3 Python快速面面观（上）

> AI领域中的Python开发 --- by 丁宁

@(SIGAI课程录制)

- 上节课：学习这门课的准备工作已完成
- 接下来：用两节课的时间过一遍Python核心基础的全貌

> **说明**：为了聚焦于Python本身，本节课仅使用**SIGAI**在线编程的**terminal**方式

第一个Python程序

进入SIGAI在线编程的**terminal**模式，输入`python`，敲击回车，进入Python交互式解释器：

```
sigai@8a5f47e78164:/$ python
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

> >>>表示Python解释器等待键入命令

此时直接输入命令可直接得到执行结果：

```
>>> print("hello world")
hello world
```

输入与输出

print()

```
>>> print("SIGAI")
SIGAI
>>> print(300)
300
>>> print(100 + 200)
300
>>> print('100 + 200 =', 100 + 200) # 注意空格哦
100 + 200 = 300
```

input()

```
>>> name = input()
SIGAI
>>> print(name)
SIGAI
>>> name = input('please enter your name: ')
please enter your name: sigai
>>> print(name)
sigai
```

Python中应知道的细节

- 大小写敏感
- 小坑：**缩进** vs **{}**
- 缩进：**Tab** vs **4个空格**

数据类型

- 整数：与数学上的写法完全一样，可处理任意大小的整数
- 浮点数：由于内部存储方式不同，整数计算永远精确，浮点数计算则不是
- 字符串：单引号或双引号括起来的任意文本，后面详细说
- 布尔值：`True` or `False`；可进行`and or not`运算
- 空值：`None` 只需记住 `None`不等于`0`即可

常量

在**Python**中通常用全部大写的变量名表示常量

```
>>> PI = 3.14159265359
```

基本运算

```
>>> 1 + 1
2
>>> 2 - 1
1
>>> 2 * 3
6
>>> 3 / 2
1.5
>>> 2 ** 3
8
>>> 3 // 2
1
>>> 3 % 2
1
>>> 3.0 // 2.0
1.0
>>> -5 // 3
-2
>>> -6 // 3
-2
>>> -7 // 3
-3
```

浅谈变量

- 变量可以是任意数据类型，因而Python是动态类型语言
- 使用变量，必须先给变量赋值
- 务必搞清楚Python变量在内存中的具体情况

```
sigai@8a5f47e78164:~$ python
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
>>> a = "SIGAI"
>>> b = a
>>> a = "sigai"
>>> print(b)
```

List

可变有序集合

List之可变

```
>>> L = ['sigai_1', 'sigai_2', 'sigai_3']
>>> print(L)
['sigai_1', 'sigai_2', 'sigai_3']
>>> L.append('sigai_4')
>>> print(L)
['sigai_1', 'sigai_2', 'sigai_3', 'sigai_4']
>>> L.pop()
'sigai_4'
>>> print(L)
['sigai_1', 'sigai_2', 'sigai_3']
>>> L2 = ['SIGAI_1', 'SIGAI_2', 'SIGAI_3']
>>> L3 = L + L2
>>> print(L3)
['sigai_1', 'sigai_2', 'sigai_3', 'SIGAI_1', 'SIGAI_2', 'SIGAI_3']
>>> LL = L * 2
>>> print(LL)
['sigai_1', 'sigai_2', 'sigai_3', 'sigai_1', 'sigai_2', 'sigai_3']
```

List之有序

```
>>> nList = [1,2,3]
>>> print(nList)
[1, 2, 3]
>>> print(nList[0])
1
>>> print(nList[2])
3
>>> print(nList[3])
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> print(nList[-1])
3
>>> print(nList[-3])
1
>>> print(nList[-4])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> len(nList)
3
>>> nList.append(4)
>>> len(nList)
4
```

List之集合

```
>>> print(L)
['sigai_1', 'sigai_2', 'sigai_3']
>>> print(nList)
[1, 2, 3, 4]
>>> L.append(nList)
>>> print(L)
['sigai_1', 'sigai_2', 'sigai_3', [1, 2, 3, 4]]
>>> print(L[3])
[1, 2, 3, 4]
>>> print(L[3][0])
1
>>> len(L)
4
>>> len(L[3])
4
>>> L = []
>>> len(L)
0
```

Tuple

初始化后不可修改的**List**就是**Tuple**

Tuple之不可变

