基础操作

```
命令模式:代码块外侧蓝色 （编辑模式—>命令模式  esc）

X剪切
C复制
V粘贴
Z撤销
A在当前代码块上方插入空代码块
B在当前代码块下方插入空代码块
DD删除

M代码模式转为markdown(标记)模式
Ymarkdown(标记)模式转为代码模式

shift选中要合并的连续的代码块 再按shift+M合并
```

```
编辑模式:代码块外侧绿色（命令模式—>编辑模式  鼠标点击代码块里面,光标放到代码块里）

ctrl+enter 执行当前代码块,执行后停在当前代码块
shift+enter 执行当前代码块并移动到下一个代码块
alt+enter 执行当前代码块并在下方插入一个空的代码块

ctrl+x 剪切
ctrl+c 复制
ctrl+v 粘贴
ctrl+z 撤销

ctrl+shift+ -  光标放在要分割的位置再按这三个键进行代码块的分割
```

基础知识

markdown基本语法

1.语法标题介绍

一级标题 # +文字

二级标题 ## +文字

三级标题 ### +文字

四级标题 #### +文字

注意：井号和文字之间需要以空格分隔

2.粗体

文本前后各两个星号

3.斜体

文本前后各一个星号

4.斜体加粗

文本前后各三个星号

5.删除线

文本前后各两个波浪线

6.引用

一个大于号表示引用

7.分割线

方式一：用三个以上的减号表示分割线，减号和文本之间需要加入空行

方式二：用三个以上的星号表示分割线，星号和文本之间不需要加入空行

8.表格语法

表头|表头|表头|

------|------|------|

内容|内容|内容|

内容|内容|内容|

注意竖线的数量

9.超链接

10.不需要运行的代码块

Python代码语法

1.查看帮助文档

```
In [ ]:  1 print() #最常用 shift+tab键调出帮助文档

In [2]:  1 ?print

In [3]:  1 help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

2.python注释功能

1.2.2 python的注释功能

注释：即你不想进行输出/执行，但感觉有必要存在于代码结构中的内容。

Tips:

这部分内容存在的主要作用是：

1. **方便程序调试，当代码整体不需要全部运行的时候，可以将没有必要执行的部分进行注释，这部分内容将不会被执行。**
2. **方便加入对代码的解释描述，这样在代码交付的时候，就不用手把手来给对方进行讲解了。**

加入注释主要有以下两种方式：

a. `'''需要被注释的内容'''` --- 适用于多行注释

b. `#需要被注释的内容` --- 适用于单行注释

- 来感受一下

小技巧：选中所有需要注释的行，然后按下"ctrl + /" 键，可以试试效果

3.全局输出控制语句

1.2.3 全局输出控制语句

In [9]:
```
# 设置全部行输出
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

上述代码表示，在输出结果时，以单元格为单位，令所有可以被输出的对象，依次进行输出。

4.python的print功能

打印，把函数的结果打印到屏幕

In [12]:
```
print('小明')
```
小明

In [13]:
```
print('a')
```
a

In [14]:
```
print(a) #没有使用引号,是变量a
```
```
NameError                                 Traceback (most recent call last)
<ipython-input-14-bca0e2660b9f> in <module>
----> 1 print(a)

NameError: name 'a' is not defined
```

In [ ]:
```
print('')
```
```
Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
```

print 内的两个参数sep和end

```
In [15]: print('大明','小明') #sep参数控制一个print内的多个值以什么进行连接,默认值是空格,当不设置的时候,一个print内的多个值以空格进行连接
大明 小明

In [16]: print('小明','大明',sep='**')
小明**大明

In [17]: print('小明','大明',sep='**') #end参数控制多个print之间以什么进行连接,默认值是'\n',也就是换行
         print('小明','大明',sep='**')
小明**大明
小明**大明

In [19]: print('你','我','他',sep='和',end='加上')
         print('你','我','他',sep='和')
你和我和他加上你和我和他

In [20]: print('你','我','他',sep='和',end='==')
         print('你','我','他',sep='和')
你和我和他==你和我和他
```

