

AI 이노베이션스퀘어

1 Python 소개, 기본 문법

2 Python 자료형(숫자형, 문자형)

3 Python 자료형 (Sequence형, Hash/Mapping형)

4 Python 제어 구문(조건, 반복, 예외처리)

5 Python 함수(정의, 호출, 반환)



- 과거의 컴퓨터
 - Compute: 계산하다
 - Computer: 계산하는 사람 또는 기계

NASA 최초의 우주 궤도 비행 프로젝트



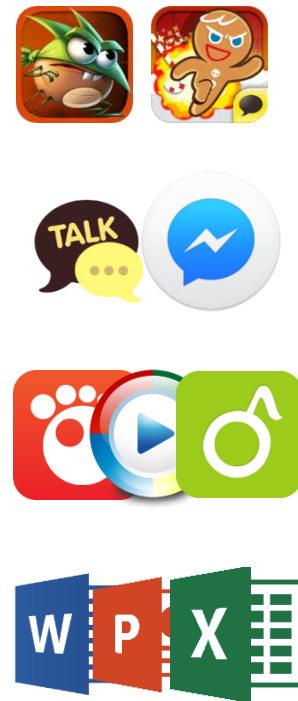
- 현재의 컴퓨터
 - 하드웨어 + 소프트웨어

```
//Scroll to solution//  
function ScrolltoAnchor($classItem, s  
$classItem.bind('click', function  
var anchor = $(this);  
{ (mobile) {  
  $('html, body').stop().an  
    scrollTop: $(anchor.a  
  }, 1000);  
} else {  
  $('html, body').stop().an  
    scrollTop: $(anchor.a  
  }, 1000);  
}
```

0/1 기반
프로그램/데이터
(Software)



전자 기기
(Hardware)



⋮
다양한 서비스

- 소프트웨어란?
 - 컴퓨터 하드웨어에게 필요한 기능을 수행할 수 있도록 하는 **프로그램**
- 프로그램이란?
 - 특정 목적을 위해 **프로그래밍**을 통해 만든 명령어들의 모임
- 프로그래밍이란?
 - 수식이나 작업을 컴퓨터가 수행하기 알맞게 정리해 순서를 정하고 컴퓨터 명령 코드인 **프로그래밍 언어**를 이용하여 **코딩**하는 작업을 총칭하여 쓰는 말
 - 코딩(Coding)이란?
 - **프로그래밍 언어를 사용하여 프로그램의 코드를 작성하는 일**
 - 알고리즘을 프로그래밍 언어로 변환하는 작업



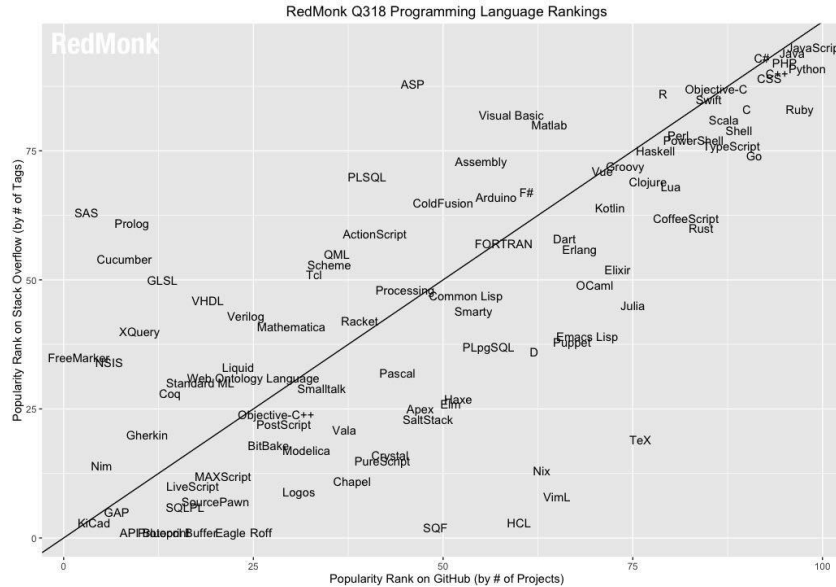
- 프로그래밍 언어란?
 - 사람과 컴퓨터 사이에 존재하는 일종의 **커뮤니케이션 수단**.



Worldwide, Sept 2018 compared to a year ago:

Rank	Change	Language	Share	Trend
1	↑	Python	24.58 %	+5.7 %
2	↓	Java	22.14 %	-0.6 %
3	↑	Javascript	8.41 %	+0.0 %
4	↓	PHP	7.77 %	-1.4 %
5		C#	7.74 %	-0.4 %
6		C/C++	6.22 %	-0.8 %
7		R	4.04 %	-0.2 %
8		Objective-C	3.33 %	-0.9 %
9		Swift	2.65 %	-0.9 %
10		Matlab	2.1 %	-0.3 %

<http://pypl.github.io/PYPL.html>



<https://redmonk.com/sograzy/2018/08/10/language-rankings-6-18/>

Sep 2018	Sep 2017	Change	Programming Language	Ratings	Change
1	1		Java	17.436%	+4.75%
2	2		C	15.447%	+8.06%
3	5	↑	Python	7.653%	+4.67%
4	3	↓	C++	7.394%	+1.83%
5	8	↑	Visual Basic .NET	5.308%	+3.33%
6	4	↓	C#	3.295%	-1.48%
7	6	↓	PHP	2.775%	+0.57%
8	7	↓	JavaScript	2.131%	+0.11%
9	-	↑↑	SQL	2.062%	+2.06%
10	18	↑↑	Objective-C	1.509%	+0.00%

<https://www.tiobe.com/tiobe-index/>

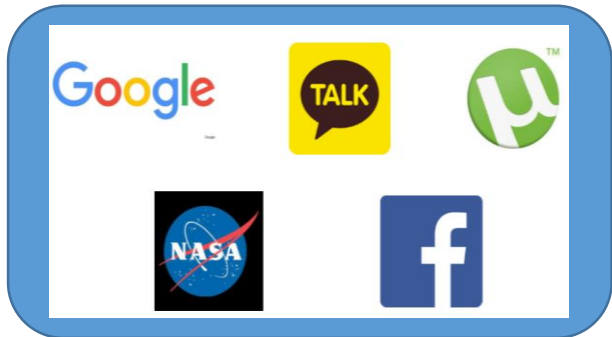
Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C++		99.7
3. Java		97.5
4. C		96.7
5. C#		89.4
6. PHP		84.9
7. R		82.9
8. JavaScript		82.6
9. Go		76.4
10. Assembly		74.1

<https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>





왜 파이썬을 쓰는 거지?