```
>>> T = ('sigai_1', 'sigai_2')
>>> print(T)
('sigai_1', 'sigai_2')
>>> print(T[0])
sigai_1
>>> T.append('sigai_3')
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
>>> T.pop()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'pop'
>>> T[1] = 'sigai_3'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Tuple之括号歧义

```
>>> T = (1)
>>> print(T)
1
>>> T = (1, 2)
>>> print(T)
(1, 2)
>>> T = (1,)
>>> print(T)
(1,)
```

Tuple之可变

```
>>> T = ('sigai', [1, 2, 3])
>>> print(T)
('sigai', [1, 2, 3])
>>> T[1].append(4)
>>> print(T)
('sigai', [1, 2, 3, 4])
```

Dict

Python中可变的**key-value**形式的数据结构，**查找速度极快**

> 用空间换时间的策略，消耗内存大 内部存放顺序与放入key的顺序无关 key必须是**不可变对象**

```
>>> D = {'sigai_1': 90, 'sigai_2': 80, 'sigai_3': 100}
>>> print(D)
{'sigai_1': 90, 'sigai_2': 80, 'sigai_3': 100}
>>> print(D['sigai_1'])
90
>>> D['sigai_4'] = 95
>>> print(D)
{'sigai_1': 90, 'sigai_2': 80, 'sigai_4': 95, 'sigai_3': 100}
>>> D['sigai_4'] = 59
>>> print(D)
{'sigai_1': 90, 'sigai_2': 80, 'sigai_4': 59, 'sigai_3': 100}
```

```
>>> print(D)
{'sigai_1': 90, 'sigai_2': 80, 'sigai_4': 59, 'sigai_3': 100}
>>> print(D['sigai_5'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'sigai_5'
>>> 'sigai_5' in D
False
>>> 'sigai_4' in D
True
>>> D.keys()
dict_keys(['sigai_1', 'sigai_2', 'sigai_4', 'sigai_3'])
>>> print(D.get('sigai_4'))
59
>>> print(D.get('sigai_5'))
None
>>> print(D.get('sigai_5', -1))
-1
>>> print(D.pop('sigai_4'))
59
>>> print(D)
{'sigai_1': 90, 'sigai_2': 80, 'sigai_3': 100}
>>> L
[]
>>> D[L] = 'list'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

Set

Dict中Key的集合

> 由于key必须**hashable**，也就是说key是**唯一**的，因此**Set中无重复的Key**

```
>>> L = ['sigai_1', 'sigai_2', 'sigai_1']
```

```
>>> print(L)
['sigai_1', 'sigai_2', 'sigai_1']
>>> S = set(L)
>>> print(S)
{'sigai_1', 'sigai_2'}
>>> S.add('sigai_3')
>>> print(S)
{'sigai_1', 'sigai_2', 'sigai_3'}
>>> S.add('sigai_3')
>>> print(S)
{'sigai_1', 'sigai_2', 'sigai_3'}
>>> S.remove('sigai_1')
>>> print(S)
{'sigai_2', 'sigai_3'}
>>> S.remove('sigai_1')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'sigai_1'
>>> S.add([1,2,3])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

Python入门小坑之Python除法

```
>>> 10 / 3
3.3333333333333335
>>> 10 // 3
3
>>> 10 % 3
1
>>> (10 - (10 % 3)) / 3 == 10 // 3
True
```

Python入门小坑之再谈字符串

单引号，双引号，三单引号，三双引号

```
>>> s1 = 'sigai'
>>> s2 = "sigai"
>>> s3 = '''sigai'''
>>> s4 = """sigai"""
>>> id(s1) == id(s2) == id(s3) == id(s4)
True
```

引号作为字符

```
>>> print('sigai')
```

```
sigai
>>> print("'sigai'")
'sigai'
>>> print('"sigai"')
"sigai"
>>> print('''''sigai' and "SIGAI"''')
'sigai' and "SIGAI"
>>> print('\'sigai\'')
'sigai'
>>> print('\"sigai\"')
"sigai"
>>> print('\"sigai\'')
"sigai'
```

转义字符

```
>>> print('\"sigai\'\nSIGAI')
"sigai'
SIGAI
>>> print('\"sigai\'\nSIGAI\\SIGAI')
"sigai'
SIGAI\SIGAI
```

raw string

```
>>> print("sigai'SIGAI\nSIGAI")
sigai'SIGAI
SIGAI
>>> print(r"sigai'SIGAI\nSIGAI")
sigai'SIGAI\nSIGAI
```

