Python基础

1. Print 输出
2. 注释+ 行与缩进
3. 基本数据类型
4. 条件控制
5. 输入输出

1. Print 输出

- print 默认输出是换行的

2. 注释+ 行与缩进

- Python中单行注释以 # 开头
- 多行注释可以用多个 # 号，还有 ''' 和 """

In [24]:

```
print ("Hello, Python!") # 第二个注释

'''
第三注释
第四注释
'''

"""
第五注释
第六注释
"""
print ("Hello, Python!")
```

```
Hello, Python!
Hello, Python!
```

- python最具特色的就是使用缩进来表示代码块，不需要使用大括号 {} 。
- 缩进的空格数是可变的，但是同一个代码块的语句必须包含相同的缩进空格数。实例如下：

In [25]:

```
if True:
    print ("True")
else:
    print ("False")
```

```
True
```

In [27]:

```
if True:
    print ("Answer")
    print ("True")
else:
    print ("Answer")
  print ("False")    # 缩进不一致，会导致运行错误
```

```
  File "<tokenize>", line 6
    print ("False")    # 缩进不一致，会导致运行错误
    ^
IndentationError: unindent does not match any outer indentation level
```

3. 基本数据类型

- Python3 中有六个标准的数据类型：
  - Number（数字）
  - String（字符串）
  - List（列表）
  - Tuple（元组）
  - Set（集合）
  - Dictionary（字典）

3.1 数字(Number)类型

- python中数字有四种类型：整数、布尔型、浮点数和复数。
  - int (整数), 如 1, 只有一种整数类型 int，表示为长整型，没有 python2 中的 Long。
  - bool (布尔), 如 True。
  - float (浮点数), 如 1.23、3E-2
  - complex (复数), 如 1 + 2j、 1.1 + 2.2j (j为根号负一)

In [28]:

```
a=29
print(type(a))
a=a+1
print(a)
a+=1
print(a)
```

```
<class 'int'>
30
31
```

In [30]:

```
5 + 4  # 加法
4.3 - 2 # 减法
3 * 7  # 乘法
2 / 4  # 除法，得到一个浮点数
2 // 4 # 除法，得到一个整数
17 % 3 # 取余
2 ** 5 # 乘方
```

Out[30]:

```
32
```

3.2 字符串

- 单引号
- 双引号
- 三连单引号或双引号
- 转义符 "
- 字符串的截取的语法格式如下：变量[头下标:尾下标:步长]

In [5]:

```
message='hello, python!'
message1="hello, python!"
print(message)
print(message1)
```

```
hello, python!
hello, python!
```

In [6]:

```
message2= 'I'm looking for someone to chat.'
```

```
  File "<ipython-input-6-c67e3b6fee61>", line 1
    message2= 'I'm looking for someone to chat.'
                   ^
SyntaxError: invalid syntax
```

In [8]:

```
message2= 'I\'m looking for someone to chat.'
message2= "I'm looking for someone to chat."
```

In [9]:

```
message3= "when you see me, you should ask"where have you been",try it."
```

```
  File "<ipython-input-9-41f598b767e2>", line 1
    message3= "when you see me, you should ask"where have you been""
                                               ^
SyntaxError: invalid syntax
```

In [10]:

```
message3= 'when you see me, you should ask"where have you been", try it.'
```

In [11]:

```
message4= """ what are you 弄啥嘞
看没看懂信息啊
上课不听讲嘛"""
```

In [29]:

```
print(message1[0:-1])           # 输出第一个到倒数第二个的所有字符
print(message1[0])              # 输出字符串第一个字符
print(message1[2:5])            # 输出从第三个开始到第五个的字符
print(message1[2:])             # 输出从第三个开始的后的所有字符
print(message1 * 2)             # 输出字符串两次
print(message1 + '你好')        # 连接字符串
```

```
hello, python
h
llo
llo, python!
hello, python!hello, python!
hello, python!你好
```

