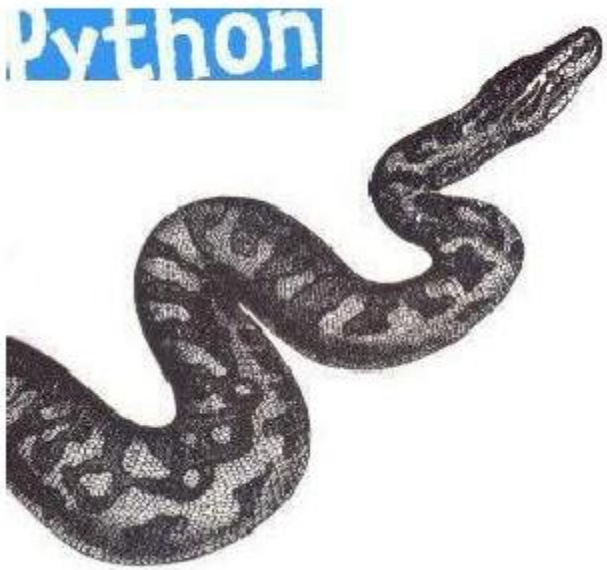
Python (파이썬)



- 김일한
- ilhank@naver.com

파이썬 개발환경설치

컴퓨터및 메모리구조

데이터타입_ 연산자 _제어문 _ 함수

- 클래스
- 파일처리, 표준함수, 데이터베이스

- 09:30 ~ 17:20
- 점심시간: 12:30~ 13:30
- 쉬는시간: 매시 20~30분
- 과제: 16:30 제출
- 평가: 금 15:30분종료 14:00

개발환경

- 파이썬 인터프리터(www.python.org)
- 아나콘다:파이썬 종합개발환경

(파이썬 인터프리터+ 각종유틸리티+라이브러리)

편집툴: 스파이터, jupyter notebook

- 파이참(편집)

Guido Van Rossum in Amsterdam

Guido Van Rossum

파이선의 장점

- 개발 시간을 단축 시켜준다
- 배우기 쉬울 뿐 아니라 개발자로
- 사용하기도 쉽다
- 코드를 이해하기 쉽다
- 코드가 짧아진다
- 팀 프로젝트에 좋다
- 확장성이 뛰어나다(C, Java)

파이선은 어떤 언어인가?

- 객체 지향 언어
- 대화기능의 인터프리터 언어
- 동적인 데이터 타입 결정 지원
- 플랫폼에 독립적
- 개발 기간 단축에 초점을 둔 언어
- 간단하고 쉬운 문법
- 고 수준의 내장 객체 데이터 형 제공
- 메모리 자동 관리
- 무료

활용 분야

- 시스템 관리 (스크립팅)
- GUI
- 인터넷 프로그래밍
- DB 프로그래밍
- 각종 텍스트 프로세싱
- 분산처리
- 수치연산, 그래픽스등등

2-1 변수명 및 예약어

변수명 만드는 법

```
[_a-zA-Z][_a-zA-Z0-9]*
```

변수명의 예

```
a, a1, my_name, your_job, MyName,
_private, __private_member
```

변수명이 될 수 없는 것들

```
1abc, @file, %x
```

2-1 변수명 및 예약어

예약어

예약어는 변수로 사용할 수 없다

예약어 확인하기

```
>>> import keyword
>>> keyword.kwlist
['and', 'assert', 'break', 'class', 'continue'
   , 'def', 'del', 'elif', 'else', 'except', '
   exec', 'finally', 'for', 'from', 'global',
   'if', 'import', 'in', 'is', 'lambda', 'not'
   , 'or', 'pass', 'print', 'raise', 'return',
    'try', 'while', 'yield']
>>> len(keyword.kwlist)
29
```

2-1 변수명 및 예약어

변수명 만들 때 조심할 점

함수의 이름이나 모듈의 이름은 피하라

```
>>> str(12345)
'12345'
>>> str = 'abc'  # 이제 str는 문자열 변수이다.
>>> str(12345)   # 함수로 사용할 수 없다.
Traceback (innermost last):
  File "<pyshell#23>", line 1, in ?
    str(12345)
TypeError: call of non-function (type string)
```

2-2 파이썬 기초문

주석문

```
# 이것은 주석입니다
```

연속라인

\ 은 다음 라인과 현재 라인을 연결

```
>>> if (a == 1) and
SyntaxError: invalid syntax
>>> if (a == 1) and
        (b == 3):
        print 'connected lines'
```

