

声明：本课程版权归华算科技所有，仅限个人学习，严禁任何形式的录制、传播和账号分享。一经发现，平台将依法保留追究权，情节严重者将承担法律责任。

# Python与机器学习

## ——Python基本使用

华算科技 黄老师  
2022年2月21日



1. 变量与运算符
2. 顺序结构，选择结构与循环结构
3. 函数
4. 案例：反应与压强的关系
5. 文件读写

1. 变量与运算符
2. 顺序结构，选择结构与循环结构
3. 函数
4. 案例：反应与压强的关系
5. 文件读写

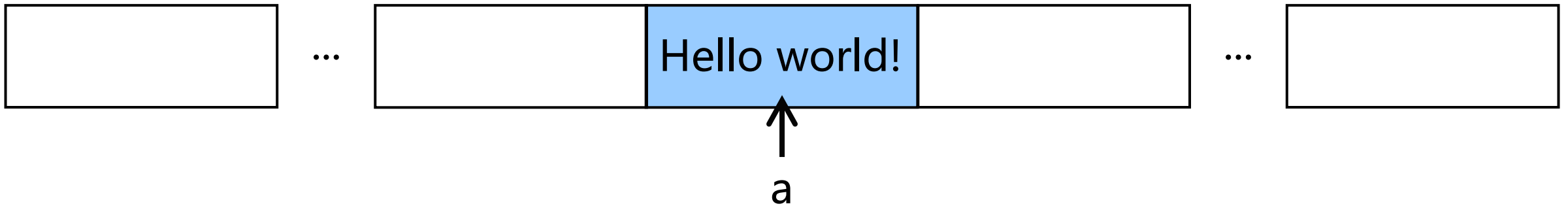
```
In [1]: print('Hello world!')
```

Hello world!

```
In [2]: a = 'Hello world!'
        print(a)
```

Hello world!

`a = 'Hello world!'`  
`print(a)`



# 变量

```
In [1]: print('Hello world!')
```

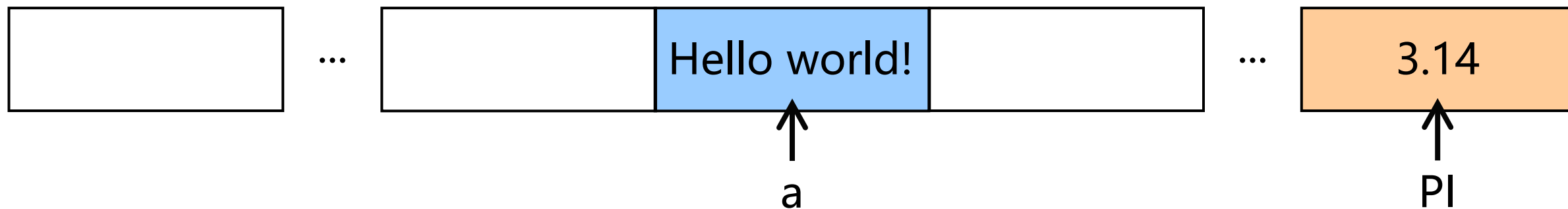
Hello world!

```
In [2]: a = 'Hello world!'
        print(a)
```

Hello world!

```
In [3]: PI = 3.14
        print(PI)
```

3.14



# 实操：问候程序

```
In [4]: name = input()  
        print('Hello, ', name, ' !')
```

Python

Hello, Python !

从输入设备（键盘）获取数据，并保存在name变量中

使用print进行输出



# 字符串

以单引号 (') 或双引号 (") 括起来的任意文本

例:

'Hello world!'    字符串数据类型

'I love China!'

"I'm fine."

'F%\*k(&^\$#@q'

'5'



任意大小的整数

例：

520     整数数据类型

0

-2021

1\_000\_000\_000

0xff00

小数

例:

3.14    浮点数数据类型

-1.23

6.02e23

1.60e-19

9.