파이썬을 사용하는 기업들

<https://docs.python.org/ko/3/tutorial/index.html>
<https://docs.python.org/ko/3/tutorial/appetite.html>

- 배우기 쉽다.
- 간결한 문법과 풍부한 표준 라이브러리를 가지고 있다.
- 자료 구조들과 객체 지향 프로그래밍에 대해 간단하고도 효과적인 접근법을 제공
- 인터프리터적인 특징들은 대부분 플랫폼과 다양한 문제 영역에서 스크립트 작성과 빠른 응용 프로그램 개발에 이상적인 환경을 제공
- 풍부한 표준 라이브러리는 소스나 바이너리 형태로 파이썬 웹 사이트, <https://www.python.org/>에서 무료로 제공, 자유롭게 배포가능
- C 나 C++ (또는 C에서 호출 가능한 다른 언어들)로 구현된 함수나 자료 구조를 쉽게 추가할 수 있음

- 웹 앱에서 인공지능까지 인기가 급상승하며 주목받는 언어
- 데이터 분석에 장점을 가진 스크립트 언어, '데이터 사이언티스트'라는 직종이 인기가 있음
- 해외에서는 웹 어플리케이션의 개발 언어로도 많이 사용되고 있으며, Python 프로그래머의 평균 연봉이 높은 것이 화제가 되기도 한다.
- 최근에는 기계학습 등 인공지능의 개발에도 많이 사용되고 있다. 버전 2.x와 3.x는 일부 호환이 되지 않지만 두 버전 모두 이용자가 많다.

Python

파이썬

웹 앱에서 인공지능까지
인기가 급상승하며 주목받는 언어

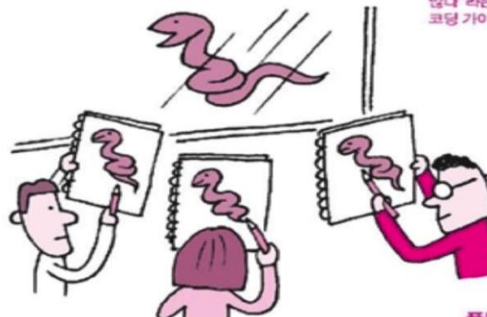
탄생
1991년
만든 사람
Guido van Rossum
주요 용도
웹 앱, 데이터 분석,
인공지능
분류
정지형 · 함수형 · 객체지
향형 / 인터프리터

이런 언어

데이터 분석에 장점을 가진 스크립트 언어로 '데이터 사이언티스트'라는 직종에 인기가 있다. 해외에서는 웹 애플리케이션의 개발 언어로도 많이 사용되고 있으며, Python 프로그래머의 평균 연봉이 높은 것이 화제가 되기도 한다. 최근에는 기계학습 등 인공지능의 개발에도 많이 사용되고 있다. 버전 2.x와 3.x는 일부 호환이 되지 않지만 두 버전 모두 이용자가 많다.

인텐트가 중요

많은 언어가 '{''}' 등으로 블록 구조를 표현하는 반면, Python에서는 인덴트(들여 쓰기)로 표현한다. "코드는 쓰는 것보다 읽는 것이 더 많다"라는 의미에서 PEP8이라는 코딩 가이드도 제공되고 있다.



다양한 구현이 있다

원래의 처리계인 'CPython'뿐만 아니라 'Jython'나 'PyPy', 'Cython'나 'IronPython' 등 다양한 구현이 있으며, 각각 특징이 있다.

풍부한 통계 라이브러리

데이터 분석과 기계학습에 사용할 수 있는 라이브러리가 풍부하게 갖추어져 있으며 무료로 사용할 수 있다. 'NumPy'나 'Pandas', 'matplotlib' 등이 특히 유명하다.

Column

The Zen of Python

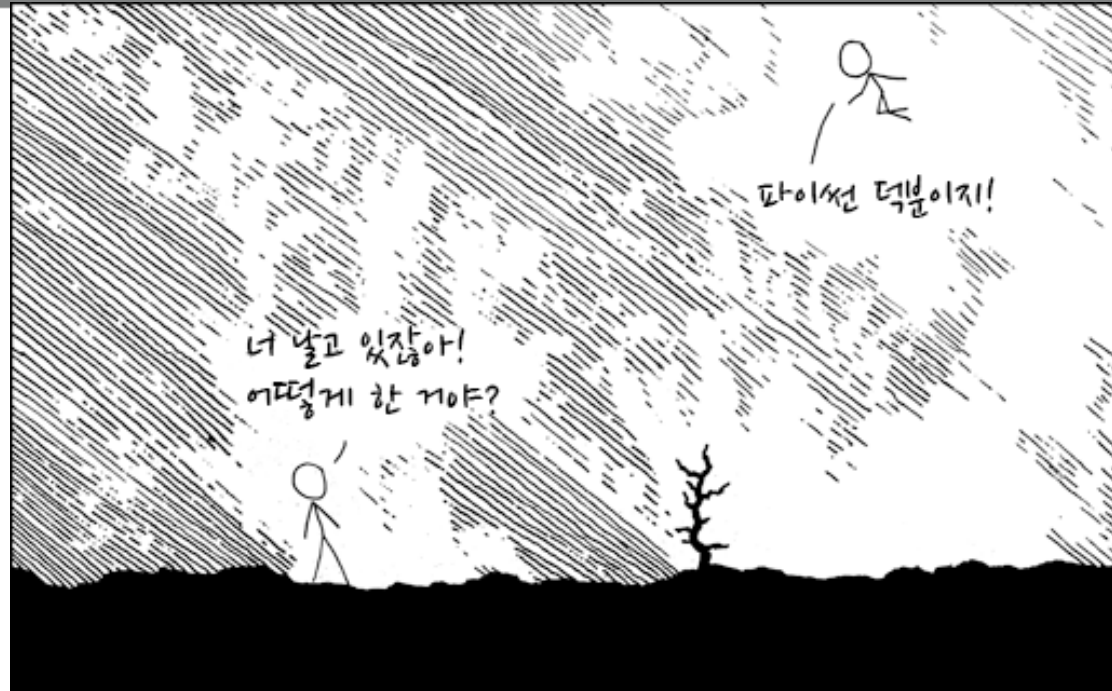
Python 프로그래머가 가져야 할 마음가짐이 정리된 말. Python으로 소스코드를 작성할 때 '단순'과 '가독성'을 실현하기 위한 19개의 문장으로 구성되어 있다.



