NPEX PYTHON 강의

서울대 전기공학부 명예교수
성원용

성원용

서울대학교 SEOUL NATIONAL UNIVERSITY

- 서울대 전기공학부 명예교수
  - 서울대 전기공학부 1989.2 ~2020.8 근무
  - 미국 University of California, Santa Barbara 박사 (1987)
  - 주 전공은 인공지능, 신호처리, 반도체설계, 병렬처리 프로그래밍 (전기공학부에서 가장 광범위)
  - 과거 연구: Synchronous DRAM architecture, NAND flash memory ECC, 음성인식, 고정소수점 변환, deep neural network
- 2020~2021: 광주과학기술원 AI대학원 석학초빙교수
- IEEE Fellow member
- Programming은 6살부터 100세까지.

WHY PYTHON?

서울대학교 SEOUL NATIONAL UNIVERSITY

- 매우 쉬운 언어 – 과거 Apple2의 BASIC 비슷 (변수 타입 선언 불필요, interpretive language, 간결)
- Python 언어 – Deep neural network 프로그래밍 언어인 TensorFlow 나 Pytorch 가 올라가는 언어 (다른 언어로 작성된 library 를 붙이기 쉽다 glue language.)
- NumPy – Python 언어에서 부르는 벡터라이브러리, 속도 향상에 중요 (Matlab 대치 가능)
- Python 사용범위 – 거의 모든 응용. 추가의 package 등을 이용. Web 부터 IoT. 아직 잘 안되는 곳 – mobile App.

- 요약 – 쓰기 쉽다. 속도는 library 해결, C/C++ function을 library로 붙이기 쉽다.

PYTHON이 C 언어보다 좋은 점

- Pointer 를 안 쓴다.
- 기본으로 제공하는 data 구조가 다양하다. (list 부터 dictionary 까지)
- 변수의 type을 미리 선언할 필요가 없다.
- 프로그램이 깔끔하다 (구조화를 indentation 으로 해결)
- Interpretive 언어이다. Compile, link 해서 돌릴 필요가 없다.
- Library 가 풍부하다. 특히 data science, deep learning

QUORA - WHY SHOULD I USE C++ INSTEAD OF PYTHON FOR SCIENTIFIC COMPUTING?

- In my beginning programming class I show students two bits of code, bubble sort in C++, and bubble sort in Python. The C++ code runs 100 times faster than python. And then I show that using the quicksort library function in Python runs 100 times faster than C++. (Note: this does not prove anything about the efficiency of python as such, but rather that libraries often contain better algorithms.)
- Now, Python is a more flexible language than C++, because a Python statement only has make sense when it's executed, while a C++ statement has to make sense to the compiler. That also makes Python slower than C++ because the C++ compiler, knowing more, can generate much much much more efficient code.
- So you should use Python if the efficiency of the code doesn't matter much, for instance because you are using library functions that do the majority of the work, and which are probably written in a low level language like C. See my sorting example.
- You should also use Python if you need library functions that don't exist in C++, for instance for file system operations or text manipulation. (Note: very recent C++ versions have included some of this functionality which has existed in Python for a decade or more.)
- But if you're writing basic code that is not available as library functions and efficiency is more important than development time, such as in much numeric code, you should use C++.

예비교육 스케쥴

서울대학교 SEOUL NATIONAL UNIVERSITY

- 월 오전 (강의): Python 변수 등, Control flow
- 월 오후 실습: Python 개발툴 깔기, 연습
- 화 오전(강의): Function, List 등
- 화 오후 실습:
- 수 오전 (강의): Recursion and dictionary, Class and object oriented programming
- 수 오후 실습: Object oriented programming
- 목 오전: Python inheritance, NumPy 소개
- 목 오후: 실습
- 금 오전: Numpy 강의
- 금 오후: 프로젝트

자료 소스 (1/2)

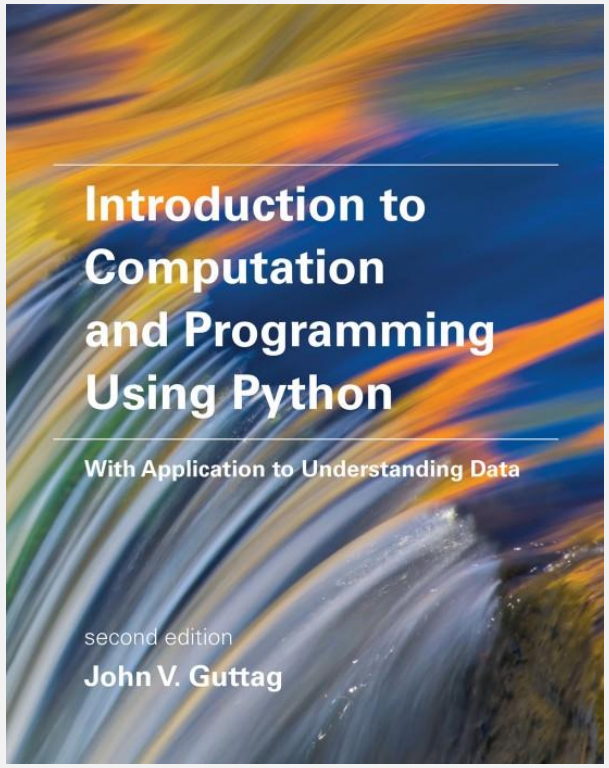
서 울 대 학 교
SEOUL NATIONAL UNIVERSITY

- MIT
- 6.0001 Introduction to Computer Science and Programming in Python
- Fall 2016

- 책: 파이썬 프로그래밍 (John V. Guttag 저
- 한국어 번역판 있음.

- 이 강의의 목적
  - (x) 모든 파이썬 문법을 잘 배우자
  - (O) 프로그램을 통한 좋은 문제해결 방법을 배우자
    Program을 divide, algorithm 및 속도

Python document
https://docs.python.org/3.8/



자료소스 (2/2)

- 세세한 문법 쉬운 설명 및 연습문제

- YouTube video (위의 점프 투 파이썬 바탕)

https://www.youtube.com/watch?v=yytWGELNeOI&t=28s&ab_channel=%EC%A1%B0%EC%BD%94%EB%94%A9JoCoding

내용 요약 (1/3)

1.컴퓨터의 구조 및 동작
- Data type (int, float, char, str 등)
- Operation (+, -, *, / 등)
- 기본 input, output

2. Program flow control
- 특별한 일이 없으면 program은 연속적으로 순서적으로 실행
- 조건문 (if, else 등)
- Loop (for, while)

```
//program for division
a = int(input('1st operand'))
b = int(input('2nd operand'))
if b!= 0:
    print('the division result=', a/b)
```

내용 요약 (2/3)

3. Function
   - 큰 program을 decompose 하여 작은 것들의 조합으로, 재활용
   - 내부에 state가 없음.
   - y = f (x0, x1, x2, …)
4. Data structure
   - List, tuple, dictionary, set
   - 미리 응용에 맞게 잘 정리해 놓은 데이터 구조
   - Program 효율 – meal-kit에 비유

```
def mag(b):
    return b[0]*b[0]+b[1]*b[1]


vector_a = [3, 4]
print (mag(vector_a))
```

내용 요약 (3/3)

서울대학교 SEOUL NATIONAL UNIVERSITY

5. Class and object oriented programming
   - Decompose, modular, 쉬운 데이터 관리
   - Function과 차별점은 내부에 state (기억장치)가 있다.
   - Class에 특화된 function 지원(method)
   - Class를 이용하여 여러 object 를 생성
     - 약간 변형된 상속 (Inherited) class 만들 수 있음.
- 6. Algorithm
   - Exhaustive/bisection search
- 7. Library 의 이용
   - NumPy library – for matrix-vector operation (Matlab 대치)
   - 쉬운 프로그래밍은 물론 속도 향상을 위해 필수적

프로그래밍을 잘 하려면 (외국어와 비슷)

