2. ~~3.~~ 기본 자료형과 연산

파이썬으로 할 수 있는 일

숫자형 ※정수, 실수만 잘사용하면 OK

파이썬의 숫자 자료형

항목	예
정수 (int)	123, 87, -7
실수 (float)	123.45, -43.21, 8.0, 3.14e5
복소수 (Complex)	3+2j, -7j
2진수 (int)	0b1101
8진수 (int)	0o34, 0o621
16진수 (int)	0x1A, 0xfff

자료형을 확인하기

- type(1.0) # 숫자 1.0의 자료형은?

왜 자료형을 알아야 할까?

생각해보는 페이지

- **왜 자료형을 구분할까?**
  - 성능과 제한은 밀접한 연관이 있다.

Q. 정수도 실수의 일부인데, 왜 정수와 실수를 구분할까?
컴퓨터 내부에 정수와 실수를 처리하는게 완전히 다르기 때문. 정수가 실수에 비해 속도가 더 빠름.
그래서 데이터를 다룰 때 정수를 다룰지, 실수를 다를지 고민!

- **컴퓨터의 계산은 누가 할까?**
  - 컴퓨터가 다룰 수 있는 자료형은 무엇일까?
  - 컴퓨터가 다룰 수 없는 자료형은 무엇일까?

- **컴퓨터가 할 수 없는 계산은 누가 할까?** → 그래서 파이썬을 배우는것! 인간알고 파이썬이 하자

- **숫자가 아닌 자료형도 있을까?** 문자/그림/음악 등....
  - 그림이나 음악은 어떤 자료형일까?

정수 동영상강의 정수와 실수

- 소수점 이하가 없는 수
- 2진수(0b111), 8진수(0o55), 16진수(0x55)도 사용 가능
  - 컴퓨터 입장의 정수

왜 2,8,16진수를?

- 수의 표현 범위는 무한대 (속도가 느릴 뿐)
  - print(2**1024) 의미 : 2^{1024}
- 자료형 변환
  - int (2.9) # 실수를 정수로 변환 -> 정수 부분 추출

실수 등명상 강의 정수와 실수

- 소수점 이하를 갖는 수 (부동소수형, float)
  - 1.2345, 3.0
- 지수형
  - 1.1e3, 0.1e-3
- 유효자리 15, 대략 10^{-308} ~10^{308} (한계가 있다)
- 무한대의 표현
  - float('inf')
  - float('inf') / 1000

floating point number

└ 유효자리 15이상이면 값이 안맞을수 있음

Q. 3.1415926을 화면에 3.14로 print하려면?
└ 고민해보기

실수는 CPU에 의존하여 계산한다.

실수

- **정수로 변환**
  - int(1.2)
  - round(1.2) # 반올림
  - ceil(1.2) # 올림
  - floor(1.2) # 내림

기본제공

math 모듈 필요 ∴ 그냥쓰면 error

math 모듈 불이는 방법

```
import math

math.ceil(1.2)
math.floor(1.2)
```

이렇게 써줘야 error나지 않음.

- **(주의) 실수는 오차 가능성이 있다.**

```
# 0.1을 1000번 더하면 100이 맞는데,

e = 0.0
for k in range(1000) :
    e += 0.1
```

0.1을 1000번 더하는 코드!

e에 99.99999999999986이 들어있다?

엉뚱한값 why? 1000번 반복해 오차가 커짐!

그럼 어떻게 ..?

실습 1

- 다음의 결과를 확인하라.

ㄱ. 12345678901234567890 * 98765432109876543210

ㄴ. 16진수 AA과 10진수 55의 합 (안해도 됨)

ㄷ. 1/3 + 1/3 + 1/3

ㄹ. 0.1 + 0.1 + 0.1

/ 는 나눗셈

ㅁ. 1.2 * 3.4 * 5.6 * 7.8 의 정수 부분의 값을 화면에 표시하기

ㅂ. 1.2 * 3.4 * 5.6 * 7.8 의 소수점 부분의 값을 화면에 표시하기

ㅅ. 0.0000000001234 * 9876

이것만 작성해서 올리세요!