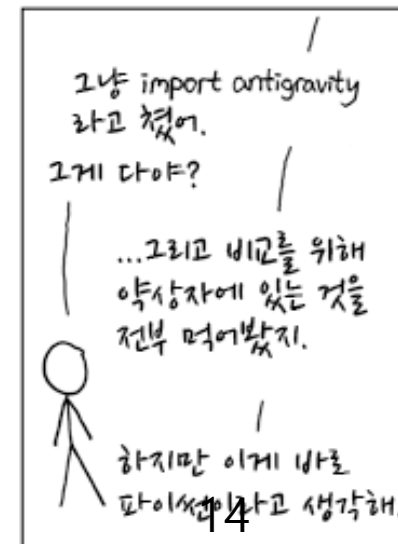
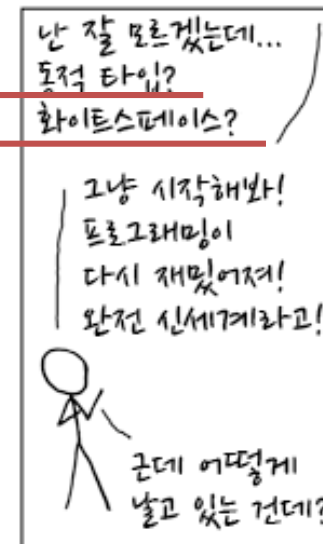
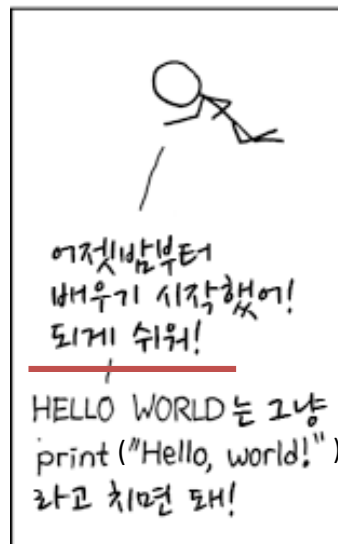
Life is too short, You need  python™

- **인터프리터**
 - 프로그래밍 언어로 작성된 소스코드를 바로 실행할 수 있는 프로그램 또는 환경
 - 컴파일러(원시코드를 기계어로 해석해주는 프로그램 또는 환경)언어보다 속도가 느린편
 - C/C++보다 5배~80배, 자바 5~20배 차이
- **생산성**
 - 간결한 문법과 풍부한 라이브러리 제공 등으로 개발하는데 있어 생산성이 뛰어남
- **멀티패러다임지원**
 - 함수형 패러다임과 객체지향 패러다임 두가지 모두를 지원(다양한 방식, 다양한 관점으로 프로그래밍할 수 있으며, 수많은 기법과 디자인 패턴을 접목시킬 수 있음)
- **Glue Language**
 - 다른 언어와의 호환성이 높음, C언어로 구현한 CPython, Java로 구현한 Jython, 파이썬으로 파이썬을 구현한 PyPy 등 대체 가능한, 특정 플랫폼에 맞춰진 구현체를 가지고 있음
- **다양한 라이브러리 제공**
 - 수많은 표준 라이브러리를 기본으로 탑재, pip를 통해 손쉽게 받을 수 있음
- **General Purpose**
 - `import antigravity`

In [] : `import antigravity`



C.f)
Domain-specific



Python 개발환경



2020년까지 bugfix만 지원 하다가 종료 예정

<https://www.python.org/dev/peps/pep-0373/>

REPL vs IDE vs Text Editor

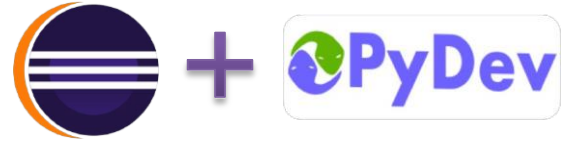
don't need
to use
the `print()` function

https://en.wikipedia.org/wiki/Read-eval-print_loop

REPL



IDE



Text Editor

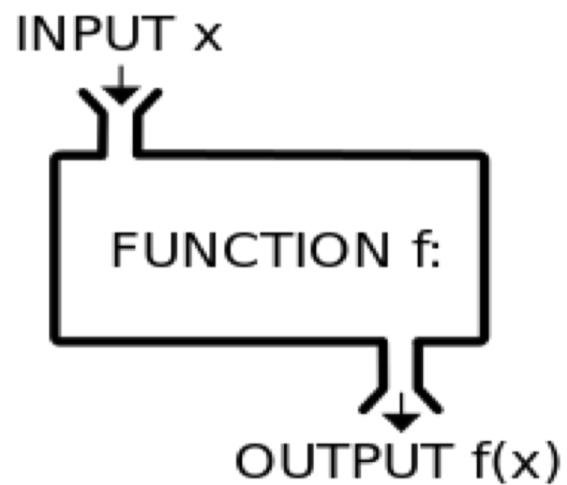


vi, emacs...

프로그래밍의 구성요소

프로그래밍

문법(키워드, 식, 문)을 이용해서
값을 입력받고, 계산/변환하고, 출력하는 흐름을 만드는 일



In []: `import this`

- 아름다운 것이 보기 싫은 것보다 좋다.
- 명시적인 것이 암묵적인 것보다 좋다.
- 간단한 것이 복잡한 것보다 좋다.
- 복잡한 것이 복잡한 것보다 좋다.
- 수평한 것이 중첩된 것보다 좋다.
- 희소한 것이 밀집된 것보다 좋다.
- 가독성이 중요하다.
- 규칙을 무시할 만큼 특별한 경우는 없다.
- 하지만 실용성이 순수함보다 우선한다.
- 예러가 조용히 넘어가서는 안된다.
- 명시적으로 조용히 만든 경우는 제외한다.
- 모호함을 만났을 때 추측의 유혹을 거부해라.
- 하나의 — 가급적 딱 하나의 — 확실한 방법이 있어야 한다.
- 하지만 네덜란드 사람(귀도)이 아니라면 처음에는 그 방법이 명확하게 보이지 않을 수 있다.
- 지금 하는 것이 안하는 것보다 좋다.
- 하지만 안하는 것이 이따금 지금 당장 하는 것보다 좋을 때가 있다.
- 설명하기 어려운 구현이라면 좋은 아이디어가 아니다.
- 설명하기 쉬운 구현이라면 좋은 아이디어다.
- 네임스페이스는 아주 좋으므로 더 많이 사용하자!