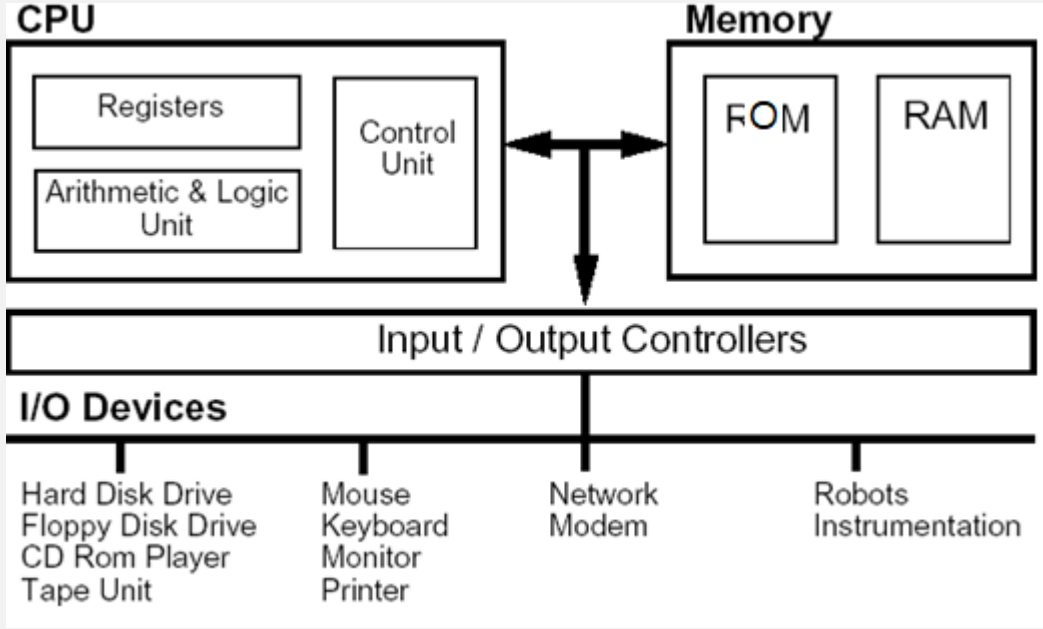
서울대학교 SEOUL NATIONAL UNIVERSITY

- 언어에 대한 기본 지속이 확실할 필요 (not 모든 명령어, function 알기 – 필요하면 Googling, YouTube)
- 직접 프로그램을 쳐 넣고 실행을 시킨다.
- Python은 굳이 compile을 하지 않아도 되기 때문에 이점에서 매우 편리하다. 학습속도 빠르다.
- 프로그램을 업무나 생활에 이용.
  - 반도체 – 반도체 소자 simulation, 간단한 yield 예측
  - 통신
  - …

Computer 구조 및 데이터 형식

COMPUTER 구조

- CPU (Central Processing Unit): PC 32bit, 64 bit
  - ALU: 연산(+, -, *, / 등)과 logical operation (AND, EXOR ..)
  - Control unit: 명령어를 하나씩 처리하면서 때로 순서를 건너 뛴다 (branch)
- Memory: DRAM PC의 경우 4~16 Gbyte 사용
- IO 장치



컴퓨터 안의 데이터 저장, 처리 – 모두 BINARY DATA (0 OR 1)

- Memory 안에는 정보가 binary 값으로 저장. 이를 쉽게 표현하기 위해서
- 2진 표현
  - 0과 1만 사용. 32비트의 표현에 32개의 0 또는 1.
  - 4비트 2진 데이터의 표현: 0~15 또는 -8 ~ +7
- Hexa-decimal 표현
  - 0, 1, …., 9, A, B, C, D, E, F
  - 32비트의 표현에 8개
  - 0xFFFF FFFF -> 0b1111 1111 1111 1111 … 1111
- 10진 표현

MEMORY 안 내용의 해석

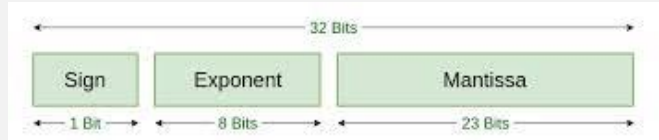
- Machine code (instruction):
  - CPU 내의 program counter 가 가리키는 번지에서 가져온다. 보통 32비트의 길이를 가지며, 이를 CPU에서 해독하여 여러가지 동작 (ALU 또는 program flow control operation)을 수행한다.
- Data
  - Computer가 load, store 에 의해서 가져오거나 쓰는 데이터. 이 데이터는 text (character 와 string), 정수, 부동소수점(실수) 등의 값을 나타낸다. 컴퓨터의 기계어 프로그램은 각 메모리 번지에서 읽어오거나 쓴 2진 숫자의 값이 이 중 어떤 종류에 해당하는지를 알고 있어야 한다. 일예로, 정수형이면 add 명령어를 실수형이면 addf 명령어를 사용하여 처리한다.
- Machine code와 Data는 동일 메모리 공간에 위치한다. 막 뒤섞여 있을 수도 있다. 이를 Von Neuman 구조라 한다.

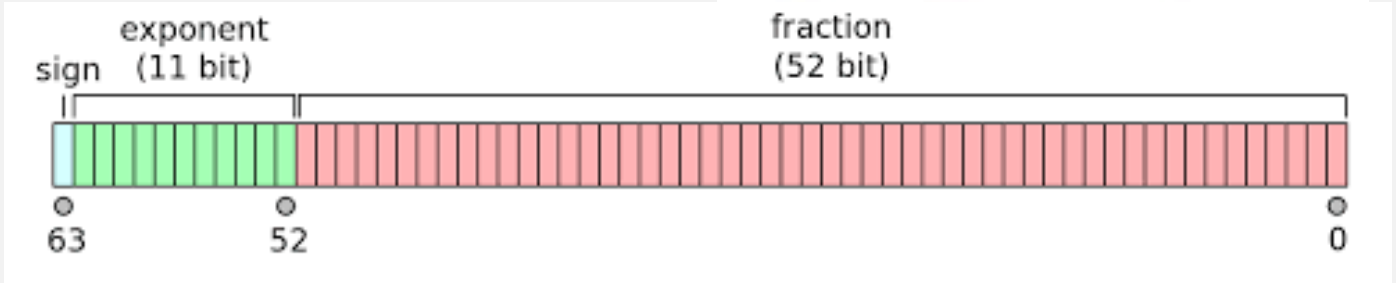
COMPUTER 내 데이터의 해석

- Integer data (two's complement)
  - 맨 앞의 bit에 따라 0이면 +, 1이면 – 값이 된다.
  - 0000…..0000 -> integer 값 0
  - 0000…..0011 -> integer 값 3
  - 0111…..1111 -> integer 값 2147483647 (=2^31-1)
  - 1111…..1111 -> integer 값 -1
  - 1000…..0000 -> integer 값 – 2147483648 (- 2^31)
  - 32bit, 64bit (long), 128bit (long long)
- Floating-point data
  - Single precision의 경우 32bit 값을 8bit는 exponent, 24bit는 sign과 값 (mantissa)을 표현하는데 사용
  - Python은 보통 64비트 double precision floating-point 사용 (11bit exponent)
- Character data (string)
  - 보통 Unicode 를 사용 (매우 많은 글자, 이모티콘 사용가능)
  - UTF-8 (자주 나오는 알파벳은 8비트를 사용)

FLOATING-POINT FORMAT

- Integer format 의 약점 – 너무 큰 수, 너무 작은 수 못 나타낸다.
- $2^{exponent} * significand$
- Single precision 32bit, double precision 64bit
- Double precision range:
  - $2^{-1022} \approx 2 \times 10^{-308}$ to approximately $2^{1024} \approx 2 \times 10^{308}$.
- 값의 정밀도: single precision: 10진 숫자 약 7개
  - double precision 약 16개





PYTHON 시작

Python의 특징

컴퓨터 언어에서 변수의 기본 개념

int, float, complex

String

Boolean

type casting

실습 : Anaconda 깔기, 이자 계산하기 등

자료 소스:

MIT

6.0001 Introduction to Computer Science and Programming in Python

Fall 2016

PYTHON 언어의 역사

서울대학교 SEOUL NATIONAL UNIVERSITY

- Created in 1990 by Guido van Rossum
- Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office … would be closed, but I had a home computer, and not much else on my hands.
- I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus).

