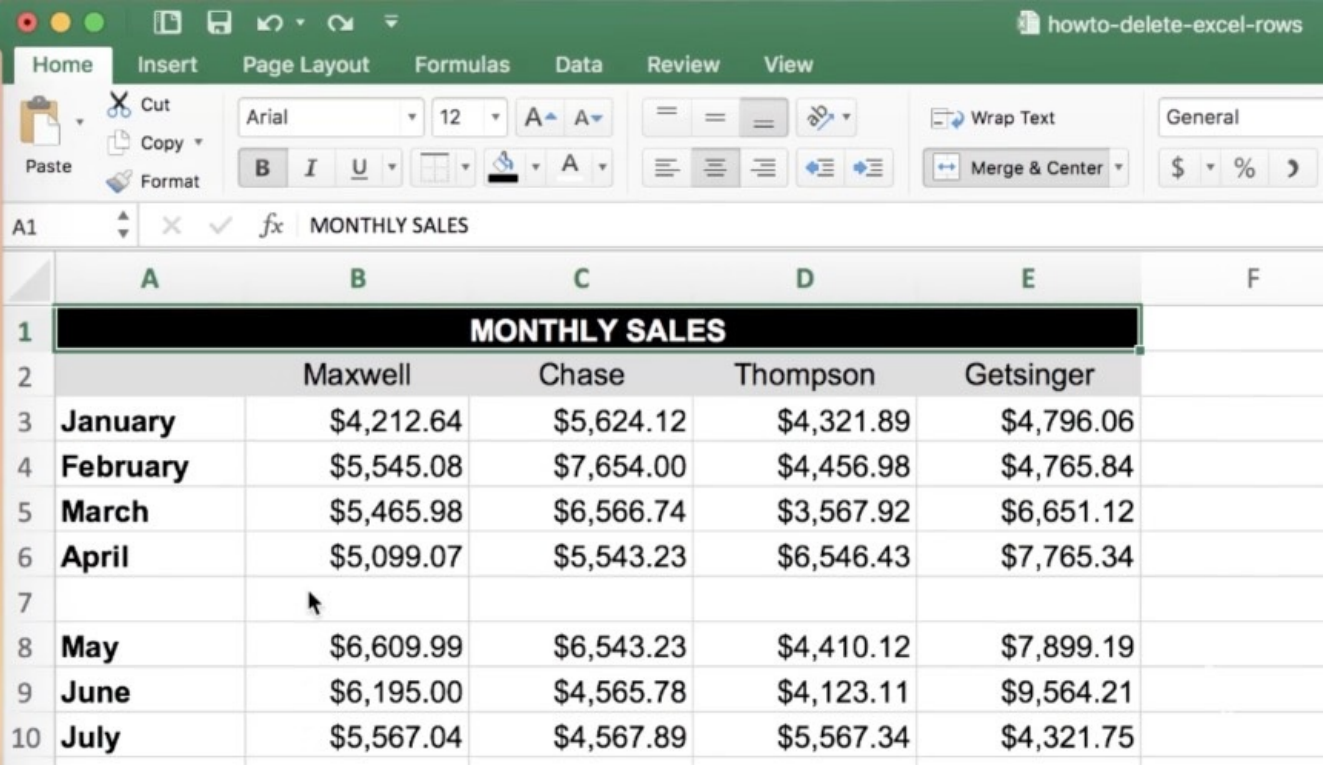


1 引子

Excel表格

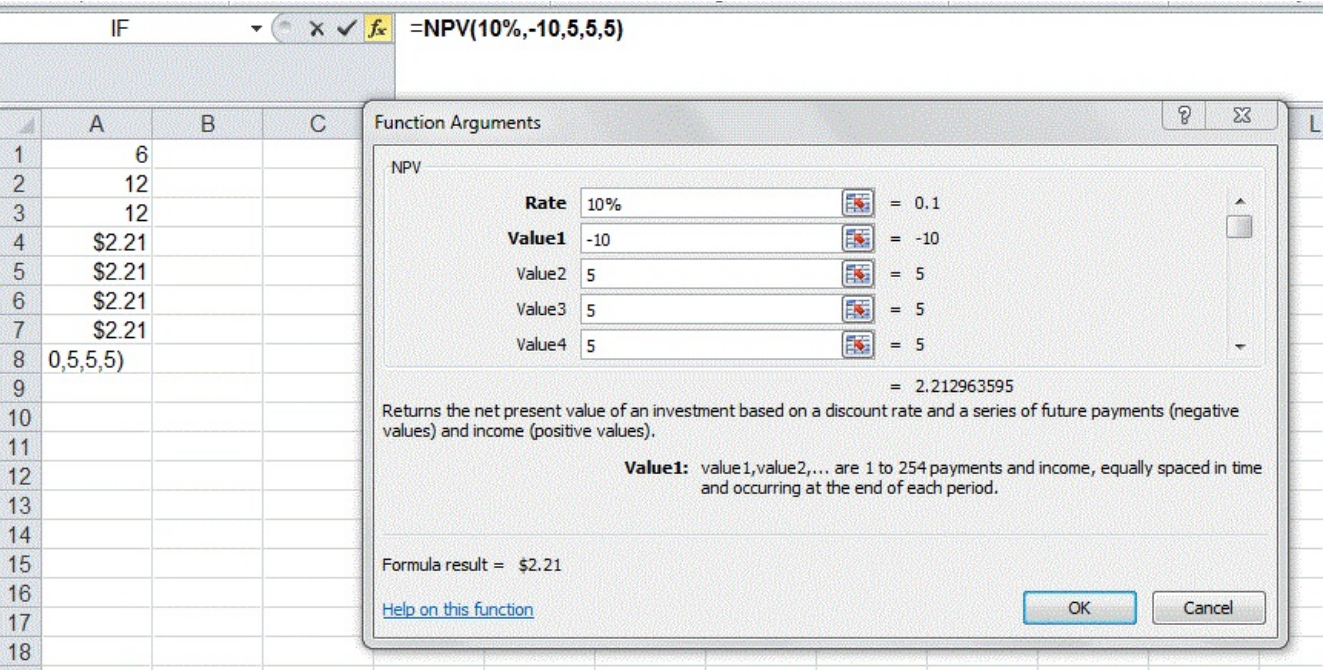
Excel表格是微软出的Office办公套件，相信大家都不陌生，尤其是在座各位在税务局工作的朋友，平时做数据处理、统计分析、辅助决策都能用得上



