一：Python基础语法

- 一基础语法
  - 输入和输出
  - 变量的命名

```
>>>number = 34
>>>name = 'forchange'
>>>list_class = ['基础语法课程'，'爬虫分析初阶'，'爬虫分析进阶'，'自动化办公']
#以上number，name，list_class都是是变量名
```

    - 1.只能是一个词；
    - 2.只能包含字母、数字和下划线；
    - 3.不能以数字开头；
    - 4.尽量描述包含的数据内容；
    - 5.不要使用python函数名和关键字。
      - python3.7.4中的关键字（不用记，熟悉即可）

```
False None True and as assert async
await break class continue def del elif
else except finally for from global if
import in is lambda nonlocal not or
pass raise return try while with yield
```

  - 条件判断

条件判断的解释：让计算机知道，在什么条件下，该去做什么。

    - 单向判断
      - if...#如果条件成立，就执行语句

```
>>>number = 6
>>>if number > 3:
...       pirnt(number)
6
#注意格式！if后面要加冒号，同时执行语句要缩进四个空格。(空格和tab，我选空格?^^)
```

    - 双向判断
      - if...else... #条件成立执行if语句 #否则执行else语句

```
number = 7
if number < 3:
   pirnt(number)
else:
   number =number - 3
print(number)
#结果输出为4
#if和else是同一层级，不需要缩进。if和else下的执行语句都需要缩进四个空格。
```

- if...else...和if...if...的区别
  - #if...else...一个条件满足后就不会进行其他判断（if代表的条件和else代表的条件是互斥的）
  - #if...if...会遍历所有条件，一个条件无论满足还是不满足，都会进行下一个条件的判断
- 多向判断
  - if...elif...else #三个及其以上条件的判断

```
grade = 65
if 80 <= grade <=100:
    print('成绩优秀')
elif 60 <= grade < 80:
    print('成绩中等')
else :
    print('成绩差')
#结果输出为成绩中等
```

- IF嵌套
  - if 嵌套 #使用if进行条件判断 #还希望在条件成立的执行语句中再增加条件判断 #即if中还有if #这两个if非平级

```
grade = 15
if 80 <= grade <=100:
    print('成绩优秀')
elif 60 <= grade < 80:
    print('成绩中等')
else :
    print('成绩差')
if 20<= grade <60:
    print('再努力一把，还有救！')
else :
    print('你要比以前更努力才行，你可以的！')
#结果输出为:
成绩差
你要比以前更努力才行，你可以的！
#注意，嵌套的第二个if缩进了4个空格，表示不同的层级。
```

- 二、数据类型
  - 数据类型

python常见的数据类型：字符串，整数型，浮点数，列表，字典，布尔值，元组。

- 字符串str：用引号括起来的文本（如：'python'、'123'、'风变编程'）
- 整数int：不带小数点的数字（如：-1、1、0、520、1314）
- 浮点数float：带小数点的数字，运算结果存在误差（如：-0.15、3.1415、1.0)
- 以下的数据结构会有一节或两节的课程介绍，可轻松上手。
  - 列表list：是一种有序的集合，可以随时增加或删除其中的元素。标识是中括号[]。

        - 元组tuple：一种类似列表的数据类型，但是不能被修改。
        - 字典dice：全称为dictionary，使用键值对（key-value）作为存储方式。标识是大括号{}。
    - 布尔值bool：表示真假的数据类型，只有两个值，True和False。
- 数据的操作
    - 字符串的拼接
        - 初阶用法：使用'+'进行字符串的拼接

```
>>>print('风变'+'编程')
风变编程
>>>name = '酱酱'
>>>begin = '我叫'
>>>print(begin + name)
我叫酱酱
```

        - 进阶用法：使用'%'进行字符串的拼接

```
>>>name = '《凤求凰》'
 >>>number = 1
>>>print('司马相如以%d曲%s打动了卓文君' %(number,name))
 司马相如以1曲《凤求凰》打动了卓文君
```