PYTHON 특징

서울대학교 SEOUL NATIONAL UNIVERSITY

- Designed to be **easy to learn** and master
  - Clean, clear syntax
  - Very few keywords
  - High productivity 2-10x compared to C, C++, Java
  - Interpreter 언어 (Compile 불필요)
- Highly portable
  - Runs almost anywhere - high end servers and workstations, down to windows CE
  - Uses machine independent byte-codes
- Extensible
  - Designed to be extensible using C/C++, allowing access to many external libraries

PYTHON 특징(2)

- Clean syntax plus high-level data types
  - Leads to fast coding (First language in many universities abroad!)
- Uses white-space to delimit blocks
  - Humans generally do, so why not the language?
  - Try it, you will end up liking it
- Variables do not need declaration
  - Although not a type-less language

```
# Simple Python program example <- this is comment
def add_list(l):   #def is the start of a function
    sum = 0.0
    for elem in l:  #for loop index needs not be a number
        if type(elem) == float:  #indentation is important
            sum +=elem           # no { } for code block
    return sum


l = [1.0, 2.0, 3.0, 'garbage', 4.0] # list can contain any type

total = add_list(l)       # You do not use pointer to use list
print('The total is=', total)  #no needs variable declaration
```

function boundary

PYTHON 과 C 언어의 차이 (1/2)

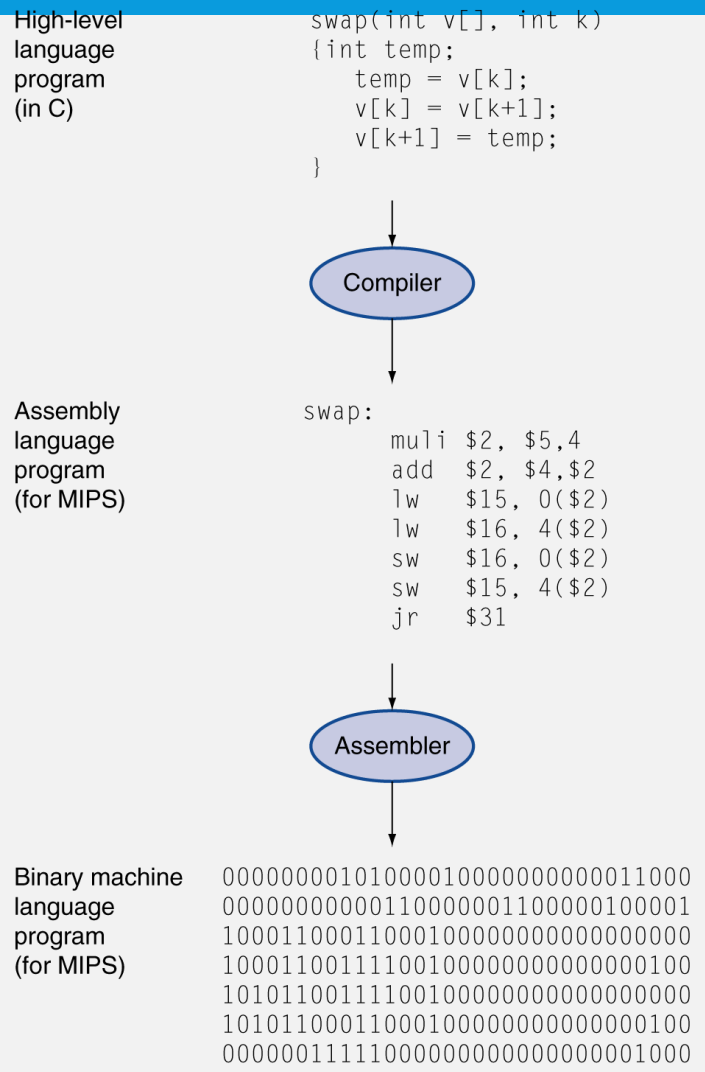
- C 언어: compile을 한다음에 run을 시킨다.

  C source file -> compile, link -> execution file (a.out 등)
  - Execution 전에 미리 compile을 해 놓아야 하니까 귀찮다.
  - Run time error가 어디에서 났는지 찾기가 어렵다.
  - Machine code를 돌리기 때문에 실행시간은 짧다.

- Python 언어: 한 줄 한 줄 source code를 읽으면서, 즉시 해독을 하면서 돈다. (Bytecode로 변환 후 실행)
  - 매 execution 시마다 해독, 실행을 중복으로 하는 셈이다.
  - 굳이 compile 을 한 다음에 돌리는 오버헤드가 없다.

C/C++ PROGRAM이 도는 구조

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
- Machine codes
  - Binary digits (bits)
  - Encoded instructions and data
  - 실제로 컴퓨터 하드웨어에서 동작



PYTHON과 C 언어의 차이(2/2)

- C 언어는 사용하는 변수의 data type을 프로그램의 시작시에 (최소한 사용전에) 미리 지정해야 한다.
  - 지정하는 까닭: 그래야 메모리에 적절한 크기를 할당한다.
  - 예: float (32비트), double (64비트), int (32비트), short (16비트), long, char, …
- Python은 사용하는 변수의 data type을 미리 지정하지 않는다.
  - 파이썬은 변수를 선언할 때 타입을 지정하지 않는다. 값을 할당하면 그때 타입이 정해집니다.
  - 연산 시 data type 또는 data의 길이가 바뀔 수 있고, 또 새로운 위치에 저장이 된다

PYTHON VS C PROGRAM

```
# Store input numbers
num1 = input('Enter first number: ')
num2 = input('Enter second number: ')

# Add two numbers
sum = float(num1) + float(num2)

# Display the sum
print(“Sum”, sum)
```

```
#include<stdio.h>
int main() {
int a, b, sum;
printf("\nEnter two no: ");
scanf("%d %d", &a, &b);
sum = a + b;
printf("Sum : %d", sum);
return(0);
. …
```

PYTHON의 구현체 CPYTHON

서 울 대 학 교
SEOUL NATIONAL UNIVERSITY

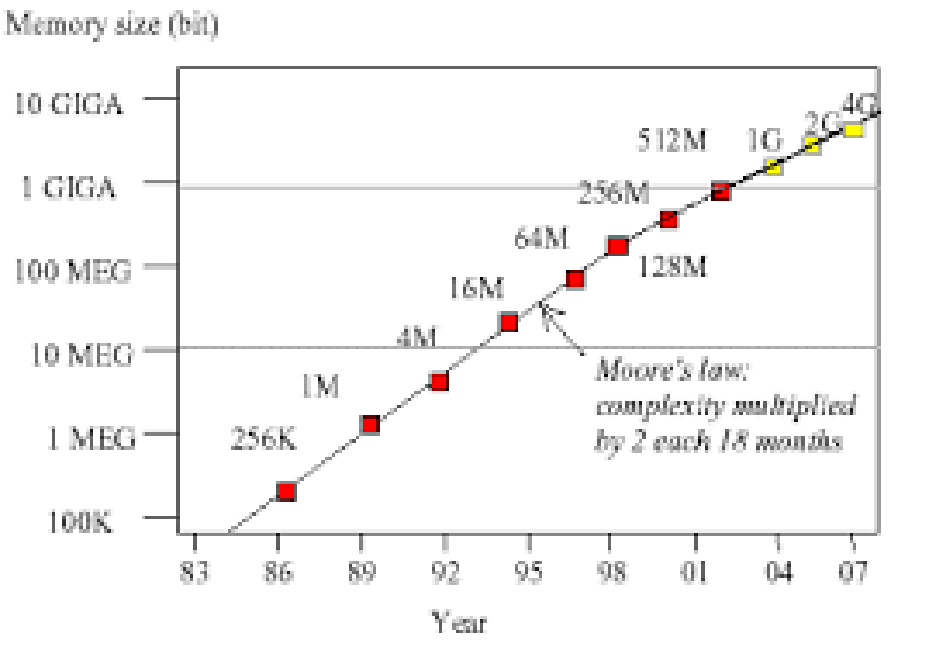
- 일반적으로 python이 C로 구현되어 있다고 알려져 있는데 그 구현체가 CPython이다. 가장 처음 만들어진 python의 구현체이고 Python.org 에서 다운 받으면 CPython을 받는 것이다. 다른 구현체들과 비교해서 언급할 때 주로 CPython이라고 표기하는데 Python을 C언어로 구현한 구현체를 의미한다.
- CPython은 인터프리터 이면서 컴파일러 이다. 우리가 작성하는 Python 코드를 bytecode로 컴파일 하고 실행한다. 다시 말해, python 코드를 C언어로 바꾸는 것이 아니라 컴파일하여 bytecode로 바꾸고 그 다음에 interpreter(virtual machine)가 실행한다.
- .py 파일을 실행하면 .pyc 라는 파일이 생성되는데 이것이 CPython이 컴파일한 bytecode가 들어있는 것이다. 그 다음 .pyc를 interpret 하는 것도 CPython이다.