MONTHLY SALES				
	Maxwell	Chase	Thompson	Getsinger
January	\$4,212.64	\$5,624.12	\$4,321.89	\$4,796.06
February	\$5,545.08	\$7,654.00	\$4,456.98	\$4,765.84
March	\$5,465.98	\$6,566.74	\$3,567.92	\$6,651.12
April	\$5,099.07	\$5,543.23	\$6,546.43	\$7,765.34
May	\$6,609.99	\$6,543.23	\$4,410.12	\$7,899.19
June	\$6,195.00	\$4,565.78	\$4,123.11	\$9,564.21
July	\$5,567.04	\$4,567.89	\$5,567.34	\$4,321.75

函数公式

平时做财务报表，难免要用到一些加减乘除数学运算，如果全靠自己手动算，然后再输入，不可避免地会出现小错误，所以我们希望报表能自动计算，于是就有了Excel公式，但Excel的公式不太好写，而且也只能处理较简单的逻辑



IF $\text{=NPV}(10\%,-10,5,5,5)$

A	B	C
1	6	
2	12	
3	12	
4	\$2.21	
5	\$2.21	
6	\$2.21	
7	\$2.21	
8	0,5,5,5)	
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		

Function Arguments

NPV

Rate: 10% = 0.1

Value1: -10 = -10

Value2: 5 = 5

Value3: 5 = 5

Value4: 5 = 5

= 2.212963595

Returns the net present value of an investment based on a discount rate and a series of future payments (negative values) and income (positive values).

Value1: value1,value2,... are 1 to 254 payments and income, equally spaced in time and occurring at the end of each period.

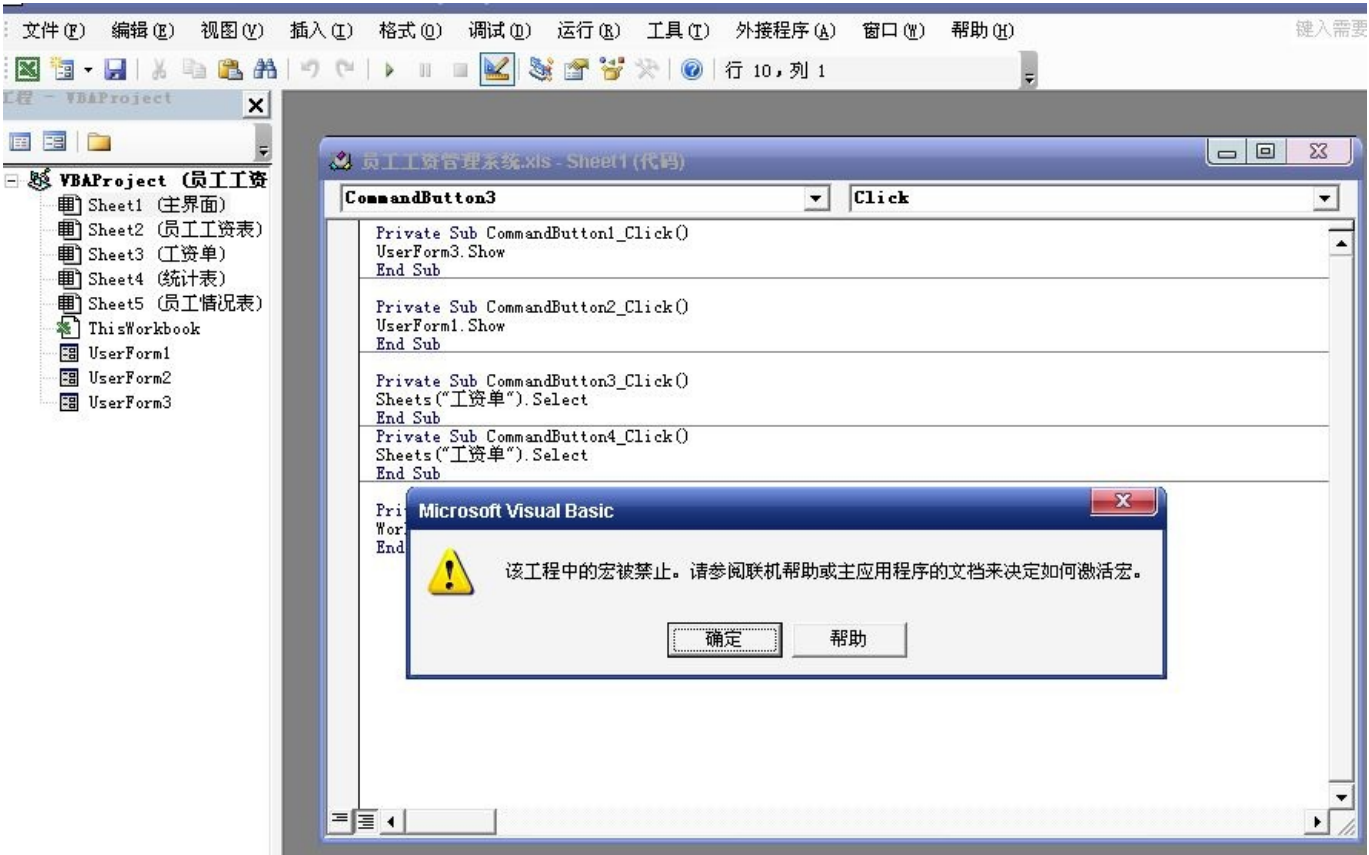
Formula result = \$2.21

[Help on this function](#)

OK Cancel

VBA编程

较复杂的逻辑就得靠VBA编程来实现了，VBA全称叫Visual Basic for Applications，这是微软开发的用来扩展Windows应用程序功能的一种语言



但现阶段学VBA并不是一个明智的选择：

- **不容易学**，没有好用的开发环境，错误调试不方便
- **性能不行**，在数据量大的情况下运行很耗时
- **适用面窄**，只能服务于Windows系统

既然要学一门编程语言，为何不与时俱进，学一门最时髦最先进的语言呢？在当前的国际形势下，自主可控的国产操作系统是大势所趋，大概率是Linux的某个发行版，Windows终将被放弃！

高级程序语言



计算机行业经过几十年的高速发展，诞生了形形色色的高级程序语言，它们都是应各种不同的需求而诞生，比如

- C / C++：高效，系统编程，科学计算
- Java：跨平台特性好，网络编程
- JavaScript / PHP：交互性好，嵌入浏览器

为什么要学Python？

网上有很多人说：“人生苦短，我用Python！”，为啥？最直接的原因是简单好用，对非程序员友好

具体来说：

- **简单轻量**：学起来快，稍有编程经验的人一周就能上手，不像C / C++那么heavy
- **功能强大**：Web开发、科学计算、网络爬虫、数据分析、量化交易，几乎无所不能
- **社区活跃**：大量的第三方库让你永远可以站在巨人的肩膀上，无需浪费时间重复造轮子
- **免费使用**：任何人、组织、机构都可以免费使用Python，不像Java那么商业化