- 四则运算
    - 运算符　　表示　　例子
        - +　加　　1 + 1 输出结果为2
        - -　减　　1 - 1 输出结果为0
        - *　乘　　3 * 2 输出结果为6
        - /　除　　2 / 1 输出结果为2
        - %　　取模-返回除法的余数　　5 % 2 输出结果为1
        - **　　幂-返回x的y次幂　　2 ** 3 输出结果为8
        - //　　取整除-返回商的整数部分　　11 // 2 输出结果为5
    - 运算优先级：从左到右顺着来，括号里的优先算，乘除排在加减前。
- 数据转换
    - type() #查看变量的数据类型

```
>>>who = 'xiaojiangjiang'
>>>print(type(who))
<class 'str'>
#结果显示这是一个字符串类型的数据
```

    - str() #将其他数据类型强制转换为字符串

```
>>>begin = '我吃了'
>>>number = 1
>>>fruit = '个水果'
```

```
>>>print(begin + str(number) +fruit)
我吃了1个水果
#进行字符串拼接时，不同数据类型不能直接使用'+'连接，需要现将整数转化为字符串类型
```

- int()
  - #将整数形式的字符串转化为整数（文本类字符串和浮点形式的字符串不能转化为整数）
  - #对浮点数直接抹零取整

```
>>>print(int(3.8))
3
```

- float() #将整数和字符串转换为浮点数 #文字类字符串无法转换

```
>>>print(float(8))
8.0
```

- list() #将数据转换为列表类型

```
>>>a = 'python小课'
>>>print(list(a))
['p', 'y', 't', 'h', 'o', 'n', '小', '课']
```

- len() #用于检查某个数据的长度

```
>>>bros = ['刘备','关羽','张飞']
>>>print(len(bros))
3
>>>emotion = 'happy'
>>>print(len(emotion))
5
```

- 数据的常用语法
  - 列表语法
    - 列表的操作可分为两种类型，
      - 一种类型为对列表元素的处理，
      - 另一种类型为对列表的处理，
      - 每种类型都有四种操作：提取，修改，增加，删除（取改增删）。
    - 偏移量：对列表元素的位置编号。#列表的偏移量从0开始计算

```
#如果要提取一段列表，需要使用切片的形式[a:b]：从a到b的元素，但不包括b（a <= X < b）；冒号某侧如果没有数字，则全取
>>>list = ['松','竹','梅']
>>>print(list[0])
>>>print(list[1:2])
>>>print(list[:2])
松
['竹']
['松','竹']
#松，竹，梅三者的偏移量分辨是0，1，2。
```

    - 列表元素的提取

```
>>>list = ['松','竹','梅']
>>>print(list[0])
松
>>>list = [['松','松树'],['竹','竹子'],['梅','梅花']]
>>>print(list[0][1])
松树
#嵌套列表的提取
```

- 列表元素的修改

```
>>>list = ['松','竹','梅']
>>>list[0] = '松树'
>>>print(list)
['松树', '竹', '梅']
```

- 列表元素的增加
  - append() #是列表的方法，在括号内添加一个元素，可以将该元素添加到列表末尾

```
>>>list = ['松','竹']
>>>list.append('梅')
>>>print(list)
['松','竹','梅']
```

    - 易错一：用append时不能对列表赋值

```
>>>list = ['松','竹']
>>>list = list.append('梅')
>>>print(list)
None
#第二行语法错误
```

    - 易错二：append后面是小括号，而非中括号

```
>>>list = ['松','竹']
>>>list.append['梅']
>>>print(list)
TypeError: 'builtin_function_or_method' object isnot subscriptable
#第二行语法错误
```

    - 易错三：append不能一次添加多个元素

```
>>>list = ['松','竹']
>>>list.append('梅','岁寒三友')
>>>print(list)
TypeError: append() takes exactly one argument (2given)
#第二行语法错误
```

- 列表元素的删除
  - del #删除命令
    - 易错一：每次只能删除一个元素，
    - 易错二：删除多个元素时，要重新计算偏移量

```
>>>list = ['松','竹','梅']
>>>del list[0]
>>>print(list)
>>>del list[0]
>>>print(list)
['竹', '梅']
['梅']
```

- 列表的切片（即列表层面的提取，一次提取若干个元素）

```
>>>list = ['松','竹','梅']
>>>print(list[1:2])
>>>print(list[:2])
['竹']
['松','竹']
#注意:列表的切片提取出来的是列表
```