https://cjh5414.github.io/about-python-and-how-python-works/

- **바이트코드**(Bytecode, portable code, p-code)는 특정 하드웨어가 아닌 가상 컴퓨터에서 돌아가는 실행 프로그램을 위한 이진 표현법이다. 하드웨어가 아닌 소프트웨어에 의해 처리되기 때문에, 보통 기계어보다 더 추상적이다.
- 역사적으로 바이트코드는 대부분의 명령 집합이 0개 이상의 매개 변수를 갖는 1바이트 크기의 동작 코드(opcode)였기 때문에 바이트코드라 불리게 되었다. 바이트코드는 특정 하드웨어에 대한 의존성을 줄이고, 인터프리팅도 쉬운 결과물을 생성하고자 하는 프로그래밍 언어에 의해, 출력 코드의 한 형태로 사용된다. 컴파일되어 만들어진 바이트코드는 특정 하드웨어의 기계 코드를 만드는 컴파일러의 입력으로 사용되거나, 가상 컴퓨터에서 바로 실행된다.
- 사람이 읽기 쉽도록 쓰인 소스 코드와 비교하면, 바이트 코드는 덜 추상적이며, 더 간결하고, 더 컴퓨터 중심적이다. 예를 들어 바이트코드는 변수의 접근 범위(지역변수 또는 전역변수 인지 여부) 등과 같은 의미 분석 단계의 결과를 부호화한다. 그래서 일반적으로 소스 코드를 직접 분석/실행하는 것보다 더 좋은 성능을 보여 준다.
- 바이트코드 프로그램은 보통 한 번에 하나의 명령어를 읽은 후 실행한다. 이러한 형태의 바이트코드 인터프리터는 높은 이식성을 갖는다. 또 다른 형태로서 실시간 번역기 또는 저스트 인 타임(just-in-time, JIT) 컴파일러라 불리는 시스템은 실행 중에 필요에 따라서 바이트코드를 기계어로 번역한다. 이 방법은 가상 컴퓨터의 이식성을 없애지만, 바이트코드 자체에 대한 이식성을 잃지는 않는다. 예를 들어, 자바와 C# 코드는 보통 바이트코드 형태로 컴파일되어 저장되며, 실행 전에 JIT 컴파일러에 의해 기계 코드로 번역된다. 이 번역으로 인해 프로그램 실행 전에 지연시간이 발생하게 되지만, 보통 인터프리터보다는 훨씬 좋은 성능을 보여 준다.

배경

- C language – 1972년, UNIX – 1971년
  - Mini computer, core memory의 시대 (메인 메모리 32Kword 등)
  - 메모리 한 바이트 한 바이트를 아껴서 써야 할 필요.
  - 높은 CPU운용 효율이 필요.
  - 어셈블리 언어 프로그래밍 대체가 주 목적
- Python – 1990년
  - 32비트 마이크로컴퓨터와 DRAM 의 시대
  - 쉬운 프로그래밍 환경이 필요
  - Library 를 이용한 빠른 SW 개발의 시대



PRINTING TO CONSOLE

- to show output from code to a user, use `print` command

```
In [11]: 3+2
Out[11]: 5

In [12]: print(3+2)
5
```

"Out" tells you it's an interaction within the shell only

No "Out" means it is actually shown to a user, apparent when you edit/run files

Python의 약점

Syntax 에러 등의 체킹 능력이 떨어진다.

variable type을 미리 declare 하지 않기 때문

In [6]:

```
a = '1234'
print(a)
b = 4
c = a + b
print(c)
```

```
1234
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-d1486e3b2a6e> in <module>
      2 print(a)
      3 b = 4
----> 4 c = a + b
      5 print(c)

TypeError: can only concatenate str (not "int") to str
```

프로그래밍 - 변수

- 데이터를 담는 '그릇'에 해당 – 구체적으로는 메모리 공간에 저장이 된다.
- 앞의 프로그램에서 num1, num2, sum
- C 언어의 경우 프로그램에서 이 그릇의 타입을 미리 지정해야 한다.
  - integer (32비트), long(64비트), float(32비트), double(64비트), character
  - 이 사이즈를 넘으면 over-flow가 날 수 있다.
  - (대개) 도중에 이 그릇의 타입을 바꾸지 못한다. (자동 변형도 있지만 찾기 힘든 에러를 만들 수 있다.)
- Python 은 이를 object (객체)라 부른다.
  - Object - any data with state (attributes or value) and defined behavior (methods).

PYTHON OBJECT TYPES

- Scalar and Non-scalar (복잡한 것)
- Scalar 객체는 더 이상 나눌 수 없는 것.
- 네 가지의 scalar object
  - int
  - float
  - bool – True and False
  - None
- Python 3.7 지금 우리 버전은 부동에 double precision (64bit) 사용

```
a = 10.0
b = 3
c = 5
d = True
print (a+b)
print (a/b)

print (not d)
if d:
    print(b+c)
if not d:
    print(b-c)
```

```
13.0
3.3333333333333335
False
8
```

PYTHON BUILT-IN DATA TYPES

서 울 대 학 교
SEOUL NATIONAL UNIVERSITY

- - **boolean** : 참(True), 거짓(False) 두가지 값.
- - None 비었다.
- - **int(integer)** : 정수
- - **float(floating-point number)** : 실수.
- - **complex number** : 복소수.
- - **str(string)** : 문자열.
- - **list** : 순서가 있는 값들의 집합.
- - **tuple** : 리스트와 같이 순서가 있는 값들의 집합이지만 읽기전용.
- - **set** : 순서가 없고 중복되지 않는 값들의 집합.
- - **dict(dictionary)** : 키와 값의 쌍으로 데이터를 저장.

왜 LIST, DICTIONARY, SET 등 여러 데이터 구조 필요?

- Data 구조가 좋으면 프로그래밍에 매우 편리.
- 각각의 구조에 맞는 많은 function (method)이 준비되어 있다.
  - 학생의 학번을 주고, 이름과 성적을 찾는다. - dict구조가 편리
  - Python 내부에 dictionary 를 빠르게 search 하는 알고리즘이 구현되어 있다.
- 예 – meal-kit를 이용한 식사준비와 비슷
- 이러한 면에서 C++의 기능을 많이 가져왔다.

PYTHON: INTEGER AND FLOAT DATA RANGE

서울대학교 SEOUL NATIONAL UNIVERSITY

```
print("This is data test")
i = 1
while i < 100:
    result = 2**i
    print (i, result)
    i = i+1
```

```
...
93 9903520314283042199192993792
94 19807040628566084398385987584
95 39614081257132168796771975168
96 79228162514264337593543950336
97 158456325028528675187087900672
98 316912650057057350374175801344
99 633825300114114700748351602688
Process finished with exit code 0
```

```
print("This is data test")
i = 1
while i < 100:
    result = 2.0**i
    print (i, result)
    i = i+1
```

```
....
93 9.903520314283042e+27
94 1.9807040628566084e+28
95 3.961408125713217e+28
96 7.922816251426434e+28
97 1.5845632502852868e+29
98 3.1691265005705735e+29
99 6.338253001141147e+29
Process finished with exit code 0
```

참고 $\log_2 10 = 3.32$
즉 32bit integer는 10진 약 9자리

Integers have unlimited precision(!). Floating point numbers are usually implemented using double in C

UNLIMITED INTEGER PRECISION의 의미

- 데이터의 크기가 미리 정해져 있지 않다.
  - 그 까닭으로 매 assignment 마다 다른 주소에 기록한다.
- Array 연산을 C언어에서는 pointer (주소)를 조작해서 쉽게 하는데, Python에서는 그렇게 하기 힘들다 (pointer 를 안 쓴다.) -> NumPy에서 지원
- Python의 List에 여러가지 data type을 자유롭게 쓸 수 있다.
- 이러한 이유로 Python은 느리다.
  - Interpretive language
  - 유연하고 복잡한 data type

PYTHON 속도문제 해결은 무엇인가?

서울대학교 SEOUL NATIONAL UNIVERSITY

- Built-in library를 이용
- NumPy library 등
  - NumPy function 은 미리 C/C++ 언어 등으로 만들어서 속도 최적화를 해 놓았다.
  - NumPy function 의 array는 미리 data 의 크기가 정해져 있다 (single precision 부동 소수점 등). 따라서 NumPy function 의 array는 Python의 list와 다르다. Homogeneous 한 data의 집합 (비유, 단독주택이 아니라 아파트이다.)
- Python에서 NumPy 에 function을 이용하면 여기에서 loop operation이 일어난다. 그 결과 대부분의 시간이 여기에서 실행이 된다. 최근의 CPU는 SIMD와 멀티코어 구조를 가지고 어레이 연산을 빠르게 한다. 그러나 타입이 정해져 있어야 한다.