# 布尔型

True或False

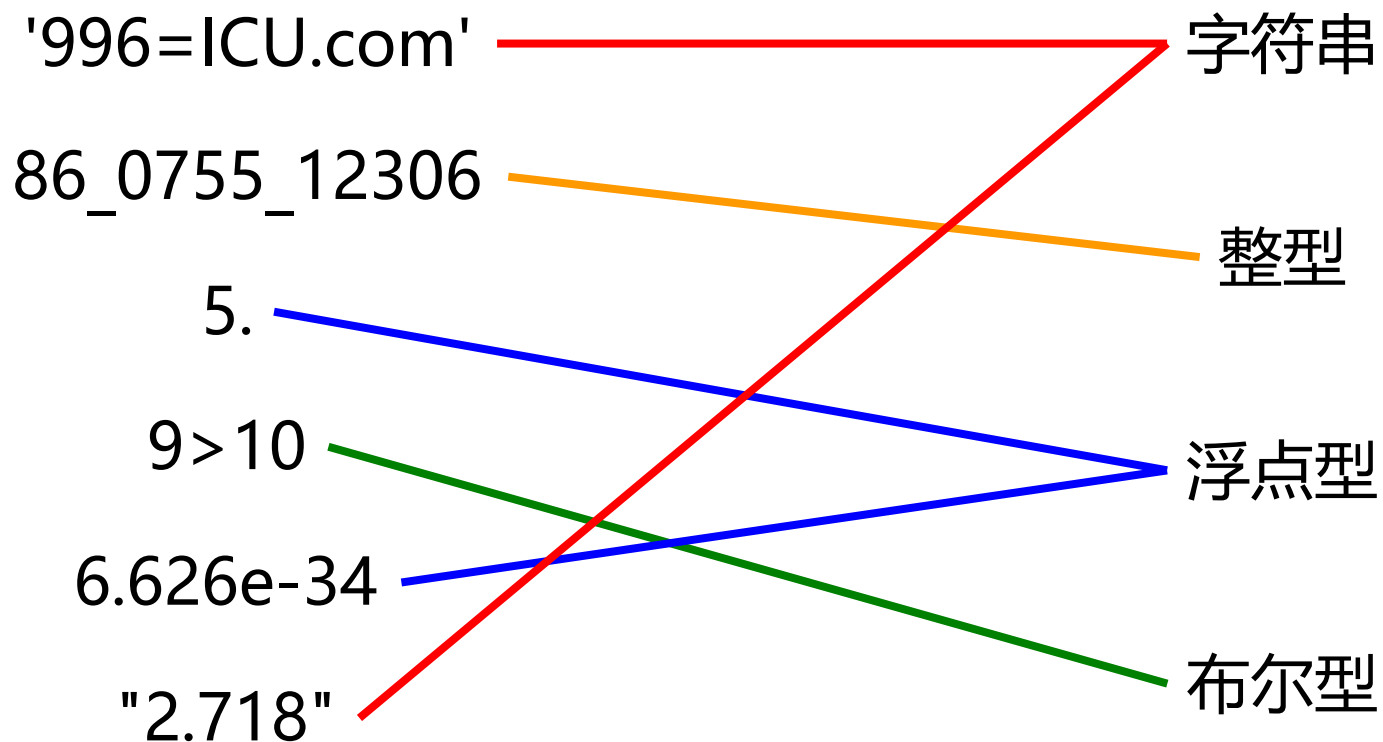
例:

True    布尔值数据类型

False

$3 > 2$

下列数据分别属于什么类型？

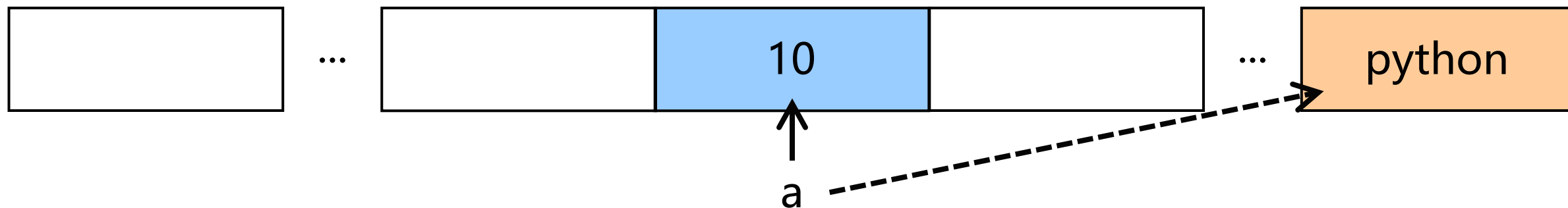


# 变量类型?

在Python中, 类型属于对象, 变量是没有类型的

`a = 10`       $\longrightarrow$     整数类型

`a = 'python'`       $\longrightarrow$     字符串类型



$1+2=3$      1与2称为操作数，+称为运算符

---



a=5, b=2, c='He', d='llo'