- 列表的修改 #同样是使用赋值语句，注意是对列表的赋值

```
>>>list = ['松','竹','梅']
>>>list[:] = ['岁寒三友']
#list[:]表示将列表的所有元素取出来
>>>print(list)
['岁寒三友']
#注意以下的错误做法:
>>>list = ['松','竹','梅']
>>>list[:] = '岁寒三友'
>>>print(list)
['岁', '寒', '三', '友']
```

- 列表的增加 #列表的增加叫作列表的合并会更合理
  - 使用符号'+'

```
符号'+'只能用在列表之间，不能用在列表和元素之间
>>>list1 = ['松']
>>>list2 = ['竹']
>>>list3 = ['梅']
>>>list = list1 + list2 +list3
>>>print(list)
['松', '竹', '梅']
```

- 列表的删除
  - del #删除命令

```
>>>list = ['松','竹','梅']
>>>del list[:2]
>>>print(list)
['梅']
```

- 通过list()函数将元组转化为列表的形式

- 字典语法

- 字典数据的提取

  #列表使用偏移量来提取，字典使用键来提取
  >>>group = {'师父':'唐三藏', '大师兄':'孙行者', '二师兄':'猪八戒', '沙师弟':'沙和尚'}
  >>>print(group['师父'])
  唐三藏

- 字典数据的修改

  >>>group = {'师父':'唐三藏', '大师兄':'孙行者', '二师兄':'猪八戒', '沙师弟':'沙和尚'}
  >>>group['师父']='唐玄奘'
  >>>print(group)
  {'师父': '唐玄奘', '大师兄': '孙行者', '二师兄': '猪八戒', '沙师弟': '沙和尚'}

- 字典数据的增加

  >>>group = {'师父':'唐三藏', '大师兄':'孙行者', '二师兄':'猪八戒', '沙师弟':'沙和尚'}
  >>>group['白龙马']='敖烈'
  >>>print(group)
  {'师父': '唐三藏', '大师兄': '孙行者', '二师兄': '猪八戒', '沙师弟': '沙和尚', '白龙马': '敖烈'}

- 字典数据的删除

  >>>group = {'师父':'唐三藏', '大师兄':'孙行者', '二师兄':'猪八戒', '沙师弟':'沙和尚'}
  >>>del group['师父']
  >>>print(group)
  {'大师兄': '孙行者', '二师兄': '猪八戒', '沙师弟': '沙和尚'}

- dict.keys()

  - 提取字典中所有的键

    >>>group = {'师父':'唐三藏', '大师兄':'孙行者', '二师兄':'猪八戒', '沙师弟':'沙和尚'}
    >>>print(group.keys())
    dict_keys(['师父', '大师兄', '二师兄', '沙师弟'])
    #打印出了所有字典的键，但是都是元组的形式
    >>>group = {'师父':'唐三藏', '大师兄':'孙行者', '二师兄':'猪八戒', '沙师弟':'沙和尚'}
    >>>print(list(group.keys()))
    ['师父', '大师兄', '二师兄', '沙师弟']

- dict.values() #提取字典中所有的值

  >>>group = {'师父':'唐三藏', '大师兄':'孙行者', '二师兄':'猪八戒', '沙师弟':'沙和尚'}
  >>>print(group.values())
  dict_values(['唐三藏', '孙行者', '猪八戒', '沙和尚'])

- dict.items() #提取字典中所有的键值对

  >>>group = {'师父':'唐三藏', '大师兄':'孙行者', '二师兄':'猪八戒', '沙师弟':'沙和尚'}
  >>>print(group.items())
  dict_items([('师父', '唐三藏'), ('大师兄', '孙行者'),('二师兄', '猪八戒'), ('沙师弟', '沙和尚')])

- 产生布尔值的表达式

  - bool() #检查数值的真假

    >>>print(bool(1))
    True
    值本身作为条件

假的 False
其他都是真的 True

- 0 5（任意整数）1.0（任意浮点数）
  - ''（空字符串） '风变编程'（字符串）
  - []（空列表） [1,2,3]
  - {}（空字典） {1:'a',2:'b'}
  - None

- 比较运算符产生布尔值

== 等于 如果两侧的值相等，条件为真
!= 不等于 如果两侧的值不相等，条件为真
> 大于 如果左侧的值大于右侧，条件为真
< 小于 如果左侧的值大于右侧，条件为真
>= 大于等于 如果左侧的值大于或等于右侧，条件为真
<= 小于等于 如果左侧的值小于或等于右侧，条件为真
注意：运算符之间不用空格，不可以写成= =、> =

