

§1 Data Type

†a Object

Python is an **object-oriented** programming language. Everything is an **object** in Python:

$$\text{object} = \begin{cases} \text{identity,} \\ \text{type / class,} \\ \text{value / state,} \\ \text{methods / behaviors / operations.} \end{cases}$$

```
# print the identity, type, and the value for 4
print(id(4), type(4), 4)
# type of any type is a type, the type itself is a type
print(type(type(4)))
print(type(type(type(4))))
```

```
140711773227544 <class 'int'> 4
<class 'type'>
<class 'type'>
```

- **Identity**: it guarantees that different objects have distinct identities at any given time.
- **Type**: objects of the same type support the same operations, and share the same properties.

†b Binding and Input

In Python, the **assignment** of $a = b$ is like making the name a pointing to the object b .

```
# an example for binding
a, b = 4, print
print(type(a), a, type(b), b,
      id(a), id(4), id(b), id(print))
b(a+5, "hello")
```

```
<class 'int'> 4 <class 'builtin_function_or_method'>
<built-in function print> 140723891816984
140723891816984 2069908885472 2069908885472
9 hello
```

The basic input in Python is through the function `input()`. The input takes ONE string as prompt, and it reads input as a string.

```
# an example for input function
n = input(f"{{a}} and hello\n")
```

```
print(type(n), n)
```

```
4 and hello
5
<class 'str'> 5
```

†c Numeric

The following are numeric types:

$\text{bool} \subset \text{int} \subset \text{float} \subset \text{complex}$

```
# an example for the above data types
print(type(True), True, type(1), 1,
      type(1.0), 1.0, type(1+0j), 1+0j)
```

```
<class 'bool'> True <class 'int'> 1 <class 'float'> 1.0
<class 'complex'> (1+0j)
```

```
# subset example
if True == 1 == 1.0 == 1+0j:
    print("Yes")
else:
    print("No")
```

```
Yes
```

We can use `bool()`, `int()`, `float()`, and `complex()` to convert a string to the corresponding data type from `input()`;

```
# input string to number
n = input("type in an integer\n")
print(type(n), n, type(int(n)), int(n))
```

```
type in an integer
17
<class 'str'> 17 <class 'int'> 17
```

identically map from a subset to a larger set, or canonically map from the supset to the restricted set:

```
# identical map and canonical map
print(int(False), float(5), int(3.7))
```

```
0 5.0 3
```

```
<class 'decimal.Decimal'>  
0.10000000000000000000000000000000000000000000000000000000000000  
0.100000000000000000555111512312578270211815834045410
```