Python的劣势：

- 运行慢？混合编程，底层计算用C / C++ / Fortran来实现
- 不适合大项目？知乎、豆瓣、Youtube、Instagram
- 版本多、依赖容易崩？Windows下可以用Anaconda管理，类Unix系统下可以用Pacman、Homebrew等包管理器

一个例子

C++ :

```
#include <iostream>

int main() {
    std::cout << "你好，华科！"; return 0;
}
```

Pyhton :

```
print("你好，华科！")
```

同样是打印“你好，华科！”到屏幕，对比可以看出Python的简洁

动手试一下：

```
In [4]:
print("你好，华科！")
```

你好，华科！

编译型语言 vs. 解释性语言

- 编译型语言：事先将高级程序语言翻译成机器语言（俗称编译），然后再运行，执行效率高
- 解释性语言：执行一句翻译一句，不需要先编译，交互性好，执行效率低

C / C++属于前者，Python属于后者，由于它有很好的交互性，所以我选择不用ppt，而是这样一个方便交互和展示的形式

2 环境搭建

Windows / Mac

1. 在[Python官网 \(https://www.python.org/downloads/release/python-380/\)](https://www.python.org/downloads/release/python-380/)分别下载 Windows x86-64 executable installer 和 macOS 64-bit installer 安装包，最新版本是Python3.8：

python.org/downloads/release/python-380/

Files

Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later
Windows help file	Windows	
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64
Windows x86 embeddable zip file	Windows	
Windows x86 executable installer	Windows	
Windows x86 web-based installer	Windows	

2. Windows下安装时注意勾选**将Python添加到环境变量**，否则在非系统路径下无法调用Python。

Linux

Linux下比较简单，各个发行版的官方软件仓库里都有Python，直接在Shell中安装即可，有的发行版甚至装机自带

- Arch系：sudo pacman -S python
- Debian系：sudo apt-get install python3.*
- RedHat系：sudo yum install python3*

讲者的电脑用的是Arch Linux，所以安装过程如下（官方软件仓库里还没更新到3.8，版本还是3.7.4）：

```
[murongxixi@murongxixi-xps] - [~] - [五 11月 08, 04:09]
[$] <> sudo pacman -S python
警告：python-3.7.4-2 已经为最新 -- 重新安装
正在解析依赖关系...
正在查找软件包冲突...

软件包 (1)  旧版本  新版本  净变化  下载大小
extra/python  3.7.4-2  3.7.4-2  0.00 MiB  34.10 MiB

下载大小：      34.10 MiB
全部安装大小：   144.87 MiB
净更新大小：     0.00 MiB

:: 进行安装吗？ [Y/n]
```

安装成功后，Windows下进入命令行，Mac / Linux进终端，输入python 回车出现如下界面就代表环境搭建成功了：

```
[murongxixi@murongxixi-xps] - [~] - [五 11月 08, 04:19]
[$] <> python
Python 3.7.4 (default, Oct  4 2019, 06:57:26)
[GCC 9.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

开发环境

直接在命令行里写代码显然是痛苦且低效的，推荐两个开发环境：

1. vscode + python + tab nine + code runner + bracket pair colorizer，vscode是现在非常流行的文本编辑器，拥有丰富的第三方插件，装上后面四个插件就可以高效地写代码了，tab nine负责智能补全，bracket pair colorizer括号配对着色，code runner右键直接运行

untitled.py ×

Notes > Talk > 191112-python-intro > files > untitled.py > ...

1 for i in range(5):

2 print(i**2)

3 print("你好，华科！")

4 print(2+3)

5

问题 输出 调试控制台 终端

[Running] python -u "/home

0

1

4

9

16

你好，华科！

5

Run Code

转到定义

查看定义

Find All References

查看引用

重命名符号

更改所有匹配项

格式化文档

格式化文档，方法是使用...

源代码操作...

剪切

1. jupyter lab / notebook，现在用的就是这个，交互性好，可以同时运行Python代码和显示Markdown文本，非常适合用来展示

3 基础知识

除了在命令行交互环境中运行外，还可以将若干条python语句写在一个.py文件中，然后用命令`python .py`运行
如下图所示，在untitled.py文件中写入了`print("你好， 华科！")`和`print(2+3)` 两条打印语句

```
[murongxixi@murongxixi-xps] - [~] - [五 11月 08, 08:56]
[$] <> echo "print(\"你好， 华科！\")" > untitled.py
[murongxixi@murongxixi-xps] - [~] - [五 11月 08, 08:56]
[$] <> echo "print(2+3)" >> untitled.py
[murongxixi@murongxixi-xps] - [~] - [五 11月 08, 08:56]
[$] <> cat untitled.py
print("你好， 华科！")
print(2+3)
[murongxixi@murongxixi-xps] - [~] - [五 11月 08, 08:56]
[$] <> python untitled.py
你好， 华科！
5
[murongxixi@murongxixi-xps] - [~] - [五 11月 08, 08:57]
[$] <> █
```

保留字

保留字都有特殊含义，不能用来命名别的变量，前面见过的用来打印输出的`print` 就是一个保留字

```
and    exec    not        assert  finally  or       break  for      pass    class
from   print   continue  global  raise    def     if      return  del     import
try    elif    in         while   else     is      with    except  lambda  yield
```

行和缩进

与C / C++不同，Python的代码块不使用大括号{} 来判断，而是用缩进来表示

动手试一下：

```
In [359]:
if 3 > 2:
    print(3)
else:
    print(2)
    print(1)  # 与else相关

3
```

```
In [356]:
if 3 > 2:
    print(3)
else:
    print(2)
print(1)  # 与else无关

3
1
```


引号

Python使用单引号'、双引号"、三引号''' 或""" 来表示字符串，引号的开始与结束必须相同

其中三引号可以由多行组成，常用于做注释，单行注释也可用# 来表示

动手试一下：

```
In [360]:  
  
"""  
  
    这个程序用来测试缩进  
    这个程序用来测试缩进  
    """  
  