- 成员运算符产生布尔值

in 属于 如果值在指定序列里，条件为真
not in 不属于 如果值不在指定序列里，条件为真

- 逻辑运算符产生布尔值

and 且 连接两个布尔值，如果两个都为真，该条件才为真
or 或 连接两个布尔值，如果有一个为真，该条件即为真
not 非 反逻辑状态，a为True，not a则为False，反之亦然

- 三、循环
  - for循环
    - for循环的基本格式是：for...in...
      - #遍历字符串

```
>>>for i in 'coding':
...  print(i)
c
o
d
i
n
g
```

      - #遍历列表

```
>>>for i in ['for','change']
...  print(i)
for
change
```

      - #遍历字典的键 for...in dict:

```
>>>list = {1:'a',2:'b',3:'c'}
>>>for i in list:
...    print(i)
1
2
3
```

- #遍历字典的值 for…in dict.values():

```
>>>list = {1:'a',2:'b',3:'c'}
>>>for i in list.values():
...    print(i)
a
b
c
```

- #遍历字典的键值对 for…in dict.items():

```
>>>list = {1:'a',2:'b',3:'c'}
>>>for k, v in list.items():
...    print(k)
...    print(v)
1
a
2
b
3
c
```

- for…in range() #处理指定次数的循环
  - range()函数
    - #range()有最基本的三种用法：
      - range(b) a：计数从a开始。不填时，从0开始
      - range(a,b) b：计数到b结束，但不包括b
      - range(a,b,c) c：计数的间隔，不填时默认为1。

```
>>>range(5)#计数依次为0，1，2，3，4>>>range(1,5)#计数依次为1，2，3，
4>>>range(2,8,2)#计数依次为2，4，6
>>>for i in range(3):
...        print('第%d遍风变编程' %i)
第0遍风变编程
第1遍风变编程
第2遍风变编程
```

- while循环
  - #当条件为真时，执行循环语句，只要条件为真，便会一直循环

```
>>>count = 3
>>>while count > 1:
...      print('happy coding')
```

```
...     count = count -1
happy coding
happy coding
```

- while循环和for循环的区别：
  - #for擅长处理固定次，自动遍历各序列
  - #while处理不定次数的循环，条件为False便停止
- 循环进阶
  - break #如果满足条件，则结束循环

```
>>>while True:
...     print('happy coding')
...     break
happy coding
#break会结束循环，如果只有前两行代码，会无限循环打印happy coding
>>>count = 3
>>>while count >1:
...     print('happy coding')
...     count = count - 1
...  if count == 2:  #当count等于2的时候，停止循环
...     break
happy coding
#对比while循环的例子，我们发现这里只打印了一次happy coding
```

  - continue #如果满足条件，则跳过当前循环的剩余语句，直接开始下一轮循环

```
count = 3
while count >1:
  print('happy')
  count = count - 1
  if count == 2:  #当count等于2的时候，跳过下列语句，重新开始新的一轮循环
    continue
  print('coding')  #由于continue语句，coding只会打印一次
#打印的结果为：
happy
happy
coding
else
#无论是否进入循环，最后都会执行esle语句，除非执行break语句跳出循环
count = 3
while count >2:
  print('在风变')
  count = count -1
else:  #无论是否进入循环都会执行else语句
  print('happy coding')
#打印结果为：
在风变
```

```
happy coding
```

- 循环嵌套·#即循环中有循环

```
>>>for i in ['风变','编程']:  #首先遍历列表元素
...   for t in i:        #然后遍历元素(字符串)
...     print(t)
风
变
编
程
```

- 四、函数
  - 函数是组织好的、可以重复使用的、用来实现单一功能的代码
  - 函数类型可分为自定义函数和内置函数，自定义函数是需要自己定义，而内置函数是python内部已经定义好的函数，比如print()、input()等
  - 函数定义的语法
    - def #定义函数
    - return #函数的返回值
  - #函数定义的格式
    - def 函数名(参数)：
      - 函数体
      - return 语句

```
#一个简单的例子
def math_func(x):
y = x + 5
print(y)
return y
math_func(2)
#打印结果为7
```