모듈의 이해

3. 모듈 import하기 다 함께 볼것!

모듈을 불러오는 방법 3가지

① import math

- math 라는 모듈이 있다. 모듈 안에는 여러 부품이 있다. 그 모듈을 내 프로그램에 붙여서 사용한다.
- 모듈 안의 부품을 사용할 때에는 모듈 이름도 함께 지정한다.
- math.sin ( )

ex) print(math.pi)

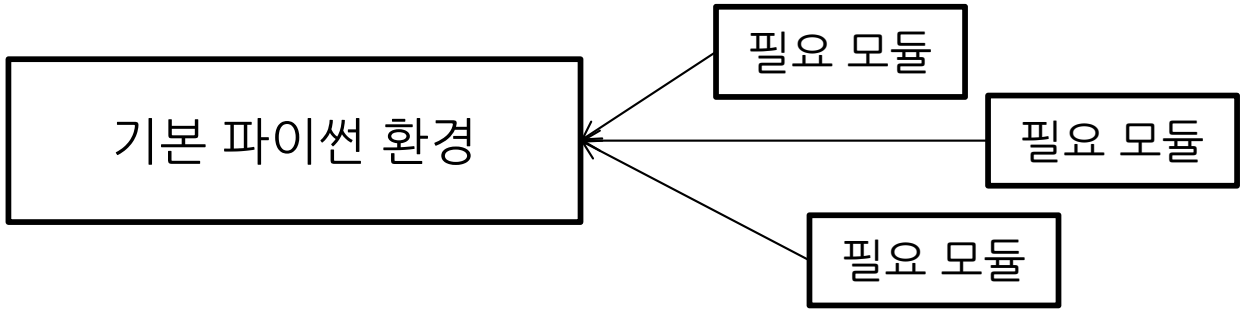
ex) print(pi)

② from math import e, pi, cos, sin

- math 라는 모듈에 포함된 부품 중 e, pi, cos, sin 부품을 가져온다.

③ from math import *

- math 라는 모듈의 모든 부품을 가져온다.



프로그램이 복잡하고 느려지니까!

왜 모든 것을 미리 import 해 놓지 않나?

분수 (fractions)

- **fractions 모듈 사용**

```
>>> from fractions import Fraction
>>> Fraction(5,7)                    # 5/7
>>> Fraction(5,7) + Fraction(2,5)
>>> float(Fraction(5,7) + Fraction(2,5) )
```

모듈이름 / 부품이름

$\frac{5}{7} + \frac{2}{5}$

한번 작성해보기!

* python 인터프리터 모드에서는 print( )를 쓰지 않아도 화면에 출력되지만 프로그램 모드에서는 print( )를 쓰지 않으면 출력되지 않는다.

5/7로 하면, 실수와 유효자리 생김 → 근사!!

왜 float 형을 쓰지 않고 분수(fraction)를 사용할까?

실습 2

- **다음의 결과를 확인하라.**
  1. 0.1 + 0.1 + 0.1 은 0.3 이지만, 결과가 다르게 나온다. 정확하게 나오게 하는 방법은?
  2. 반지름이 5.2인 원의 면적은 얼마인가?

[redacted]

원의 면적 공식 = πr^2 (r = 반지름)

할은 뒷장!

수와 연산자 4연산자 함께 보기

- +, -, *, / (사칙연산)
- / , //
  - /는 나누기 연산. 따라서 4/3은 1.333333
  - //는 나누기 연산의 몫(정수). 따라서 5 // 2은 2
- ** (제곱)
  - x ** y x^y
- % (나머지)
  - 7%3
  - 요일 계산, 홀수짝수 판단

이걸코딩으로 할수 있을까!