# 这个程序用来测试缩进  
if 3 > 2:  
    print('你好')  
else:  
    print("华科")  
  
你好
```

4 基础语法

变量

- 数值型变量

动手试一下：

```
In [21]:  
  
a = 2  
b = 3  
print(a+b)  
  
5
```

- 字符串变量

动手试一下：

```
In [18]:  
  
a = "你好"  
b = "华科"  
print(a + "，" + b + "！")  
  
你好，华科！
```

```
In [29]:  
  
str = "你好，华科！"      # 一共6个字符 0:你 1:好 2:， 3:华 4:科 5:!  
print(str)                # 完整字符串  
print(len(str))           # 字符串长度  
print(str[0])             # 字符串中的第一个字符  
print(str[2:5])           # 字符串中第3~5个  
print(str[2:])            # 从第3个字符开始到结束  
print(str * 2)            # 输出字符串两次  
print(str + "你好，武汉！") # 连接字符串 上一个例子已经用过
```

```
你好，华科！  
6  
你  
， 华科  
， 华科！  
你好，华科！你好，华科！  
你好，华科！你好，武汉！
```


列表

字符串是一种特殊的列表，列表允许元素不同类型

动手试一下：

```
In [18]:  
  
list1 = ["你好", 2, "华科", 3] # 列表用中括号[]  
list2 = ["Hello", 'HUST']  
  
print(list1)           # 完整列表  
print(list1[0])         # 列表的第一个元素  
print(list1[1:3])       # 第2~3个元素  
print(list1[2:])        # 从第3个元素开始到结束  
print(list1 * 2)        # 输出列表两次  
print(list1 + list2)    # 连接列表  
print([list1, list2])   # 列表的元素可以还是列表 即嵌套  
  
['你好', 2, '华科', 3]  
你好  
[2, '华科']  
['华科', 3]  
['你好', 2, '华科', 3, '你好', 2, '华科', 3]  
['你好', 2, '华科', 3, 'Hello', 'HUST']  
[['你好', 2, '华科', 3], [['Hello', 'HUST']]]
```

排序，求和

动手试一下：

```
In [246]:  
  
list = [2, 4, 1, 3]  
list.sort() # 默认升序  
print(list)  
  
list = [2, 4, 1, 3]  
list.sort(reverse = True) # 降序  
print(list)  
  
print(sum(list)) # 求和  
  
[1, 2, 3, 4]  
[4, 3, 2, 1]  
10
```

字典

列表是有序的对象集合，字典是无序的对象集合，由索引（key）和它对应的值（value）组成

字典当中的元素是通过键来存取的，而不是通过下标存取

动手试一下：

```
In [31]:  
  
dict = {} # 字典用大括号{}  
dict['Hello'] = "你好"  
dict["HUST"] = "华科"  
  
print(dict['Hello']) # 输出键为Hello的值  
print(dict['HUST']) # 输出键为HUST的值  
print(dict.keys())  # 输出所有键  
print(dict.values()) # 输出所有值  
  
你好  
华科  
dict_keys(['Hello', 'HUST'])  
dict_values(['你好', '华科'])
```

5 运算

算数运算

- 加法，减法，乘法，除法，乘方

动手试一下：

```
In [65]:
1 + 2, 2 - 1, 2 * 3, 3 / 5, 2**3, 4**(1/2)

(3, 1, 6, 0.6, 8, 2.0)
```

- 取模，取整

动手试一下：

```
In [365]:
a = 13%3; b = 13%5; c = 13//3; d = 13//5
print(a)
print(b)
print(c)
print(d)

1
3
4
2
```

- 对数

动手试一下：

```
In [361]:
'''
引入Python自带的math模块
import也是Python保留字
'''
import math
math.log2(8), math.log10(100), math.log(math.exp(5))

(3.0, 2.0, 5.0)
```

通过按tab键提示可以看到math模块里还有（反）三角函数

```
[66... import math      # 引入math模块
math.log2(8), math.log10(100), math.log(math.exp(5))
math.

[66... (3.141592653589793, 2.81828459045)
三角 f acos      function
      f acosh     function
      f asin      function
      f asinh     function
[68... math f atan      function      cos(math.pi/2), math.tan(math
      f atan2     function
[68... (1.0, 0.9999999999999999)
向_ f ceil      function
      f convsign  function
```

- 正弦，余弦，正切

动手试一下：

```
In [362]:
math.sin(math.pi/2), math.cos(math.pi/2), math.tan(math.pi/4)

(1.0, 6.123233995736766e-17, 0.9999999999999999)
```

- 向上取整，向下取整

动手试一下：

```
In [69]:
math.ceil(3.6), math.floor(3.6)

(4, 3)
```

比较运算

判断某条语句是否成立


```
In [366]:  
  
print(a,b,c,d)  # 上面取模取整的输出  
if a < b:  
    print("a小于b")  
if a <= b:  
    print("a小于等于b")  
if a == 1:  
    print("a等于1")  
if a != b:  
    print("a不等于b")
```

1 3 4 2
a小于b
a小于等于b
a等于1
a不等于b

大于> 和大于等于>= 同理，不赘述

逻辑运算

- 与：全部都成立才算成立

动手试一下：

```
In [47]:  
  
if a < b and c > d:  
    print("a小于b 和 c大于d 同时成立")  
if a < b and c <= d:  
    print("a小于b 和 c小于等于d 同时成立")
```

a小于b 和 c大于d 同时成立

- 或：有一个成立就成立

动手试一下：

```
In [49]:  
  
if a < b or c <= d:  
    print("a小于b 和 c小于等于d 同时成立")  
if a > b or c <= d:  
    print("a大于b 和 c小于等于d 同时成立")
```

a小于b 和 c小于等于d 同时成立

- 非：取反

动手试一下：

```
In [50]:  
  
if not a < b:  
    print("a小于b 不成立")  
if not a >= b:  
    print("a大于等于b 不成立")
```

a大于等于b 不成立

成员运算

判断列表是否包含某个元素

动手试一下：

```
In [51]:  
  
list1 = ["你好", 2, "华科", 3]  # 列表用中括号[]  
list2 = ["Hello", 'HUST']  
if 2 in list1:  
    print("2在列表1中")  
else:  
    print("2不在列表1中")  
  
if 'H' not in list2[0]:  
    print("H不在列表2中")  
else:  
    print("H在列表2中")
```

2在列表1中
H在列表2中

6 循环

- while循环

动手试一下：

```
In [83]:  
  
i = 0  
while i < 5:  
    print(i)  
    i += 1    # i = i + 1
```

0
1
2
3
4

- for循环

动手试一下：