3.3 List（列表）

- List（列表） 是 Python 中使用最频繁的数据类型。列表中元素的类型可以不相同，它支持数字，字符串甚至可以包含列表（所谓嵌套）。
- 列表是写在方括号 [] 之间、用逗号分隔开的元素列表。
- 和字符串一样，列表同样可以被索引和截取，列表被截取后返回一个包含所需元素的新列表。
- 列表截取：列表变量[头下标:尾下标]
- 列表中的元素是可以改变的

In [32]:

```
list1 = [ 'abc','bcd','efg','hij' ]#相同类型元素
list2 = [ 'abcd', 786 , 2.23, 'runoob', 70.2 ]#不同类型元素
list3 = [ 1,2,3,4,5,6,7,8]
print (list1)             # 输出完整列表
print (list2[0])          # 输出列表第一个元素
print (list3[1:3])        # 从第二个开始输出到第三个元素
print (list3[2:])         # 输出从第三个元素开始的所有元素
print (list3 * 2)         # 输出两次列表
print (list1 + list3)     # 连接列表
list3[4]=10               # 列表中的元素是可以改变的
print(list3)
```

```
['abc', 'bcd', 'efg', 'hij']
abcd
[2, 3]
[3, 4, 5, 6, 7, 8]
[1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8]
['abc', 'bcd', 'efg', 'hij', 1, 2, 3, 4, 5, 6, 7, 8]
[1, 2, 3, 4, 10, 6, 7, 8]
```

3.4 Tuple（元组）

- 元组（tuple）与列表类似，不同之处在于元组的元素不能修改。元组写在小括号 () 里，元素之间用逗号隔开。

In [34]:

```
t1 = ( 'abcd', 786 , 2.23, 'runoob', 70.2  )
t2 = (123, 'runoob')

print (t1)               # 输出完整元组
print (t1[0])            # 输出元组的第一个元素
print (t1[1:3])          # 输出从第二个元素开始到第三个元素
print (t1[2:])           # 输出从第三个元素开始的所有元素
print (t2 * 2)       # 输出两次元组
print (t1 + t2) # 连接元组
t2[0]=11
```

```
('abcd', 786, 2.23, 'runoob', 70.2)
abcd
(786, 2.23)
(2.23, 'runoob', 70.2)
(123, 'runoob', 123, 'runoob')
('abcd', 786, 2.23, 'runoob', 70.2, 123, 'runoob')
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-34-6a580e6a3e42> in <module>
      8 print (t2 * 2)       # 输出两次元组
      9 print (t1 + t2) # 连接元组
---> 10 t2[0]=11

TypeError: 'tuple' object does not support item assignment
```

3.5 Set（集合）

- 集合（set）是由一个或数个形态各异的大小整体组成的，构成集合的事物或对象称作元素或是成员。
- 基本功能是进行成员关系测试和删除重复元素。
- 可以使用大括号 { } 或者 set() 函数创建集合，注意：创建一个空集合必须用 set() 而不是 { }，因为 { } 是用来创建一个空字典。

In [35]:

```
student = {'Tom', 'Jim', 'Mary', 'Tom', 'Jack', 'Rose'}

print(student)   # 输出集合，重复的元素被自动去掉
```

```
{'Tom', 'Jack', 'Mary', 'Rose', 'Jim'}
```

In [36]:

```
# 成员测试
if 'Rose' in student :
    print('Rose 在集合中')
else :
    print('Rose 不在集合中')


# set可以进行集合运算
a = set('abracadabra')
b = set('alacazam')

print(a)

print(a - b)     # a 和 b 的差集

print(a | b)     # a 和 b 的并集

print(a & b)     # a 和 b 的交集

print(a ^ b)     # a 和 b 中不同时存在的元素
```

```
Rose 在集合中
{'r', 'b', 'a', 'd', 'c'}
{'b', 'd', 'r'}
{'l', 'r', 'b', 'a', 'd', 'm', 'z', 'c'}
{'a', 'c'}
{'b', 'l', 'd', 'm', 'z', 'r'}
```

3.6 Dictionary（字典）

- 字典（dictionary）是Python中另一个非常有用的内置数据类型。
- 列表是有序的对象集合，字典是无序的对象集合。两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。
- 字典是一种映射类型，字典用 { } 标识，它是一个无序的 键(key) : 值(value) 的集合。
- 键(key)必须使用不可变类型。
- 在同一个字典中，键(key)必须是唯一的。