  - 变量作用域 #变量作用域可认为是变量作用的范围
    - 局部变量：只能函数内或者一定代码块内生效
    - 全局变量：在全局内生效的变量
      - global #将局部变量转化为局部变量
      - python内置函数
- 五、类与对象
  - 类：具有相同属性和方法的对象的抽象 实例：类的个例
  - 对象：Python中的对象是类和实例的集合，类可以看作是对象，实例也可以看作是对象
  - 基本语法
    - class #定义类，注意类名需要大写

- class MyClass: #定义类MyClass
- i =12345 #定义类的属性(变量)

```
def f(self):  #定义类的方法
    return('hello world')  #执行这个方法会返回'helloword'这个字符串
    x = MyClass()  #创建类的实例x
    print(x.i)  #打印实例x的属性
    print(x.f())  #打印实例x的f方法
#输出的结果为
12345
hello world
class A(B)
```

- #定义B类的子类A类，A类具有B类的属性和方法，也将B类称为A类的父类
  - class SecondClass(MyClass):
  - pass
  - #定义SecondClass是MyClass的子类，SecondClass可以调用MyClass的属性和方法

```
x = SecondClass()
print(x.i)
print(x.f())
#输出的结果为
12345
hello world
class A(B, C)
#多重继承，A类同时是B类和C类的子类，A类在调用属性和方法的时候，会优先调用位于左侧的类
class B:
i =123  #B类的属性是i=123
class C:
i =12345  #C类的属性是i=12345
class A(B,C):  #A类是B类和C类的子类
pass
x = A()  #创建A类的实例x
print(x.i)  #调用属性,会优先调用B类的属性
#结果输出为
123
```

- def __init__(self):
  - #创建类的初始化方法，只要调用类，便自动调用初始化方法的语句，常用于创建实例属性

```
>>>class A:
...  def __init__(self):  #只要创建实例，便会自动执行初始化方法下的语句
...      print('hello world')
>>>x = A()
hello world  #只要创建实例就会调用方法,打印helloworld
```

#对比以下没有初始化的方法：
>>>class A:
... def f(self):
... print('hello world')
>>>x = A()
#不使用初始化方法，创建实例无任何输出

- super() #在子类的方法里调用父类的方法，使子类的方法可以在继承父类方法的基础上进行扩展

1 2 3 def super(cls, inst): mro = inst.__class__.mro() return mro[mro.index(cls) + 1]
cls代表类，inst代表实例，可以看出上面的代码做了两件事：
· 获取inst的MRO列表。
· 查找cls在MRO的index，并返回它的下一个类，即mro[index + 1]
当你使用super(cls,inst)时，python会在inst的MRO列表上搜索下cls的下一个类。

- 六、模块与库
  - 模块类型
    - 内置模块 #python官方组织编写和维护的模块
    - 自定义模块 #自己写代码，然后将代码块保存为 .py 文件
    - 第三方模块 #从自定义模块而来，代码写作者公开自己的代码
  - 根据模块的组织形式的不同，也可分为单个模块文件、模块包、模块库
  - 模块和模块对象导入方法
    - import A #导入模块A
      - #现在可以调用模块里函数和变量，但是必须通过【模块名.函数名()】和【模块名.变量名】的方式调用
      - #创建类实例的时候，需要使用【实例名 = 模块名.类名()】进行创建，创建实例后调用类方法和属性可以使用【实例名.函数名()】和【实例名.变量名】

import A as a
#导入模块A，并将模块A重新命名为a
#调用模块中的类、函数和变量如上述操作一样
from A import B
#导入模块A中的对象B
#调用对象B中的函数和变量可以不加模块名
from A import B，C，D
#导入模块A中的多个对象B，C，D
from A import *
#导入模块A中的所有对象
if __name__=="__main__":
#当.py文件被直接运行时，if__name__=="__main__":之下的代码块将被运行
#当.py文件以模块形式被导入时，if __name__=="__main__":之下的代码块不被运行

- 七、文件读写
  - 文件读写三步骤
    - 第一步，打开文件

- 第二步，读（写）文件
- 第三步，关闭文件

- 打开文件语法

```
open(file, mode, encoding)
#打开文件
f = open('/letter.txt', 'r', encoding = 'UTF-8')
with open() as…
#使用这种方式打开文件，可以不使用close()关闭文件
with open('/letter.txt', 'r', encoding = 'UTF-8')as f:
```