在使用正则表达式的时候**raw string**会非常方便

三引号?

```
>>> print('''sigai\n'sigai'\n"sigai"''')
sigai
'sigai'
"sigai"
>>> print('''sigai
... 'sigai'
... "sigai"
... ''')
sigai
'sigai'
"sigai"
```

常用于多行输入或多行字符或多行注释

Python入门小坑之字符编码

- ASCII编码省空间，但是容易出现乱码
- Unicode统一了各种语言的编码，但可能存在大量空间冗余
- UTF-8：可变长的Unicode编码
- ASCII可被认为是UTF-8的一部分

字符编码常见工作模式

- 内存中：Unicode
- 存储时：UTF-8
- 传输时：UTF-8

Python中的字符串

> 内存中默认的字符串是str类型，以Unicode编码 存储或传输时用以字节为单位的bytes类型

```
>>> print('SIGAI在线编程平台')
SIGAI在线编程平台
>>> type('SIGAI在线编程平台')
<class 'str'>
>>> print(b'SIGAI在线编程平台')
  File "<stdin>", line 1
SyntaxError: bytes can only contain ASCII literal characters.
>>> print(b'SIGAI')
b'SIGAI'
>>> type(b'SIGAI')
<class 'bytes'>
```

字符串的编解码（decode & encode）

- 纯英文可用ASCII将str编码为bytes
- 含有中文则可用UTF-8将str编码为bytes
- 从网络或磁盘上读取的字节流为bytes

```
>>> s_u = 'sigai'
>>> s_b = b'sigai'
>>> type(s_u)
<class 'str'>
>>> type(s_b)
<class 'bytes'>
>>> type(s_b.decode('ascii'))
<class 'str'>
>>> type(s_u.encode('ascii'))
<class 'bytes'>
>>> s_u = 'sigai在线编程'
>>> print(s_u.encode('utf-8'))
b'sigai\xe5\x9c\xa8\xe7\xba\xbf\xe7\xbc\x96\xe7\xa8\x8b'
```

```
>>> print(s_u.encode('ascii'))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeEncodeError: 'ascii' codec can't encode characters in position 5-8:
ordinal not in range(128)
>>>
print(b'sigai\xe5\x9c\xa8\xe7\xba\xbf\xe7\xbc\x96\xe7\xa8\x8b'.decode('utf-
8'))
sigai在线编程
```

Python入门小坑之再谈变量

变量指向一个对象，而对象有可变与不可变之分

可变类型与不可变类型

```
>>> a = "SIGAI"
>>> id(a)
140182492168632
>>> a = "sigai"
>>> id(a)
140182492168688
>>> a = ["sigai_1", "sigai_2"]
>>> id(a)
140182492190088
>>> a.append("sigai_3")
>>> a
['sigai_1', 'sigai_2', 'sigai_3']
>>> id(a)
140182492190088
```

可变对象与不可变对象

```
>>> L = ['sigai_2', 'sigai_3', 'sigai_1']
>>> print(sorted(L))
['sigai_1', 'sigai_2', 'sigai_3']
>>> print(L)
['sigai_2', 'sigai_3', 'sigai_1']
>>> L.sort()
>>> print(L)
['sigai_1', 'sigai_2', 'sigai_3']

>>> s = 'sigai'
>>> print(s.replace('s', 'S'))
Sigai
>>> print(s)
sigai
```

> 务必搞清楚，你改变的是对象本身，还是仅得到了一个中间结果

- 变量无类型，对象有类型
- 对象是内存中存储数据的实体，变量则指向对象的指针

Python中的引用与拷贝

可变类型对象的赋值，传递的是引用，类似于C语言中的指针

```
>>> a
['sigai_1', 'sigai_2', 'sigai_3']
>>> id(a)
140182492190088
>>> b = a
>>> id(b)
140182492190088
>>> a.append("sigai_4")
>>> a[0] = "sigai_5"
>>> a
['sigai_5', 'sigai_2', 'sigai_3', 'sigai_4']
>>> b
['sigai_5', 'sigai_2', 'sigai_3', 'sigai_4']
>>> id(a) == id(b)
True
```

如果不想传递引用，需要使用拷贝的方式

```
>>> b = a[:]
>>> id(a) == id(b)
False
>>> a.pop()
'sigai_4'
>>> a[0] = "sigai_1"
>>> a
['sigai_1', 'sigai_2', 'sigai_3']
>>> b
['sigai_5', 'sigai_2', 'sigai_3', 'sigai_4']
```

本节课程就到这里，下节课开始，就要使用脚本模式写Python代码了，赶紧进入下一节课吧~