In [15]:

```
import numpy as np
import time
a = [float(i) for i in range(10000)];
b = [float(i) for i in range(10000)];

tic = time.time()
dot = 0.0;
for i in range(len(a)):
    dot += a[i] * b[i]
toc = time.time()

print("dot_product = "+ str(dot));
print("Computation time = " + str(1000*(toc - tic )) + "ms")

a_array=np.array(a)
b_array=np.array(b)
n_tic = time.time()
n_dot_product = np.dot(a_array, b_array)
n_toc = time.time()

print("₩nn_dot_product = "+str(n_dot_product))
print("Computation time = "+str(1000*(n_toc - n_tic ))+"ms")
```

```
dot_product = 333283335000.0
Computation time = 10.971307754516602ms

n_dot_product = 333283335000.0
Computation time = 0.9603500366210938ms
```

PYTHON 버전

- 현재 버전 Python 3.9 (Oct 2020)
- Python 2.* version은 현재 사용 안함.
- 대표적으로 2.x에선는 print가 statemen로 쓰임. 3.x에서는 function

print “Hello World”

print(“Hello World”)

```
import sys
print("You are using Python {}.{}.".format(sys.version_info.major, sys.version_info.minor))
```

You are using Python 3.7.

객체(OBJECT)

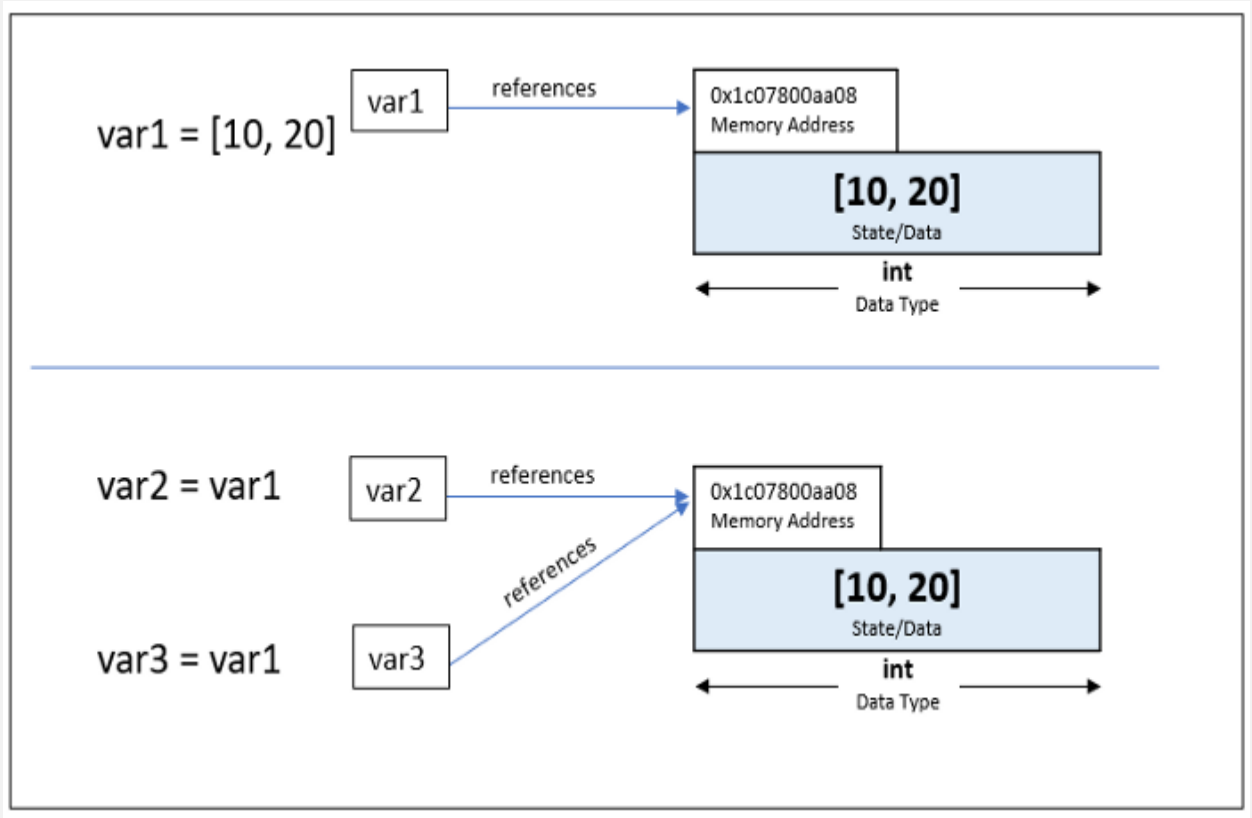
서울대학교 SEOUL NATIONAL UNIVERSITY

- *객체 (Objects)*는 파이썬이 데이터(data)를 추상화한 것(abstraction)입니다. 파이썬 프로그램의 모든 데이터는 객체나 객체 간의 관계로 표현됩니다.
- 객체에는 값(value)·유형(type)·정체성(identity)이라는 세 특성이 있다.
- 값은 메모리에 기록된 내용이다.
- 유형은 데이터의 종류로, 유형에 따라 그 값을 어떻게 읽고 다루어야 할지가 결정된다.
- 정체성은 각각의 객체를 식별하기 위한 고유번호(id)로, 객체가 메모리 속에서 위치한 주소 값이기도 하다.

REFERENCE COUNTING IN PYTHON

서 울 대 학 교
SEOUL NATIONAL UNIVERSITY

- Reference counting is one of the memory management technique in which the objects are deallocated when there is no reference to them in a program.
- Variables in Python are just the references to the objects in the memory. In the below example, when Python executes var1 = [10, 20], the integer object [10, 20] is stored in some memory location 0x20bfa819cc8, and var1 is only the reference to that object in the memory. This means var1 doesn't contain the value [10, 20] but references to the address 0x20bfa819cc8 in the memory.
- Python 객체의 reference count는 객체가 참조될 때마다 증가하고 객체의 참조가 해제될 때 감소한다. 객체의 reference count가 0이 되면 객체의 메모리 할당이 해제된다.



https://towardsdatascience.com/understanding-reference-counting-in-python-3894b71b5611

>>> year = 1789

객체를 만들어 변수에 대입

>>> year

객체의 값 (객체 자신) 구하기

1789

>>> type(year) # 객체의 유형 (클래스) 구하기

<class 'int'>

>>> id(year) # 객체의 정체성 (고유번호) 구하기

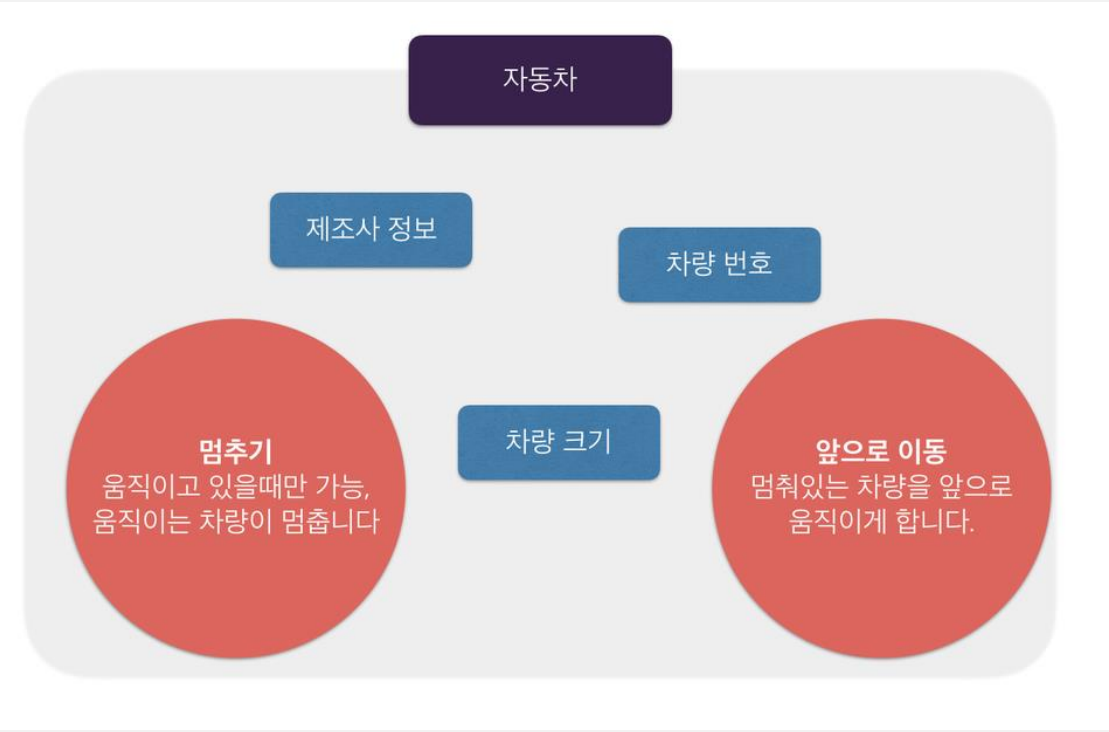
140711867085328

```
year = 1789
new_year = 1789
print(year == new_year)                 #True
print(type(year) == type(new_year))     #True
print(id(year) == id(new_year))         #False
same_year = year
print(id(year) == id(same_year))        #True
same_year = 2020
print(same_year, year)                  #2020, 1789
print(id(year) == id(same_year))        #False
id_year = id(year)
year = year +1
print(id_year, id(year))                #2920727913136 2920727913200
```

