능한 형태인 경우에만 형변환을 할 수 있다.

정수(integer)를 실수(float)로, 실수(float)를 정수(integer)로 변환할 수 있다.

In [ ]:

```
x = 5            # 정수
print(type(x))   # x는 정수다.
```

In [ ]:

```
x = float(x)     # 정수를 실수로 형변환한다.
print(type(x))   # x는 이제 실수다.
```

In [ ]:

```
x = int(x)       # 실수를 정수로 형변환한다.
print(type(x))   # x는 다시 정수로 바뀌었다.
```

숫자(정수와 실수)를 문자열로 변환할 수 있지만, 문자열은 숫자로 변환 가능한 것만 숫자로 변환할 수 있다.

In [ ]:

```
x = 5           # 정수
print(type(x))  # x는 정수다.
```

In [ ]:

```
x = str(x)      # 숫자(정수와 실수)는 문자열로 형변환할 수 있다.
print(type(x))  # x는 문자열이다.
```

In [ ]:

```
x = float(x)    # 문자열 중 숫자로 변환이 가능한 문자열은 숫자로 변환할 수 있다.
print(type(x))  # x는 이제 실수다.
```

만약 자료형이 서로 변환 가능한 형태가 아니면 오류가 발생한다.

In [ ]:

```
x = 'drum'      # 문자열
print(type(x))  # x는 문자열이다.

x = float(x)    # error
```

주석 달기

주석(comment)이란?

- 중요한 점이나 다시 확인해야 하는 부분(reminder)을 표시하는 것이다.
- 컴퓨터는 주석을 인식하지 않는다.
- 즉, 사용자만을 위한 것이다.

가장 중요한 습관

- 개발자에게 있어서 주석을 다는 습관은 매우 중요한 일 중 하나다.
- 주석을 잘 작성하면 다른 사용자뿐만 아니라 본인도 나중에 작성한 코드를 쉽게 이해할 수 있고 코드의 분석 및 수정이 용이하므로 반드시 코딩하면서 동시에 주석다는 습관을 들이는 것이 중요하다.

파이썬에서 주석다는 법

- 주석으로 처리될 내용 맨 앞에 **#** 를 입력한다.
- 주석은 한 줄을 온전히 차지할 수도 있고, 그 줄의 코드 뒷 부분에 작성할 수도 있다.
- 그 줄의 **# 뒤에 작성되는 모든 내용은 주석으로 처리**된다.

한 줄 전체를 주석 처리하는 예

```
# 이름을 출력합니다.
print('Python')
```

그 줄의 마지막 부분에 작성하는 예

```
print('Python') # 이름을 출력합니다.
```

주석은...

- 코드 실행에 영향을 미치지 않을 뿐만 아니라,
- 프로그램 실행 속도를 느리게 하지도 않고,
- 실행 프로그램의 용량을 늘리지도 않는다.