In [66]:

```
d={}
d['a']="AAA"
d[1]=111
s={'name': 'abc','code':1, 'site': 'www.123.com'}


print (d['a'])       # 输出键为 'one' 的值
print (d[1])           # 输出键为 2 的值
print (s)          # 输出完整的字典
print (s.keys())   # 输出所有键
print (s.values()) # 输出所有值
```

```
AAA
111
{'name': 'abc', 'code': 1, 'site': 'www.123.com'}
dict_keys(['name', 'code', 'site'])
dict_values(['abc', 1, 'www.123.com'])
```

In [69]:

```
len(d[1]) #len()函数。可以判断各种数据类型的长度,整数和浮点数需要先转换成字符串
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-69-580d4290816c> in <module>
----> 1 len(d[1]) #len()函数。可以判断各种数据类型的长度,整数和浮点数需要先转换成
字符串

TypeError: object of type 'int' has no len()
```

3.7 Python数据类型转换

- int(x [,base]) #将x转换为一个整数
- float(x) #将x转换到一个浮点数
- str(x) #将对象 x 转换为字符串
- list(s) #将序列 s 转换为一个列表
- set(s) #转换为可变集合

In [64]:

```
x=109.01
int(x) #将x转换为一个整数
float(x) #将x转换到一个浮点数
str(x) #将对象 x 转换为字符串
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-64-00bdb7028061> in <module>
      3 float(x) #将x转换到一个浮点数
      4 str(x) #将对象 x 转换为字符串
----> 5 len(x) #len()函数。可以判断各种数据类型的长度

TypeError: object of type 'float' has no len()
```

4 条件控制

4.1 if 语句

if condition_1:

    statement_block_1

elif condition_2:

    statement_block_2

else:

    statement_block_3

- 如果 "condition_1" 为 True 将执行 "statement_block_1" 块语句
- 如果 "condition_1" 为False，将判断 "condition_2"
- 如果"condition_2" 为 True 将执行 "statement_block_2" 块语句
- 如果 "condition_2" 为False，将执行"statement_block_3"块语句

Python 中用 elif 代替了 else if，所以if语句的关键字为：if – elif – else。

In [45]:

```
age = int(input("请输入你家狗狗的年龄: "))
print("")
if age <= 0:
    print("你是在逗我吧!")
elif age == 1:
    print("相当于 14 岁的人。")
elif age == 2:
    print("相当于 22 岁的人。")
elif age > 2:
    human = 22 + (age -2)*5
    print("对应人类年龄: ", human)
```

请输入你家狗狗的年龄: 10

对应人类年龄: 62

4.2 比较运算符

== 等于 - 比较对象是否相等 (a == b) 返回 False。

!= 不等于 - 比较两个对象是否不相等 (a != b) 返回 True。

> 大于 - 返回x是否大于y (a > b) 返回 False。

< 小于 - 返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。注意，这些变量名的大写。 (a < b) 返回 True。

>= 大于等于 - 返回x是否大于等于y。 (a >= b) 返回 False。

<= 小于等于 - 返回x是否小于等于y。 (a <= b) 返回 True。

In [46]:

```
if ( a == b ):
   print ("1 - a 等于 b")
else:
   print ("1 - a 不等于 b")
```

1 - a 不等于 b

4.3 逻辑运算符

- and
- or
- not

In [47]:

```
a = 10
b = 20

if ( a and b ):
   print ("1 - 变量 a 和 b 都为 true")
else:
   print ("1 - 变量 a 和 b 有一个不为 true")

if ( a or b ):
   print ("2 - 变量 a 和 b 都为 true，或其中一个变量为 true")
else:
   print ("2 - 变量 a 和 b 都不为 true")
```

```
1 - 变量 a 和 b 都为 true
2 - 变量 a 和 b 都为 true，或其中一个变量为 true
```

4.4 成员运算符

- in
  - 如果在指定的序列中找到值返回 True，否则返回 False。 x 在 y 序列中 , 如果 x 在 y 序列中返回 True。
- not in
  - 如果在指定的序列中没有找到值返回 True，否则返回 False。 x 不在 y 序列中 , 如果 x 不在 y 序列中返回 True。