객체(OBJECT)의 이해

서 울 대 학 교
SEOUL NATIONAL UNIVERSITY

- Any data with state (attributes or value) and defined behavior (methods).
- Also the ultimate base class of any new-style class.
- 객체는 어떠한 속성값과 행동을 가지고 있는 데이터입니다.
- 파이썬의 모든것들(숫자, 문자, 함수 등)은 여러 속성과 행동을 가지고 있는 데이터입니다.
- 실제 세상에서도 object는 그 본연의 추상적인 개념만 가지고 있는 것이 아니라, 다양한 정보와 행동을 가지고 있습니다.



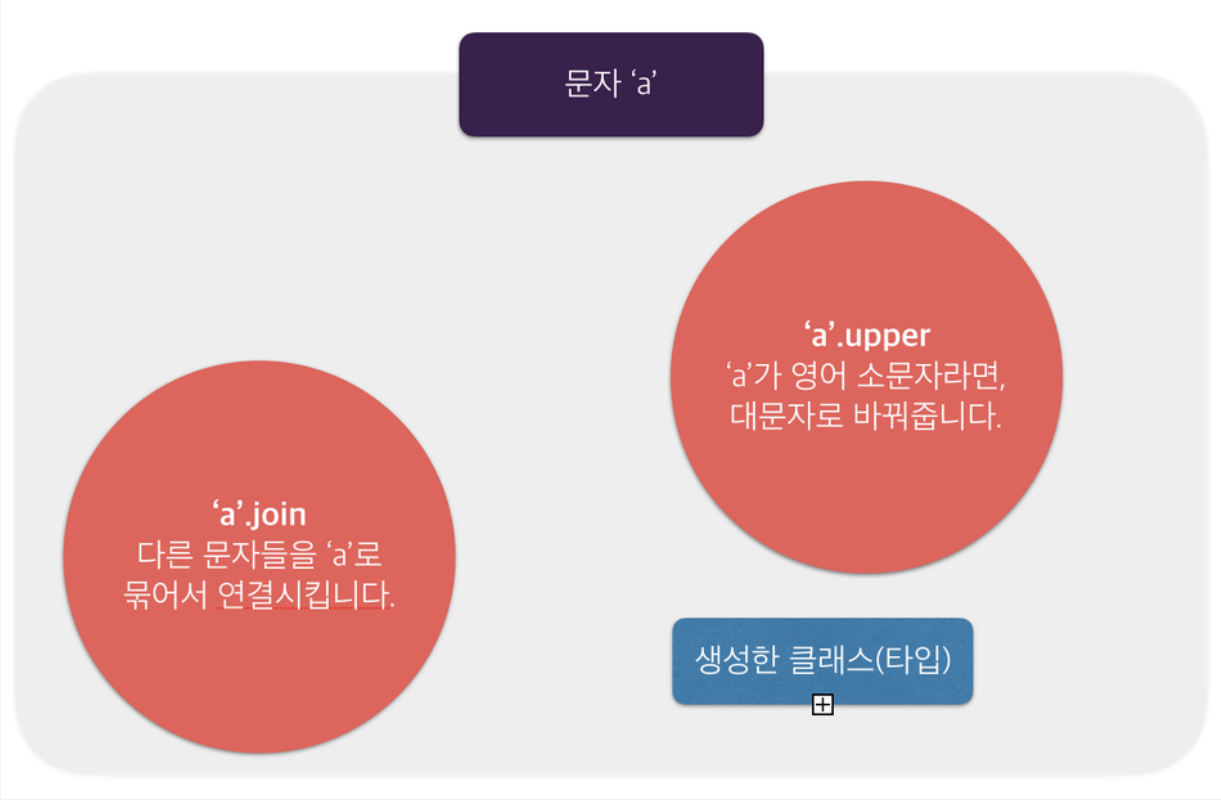
- 자동차를 가지고 한번 생각을 해보겠습니다.
- 자동차를 원동기를 장치하여 그 동력으로 바퀴를 굴려서 철길이나 가설된 선에 의하지 아니하고 땅 위를 움직이도록 만든 차. 라는 사전적 의미만을 담는것이 아니라
- 내가 탈 수 있고, 앞으로 뒤로 움직이는 행동을 할 수 있고 차 앞뒤에 있는 차량번호 정보와 디자인과 제조사, 모델명 정보도 있습니다.
- 이러한 정보와 행위를 묶은 데이터를 하나의 자동차 객체로 볼 수 있습니다.

METHOD – 특정 종류의 OBJECT에 적용되는 함수

- object이름.method(**) 의 형태

```
abc = 'abc'
print(abc.upper())
print(''.join(['c', 'd', 'e']))
print('_'.join(['c', 'd', 'e']))
print(abc.join(['c', 'd', 'e']))

ABC
cde
c_d_e
cabcdabce
```



STRING METHOD

- They return the new value

Method	Description
capitalize()	Converts the first character to up
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specifie
encode()	Returns an encoded version of the stri
endswith()	Returns true if the string ends with the
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value an
format()	Formats specified values in a string
format_map()	Formats specified values in a strir
index()	Searches the string for a specified valu
isalnum()	Returns True if all characters in the stri
isalpha()	Returns True if all characters in the stri
isascii()	Returns True if all characters in the stri
isdecimal()	Returns True if all characters in th
isdigit()	Returns True if all characters in the stri
isidentifier()	Returns True if the string is an ide
islower()	Returns True if all characters in the stri
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Converts the elements of an iterable into a string
ljust()	Returns a left justified version of the string
lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value
rfind()	Searches the string for a specified value and returns the last position of where it was found
rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case
zfill()	Fills the string with a specified number of 0 values at the beginning

w3schools.com/python/python_ref_string.asp

PYTHONTUTOR.COM

VARIABLE NAMING

- lowercase
- underscore_between_words
- don'tstartwithnumbers
- SeePEP8

- A variable name must start with a letter or the underscore character

- A variable name cannot start with a number (숫자로 시작하면 안된다)
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Type	Naming Convention	Examples
Variable	Use a lowercase single l etter, word, or words. S eparate words with und erscores to improve rea dability.	x , var , my_variable
Class	Start each word with a c apital letter. Do not sep arate words with unders cores. This style is called camel case.	Model , MyClass

RESERVED WORDS (KEY WORDS)

서울대학교 SEOUL NATIONAL UNIVERSITY

Following is the list of reserved keywords in Python 3

and	except	lambda	with
as	finally	nonlocal	while
assert	false	None	yield
break	for	not	
class	from	or	
continue	global	pass	
def	if	raise	
del	import	return	
elif	in	True	
else	is	try	

Python 3 has 33 keywords while Python 2 has 30. The print has been removed from Python 2 as keyword and included as built-in function.

