Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.

Seoul National University

Dec 16 – 20, 2019

Basic Data Types

Data Types

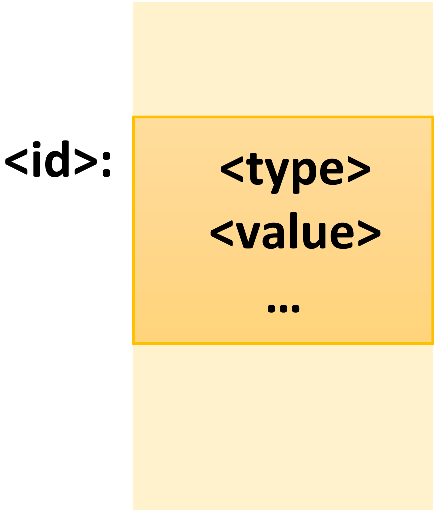
- Basic data types
  - Boolean
  - Integer
  - Floating point
  - String
- Container data types
  - List
  - Dictionary
  - Tuple
  - Set

- User-defined data types (classes)
  - Automobile
  - Monster
  - Pixel
  - …
- Libraries
  - array
  - math
  - random
  - urllib
  - …

Object-oriented Data Model

- Objects are Python's abstraction for data

- Each object has:
  - An identify (e.g., memory address) – `id(x)`
  - A type (or class) – `type(x)`
  - A value



- The '`is`' operator compares the identity of two objects
- Objects can be immutable (e.g., numbers, strings, tuples, …)
- Different variables can refer to the same object

Constants and Variables

- Constant
  - An immutable object with a fixed value (its value cannot be changed)
  - Boolean constants: `True`, `False`
  - Numeric constants: `0`, `12`, `3.14159`
  - String constants: `'this is a string'`, `"hello"`
- Variable
  - A "name" for an object
  - A variable refers to an object (mutable/immutable)
- Python is a dynamically-typed language
  - Variable names can be bound to different values, possibly of varying types (or classes)

Naming Variables

- Must start with a letter or underscore ('_')
- Must consists of letters, numbers, and underscores
- Case sensitive: `spam`, `Spam`, `SPAM` (all different variables)

- Wrong examples: `2spam` `#hello` `x.15`
- Bad examples: `a` `x9gbzlwi` `var1`
- Good examples: `name` `age` `student_id`

Example

```
xlq3z9ocd = 35.0
xlq3z9afd = 8.0
xlq3z9afd = xlq3z9ocd * xlq3z9afd
print(xlq3z9afd)
```

```
a = 35.0
b = 8.0
c = a * b
print(c)
```

```
rate = 35.0
hours = 8.0
pay = rate * hours
print(pay)
```

Reserved Words

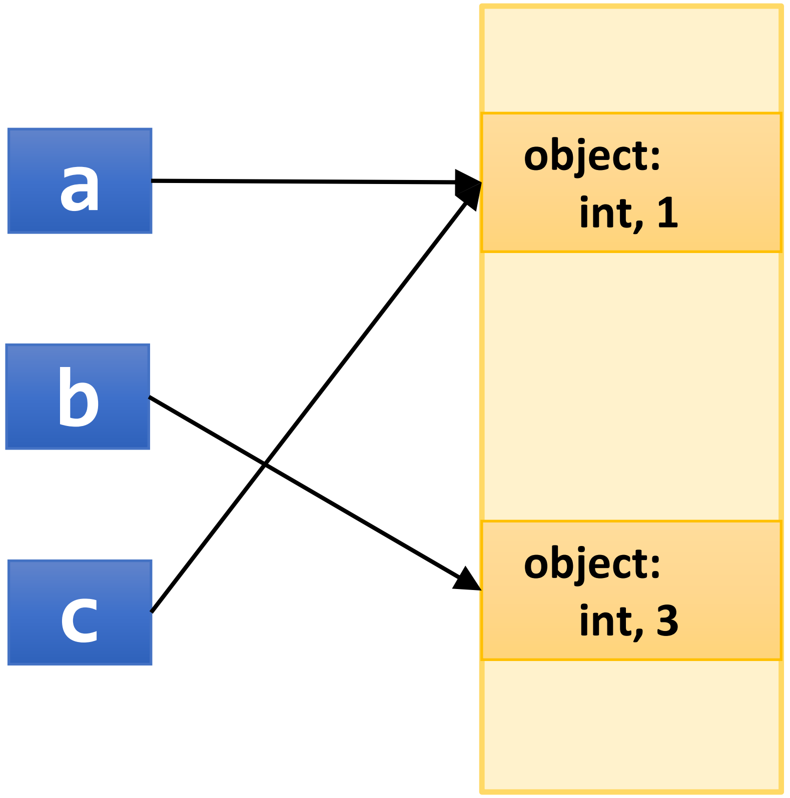
- You cannot use reserved words for variable or function names