- 读写模式mode

```
模式mode 操作 若不存在 是否覆盖
r 只能读不能写 报错 -
rb 二进制只读 报错 -
r+ 可读可写 报错 是
rb+ 二进制读写 报错 是
w 只能写不能读 创建文件 是
wb 二进制只写 创建文件 是
w+ 可读可写 创建文件 是
wb+ 二进制读写 创建文件 是
a 追加不能读 创建文件 否，追加写
ab 二进制追加不能读 创建文件 否，追加写
a+ 可读可写 创建文件 否，追加写
ab+ 二进制追加可读可写 创建文件 否，追加写
```

- 读写文件语法
  - read() #读取文件内容

```
with open('/letter.txt','r',encoding ='UTF-8') as f:
content = f.read()  #以字符串的形式读取文件内容，将文件内容赋值给变量content
```

  - readlines() #以列表的方式读取文件内容

```
with open('/letter.txt','r',encoding = 'UTF-8') asf:
# = f.readlines()
#以列表的形式读取文件内容，将文件内容赋值给变量content
```

  - write() #清空文件内容，并写入字符串入内容

```
with open('/letter.txt','r',encoding ='UTF-8') as f:
f.write('python')
writelines()
#清空文件内容，以列表的方式写入
with open('/letter.txt','r',encoding ='UTF-8') as f:
f.writelines('python')
```

- 关闭文件语法
  - close() #关闭文件
- csv文件读写的相关函数
  - reader() #读取csv文件的函数

```
import csv  #导入csv模块
with open('letter.csv') as f:
reader =csv.reader(f)  #读取csv文件，将文件内容赋值到reader
```

- writer() #将内容写入csv文件
  - writerow() #写入一行内容
  - writerows() #一次写入多行csv文件

```
import csv  #导入csv模块
with open('letter.csv','w',newline = '') as f:
writer =csv.writer(f)  #写入csv文件
writer.writerow(['python小课', '风变编程'])  #写入一行内容
data =[['交互式学习', '更简单'],['助教酱酱','为你答疑解惑']]
writer.writerows(data)  #写入多行内容
```

- os模块
  - os.getcwd() #返回当前的工作目录

- 八、debug
  - try...except...语句 用于处理

```
for i in range(6):
try:
print(6/i)
#使用6依次除于0，1，2，3，4，5，并打印
```

  - except ZeroDivisionError #除非发生ZeroDivisionError类型的错误，执行下列语句

```
print('0是不能做除数的！')
```

- 九、其他
  - str.spilt()#返回一个由字符串内单词组成的列表

```
>>>'1,2,3'.split()
['1',',','2',',','3']
>>>'1,2,3'.split('?,')
['1','2','3']
```

  - random模块 #随机模块

```
import random
#需要先导入random模块，然后再调用相应方法
print(random.randint(1,10))      # 产生 1 到 10 的一个整数型随机数
print(random.random())          # 产生 0 到 1 之间的随机浮点数
print(random.uniform(1.1,5.4))   # 产生  1.1 到 5.4 之间的随机浮点数，区间可以不是整数
print(random.choice('tomorrow')) # 从序列中随机选取一个元素
print(random.randrange(1,100,2)) # 生成从1到100的间隔为2的随机整数
```

  - 转义字符

```
转义字符 意义
\a 响铃(BEL)
\b 退格(BS)，将当前位置移到前一列
\f 换页(FF)，将当前位置移到下页开头
\n 换行(LF)，将当前位置移到下一行开头
```

\r 回车(CR)，将当前位置移到本行开头
\t 水平制表(HT) (跳到下一个TAB位置)
\v 垂直制表(VT)
\\ 代表一个反斜杠字符"\"
\' 代表一个单引号(撇号)字符
\" 代表一个双引号字符
\? 代表一个问号
\0 空字符(NUL)
\ddd 1到3位八进制所代表的任意字符
\xhh 1到2位十六进制所代表的任意字符
注意1：区分斜杠“/”和反斜杠“\”，此处不可互换。 注意2：以上表格内容也不需要硬记。