문법(키워드, 식, 문)을 이용해서
값을 입력받고, 계산/변환하고, 출력하는 흐름을 만드는 일

keyword


```
In [ ]: import keyword  
keyword.kwlist
```

- 예약어(reserved word, keyword)

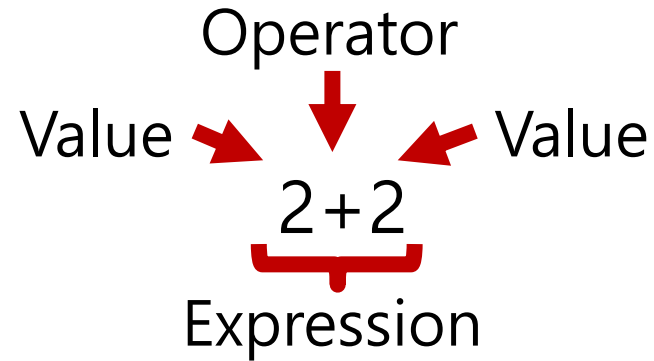
False**None****True****and****as****assert****break****class****continue****def****del****elif****else****except****finally****for****from****global****if****import****in****is****lambda****nonlocal****not****or****pass****raise****return****try****while****with****yield**

문법(키워드, 식, 문)을 이용해서
값을 입력받고, 계산/변환하고, 출력하는 흐름을 만드는 일

Expression

- 표현식(expression) = 평가식 = 식
 - 값
 - 값들과 연산자를 함께 사용해서 표현한 것
 - 이후 “평가”되면서 하나의 특정한 결과값으로 축약
 - 수
 - $1 + 1$
 - » $1 + 1$ 이라는 표현식은 평가될 때 2라는 값으로 계산되어 축약
 - 0과 같이 값 리터럴로 값을 표현해놓은 것
 - 문자열
 - 'hello world'
 - "hello" + ", world"
 - 궁극적으로 “평가”되며, 평가된다는 것은 결국 하나의 값으로 수렴한다는 의미
 - python에서는 기본적으로 left-to-right로 평가

■ 수식 표현



2+2	Addition
2-2	Subtraction
2*2	Multiplication
2/2	Division
2%2	Modulus
2**2	Square

- +, -, *, / and % are called **operators**.
- * sign is called an **asterisk**.

```
In [ ] : 2+2
```

```
Out[ ] :
```

```
In [ ] : 5-7
```

```
Out[ ] :
```

```
In [ ] : 8*9
```

```
Out[ ] :
```

```
In [ ] : 10/2
```

```
Out[ ] :
```

```
In [ ] : 3**2
```

```
Out[ ] :
```

```
In [ ] : 2 + 2
```

```
Out[ ] :
```

- 문자열 표현
 - 문자열 : 텍스트의 묶음
 - 문자열은 홑 따옴표 또는 쌍 따옴표로 묶어준다.
 - 예: 'Good', "good"
 - 문자열 + 문자열
 - 문자열도 숫자처럼 덧셈 연산으로 더할 수 있음
 - 여기서, 문자열의 덧셈은 문자열의 나열을 의미함

```
In [ ] : 'Hello' + 'World!'
```

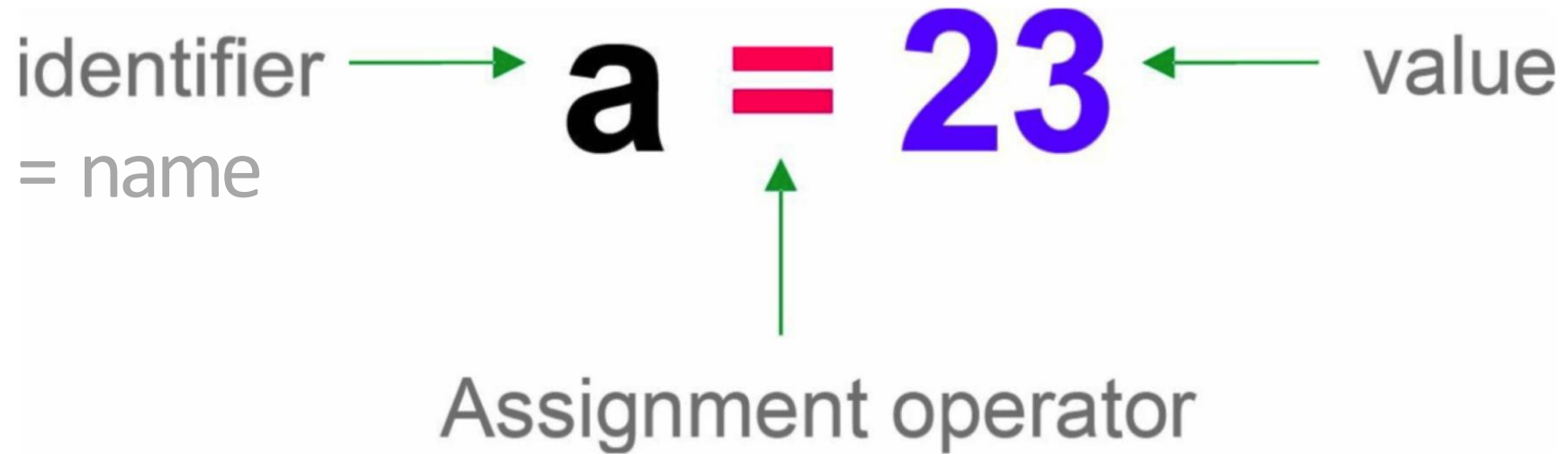
```
Out[ ] :
```

```
In [ ] : 'Hello ' + 'World!'
```

```
Out[ ] :
```

Assignment

binding



■ 변수의 일반적 의미

- 아직 알려지지 않거나 어느 정도까지만 알려져 있는 양이나 정보에 대한 상징적인 이름
 - 대수학 : 수식에 따라서 변하는 값
 - 상수 : 변하지 않는 값

■ 프로그래밍에서의 변수

- 값을 기억해 두고 필요할 때 활용할 수 있음
 - 중간 계산값을 저장하거나 누적 등

■ Python

- 값(객체)을 저장하는 메모리 상의 공간을 가르키는(object reference) 이름
 - A variable is a name for an object within a scope
 - **None** is a reference to nothing
- Python은 모든 것이 객체이므로 변수보다는 **식별자**로 언급. 변수로 통용해서 사용하기도 함
 - 변수, 함수, 사용자 정의 타입 등에서 다른 것들과 구분하기 위해서 사용되는 변수의 이름, 함수의 이름, 사용자 정의 타입의 이름 등 '이름'을 일반화 해서 지칭하는 용어
- 변수의 경우, 선언 및 할당이 동시에 이루어져야 함
- case-sensitive names
 - E.g., functions, classes and variables