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

Assignments

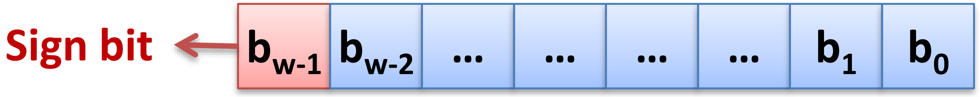
- Assignment operator (=) assigns a value (or an object) to a variable

```
>>> a = 1

>>> b = 3

>>> c = a

>>> print(id(a), id(b), id(c))
```

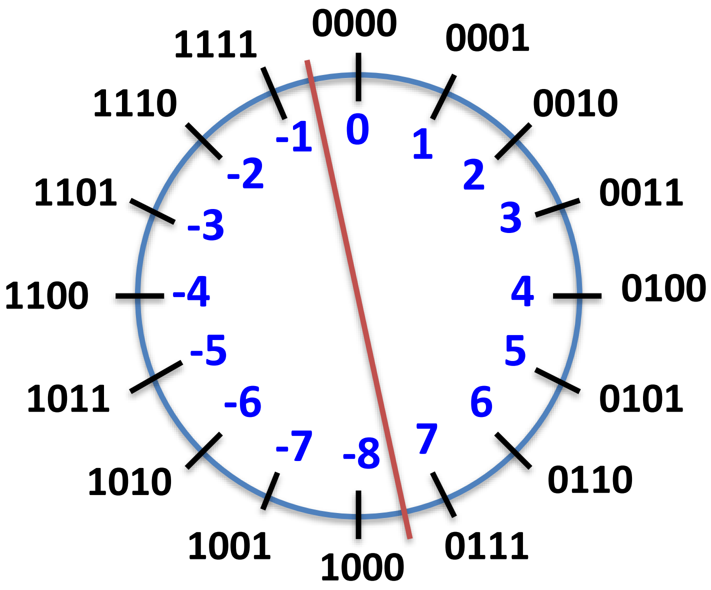


Integer

- `int`
  - Integer numbers in an unlimited range
  - Negative numbers are represented in two's complement format



$$O(B) = -b_{w-1} \cdot 2^{w-1} + \left(\sum_{i=0}^{w-2} b_i \cdot 2^i \right)$$



  - Some small integers are shared (implementation-specific)

Representing Integer Constants

- An integer constant should start with a non-zero digit (except zero)
- Use prefixes (0b, 0o, 0x) to denote binary/octal/hexadecimal values
- A single underscore('_') can be placed between digits

```
>>> print(1011)
>>> print(0b1011)
>>> print(0o1011)
>>> print(0x1011)
>>> print(10_11)
>>> print(0100)
>>> print(10__11)
```

```
>>> print(2_7_8_9_0)
>>> print(27_890)
>>> print(2_7890)
>>> print(0b0110_1100_1111_0010)
>>> print(0x6cf2)
>>> print(0o66362)
```

Integer Example

```
>>> print(100**100)

>>> a = 5

>>> b = 3

>>> c = 2

>>> d = b + c

>>> print(a, d)

>>> print(id(a), id(d))
```

Boolean

- bool
  - False or True
  - A subtype of the integer type
  - Boolean values behave like the values 0 and 1, respectively
  - When converted to a string, 'False' or 'True' are returned, respectively

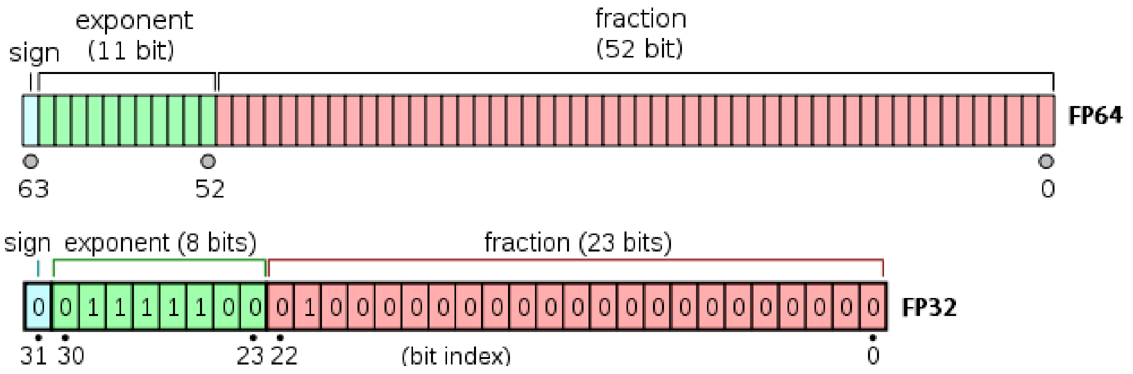
```
>>> t = False
>>> print(t)

>>> a = 100
>>> b = bool(a)
>>> print(a, b)
```

```
>>> t = True
>>> f = False
>>> x = 10
>>> print(x + t)
>>> print(t * f)
>>> print(True == 1)
```