5.获取用户输入

6.python中的变量

1.2.6.1 单个变量的赋值

```
In [32]: b=2

In [33]: print(b)
2

In [35]: d
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-35-e983f374794d> in <module>
----> 1 d

NameError: name 'd' is not defined
```

1.2.6.2 多变量同时赋值

```
In [36]: x=y=z=2 #链式赋值,值相同

In [37]: x
         y
         z
Out[37]: 2
Out[37]: 2
Out[37]: 2

In [38]: id(x)
         id(y)
         id(z)
Out[38]: 140736085763904
Out[38]: 140736085763904
Out[38]: 140736085763904
```

拆包式赋值

```
In [39]: m,n,j=5,8,'abc' #拆包式赋值,按照位置一一对应

In [40]: m
         n
         j
Out[40]: 5
Out[40]: 8
Out[40]: 'abc'
```

1.2.6.3 交换两个变量的赋值

```
In [43]:
1 m, n=n, m
```

```
In [44]:
1 m
2 n
```

```
Out[44]: 5
```

```
Out[44]: 8
```

1.2.6.4 变量可以被重复赋值

对同一个变量可以重复赋值,而且重新赋值的数据类型可以和开始赋值的数据类型不同

注意,由于对同一个变量可以重复赋值,所以在使用变量前,确定好变量的赋值内容

变量可以被重复赋值的原因:变量赋值实际上是变量对值的一个引用,而不是变量的取值

```
In [45]:
1 a
```

```
Out[45]: 1
```

```
In [46]:
1 a=5
2 a
```

```
Out[46]: 5
```

```
In [47]:
1 a='你好'
2 a
```

```
Out[47]: '你好'
```

1.2.6.5 变量名区分大小写

```
In [48]:
1 A
```

```
NameError                                 Traceback (most recent call last)
<ipython-input-48-7d157d7c000a> in <module>
----> 1 A

NameError: name 'A' is not defined
```

```
In [49]:
1 A=9
```

```
In [50]:
1 a
2 A
```

```
Out[50]: '你好'

Out[50]: 9
```

一行多句

```
In [65]:
1 a=1 ; b=2
```

一句多行

```
In [66]:
1 print("中心极限定理说明，在适当的条件下，大量相互独立随机变量的均值经适当标准化后依分布收敛于正 态分布。这组定理是数理统计学和误差分
```

中心极限定理说明，在适当的条件下，大量相互独立随机变量的均值经适当标准化后依分布收敛于正 态分布。这组定理是数理统计学和误差分析的理论基础，指出了大量随机变量之和近似服从正态分布的条件。

```
In [69]:
1 print("中心极限定理说明，在适当的条件下，\
2 大量相互独立随机变量的均值经适当标准化后依分布收敛于正 态分布。\
3 这组定理是数理统计学和误差分析的理论基础，指出了大量随机变量之和近似服从正态分布的条件。") #'\'续行符
```

中心极限定理说明，在适当的条件下，大量相互独立随机变量的均值经适当标准化后依分布收敛于正 态分布。这组定理是数理统计学和误差分析的理论基础，指出了大量随机变量之和近似服从正态分布的条件。

```
In [73]:
1 print("""中心极限定理说明，在适当的条件下，
2 大量相互独立随机变量的均值经适当标准化后依分布收敛于正 态分布。
3 这组定理是数理统计学和误差分析的理论基础，
4 指出了大量随机变量之和近似服从正态分布的条件。""")
```

```
中心极限定理说明，在适当的条件下，
大量相互独立随机变量的均值经适当标准化后依分布收敛于正 态分布。
这组定理是数理统计学和误差分析的理论基础，
指出了大量随机变量之和近似服从正态分布的条件。
```