Q. 만날 목요일이다. 오늘부터 100일 뒤는 무슨 요일일까?

→ 그때 100/7 해주면 나머지는 3일

목 금 토 이틀후는 토요일!

오늘

따라서 답은 토요일.

연산자는 계산 가능한 자료형에만 써야 한다.

연산

- **동일한 자료형 또는 호환 가능한 자료형만 연산이 된다.**
  - 따라서 형 변환이 필요한 경우도 있다.

왜 같은 자료형이어야 할까?
계산은 누가 할까?

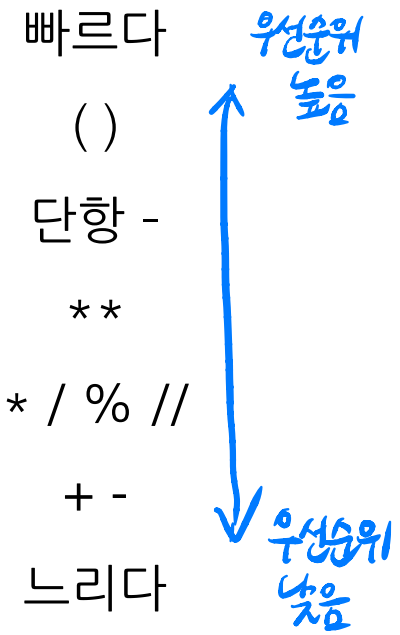
```
>>> 3 + 4.1          (O)        정수 + 실수
>>> 123 + '456'      (X)        정수 + 문자열
>>> 123 + 'abc'      (X)
>>> 123 + int('456')  (O)       정수 + 정수(문자열)
>>> 123 + int('abc') (X)
>>> 123 + ( 456+789j )  (O)
>>>  'abc' + '456'    (O)
```

숫자를 자료형으로 묶으면 해당 자료형으로 바뀐다.

연산자 우선순위

- *, / 가 +, - 보다 빠르다.
- %, //는 *, / 와 같다.
- **는 훨씬 빠르다.
- 단항 연산자 -가 더 빠르다.
  - 변수 앞에 붙이는 - 이다. (b = -a)
- ( ) 는 가장 빠르게 우선순서를 바꿀 수 있다.
- 2 + 3 + 4 와 2 + 3 ** 4 의 실행 순서는?
  - 2 ** 3 ** 4 는?

연산자 우선순위



실습3

- **다음의 수식의 결과는? Python을 이용하라.**

① 1000을 99로 나눈 나머지를 8로 나눈 나머지를 5로 나눈 나머지

② $4^{4^4} + 3^3 + 2$

③ $\frac{3^4}{5^{-2}}$

④ 오늘은 월요일이다. 오늘로부터 137일 후는 무슨 요일일까?

⑤ 원주율(π)은 3.1415926535 이다. 이 값을 소수점 이하 3자리까지만 화면에 출력하라.
(힌트 : 정수와 실수의 관계)

논리형 Boolean

- **참과 거짓을 표현하는 자료형**
  - 컴퓨터의 판단에는 참과 거짓은 매우 중요하다.
  - 컴퓨터는 계산만 하는 것이 아니라 판단도 할 수 있다.

- **논리형의 값 True , False**

```
>>> True          # 대소문자 구별
>>> 4 < 9         # < 나 > 는 크기를 판단하는 연산 기호
>>> a = 4 < 9
```

True False 라는것!

true false라쓰면 Error!

- **숫자와 논리형**
  - 값이 0 이면 False, 0 이 아니면(양수, 음수) True

문자형 String

- **문자(알파벳, 숫자, 기호)의 구성**
- **사람이 가장 많이 사용하는 형태의 자료형**
  - 계산을 위한 자료형이 아니라, 기억(저장)을 위한 자료형이다.
- **“ ” 으로 감싸서 사용**
  - “Alphabet”, “b”, “number 1234”
  - ‘ ’ 으로 감싸서 사용할 수도 있다.

이 외의 문자형 내용은 뒤에서