Floating Point

- `float`
  - Python only supports double-precision floating point numbers
  - The benefit of supporting single-precision is not that great due to the overhead of using objects in Python

- IEEE754 floating point representation standard
  - Single precision: 32-bit
    ($1.4 \times 10^{-45} \sim 3.4 \times 10^{38}$)
  - Double precision: 64-bit
    ($4.9 \times 10^{-324} \sim 1.8 \times 10^{308}$)



Representing FP Constants

- Represented with or without exponent
- Integer and exponent parts are always interpreted using radix 10
- A single underscore ('_') can be placed between digits

```
>>> print(3.14)
>>> print(10.)
>>> print(0.001)
>>> print(1e100)
>>> print(3.14e-10)
>>> print(0e0)
```

```
>>> print(3.14_15_92)
>>> print(1_234.005_694)
>>> print(e100)
>>> print(0b1000.0011)
>>> print(0o1234.56)
>>> print(0xdead.beef)
```

Floating Point Example

```
>>> pi = 3.14159
>>> print(pi)
>>> print(2*pi)

>>> d = 0.1
>>> print(d+d+d+d+d+d+d+d+d+d)

>>> VeryLarge = 1e20
>>> x = (pi + VeryLarge) - VeryLarge
>>> y = pi + (VeryLarge – VeryLarge)
>>> print(x, y)
```

String

- `str`
  - A sequence of characters
  - Python 3 natively supports Unicode characters (even in identifiers)
  - No difference in single (e.g., `'hello'`) or double-quoted strings (e.g., `"hello"`)

```
>>> print("I'm your father")
>>> print('Where is "spam"?')

>>> s1 = "What is the"
>>> s2 = 'spam'
>>> print(s1 + s2)
```

```
>>> 이름 = '홍길동'
>>> print("안녕" , 이름)
>>> print("안녕" + 이름)
```

Integer Operations (1)

Operation	Operator	Examples		Priority
Power	`**`	`>>> 2**8`	`>>> -3**2`	Power is higher (or lower) than unary operators on its left (or right). Right-to-left among them.
Unary minus	`-`	`>>> -2-2`	`>>> 3**-2`	
Unary invert	`~`	`>>> ~2`	`>>> -~2`	
Multiplication	`*`	`>>> 2*3`	`>>> -2*3`	Lower than power and unary operators. Left-to-right among them.
Division (yields `float`)	`/`	`>>> 8/3`	`>>> -3/2`	
Floor division (yields `int`)	`//`	`>>> 8//3`	`>>> -3//2`	
Modulo	`%`	`>>> 8%3`	`>>> -3%2`	
Addition	`+`	`>>> 100+1`	`>>> 24+-2`	Lower than *, /, //, %. Left-to-right among them.
Subtraction	`-`	`>>> 100-1`	`>>> 24--2`	

Integer Operations (2)

Operation	Operator	Examples		Priority
Shift left	<<	>>> 2<<3	>>> -1<<2	Lower than +, -. Left-to-right among them.
Shift right	>>	>>> 9>>3	>>> -1>>3	
Bitwise AND	&	>>> 15&5	>>> 3+4&3	Lower than shift.
Bitwise XOR	^	>>> 15^5	>>> 12^15&7	Lower than AND.
Bitwise OR	|	>>> 15|5	>>> 10^5|3	Lower than OR.
Comparisons	<, >, ==, !=, <=, >=	>>> 3>-1	>>> 3<5<6	Lower than OR. Left-to-right among them.
Logical NOT	Not	>>> not True	>>> not 0	Lower than comparisons.
Logical AND	And	>>> 2<1 and 4<9	>>> 3<5 and 5<6	Lower than NOT
Logical OR	or	>>> 2<1 or 4<9	>>> 3<5 or 5<6	Lower than AND

Type Conversion

- int()
- float()
- str()

```
>>> int(3.14)
>>> int('3.14')
>>> int(True)
>>> int('0xcafe')
>>> int('cafe', 16)
>>> int('0xcafe', 0)
```

```
>>> float(3)
>>> float('  -3.14\n')
>>> float('1e10')
>>> str(2020)
>>> str(0xcafe)
>>> str(3.141592)
```

Getting a User Input

- input(prompt)
  - If a prompt argument is present, it is written to standard output
  - Then, reads a line from input, converting it to a string (stripping a trailing newline)

```
>>> name = input('Your name: ')
Your name: Spam
>>> age = input('Your age: ')
Your age: 20
>>> print('Hello,', name)
Hello, Spam
>>> print('You will be', int(age)+1, 'next year!')
You will be 21 next year!
```