To check the keyword list, type following commands in interpreter −

```
>>> import keyword
>>> keyword.kwlist
```

assert	For debugging
break	To break out of a loop
class	To define a class
continue	To continue to the next iteration of a loop
def	To define a function
del	To delete an object
elif	Used in conditional statements, same as else if
else	Used in conditional statements
except	Used with exceptions, what to do when an exception occurs
False	Boolean value, result of comparison operations
finally	Used with exceptions, a block of code that will be executed no matter if there is an exception or not
for	To create a for loop
from	To import specific parts of a module
global	To declare a global variable
in	To check if a value is present in a list, tuple, etc.
is	To test if two variables are equal
lambda	To create an anonymous function
None	Represents a null value
nonlocal	To declare a non-local variable
not	A logical operator
or	A logical operator
pass	A null statement, that will do nothing
raise	To raise an exception
return	To exit a function and return a value
True	Boolean value, result of comparison
try	To make a try...except statement
while	To create a while loop
with	Used to simplify exception handling
yield	To end a function, returns a generator

https://www.w3schools.com/python/python_ref_keywords.asp

- False True None
- and or not is
- for in if elif else while continue break pass try except raise assert yield
- def lambda return global nonlocal
- class del
- import from as
- with
- async await

SCALAR OBJECTS

- `int` – represent **integers**, ex. `5` (어떤 정밀도를 지시하지는 않는다!)
- `Float` – represent **real numbers**, ex. `3.27`
- `Bool` – represent **Boolean** values `True` and `False`
- `None type` – **special** and has one value, `None`
- Can use `type()` to see the type of an object

```
>>> type(5)
int
>>> type(3.0)
float
```

what you write into the Python shell

what shows after hitting enter

- Text Type: str
- Numeric Types: int, float, complex
- Sequence Types: list, tuple, range
- Mapping Type: dict
- Set Types: set, frozenset
- Boolean Type: bool
- Binary Types: bytes, bytearray, memoryview

BOOL

- 논리값을 나타내는 자료형
- False와 True값 중 하나를 가짐
- a = False와 같이 이용
- True와 False가 아닌 값을 Bool로 참, 거짓을 판단하는 상황에선 일반적으로 0 및 공백이 False, 0이 아닌 값을 True로 인식함
- ex) a=1

```
if a:
    실행됨
```

a=0

```
if a:
    실행되지 않음
```

TYPE 함수와 TYPE 변환

- type(변수) : 변수의 type을 출력하는 함수. ‘int’ ‘float’ ‘str’과 같이 어떤 타입인지 나타낸다. string으로 출력하는 것이 아니라 type이라는 자료형을 나타내는 자료형으로 출력됨
- 파이썬 기본 함수 가운데 str(), int() 등을 활용해 자료형을 바꿀 수 있다.
- 두개의 string을 + 로 연결하면, concatenate 가 된다.

ex) a = 3 -> print(str(a)+’seconds’)

s = ‘3’ -> a = a+int(s)

```
>>> a = 1
>>> type(a)    -> int
   >>> a = "Hello"
   >>> type(a)   -> string
   >>> type(1.0)  -> float
```

FLOAT

- 실수를 표현하는 변수 (double precision 64bit)
- f = 1.0과 같은 방식으로 선언하여 사용할 수 있음
- f = 1e0(1.0*10^0)과 같은 방식으로도 마찬가지로 사용 가능
- round(a) : 반올림하여 정수를 반환
- abs(a) : 절댓값을 반환

```
>>> 3/4 0
>>> 3/4. 0.75
```

TYPE CONVERSIONS (CAST)

- can **convert object of one type to another**
- `float(3)` converts integer `3` to float `3.0`
- `int(3.9)` truncates float `3.9` to integer `3`

질문
int(-3.9)?

COMPLEX

- 복소수를 표현하는 변수
- c = 1+0j 와 같이 평범하게 복소수 적듯이 선언
- c.conjugate()와 같은 복소수에 사용할 수 있는 함수 (method)가 별도로 존재함.

```
a= 1+3j
b= 2-2j
print(a.real, a.imag)
print(a.conjugate())
print(abs(a))
print(a*b)
-----
```

```
1.0 3.0
(1-3j)
3.1622776601683795
(8+4j)
```

```
a= -1
b= 2-2j
print(a.real, a.imag)
print(a.conjugate())
print(abs(a))
print(a*b)

-1 0
-1
1
(-2+2j)
```

```
>>> complex(3, 2) == 3 + 2j
True
```

```
print(type(1+j))
---------------------------------------------------------------------------
NameError                 Traceback (most recent call last)
<ipython-input-11-34d0241085a0> in <module>
----> 1 print(type(1+j))

NameError: name 'j' is not defined
```

ISINSTANCE(OBJECT, TYPE)

서울대학교 SEOUL NATIONAL UNIVERSITY

코드 8-15 한 객체는 동시에 여러 클래스의 인스턴스일 수 있다

```
>>> isinstance(1789, int)     # 정수의 인스턴스인가?
True

>>> isinstance(1789, float)   # 실수의 인스턴스인가?
False

>>> isinstance(1789, object)  # object의 인스턴스인가?
True
```

1789를 isinstance() 함수로 확인해보니, int의 인스턴스이기도 하고 object의 인스턴스이기도 하다. object 클래스는 모든 클래스의 최상위 범주인 특별한 클래스다.

EXPRESSIONS

- **combine objects and operators** to form expressions
- an expression has a **value**, which has a type
- syntax for a simple expression

  `<object> <operator> <object>`

  - `c = a + b`

OPERATORS ON INTS AND FLOATS

- `i+j` → the **sum**
- `i-j` → the **difference**
- `i*j` → the **product**

if both are ints, result is int
if either or both are floats, result is float

- `i/j` → **division** → result is float

i//j integer 로 된 결과 (1.0)

- `i%j` → the **remainder** when `i` is divided by `j`
- `i**j` → `i` to the **power** of `j`

```
a = 3
b = 1.2
print(a//b)
print(type(a//b))
print(a%b)
print(a/b)

2.0
<class 'float'>
0.6000000000000001
2.5
```

SIMPLE OPERATIONS

- parentheses used to tell Python to do these operations first
- **operator precedence** without parentheses
  - **
  - *
  - /
  - + and – executed left to right, as appear in expression

BINDING VARIABLES AND VALUES

- equal sign is an **assignment** of a value to a variable name


```
pi_approx = 22/7
```

- value stored in computer memory
- an assignment binds name to value
- retrieve value associated with name or variable by invoking the name, by typing `pi`

ABSTRACTING EXPRESSIONS

- why **give names** to values of expressions?
- to **reuse names** instead of values
- easier to change code later

```
pi = 3.14159
radius = 2.2
area = pi*(radius**2)
```

PROGRAMMING VS MATH

- in programming, you do not “solve for x”

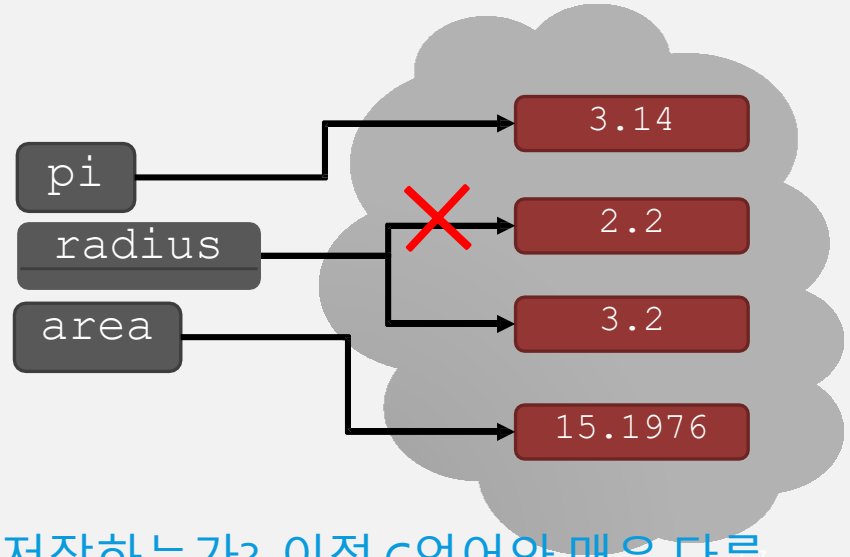
```
pi = 3.14159
radius = 2.2
# area of circle
area = pi*(radius**2)
radius = radius+1
```

an assignment
* expression on the right, evaluated to a value
* variable name on the left
* equivalent expression to `radius = radius + 1` is `radius += 1`

CHANGING BINDINGS

- can **re-bind** variable names using new assignment stat ements
- previous value may still stored in memory but lost the han dle for it (garbage)
- value for area does not change until you tell the co mputer to do the calculation again

```
pi = 3.14
radius = 2.2
area = pi*(radius**2)
radius = radius+1
```



왜 Python은 새로운 번지에 값이 바뀐 radius 를 저장하는가? 이점 C언어와 매우 다름.

ID(OBJECT)

서울대학교 SEOUL NATIONAL UNIVERSITY

1. 아이디 연산자(Identity Operators)

is : 양쪽 Operand가 동일한 Object를 가리키는지 아닌지를 검사합니다.

is not : 양쪽 Operand가 다른 Object를 가리키는지 아닌지를 검사합니다.