```
In [116]:  
  
for i in range(5):  # range可以很方便的构建列表 range(5) = [0, 1, 2, 3, 4]  
    print(i**2)
```

0
1
4
9
16

```
In [87]:  
  
for i in range(0,5,2):  # range(0,5,2) = [0, 2, 4] 0是开始数 5是结束数（不包括） 相邻两数差为2  
    print(i)
```

0
2
4

- break 跳出循环，循环的剩余部分不再执行

动手试一下：

```
In [138]:  
  
for i in range(5): # range(5) = [0, 1, 2, 3, 4]  
    print(i)  
    if i == 3:  
        break  
  
0  
1  
2  
3
```

- continue 忽略本轮循环剩余部分

动手试一下：

```
In [90]:  
  
for i in range(5): # range(5) = [0, 1, 2, 3, 4]  
    if i == 3:  
        continue  
    print(i)  
  
0  
1  
2  
4
```

7. 字符串

前面已经见过字符串的简单操作

```
In [137]:  
  
str = "你好，华科！" # 一共6个字符 0:你 1:好 2:， 3:华 4:科 5:!  
print(str) # 完整字符串  
print(len(str)) # 字符串长度  
print(str[0]) # 字符串中的第一个字符  
print(str[2:5]) # 字符串中第3~5个  
print(str[2:]) # 从第3个字符开始到结束  
print(str * 2) # 输出字符串两次  
print(str + "你好，武汉！") # 连接字符串 上一个例子已经用过
```

```
你好，华科！  
6  
你  
， 华科  
， 华科！  
你好，华科！你好，华科！  
你好，华科！你好，武汉！
```

替换

动手试一下：

```
In [136]:  
  
str = "你好，华科！你好，武汉！"  
str2 = str.replace("你好", "Hello")  
print(str2)  
str3 = str.replace("你好", "Hello", 1) # 第三个参数指定替换次数  
print(str3)  
  
Hello，华科！Hello，武汉！  
Hello，华科！你好，武汉！
```


同样通过按tab键提示可以看看字符串还有哪些函数

```
[ ]: str = "你好，华科！你好，武汉！"  
      str2 = str.|
```

f	capitalize	function
f	casefold	function
f	center	function
f	count	function
f	encode	function
f	endswith	function
f	expandtabs	function
f	find	function
f	format	function

查找

动手试一下：

```
In [106]:  
  
str = "你好，华科！你好，武汉！你好，湖北！你好，中国！"  
print(str.find("华科"))      # 找到子串则返回子串首字符在原字符串中的下标  
print(str.find("华科大"))    # 找不到返回-1
```

```
3  
-1
```

分割

动手试一下：

```
In [382]:  
  
str = "你好，华科！你好，武汉！你好，湖北！你好，中国！"  
str2 = str.split('! ') # 以！对字符串str进行分割  
for a in str2:  
    print(a)
```

```
你好，华科  
你好，武汉  
你好，湖北  
你好，中国
```

合并

动手试一下：

```
In [384]:  
  
str_list = ["你好，华科", "你好，武汉", "你好，湖北", "你好，中国", ""]  
"! ".join(str_list) # 以！为分隔符，将str_list中的所有字符串合并起来 多用于将路径合并成完整路径
```

```
'你好，华科！你好，武汉！你好，湖北！你好，中国！'
```

正则表达式

正则表达式 (Regular Expression) 是描述某种规则的表达式，通常被用来检索或替换那些符合某个模式的文本，比上面的精确匹配更强大

下面是一个识别合法ip的例子，注意ip由4个0-255的数字组成，因此每个数字有三种可能：

- 非零个位数，因此是 $[1-9]$
- 两位是，十位不能是0，因此是 $[1-9][0-9]$
- 三位数，百位只能是1或2，因此是 $[1-2][0-9][0-9]$

动手试一下：

```
in [374]:  
  
import re # 导入python自带的re模块  
  
str = '192.168.1.100'  
m = re.match('([1-9]|[1-9][0-9]|[1-2][0-9][0-9])\.([1-9]|[1-9][0-9]|[1-2][0-9][0-9])\.([1-9]|[1-9][0-9]|[1-2][0-9][0-9])\.([1-9]|[1-9][0-9]|[1-2][0-9][0-9])$', str)  
m.groups()
```

| 表示或，因为每个数字有三种合法的形式；\. 表示点号本身，因为点号在正则表达式中表示任意字符，需要反斜杠转义

下面是一个从字符串列表中提取手机号和身份信息例子，由于有的手机号有区号，有的没区号，身份信息有的用身份证，有的用护照，所以只能求助于正则表达式

动手试一下：

```
list = ["小平的手机号1389876543，身份证号320811190012344321。",
        "小强的手机号+861591234567，身份证号40140511200056788765。",
        "Trump的手机号+1-222-333-4444，护照号E12345789。"]

for str in list:
    print(str)
```

小平的手机号1389876543，身份证号320811190012344321。

小强的手机号+861591234567，身份证号40140511200056788765。

Trump的手机号+1-222-333-4444，护照号E12345789。

```

In [373]:
pattern = re.compile(r'([1+][0-9-]*)', ' ') # 手机号匹配模式
for str in list:
    r = pattern.findall(str) # findall查找所有的匹配
    print(r)
print("-----")
pattern = re.compile(r'号([0-9]{18}|[A-Z][0-9]{8})') # 身份信息匹配模式 {}中的数字表示前面[]中的模式出现的次数
for str in list:
    r = pattern.findall(str) # findall查找所有的匹配
    print(r)

```

```

['1389876543']
['+861591234567']
['+1-222-333-4444']

```

```

['320811190012344321']
['401405112000567887']
['E12345789']

```

- 手机号匹配模式是([1+][0-9-]*), , 表示以1 或+ 起始, 以, 结束, 由数字0-9 和- 构成的字符串序列
- 身份信息匹配模式是([0-9]{18}), 表示18位身份证数字; 或者([A-Z][0-9]{8}), 表示1位字母加8位数字的9位护照

下面是一个删掉注释的例子

动手试一下：