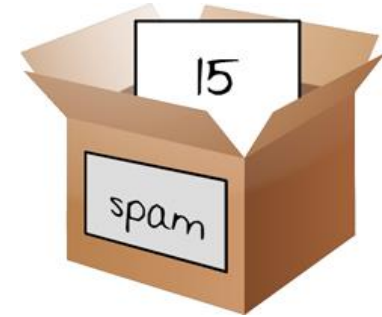
- 변수(Variables)

- 변수는 어떤 값을 갖고 있는 박스와 같다.
- = (대입 연산자)를 이용하여 값, 또는 연산 결과를 저장한다.
- 예를 들어, 변수 spam에 값 15를 저장하려면

```
In [ ] : spam = 15
```

```
In [ ] : spam
```

```
Out[ ] :
```



- 변수에 값을 저장하면 메모리에 저장하게 되는데, 메모리는 고유의 주소값을 가짐
- 주소값을 확인하려면 id()함수를 사용

```
In [ ] : id(spam)
```

```
Out[ ] :
```

- %whos : 현재 메모리에 올라간 변수들의 목록을 확인

```
In [ ] : a = 10  
         b = 'hi'  
         c = 3.14
```

```
In [ ] : %whos
```

Variable	Type	Data/Info
a	int	10
b	str	hi
c	float	3.14

- 변수를 더 이상 사용하고 싶지 않다면 del

```
In [ ] : del a
```

```
In [ ] : a
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-4-3f786850e387> in <module>  
----> 1 a  
  
NameError: name 'a' is not defined
```

```
In [ ] : %whos
```

Variable	Type	Data/Info
b	str	hi
c	float	3.14

- 변수 이름을 지을 때...
 - 의미 있는 이름을 사용(권장)
 - 소문자와 대문자는 서로 다르게 취급
 - 변수의 이름은 **영문자와 숫자, 밑줄(_)**로 이루어짐
 - 변수의 이름 중간에 **공백이 들어가면 안됨**. 단어를 구분하려면 밑줄(_)을 사용함
 - 숫자로 시작할 수 없음
 - 낙타체(권장)
 - 변수의 첫 글자는 소문자, 나머지 단어의 첫 글자는 대문자로 적는 방법
 - 예) myNewCar

- Quiz
 - 변수 이름으로 사용하기에 적절할까요?

sum

_count

number_of_pictures

King3

2nd_base

money#

while

- Quiz
 - 변수 이름으로 사용하기에 적절할까요?

sum	(O) 영문 알파벳 문자로 시작
_count	(O) 밑줄 문자로 시작할 수 있다.
number_of_pictures	(O) 중간에 밑줄 문자를 넣을 수 있다.
King3	(O) 맨 처음이 아니라면 숫자도 넣을 수 있다.
2nd_base	(X) 숫자로 시작할 수 없다.
money#	(X) #과 같은 기호는 사용할 수 없다.
while	(X) 예약어는 사용할 수 없다.