In [49]:

```
a = 10
b = 20
list = [1, 2, 3, 4, 5 ];
if ( a in list ):
   print ("1 - 变量 a 在给定的列表中 list 中")
else:
   print ("1 - 变量 a 不在给定的列表中 list 中")

if ( b not in list ):
   print ("2 - 变量 b 不在给定的列表中 list 中")
else:
   print ("2 - 变量 b 在给定的列表中 list 中")
```

```
1 - 变量 a 不在给定的列表中 list 中
2 - 变量 b 不在给定的列表中 list 中
```

4.5 循环语句

- while语句
- for循环
- range()函数
- break和continue语句

In [50]:

```
#while用于条件判断，特别是数量一定时
count = 0
while count < 5:
   print (count, " 小于 5")
   count = count + 1
else:
   print (count, " 大于或等于 5")
```

```
0  小于 5
1  小于 5
2  小于 5
3  小于 5
4  小于 5
5  大于或等于 5
```

In [61]:

```
# for循环可以遍历任何序列的项目，如一个列表、一个字符串、一个字典。
languages = ["C", "C++", "Perl", "Python"]
language_dict={1:"C",2:"C++",3:"Perl",4:"Python"}
for x in languages:
    print (x)
for x in language_dict:
    print (x,end=" ")
    print(language_dict[x])
```

```
C
C++
Perl
Python
1 C
2 C++
3 Perl
4 Python
```

In [74]:

```
#如果你需要遍历数字序列，可以使用内置range()函数。它会生成数列，例如:
for i in range(3) :#从零开始
    print(i,end=" ")
print()
for i in range(5,9) :#可以指定从哪开始
    print(i,end=" ")
print()
for i in range(5,10,2) :#可以指定每次增量的多少
    print(i,end=" ")
```

```
0 1 2
5 6 7 8
5 7 9
```

In [75]:

```
#break 语句可以跳出 for 和 while 的循环体。
for letter in 'zhaiyujia':     # 第一个实例
   if letter == 'a':
      break
   print ('当前字母为 :', letter)
```

```
当前字母为 : z
当前字母为 : h
```

In [76]:

```
#continue语句被用来告诉Python跳过当前循环块中的剩余语句，然后继续进行下一轮循环。
for letter in 'zhaiyujia':     # 第一个实例
   if letter == 'i':         # 字母为 o 时跳过输出
      continue
   print ('当前字母 :', letter)
```

```
当前字母 : z
当前字母 : h
当前字母 : a
当前字母 : y
当前字母 : u
当前字母 : j
当前字母 : a
```

5. 输入输出

1.读和写文件

- open() 将会返回一个 file 对象，基本语法格式如下:open(filename, mode)
  - filename：包含了你要访问的文件名称的字符串值。
  - mode：决定了打开文件的模式：只读，写入，追加等。
    - 'w'表示写入
    - 'r'表示读取
    - 'a'表示添加

- f.readline()从文件中读取单独的一行。换行符为 '\n'。f.readline() 如果返回一个空字符串, 说明已经已经读取到最后一行。
- f.readlines() 将返回该文件中包含的所有行。
- f.write(string) 将 string 写入到文件中, 然后返回写入的字符数。

In [77]:

```
with open("text.txt",'w') as f:
    f.write('a')
    f.write('b'+"\n")
    f.write('cde'+"\n")
```

2. Python3 OS 文件/目录方法

- os 模块提供了非常丰富的方法用来处理文件和目录。
  - os.getcwd() 得到当前工作目录，即当前Python脚本工作的目录路径

- os.listdir(path) 返回指定目录下的所有文件和目录名
- os.path.isdir(file) 判断是否是文件夹
- os.chdir(path) 改变当前工作目录到指定的路径

In [78]:

```
import os
os.getcwd()
os.listdir(os.getcwd()+"\\AppData")
os.path.isdir(os.getcwd())
os.chdir(os.getcwd()+"\\AppData")
```

Out[78]:

```
'C:\\Users\\zhaiy'
```