```
In [376]:  
  
list = ["小平的手机号1389876543，身份证号320811190012344321。",  
        "小强的手机号+861591234567，身份证号40140511200056788765。 # +86是中国区号",  
        "Trump的手机号+1-222-333-4444，护照号E12345789。 # 外国友人没有身份证"]  
  
for str in list:  
    print(str)  
  
print("-----")  
pattern = re.compile(r'#.*$')    # .表示任意字符 #.*表示#打头任意长的字符串  
for str in list:  
    print(pattern.sub("", str))  # sub将所有匹配替换为空字符串 即删掉注释
```

小平的手机号1389876543，身份证号320811190012344321。
小强的手机号+861591234567，身份证号40140511200056788765。 # +86是中国区号
Trump的手机号+1-222-333-4444，护照号E12345789。 # 外国友人没有身份证

小平的手机号1389876543，身份证号320811190012344321。
小强的手机号+861591234567，身份证号40140511200056788765。
Trump的手机号+1-222-333-4444，护照号E12345789。

. 表示任意字符，\$ 表示字符串结尾，因此#.*\$ 表示# 打头任意长的字符串

8. 文件操作

文本文件

- 读文件

动手试一下：

```
In [179]:  
  
with open("files/untitled.py", "r", encoding="utf-8") as read_file:    # r表示以只读的方式打开  
    for line in read_file:    # 一行一行读  
        print(line)  
  
print("你好，华科！")  
  
print(2+3)
```

- 写文件

动手试一下：

```
In [182]:  
  
with open("files/untitled.py", "w+", encoding="utf-8") as write_file:    # w表示以可写入的方式打开  
    write_file.write("for i in range(5):\n")  
    write_file.write("    print(i**2)\n")
```



```

[murongxixi@murongxixi-xps] - [~/Notes/Talk/1
[$] <git:(master*)> cat files/untitled.py
for i in range(5):
    print(i**2)
[murongxixi@murongxixi-xps] - [~/Notes/Talk/1
[$] <git:(master*)> python files/untitled.py
0
1
4
9
16

```

不难发现原来的两条打印语句被覆盖了，只剩下新写入的两条语句了，如果想保留原有内容，要用追加模式打开文件动手试一下：

```

In [385]:
with open("files/untitled.py", "a+", encoding="utf-8") as write_file: # a+ 追加模式 append
    write_file.write("print(\"你好，华科！\")\n") # \"对双引号转义
    write_file.write("print(2+3)\n")

```

```

[murongxixi@murongxixi-xps] - [~/Notes/Talk/1
[$] <git:(master*)> cat files/untitled.py
for i in range(5):
    print(i**2)
print("你好，华科！")
print(2+3)
[murongxixi@murongxixi-xps] - [~/Notes/Talk/1
[$] <git:(master*)> python files/untitled.py
0
1
4
9
16
你好，华科！
5

```

Excel表格

Python可以处理Excel表格，这需要依赖第三方库 python-xlrd 和 python-xlwt，分别用来读Excel和写Excel

- Windows下可以通过Anaconda安装，**不推荐用pip安装**
- 类Unix系统下用系统自带的包管理器安装即可，例如Arch系用 `sudo pacman -S python-xlrd python-xlwt`

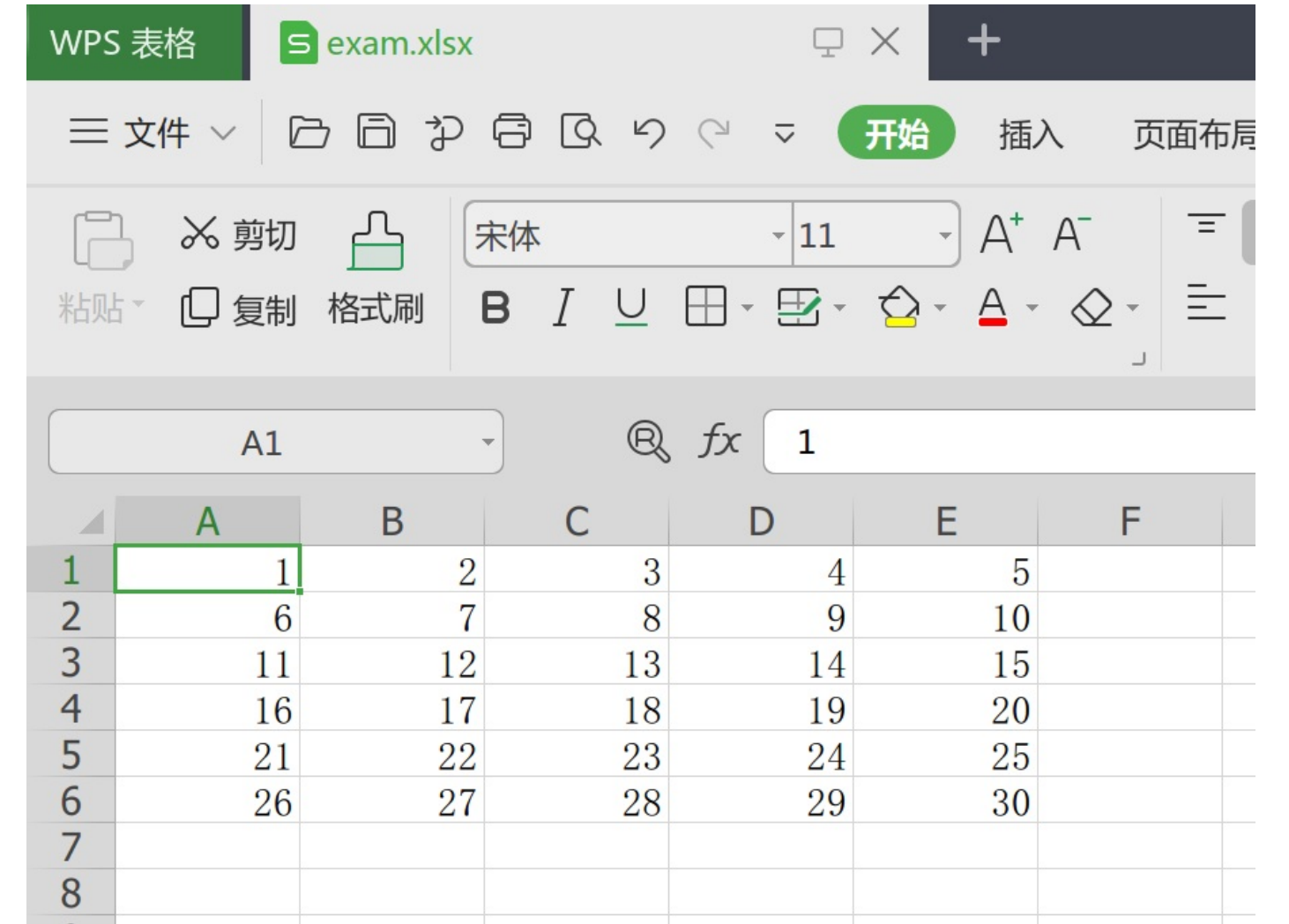
```
[murongxixi@murongxixi-xps] - [~/Notes/Talk/191112-python-i
[$] <git:(master*)> sudo pacman -S python-xlrd python-xlwt
正在解析依赖关系...
正在查找软件包冲突...