Object Reference

C



```
int a = 1;
```



```
a = 2;
```

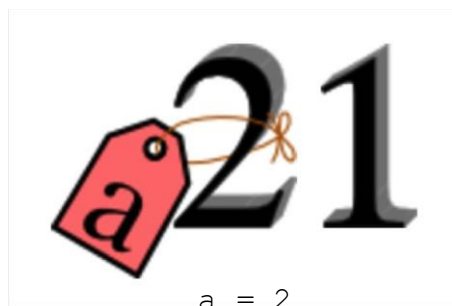


```
int b = a;
```

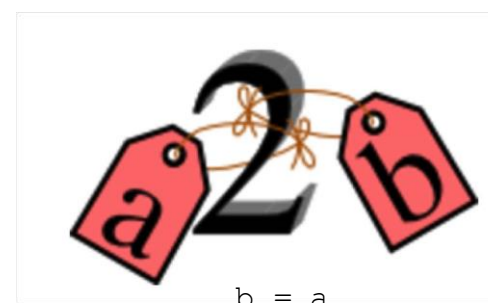
Python



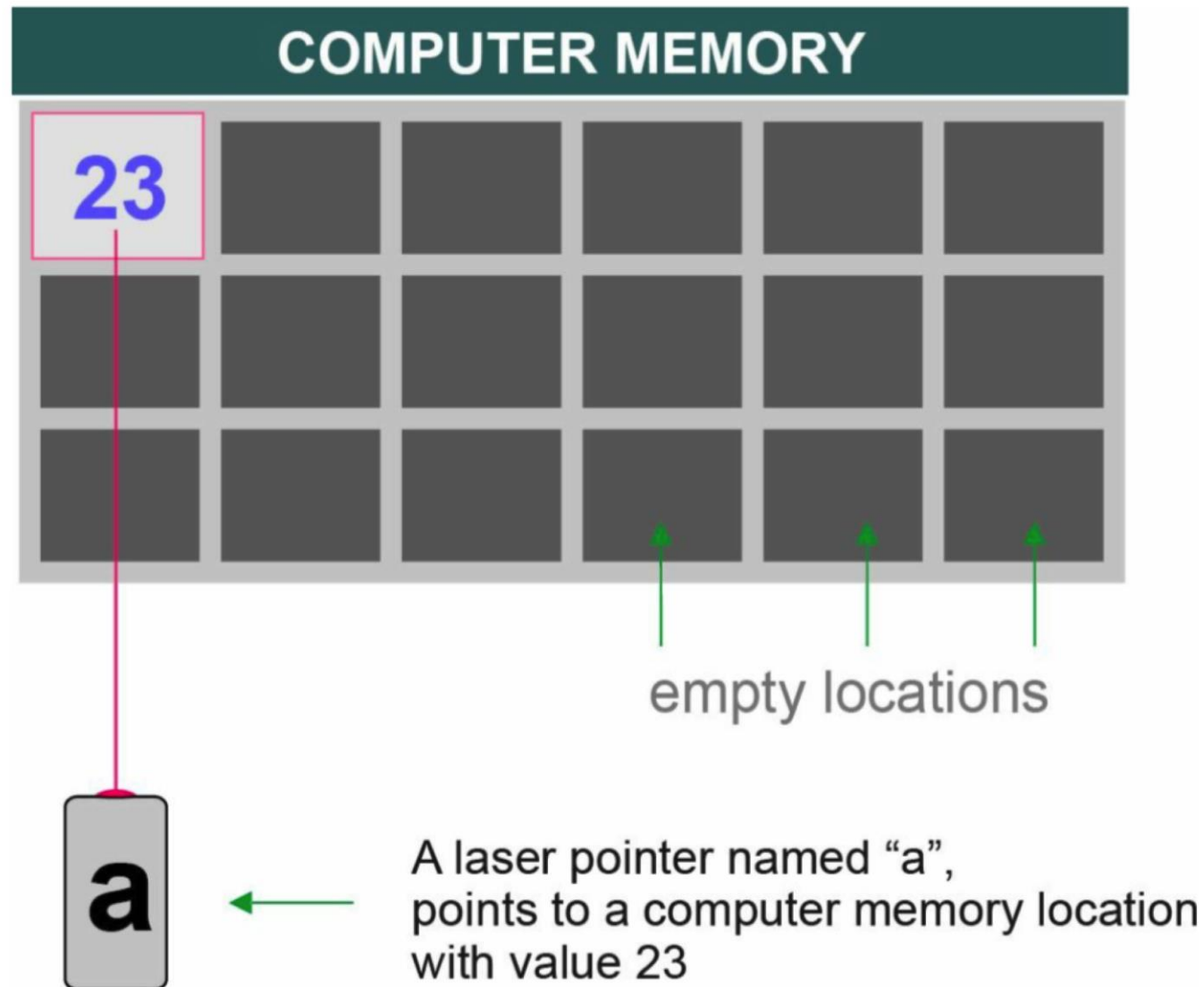
```
a = 1
```



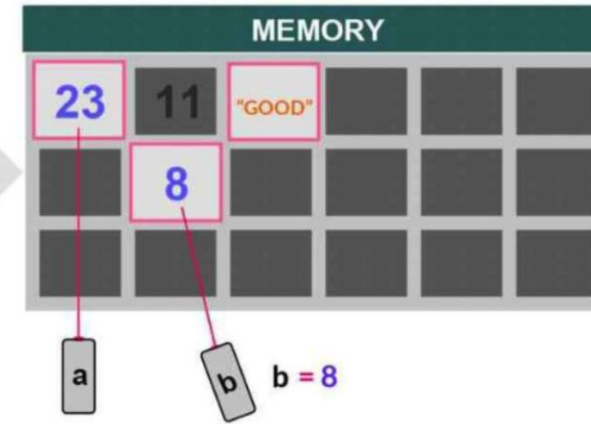
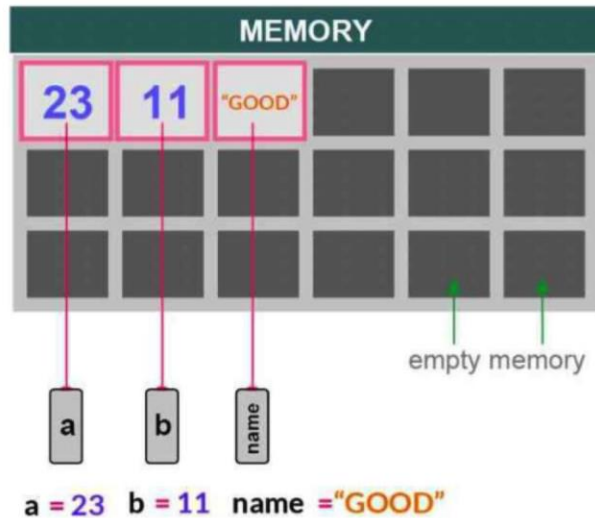
```
a = 2
```



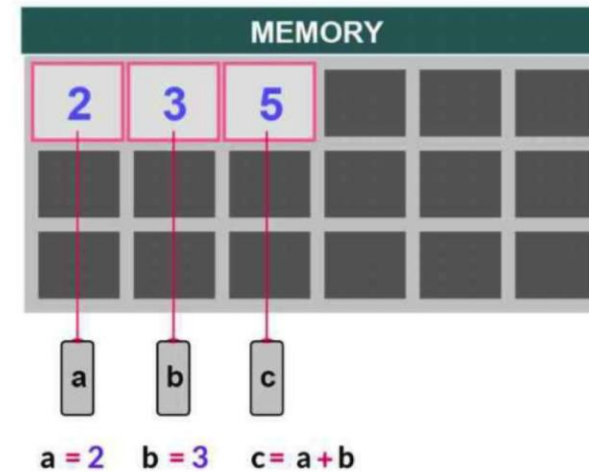
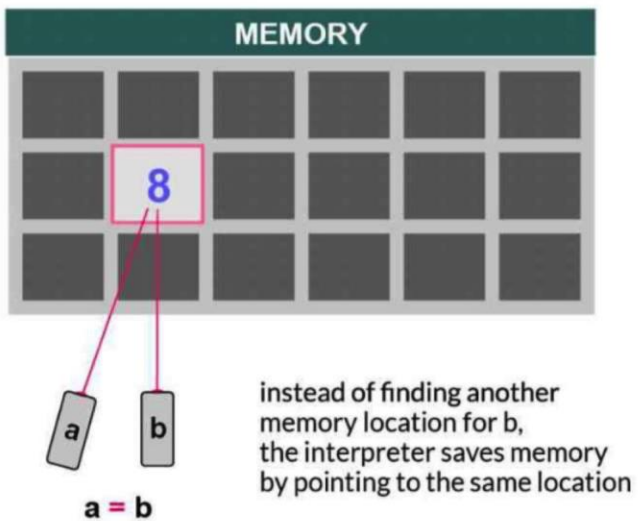
```
b = a
```



lets change the value of b from 11 to 8



Because numbers are immutable, "b" changes location to the new value.
When there is no reference to a memory location the value fades away and the location is free to use again.
This process is known as garbage collection



Variable Creation

Code

```
status = "off"
```

What Computer Does

Variables

Objects

Id: 2e6a	Id
"off"	Value
Type: String	Type

Step 1: Python creates an object

```
status = "off"
```

Variables

Objects

status →

Id: 2e6a	Id
"off"	Value
Type: String	Type

Step 2: A variable is created

Rebinding Variables

Code

```
a = 400
```

What Computer Does

Variables

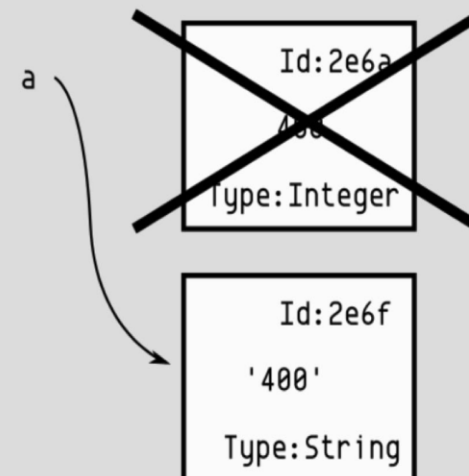
Objects



```
a = '400'
```

Variables

Objects



Old Object is Garbage Collected

- Quiz

```
In [ ] : spam = 15
```

```
In [ ] : spam + 5
```

```
Out[ ] :
```

```
In [ ] : spam = 3
```

```
In [ ] : spam + 5
```

```
Out[ ] :
```

```
In [ ] : spam = 5 + 7
```

```
In [ ] : spam
```

```
Out[ ] :
```

```
In [ ] : spam = 15
```

```
In [ ] : spam + spam
```

```
Out[ ] :
```

```
In [ ] : spam - spam
```

```
Out[ ] :
```