동일한 객체 여부를 판별하는 연산자입니다.

id() 함수는 객체를 입력값으로 받아서 객체의 고유값(레퍼런스)을 반환하는 함수입니다.

id는 파이썬이 객체를 구별하기 위해서 부여하는 일련번호입니다. 숫자로서 의미는 없습니다.

id는 동일한 객체 여부를 판별할 때 사용합니다.

In [16]:

```
pi = 3.14
radius = 2.2
area = pi*(radius**2)
radius = radius+1
print(area, radius)
```

15.197600000000003 3.2

In [21]:

```
pi = 3.14
radius = 2.2
print("id_radius",id(radius))
area = pi*(radius**2)
print("id_area", id(area))
radius = radius+1
print("id_radius", id(radius))
b_area=area
print("id_b_area",id(b_area))
area = area + 1.0
print("id_area", id(area))
print(area, radius)
```

```
id_radius 2319867072336
id_area 2319877222960
id_radius 2319876812272
id_b_area 2319877222960
id_area 2319877222928
16.1976 3.2
```

In [25]:

```
sum = 0.0
print("-1",id(sum))
for i in range(3):
        sum +=i**2
        print(i, id(sum))
print("sum=", sum)
```

```
-1 2319876808816
0 2319876811184
1 2319867071440
2 2319876811184
sum= 5.0
```

Garbage collection

STRINGS

- letters, special characters, spaces, digits
- enclose in **quotation marks(“) or single quotes(‘)**

```
hi = "hello there"
print(“This is Tom’s car”)
print(‘He said, “I will do it.”’)
```

- **concatenate** strings

```
name = "ana"

greet = hi + name

greeting = hi + " " + name
```

- do some **operations** on a string as defined in Python docs

```
silly = hi + " " + name * 3
```

STRING(STR)

서 울 대 학 교
SEOUL NATIONAL UNIVERSITY

- 문자열을 표현하는 자료형.
- s = ‘hello world’의 방식으로 선언함.
- s = ‘1.0’ 혹은 s=str(1.0)등의 방식으로 다른 자료형을 string으로 선언할 수 있음
- ‘” 내용 “’ 을 사용하여 “나 ‘, 줄바꿈 등을 포함한 string을 표현할 수 있음
- ex) s = ‘” “hello”

‘world’ ‘”

STRING(STR)

서울대학교 SEOUL NATIONAL UNIVERSITY

- string 길이를 반환하는 len('string')등 수많은 내장함수가 있음
- array의 성질을 가지고 있기 때문에 s[위치]를 통해 string의 원하는 자리에 있는 문자만 가져올 수 있음(차후에 다룰 것)
- ex) s = 'hello world' print(s[1]) -> e가 출력됨

*0,1,2,3...의 숫자를 이용해 앞에서부터의 몇번째인지, -1,-2,-3으로 뒤에서부터 몇번째 위치인지를 가리킴.

기호의 표시 – ESCAPE CODE \ (BACK SLASH)

\n 줄바꿈

\t 문자열 사이에 tab

\\ back slash 1개

\’ 작은 따옴표

\” 큰 따옴표

\a 스피커에서 삑소리

(참고, 점프투파이썬 페이지 48)

STRING ENCODING

- String은 각 글자들의 연속이고, 각 글자는 code로 나타내지는데 일반적으로 UTF-8을 사용한다.

- UTF-8은 유니코드를 위한 가변 길이 문자 인코딩 방식 중 하나로, 켄 톰프슨과 롭 파이크가 만들었다. UTF-8은 Universal Coded Character Set + Transformation Format – 8-bit 의 약자이다.
- 파일 맨 위에 아래와 같이 comment로 적어둔다

```
# -*- coding: utf- -*-
```

ASCII ENCODING (BYTES, BYTEARRAY)

서울대학교 SEOUL NATIONAL UNIVERSITY

- 파이썬에서 b'hello'처럼 ' '(작은따옴표)나 " "(큰따옴표) 앞에 b를 붙이면 바이트(bytes) 객체가 됩니다. 간단하게 'hello'와 문자열 str의 관계라고 생각하면 됩니다.
- >>> x = bytearray(b'hello')
- >>> x[0] = ord('a') # ord는 문자의 ASCII 코드를 반환
- >>> x
- bytearray(b'aello')

INPUT/OUTPUT: PRINT ( )


```
x = 7
x_str = str(x)
print("my favorite number is", x, ".")
print("my favorite number is " + x_str + ".")
print("my favorite number is %d." %x)
print("my favorite number is %s." %x_str)
print("my favorite number is %s." %7)
print("my favorite number is {0}.".format(x))
print("my favorite number is {0}.".format(x_str))
print("my favorite number is {0} and {1}.".format(x, x_str))
print("my favorite number is {x} and {x_str}.".format(x=x, x_str=x_str))
```

```
my favorite number is 7 .
my favorite number is 7.
my favorite number is 7.
my favorite number is 7.
my favorite number is 7.
my favorite number is 7.
my favorite number is 7.
my favorite number is 7 and 7.
my favorite number is 7 and 7.
```

print('This is string :', var_char, 'This is number :' ,var_num,var_num2)

콤마(,)로 구분하면, 문자, 숫자 상관없이 일정공백을 사이에 두고 출력됩니다.

print('variable',variable,end='')

한루프당 출력결과가 많을때, 혹은 여러가지 값을 찍어보고 싶을 때 프린트문 끝에 end='' 이라는 옵션을 추가해주면, 그다음 출력문이, 오른쪽에 이어서 표시됩니다. 공백이나 콤마등으로 구분을 주고 싶으면 end=' ' 나 end=','처럼 따옴표안에 문자열을 지정해주면 됩니다.

FORMAT 을 주어서 PRINT

- 실습

STR.ZFILL(WIDTH)

- str.zfill(width)
- Parameters
  - width − This is final width of the string. This is the width which we would get after filling zeros.

```
a = "hello"
b = "welcome to the jungle"
c = "10.000"

print(a.zfill(10))
print(b.zfill(10))
print(c.zfill(10))

--------------------------

00000hello
welcome to the jungle
000010.000
```

```
print('{0} and {1}'.format('spam', 'eggs'))
print('{1} and {0}'.format('spam', 'eggs'))
num1=11;num2=12; num3=13
print('{0} {1}'.format(num1,num2))
print('{0}{1}{2}'.format(num1, num2, num3))
```

```
spam and eggs
eggs and spam
11 12
111213
```

```
var_char= 'test'
var_num = 10
float_num=5e-3
print('Text %s'%var_char, 'Number %d'%var_num,'Number %f'%float_num, '₩n', end='')
print('Number %d'%var_num)
```

```
Text test Number 10 Number 0.005000
Number 10
```

INPUT/OUTPUT: `INPUT("")`

- prints whatever is in the quotes
- user types in something and hits enter
- binds that value to a variable

```
text = input("Type anything... ")
print(5*text)
```

- `input` **gives you a string** so must cast if working with numbers

```
num = int(input("Type a number... "))
print(5*num)
```

>>> a = 'I Love Python'

>>> len(a)

13

min(), max() : 문자열 내 문자, 혹은 숫자의 최소값, 최대값 (알파벳 순서, 숫자 순서 기반)

max(str), min(str) : Returns the max, min alphabetical character from the string str

>>> d = 'abc'

>>> f = '123'

>>>

>>> min(d)

'a'

https://rfriend.tistory.com/327

count() : 문자열 안에서 매개변수로 입력한 문자열이 몇 개 들어있는지 개수를 셈

(begin, end 위치 설정 가능)

count() : Counts how many times str occurs in string

>>> a = 'I Love Python'

>>> a.count('o')

2

find() : 문자열에 매개변수로 입력한 문자열이 있는지를 앞에서 부터 찾아 index 반환, 없으면 '-1' 반환

find() : Search forwards, Determine if str occurs in string and return the index

>>> a = 'I Love Python'

>>> a.find('o')

3

실습

- Format 을 주어서 프린트하기
- 순서대로 두 값을 입력받아 원하는 사칙연산 결과를 print하기
- input함수와 print함수 활용

- Python built-in 함수

```
eval('3+4*2')
```

과제

Find out the total money to return

The interest rate(%)

The period

The money to borrow

- #magnitude square of two numbers
- x_real, x_imag = input("type in a complex number").split()

PYTHON IDE 깔기

서 울 대 학 교
SEOUL NATIONAL UNIVERSITY