运算符	描述	样例
+	相加	a+b结果为7, c+d结果为'Hello'
-	相减	a-b结果为3, 字符串不能相减
*	相乘	a*b结果为10, c*b结果为'HeHe'
/	相除	a/b结果为2.5
%	取余	a%b结果为1
**	乘方	a**b结果为25
//	向下取相除后接近商的整数	a//b结果为2

# 比较运算符

a=5, b=2

运算符	描述	样例
==	相等	a==b返回False
!=	不等	a!=b返回True
>	大于	a>b返回True
>=	大于等于	a>=b返回True
<	小于	a<b返回False
<=	小于等于	a<=b返回False



# 赋值运算符

运算符	描述	样例
=	赋值	$c = a + b$
+=	相加赋值	$c += a$ , 即 $c = c + a$
-=	相减赋值	$c -= a$ , 即 $c = c - a$
*=	相乘赋值	$c *= a$ , 即 $c = c * a$
/=	相除赋值	$c /= a$ , 即 $c = c / a$
...		

# 逻辑运算符

a=True, b=False

运算符	描述	样例
and	与	a and b为False
or	或	a or b为True
not	非	not a为False

不同类型的数据之间常常需要进行转换

例：

转为整数

`int(x)`

转为浮点数

`float(x)`

转为字符串

`str(x)`

```
In [6]: a='5'
        b=1
        print(int(a)+b)
        6
```

# 列表 (list)

一组有序，可变的变量序列

Python中使用非常  
频繁的数据类型

a = [1, 2, 3, 4, 5]

列表： ' ' 将各个元素分开

b = ['x', 2.2]

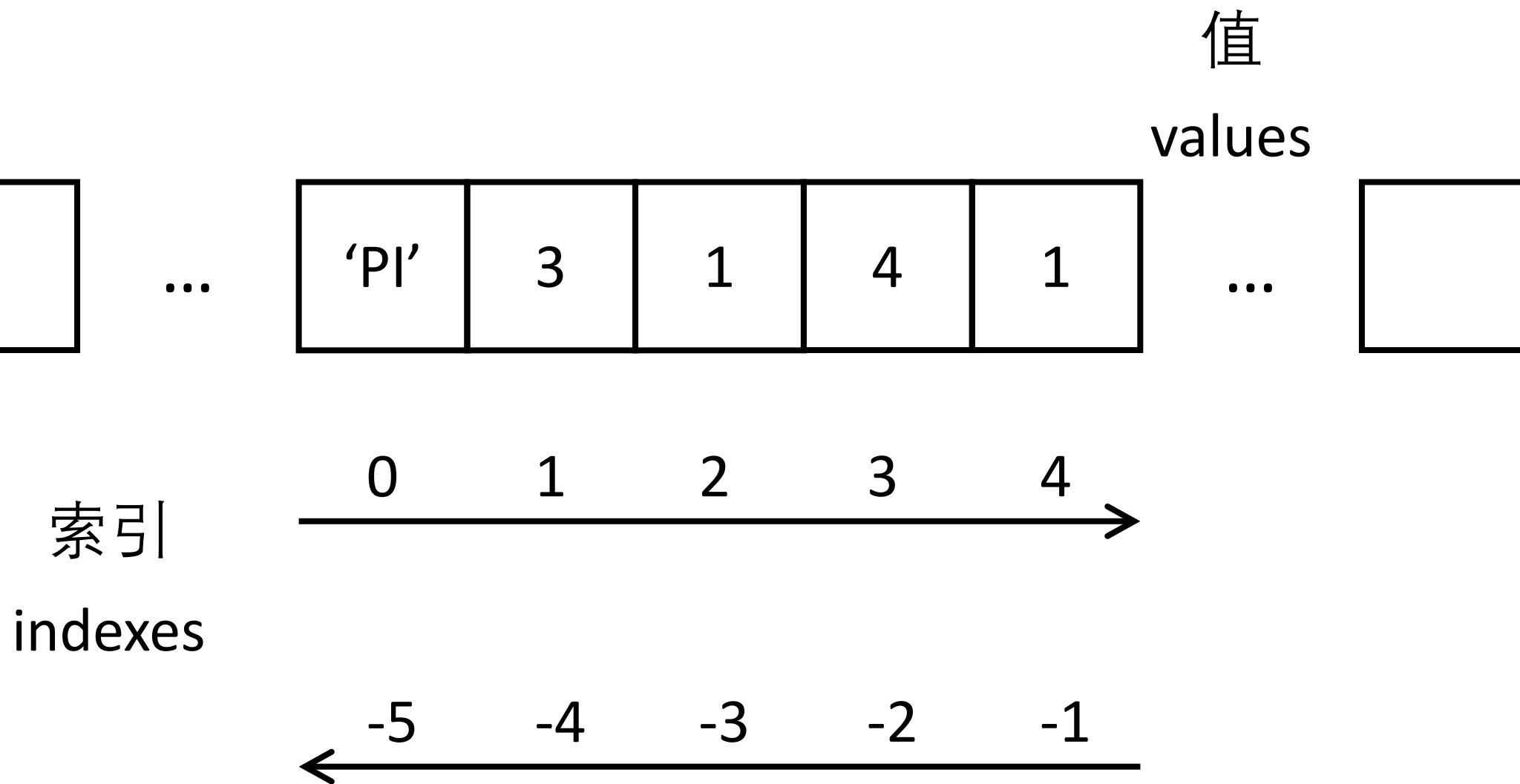
一个列表中可以同时包含不同的类型

```
In [11]: a=[1, 2, 3, 4, 5]
          print(a)
          print(a[0])
```

```
[1, 2, 3, 4, 5]
```

```
1
```