- Quiz

```
In [ ] : spam = 15
In [ ] : spam = spam + 5
In [ ] : spam
Out[ ] :
```

```
In [ ] : spam = 15
In [ ] : spam = spam + 5
In [ ] : spam = spam + 5
In [ ] : spam = spam + 5
Out[ ] :
```


- 문자열
 - 문자열은 문자뿐만 아니라 공백, 숫자, 특수문자를 포함할 수 있다.

```
In [ ] : spam1 = 'hello'
```

```
In [ ] : spam2 = 'Hi there!'
```

```
In [ ] : spam3 = 'KITTENS'
```

```
In [ ] : spam4 = '7 apples, 14 oranges, 3 lemons'
```

```
In [ ] : spam5 = 'Anything not pertaining to elephants is irrelephant.'
```

```
In [ ] : spam6 = 'O*&#wY%*&OCfsdYO*& gfC%YO'
```

- Quiz

```
In [ ] : spam = 'Hello'
```

```
In [ ] : fizz = 'World!'
```

```
In [ ] : 'spam' + 'fizz'
```

```
Out[ ] :
```

```
In [ ] : spam = 'Hello'
```

```
In [ ] : fizz = 'World!'
```

```
In [ ] : spam + 'fizz'
```

```
Out[ ] :
```

```
In [ ] : spam = 'Hello'
```

```
In [ ] : fizz = 'World!'
```

```
In [ ] : 'hi' + spam + fizz
```

```
Out[ ] :
```

- Quiz

```
In [ ] : 'Hello' + '5'
```

```
Out[ ] :
```

```
In [ ] : '5' + '5'
```

```
Out[ ] :
```

```
In [ ] : spam = 5
```

```
In [ ] : 'Hello' + spam
```

```
Out[ ] :
```

```
In [ ] : spam = 'Hello'
```

```
In [ ] : spam = 'World!'
```

```
In [ ] : spam + spam
```

```
Out[ ] :
```

```
In [ ] : a=1  
        a=b=c=a
```

```
In [ ] : lhs, rhs = lhs, rhs
```

```
In [ ] : counter = 0  
        counter += 1
```

```
In [ ] : a, b = 1, 2
```

```
In [ ] : a
```

```
Out[ ] :
```

```
In [ ] : b
```

```
Out[ ] :
```

```
In [ ] : c
```

```
Out[ ] :
```

Comments

■ 주석

- 해시 문자(#)로 시작하고 줄의 끝까지 이어집니다.

- 주석은 줄의 처음에서 시작할 수도 있고, 공백이나 코드 뒤에 나올 수도 있음
- 하지만 문자열 리터럴 안에는 들어갈 수 없습니다. 문자열 리터럴 안에 등장하는 해시 문자는 주석이 아니라 해시 문자일 뿐
- 주석은 코드의 의미를 정확히 전달하기 위한 것이고, 파이썬이 해석하지 않는 만큼, 예를 입력할 때는 생략해도 됨

■

- Line-based, i.e., but independent of indentation (but advisable to do so)

- There are no multi-line comments

- `''' '''`

■ # -*- coding: <encoding-name> -*-

- https://docs.python.org/3/reference/lexical_analysis.html#encoding-declarations

```
In [6]: #This program says hello and asks for my name  
print ('hello world!')  
print('Input your name.')  
myName = input()  
print ('Welcome, ' + myName + '.')
```

```
hello world!  
Input your name.  
Haesun  
Welcome, Haesun.
```

이렇게 줄 맨 앞에 #이 붙어 있으면
IDLE은 이 줄 전체를 그냥 무시

보통은, 이 코드를 볼 다른 친구,
또는 미래의 나를 위해
이 코드에 대한 설명을 달기 위해 사용

```
In [7]: #This program says hello and asks for my name  
print ('hello world!')  
print('Input your name.')
```

```
hello world!  
Input your name.
```

특정 코드를 잠시 실행시키고 싶지 않을 때도 사용

- python에서는 여러 줄 주석이 없음

Value

문법(키워드, 식, 문)을 이용해서
값을 입력받고, 계산/변환하고, 출력하는 흐름을 만드는 일

Value

- 값에 대한 type이 중요함
- 값에 따라 서로 다른 기술적인 체계가 필요
 - 지원하는 연산 및 기능이 다르기 때문
- 컴퓨터에서는 이진수를 사용해서 값을 표현하고 관리
 - 정확하게 표현하지 못할 수가 있음
 - 숫자형 = numeric
 - 산술 연산을 적용할 수 있는 값
 - 정수 : 0, 1, -1 과 같이 소수점 이하 자리가 없는 수. 수학에서의 정수 개념과 동일. (int)
 - 부동소수 : 0.1, 0.5 와 같이 소수점 아래로 숫자가 있는 수. (float)
 - 복소수 : Python에서 기본적으로 지원
 - 문자, 문자열
 - 숫자 "1", "a", "A"와 같이 하나의 낱자를 문자라 하며, 이러한 문자들이 1개 이상있는 단어/문장과 같은 텍스트
 - Python에서는 낱자와 문자열 사이에 구분이 없이 기본적으로 str 타입을 적용
 - byte, bytearray
 - 불리언 = boolean
 - 참/거짓을 뜻하는 대수값. 보통 컴퓨터는 0을 거짓, 0이 아닌 것을 참으로 구분
 - True와 False 의 두 멤버만 존재 (bool)
 - Python에서는 숫자형의 일부

○ Compound = Container = Collection

- 기본적인 데이터 타입을 조합하여, 여러 개의 값을 하나의 단위로 묶어서 다루는 데이터 타입
- 논리적으로 이들은 데이터 타입인 동시에 데이터의 구조(흔히 말하는 자료 구조)의 한 종류. 보통 다른 데이터들을 원소로 하는 집합처럼 생각되는 타입들
- Sequence
 - list : 순서가 있는 원소들의 묶음
 - tuple : 순서가 있는 원소들의 묶음. 리스트와 혼동하기 쉬운데 단순히 하나 이상의 값을 묶어서 하나로 취급하는 용도로 사용
 - range
- Lookup
 - mapping
 - » dict : 그룹내의 고유한 이름인 키와 그 키에 대응하는 값으로 이루어지는 키값 쌍(key-value pair)들의 집합.
 - set : 순서가 없는 고유한 원소들의 집합.