软件包 (2)                新版本  净变化  下载大小

community/python-xlrd    1.2.0-2  0.82 MiB
community/python-xlwt    1.3.0-3  1.04 MiB  0.15 MiB

下载大小:   0.15 MiB
全部安装大小:  1.87 MiB

:: 进行安装吗? [Y/n]
:: 正在获取软件包.....
python-xlwt-1.3.0-3-any
```



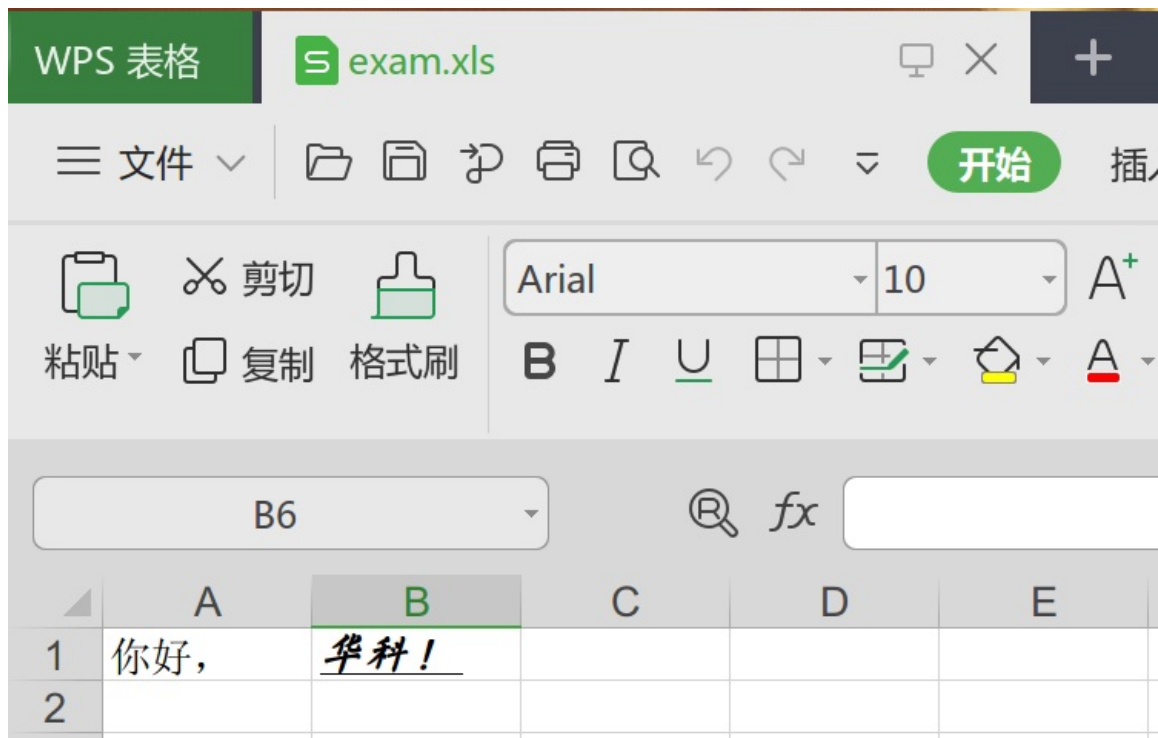
动手试一下：

```
In [196]:  
  
import xlrd  
data = xlrd.open_workbook('files/exam.xlsx')  
table = data.sheet_by_name('Sheet1')    # 获取sheet 1  
print(table.nrows, table.ncols)         # 行数 列数  
  
for i in range(table.nrows):  
    print(table.row_values(i))           # 打印第i行的值  
print("-----")  
for j in range(table.ncols):  
    print(table.col_values(j))           # 打印第j列的值  
print("-----")  
table.cell(2,3).value                    # 打印第3行第4列的单元格值
```

```
6 5  
[1.0, 2.0, 3.0, 4.0, 5.0]  
[6.0, 7.0, 8.0, 9.0, 10.0]  
[11.0, 12.0, 13.0, 14.0, 15.0]  
[16.0, 17.0, 18.0, 19.0, 20.0]  
[21.0, 22.0, 23.0, 24.0, 25.0]  
[26.0, 27.0, 28.0, 29.0, 30.0]  
  
[1.0, 6.0, 11.0, 16.0, 21.0, 26.0]  
[2.0, 7.0, 12.0, 17.0, 22.0, 27.0]  
[3.0, 8.0, 13.0, 18.0, 23.0, 28.0]  
[4.0, 9.0, 14.0, 19.0, 24.0, 29.0]  
[5.0, 10.0, 15.0, 20.0, 25.0, 30.0]
```

14.0

```
In [206]:  
  
import xlwt  
workbook = xlwt.Workbook()  
worksheet = workbook.add_sheet('My Sheet')  
font = xlwt.Font() # Create the Font  
font.name = '方正苏新诗柳楷简体-yolan'  
font.bold = True  
font.underline = True  
font.italic = True  
style = xlwt.XFStyle()           # 创建字体样式  
style.font = font  
worksheet.write(0, 0, '你好，')  
worksheet.write(0, 1, '华科!', style) # 应用字体样式到该单元格  
workbook.save('files/exam.xls')
```



9. 常用第三方库

NumPy

NumPy (Numerical Python) 支持高维数组和矩阵运算，提供大量的数学函数库，安装方法与前面的python-xlrd类似和，均值，标准差

动手试一下：

```
In [267]:
import numpy as np

a = np.array([2, 4, 1, 3]) # 将列表转换成数组
sum(a), np.mean(a), np.std(a)

(10, 2.5, 1.118033988749895)
```

```
In [268]:
b = np.argsort(a)
b

array([2, 0, 3, 1])
```

numpy数组的排序功能比Python自带的列表要强，argsort 返回的是元素的在排序后新列表中的位置，例如第二个元素0代表4是最大的，应该排在新列表中的最前面

二维数组

动手试一下：