2-2 파이썬 기초문

치환문

```
>>> a = 1
>>> b = a
>>> 1+3 = a
SyntaxError: can't assign to operator
>>> a = 1
>>> a = a + 1
```

등호는 별도로 있음

```
>>> 4 == 5
0
>>> 4 == 4
1
```

2-2 파이썬 기초문

다양한 형태의 치환문

```
>>> c, d = 3, 4    # 여러 개를 한꺼번에 치환
>>> x = y = z = 0
>>> e = 3.5; f = 5.6   # ; 로 문들을 구분
>>> print a, b, c, d, e, f
1 1 3 4 3.5 5.6

>>> e, f = f, e # 값의 교환
>>> print e, f
5.6 3.5
```

2-2 파이썬 기초문

확장 치환문(2.0)

- +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=, **=
- x op= y 의 의미는 x = x op (y)와 같다

```
>>> a = 1
>>> a += 4
>>> a
5
>>> a -= 3
>>> a
2
>>> a *= 2+3
>>> a
```

문자열로 된 파이썬 코드 실행

- 파이썬 식 실행 – **eval()**
  - **eval(‘ a + 4’**
- 문 수행 – **exec**
  - **exec ‘ a += 4’**

콘솔 입출력

콘솔 입력

raw_input – 문자열 입력

Input - 식입력

콘솔 출력

print a,b

print(a, b)

자료형의 종류

주요 내장 자료형

- 수치형
- 문자열
- 리스트
- 튜플
- 사전

내장 자료형의 분류

자료형	저장모델	변경가능성	접근방법
수치형	리터럴	불가	직접
문자열	리터럴	불가	시퀀스
리스트	저장형	가능	시퀀스
튜플	저장형	불가	시퀀스
사전	저장형	가능	매핑

자료형 확인

type 사용
types 모듈

dir(types)

3-1 수치 자료형

정수형 상수

- 10진, 8진, 16진 상수

실수형 상수

- 소수점을 포함하건 e, E가 포함된 수
- 64비트로 표현
- 유효자리 17, 지수부 10의 –308~308정도

롱형 상수

- 정수형으로 표현할 수 없는 경우
- 무한 자리 수 표현

복소수형 상수

- 실수부와 허수부로 표현
- 각각 실수형으로 표현된다

3-2 파이썬 연산자

- 산술 연산자
- 관계 연산자
- 논리 연산자
- 비트단위 연산자

산술 연산자

+, -, *, /, //, **, %

/ 인 경우는 정수 / 정수 에 주의

관계 연산자

객체의 대소를 비교

참이면 1, 거짓이면 0을 돌려준다

연산자의 종류

>, <, >=, <=, ==, !=

논리 연산자

- 종류
  - **not x**
  - **x and y**
  - **x or y**
- 진리 값의 결과
  - 참이면 **1**, 거짓이면 **0**
- 객체의 진리 값
  - **0** 혹은 빈 객체이면 거짓
    아니면, 참
  - **None, 0, 0.0, 0L, 0.0+0.0j, [], (), {}**

논리 연산자

논리식 계산 순서

결과가 알려지는 시점까지만 계산
최종 계산 시점의 객체를 리턴

1 and 2

3 or 4

b = a > 4 and 10 or 20

수치 연산 함수

수치 연산을 위한 모듈

math – 실수 연산
cmath – 복소수 연산

```
>>> import math
>>> dir(math)
['__doc__', '__name__', 'acos', '    ', 'atan'
   , 'atan2', 'ceil', 'cos    cosh', 'e', 'exp
   ', 'fabs', 'floor', '        frexp    hypot
   ', 'ldexp', 'log', 'log10', 'modf', 'pi', '
   pow                ', 'sqrt', 'tan', 'tanh
>>> math.pi
3.1415926535897931
>>> math.e
2.7182818284590451
>>> math.sin(1.0)
0.8414709848078965
>>> math.sqrt(2)
1.4142135623730951
```

파이썬 제어문

if 문

```
if 조건식1:
    문들1
elif 조건식2:
    문들2
else:
    문들3
```

파이썬 제어문

for 문

```
before……
for x in [… …]:
        continue
        break

else:
    ……
after……
```

루프를 정상적으로
다 끝냈으면

파이썬 제어문

while 문

```
before.....
while 조건::
        continue
        break

else:
    .....
after.....
```