○ None

- 존재하지 않음을 표현하기 위해서 “아무것도 아닌 것”을 나타내는 값
- 어떤 값이 없는 상태를 가리킬만한 표현이 마땅히 없기 때문에 “아무것도 없다”는 것으로 약속해놓은 어떤 값을 하나 만들어 놓은 것
- None 이라고 대문자로 시작하도록 쓰며, 실제 출력해보아도 아무것도 출력되지 않음.
- 값이 없지만 False 나 0 과는 다르기 때문에 어떤 값으로 초기화하기 어려운 경우에 쓰기도 함

immutable

1. Numeric

1. int
2. float
3. complex
4. bool

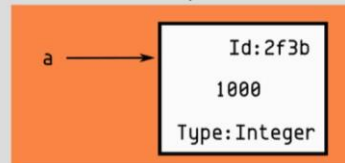
Immutable Integers

Code

```
a = 1000
```

What Computer Does

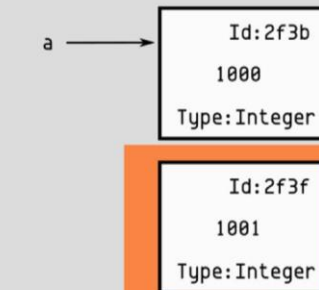
Variables Objects



Step 1: Python creates an integer

```
a = a + 1
```

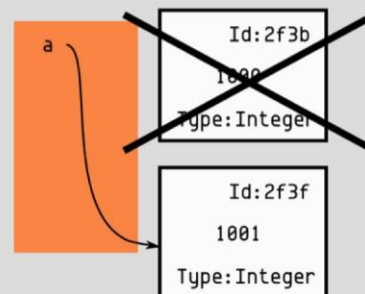
Variables Objects



Step 2: Python adds 1 to a and creates an integer

```
a = a + 1
```

Variables Objects



Step 3: Python rebinds a, garbage collects old object

Type		예시
int	int	14, -14
	int (Hex)	0xe
	int (Octal)	0o16
	int (Binary)	0b1110
float	float	14.0, 0.5, .3, 1.
	float	1.4e1, 1.79e+308 1.8e-308
	infinity	case insensible float("inf"), "Inf", "INFINITY" 및 "iNfINity"는 모두 가능
	Not a Number	case insensible float('nan')
complex	complex	14+0j
bool	bool	True, False
Underscore (readability)		1_000

■ 수를 표현하는 기본 리터럴

○ 정수

- 0, 100, 123 과 같은 표현

○ 실수

- 중간에 소수점 0.1, 4.2, 3.13123 와 같은 식
- 0.으로 시작하는 실수값에서는 흔히 앞에 시작하는 0 제외 가능. .5는 0.5를 줄여쓴 표현

○ 부호를 나타내는 -, +를 앞에 붙일 수 있다. (-1, +2.3 등)

■ 기본적으로 그냥 숫자만 사용하는 경우, 이는 10진법 값으로 해석.

○ 10진법외에도 2진법, 8진법, 16진법이 존재.

- 2진법 숫자는 0b로 시작(대소문자를 구분하지 않음)
- 8진법 숫자는 0o로 시작(대소문자를 구분하지 않음)
- 16진법숫자는 0x로 시작(대소문자를 구분하지 않음)

■ 참/거짓을 의미하는 부울대수값.

○ 자체가 키워드로 True/False를 사용하여 표현

```
In [ ] : a = 10
```

```
In [ ] : a
```

```
Out[ ] :
```

```
In [ ] : type(a)
```

```
Out[ ] :
```

```
In [ ] : b = -2
```

```
In [ ] : b
```

```
Out[ ] :
```

```
In [ ] : type(b)
```

```
Out[ ] :
```

```
In [ ] : c = int(10)
```

```
In [ ] : c
```

```
Out[ ] :
```

```
In [ ] : type(c)
```

```
Out[ ] :
```

```
In [ ] : d = int(-32)
```

```
In [ ] : d
```

```
Out[ ] :
```

```
In [ ] : type(d)
```

```
Out[ ] :
```


- Quiz

```
In [ ] : c = int('10')
```

```
In [ ] : c
```

```
Out[ ] :
```

```
In [ ] : type(c)
```

```
Out[ ] :
```

```
In [ ] : d = int('ten')
```

```
In [ ] : d
```

```
Out[ ] :
```

```
In [ ] : type(d)
```

```
Out[ ] :
```

- 정수형의 최대 범위

```
In [ ] : import sys  
         sys.maxsize
```

Out[] :

- python에서는 최대값을 넘어가면 메모리를 동적으로 늘려줌
- 즉, python에서의 정수형은 오버플로우가 없음

```
In [ ] : 9223372036854775809
```

Out[] :

```
In [ ] : a=1.2
```

```
In [ ] : a
```

```
Out[ ] :
```

```
In [ ] : type(a)
```

```
Out[ ] :
```

```
In [ ] : b=-5.
```

```
In [ ] : b
```

```
Out[ ] :
```

```
In [ ] : type(b)
```

```
Out[ ] :
```

```
In [ ] : c = float(3.14)
```

```
In [ ] : c
```

```
Out[ ] :
```

```
In [ ] : type(c)
```

```
Out[ ] :
```

```
In [ ] : d = 3.14-1
```

```
In [ ] : d
```

```
Out[ ] :
```

```
In [ ] : type(d)
```

```
Out[ ] :
```

- 실수형의 최대 범위

```
In [ ] : sys.float_info
```

```
Out[ ] :
```

- python에서는 실수형의 최대 범위를 넘어가게 되면 inf(무한대)로 변함

```
In [ ] : 1.7976931348623157e+310
```

```
Out[ ] : inf
```

```
In [ ] : e=float('inf')
```

```
In [ ] : e
```

```
Out[ ] :
```

- 실수형은 계산방식이 다르기 때문에 유의해야함

```
In [ ] : 0.1+0.1+0.1 == 0.3
```

```
Out[ ] : False
```

```
In [ ] : 0.1+0.1+0.1
```

```
Out[ ] : 0.30000000000000004
```

```
In [ ] : 1.7976931348623157e+308 == 1.7976931348623157e+308+1
```

```
Out[ ] : True
```

- 실수값은 어림값으로 계산하기 때문

- Quiz

```
In [ ] : c = float('10')
```

```
In [ ] : c
```

```
Out[ ] :
```

```
In [ ] : type(c)
```

```
Out[ ] :
```

```
In [ ] : d = float('10.0')
```

```
In [ ] : d
```

```
Out[ ] :
```

```
In [ ] : type(d)
```

```
Out[ ] :
```