```
In [264]:
a = np.array([[2, 4, 6], [5, 3, 1]]) # 将列表转换成数组
a

array([[2, 4, 6],
       [5, 3, 1]])
```


动手试一下：

改变形状

动手试一下：

有了强大的二维数组处理工具，可以将Excel表格中的数据全部导入Python中，用NumPy进行复杂的逻辑处理后，再写回到Excel中

最常用的绘图库

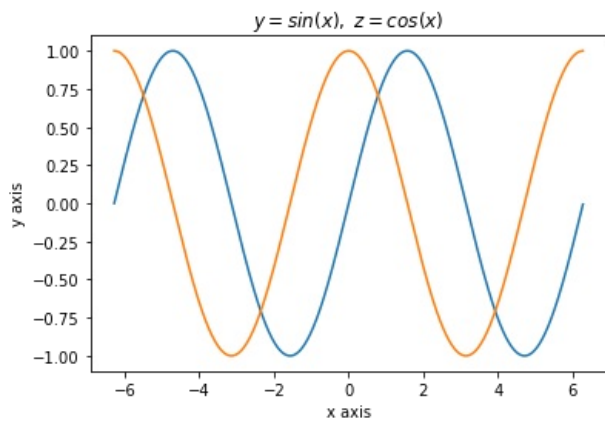
动手试一下：

```
In [313]:
# 画曲线走势
%matplotlib inline
import math
import numpy as np
from matplotlib import pyplot as plt

x = np.arange(-2*math.pi, 2*math.pi, 0.01)
y = []
z = []

for i in range(len(x)):
    y.append(math.sin(x[i]))
    z.append(math.cos(x[i]))
    i += 1

plt.title("$y = \sin(x), z = \cos(x)$")
plt.xlabel("x axis")
plt.ylabel("y axis")
plt.plot(x,y)
plt.plot(x,z)
plt.show()
```

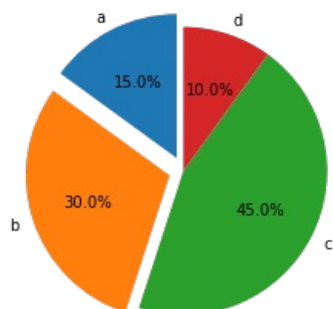


```
In [306]:
# 画饼状图
%matplotlib inline
import matplotlib.pyplot as plt

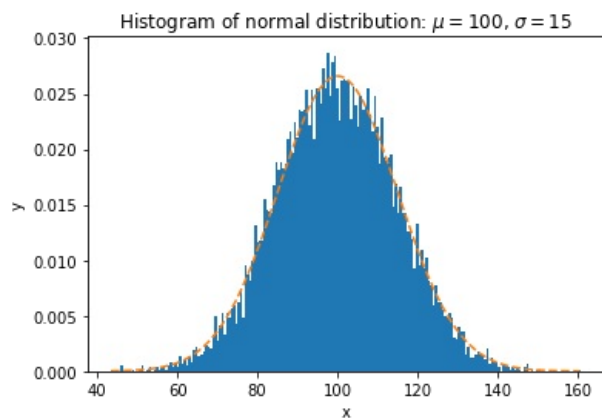
labels = 'a', 'b', 'c', 'd'
sizes = [15, 30, 45, 10]    # 百分比
explode = (0.1, 0.1, 0, 0) # a, b 饼块突出显示

plt.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=False, startangle=90)
plt.axis('equal')

plt.show()
```




```
In [339]:  
  
# 画条形图  
%matplotlib inline  
import matplotlib  
import numpy as np  
import matplotlib.pyplot as plt  
  
np.random.seed(19680801)  
  
mu = 100    # 正态分布均值  
sigma = 15  # 正态分布标准差  
x = mu + sigma * np.random.randn(10000)  
  
num_bins = 200  
  
n, bins, patches = plt.hist(x, num_bins, density=1)  
  
y = ((1 / (np.sqrt(2 * np.pi) * sigma)) * np.exp(-0.5 * (1 / sigma * (bins - mu))**2)) # 正态分布函数  
plt.plot(bins, y, '--')  
plt.xlabel('x')  
plt.ylabel('y')  
plt.title(r'Histogram of normal distribution:  $\mu=100$ ,  $\sigma=15$ ')  
plt.show()
```



SciKit-Learn

最流行的机器学习库，几乎包含所有经典的机器学习算法

动手试一下：

```
In [378]:
```

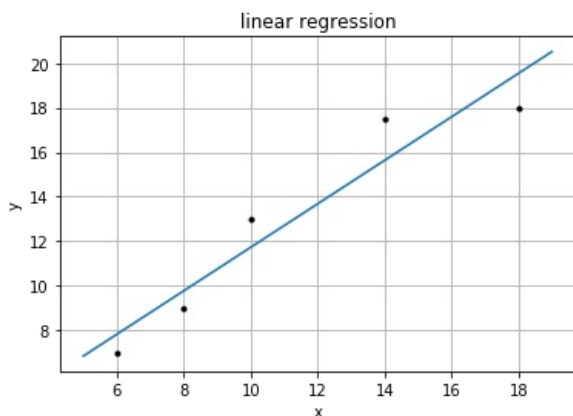
```
# 线性回归：用线性函数拟合数据
%matplotlib inline
import sklearn
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

x = np.array([6, 8, 10, 14, 18]).reshape(-1, 1) # 5个样本点构成的数据集
y = [7, 9, 13, 17.5, 18]

model = LinearRegression() # 选择学习模型：线性回归
model.fit(x, y)            # 拟合数据

xx = np.arange(5, 20, 1).reshape(-1, 1)
yy = []
for i in range(len(xx)):
    yy.append(model.predict(np.array([xx[i]]))) # 线性回归模型
    i += 1

plt.figure()
plt.title("linear regression")
plt.xlabel("x")
plt.ylabel("y")
plt.plot(x, y, "k.")
plt.plot(xx, yy)
plt.grid(True)
plt.show()
```



10. 总结

在短短的几小时里想讲清楚Python的方方面面是不可能的，所以这个slides只选择性地介绍了一些我觉得会对大家有用的内容。有些不难理解的内容比如函数，它的作用只是提高代码复用率，这里就略过了；比较有难度的内容大多也是浅尝辄止没有深入，比如正则表达式，还有最后这几个经典的第三方库。如果把一门科普课上成劝退课，就有违初衷了。

俗话说“师父领进门修行在个人”，今后还要靠大家自己深入钻研。一门语言初接触肯定会有些许不顺手，但此时千万不能知难而退，一旦退了就永远也学不会了。挺过前面短暂的阵痛期，后面就是康庄大道。

祝在座的各位朋友今后都成为能熟用Python解决问题的业务能手！