Section 2 Expressions, Variables and Objects

There is one famous saying: *Everything is an object in Python!*

In Python, each object has

- an identity,
- a type, and
- a value

Identity and id( )

Roughly speaking, the id( ) function returns an integer called identity, representing the unique memory address of an object.

In [4]:
```
id(3) # integer 3 has an identity
```
Out[4]:
```
4368389312
```

In [3]:
```
id(5.0) # float 5 has different identity
```
Out[3]:
```
140351628649552
```

In [4]:
```
id("python")# string 'python' has an identity. Btw, there is no difference between "" and ''
```
Out[4]:
```
140351609633840
```

In [5]:
```
id([1,2,3]) # list [1,2,3] has another identity
```
Out[5]:
```
140351628742560
```

In [6]:
```
id(abs) # built-in function abs also has an unique indentity!
```
Out[6]:
```
140351525722544
```

In [7]:
```
a = 3
id(a)
```
Out[7]:
```
4440835264
```

Type and type( )

Below are the common built-in types of Python. We're going to define our own types later using Class in Python. Popular data science packages also define their own types.

In [8]:
```
type(3)
```
Out[8]:
```
int
```

In [9]:
```
type(True)
```
Out[9]:
```
bool
```
In [10]:
```
type(5.)
```
Out[10]:
```
float
```
In [10]:
```
type('python')
```
Out[10]:
```
str
```
In [11]:
```
type([1,2,3])
```
Out[11]:
```
list
```
In [12]:
```
type(abs)
```
Out[12]:
```
builtin_function_or_method
```

Expression, Variable, Value and Object

Compared with the concept of *object*, perhaps you're more familiar with the notion of *variables* and *values* in Matlab. With the assignment operators (=), you can assign the *values* to *variables* through expressions in Matlab.

Formally, similar things happen in Python.

In [11]:
```
string = 'python'

print(id(string))

type(string)
```
```
140351609633840
```
Out[11]:
```
str
```

Below we're going to develop a deep understanding of what happens after executing the expression **variable = value** in Python -- dig deep into your computer memory space!

The basic conclusion can be stated as follows: **In Python, variables are just the references to objects.**

Instead of saying that we *assign values to variables* in python, perhaps it's more rigorous to say that *we use variables to point toward objects with certain values*.

In fact, it is even not the most accurate way to use the word "variables". The more appropriate word in Python might be "names" or "identifiers".

In [1]:
```
a = 3
print(id(a))
a = 1
print(id(a))
```
```
4368389312
4368389248
```

In [5]:

```python
a = 1000 # creating an int object with value 1000, and use variable a as the reference
print(id(a))
b = a  # link the SAME object to b
print(id(b))
```

```
140441832405136
140441832405136
```

In [6]:

```python
a = 1000 # creating an int object with value 1000, and use variable a as the reference
print(id(a))
b = a # link the SAME object to b -- now a and b refers to exactly the same object !
print(id(b))
b = 1 # creating a new int object with value 1, and use variable b as the reference
print(id(b))
```

```
140441832405168
140441832405168
4368389248
```