Lecture 3 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 [65]:
```
id(3) # integer 3 has an identity
```
Out[65]:
```
4426056896
```

In [64]:
```
id(5.) # float 5 has different identity
```
Out[64]:
```
140368422983824
```

In [66]:
```
id('python')# string 'python' has an identity. Btw, there is no difference between "" and '
```
Out[66]:
```
140368403558256
```

In [67]:
```
id([1,2,3]) # list [1,2,3] has another identity
```
Out[67]:
```
140368423235904
```

In [68]:
```
id(abs) # built-in function abs also has an unique indentity!
```
Out[68]:
```
140368319715760
```

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 [59]:
```
type(3)
```
Out[59]:
```
int
```

In [70]:
```
type(True)
```
Out[70]:
```
bool
```

In [60]:
```
type(5.)
```
Out[60]:
```
float
```

In [61]:
```
type('python')
```
Out[61]:
```
str
```

In [62]:
```
type([1,2,3])
```

Out[62]:
```
list
```

In [63]:
```
type(abs)
```

Out[63]:
```
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 [73]:
```
string = 'python'

print(id(string))

type(string)
```

```
140368403558256
```

Out[73]:
```
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 [42]:
```
a = 3
print(id(a))
a = 1
print(id(a))
```

```
140368422436784
4426060000
```

In [51]:
```
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))
```

```
140368422983408
140368422983408
```

In [54]:
```
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))
```

```
140368422985552
140368422985552
4426056832
```