# 列表



# 列表的使用

```
In [13]: list_a = [ 'uv', 255, 345, 'cm-1', 10 ]  
list_b = ['res', 123]
```

```
print (list_a)  
print (list_a[0])  
print (list_a[1:3])  
print (list_a[:2])  
print (list_a * 2)  
print (list_a + list_b)
```

```
['uv', 255, 345, 'cm-1', 10]
```

```
uv
```

```
[255, 345]
```

```
['uv', 255]
```

```
['uv', 255, 345, 'cm-1', 10, 'uv', 255, 345, 'cm-1', 10]
```

```
['uv', 255, 345, 'cm-1', 10, 'res', 123]
```

输出整个列表

输出单个元素

截取列表

输出列表2次

列表拼接

## 练习：删除列表项

删除'density'项，下划线处应填什么？

```
list_a = ['density', 1.2, 1.4, 1.3, 1.9]
```

```
list_a = list_a[____:]
```

```
In [14]: list_a = ['density', 1.2, 1.4, 1.3, 1.9]
```

```
list_a = list_a[1:]
```

```
list_a
```

```
Out[14]: [1.2, 1.4, 1.3, 1.9]
```

# 列表的使用

## 创建列表、添加元素：

```
In [18]: list_a = list(range(1, 10, 2))  
print(list_a)
```

```
list_a.append(11)  
print(list_a)
```

```
list_a.insert(3, 6)  
print(list_a)
```

```
[1, 3, 5, 7, 9]
```

```
[1, 3, 5, 7, 9, 11]
```

```
[1, 3, 5, 6, 7, 9, 11]
```

## 删除元素：

```
In [25]: list_a.append(7)  
list_a.remove(7)  
print(list_a)
```

```
list_a.pop(2)  
print(list_a)
```

```
[1, 3, 5, 6, 9, 11, 7]
```

```
[1, 3, 6, 9, 11, 7]
```



## 填空

```
values = _____  
values._____ (1)  
values._____ (3)  
values._____ (5)  
print('output1:', values)  
values = values[_____]  
print('output2:', values)
```

output1: [1, 3, 5]  
output2: [3, 5]

In [46]:

```
values = []  
values.append(1)  
values.append(3)  
values.append(5)  
print('output1:', values)  
values = values[1:]  
print('output2:', values)
```

output1: [1, 3, 5]  
output2: [3, 5]

# 元组 (tuple)

与列表类似，使用()，区别在于元组的元素不可更改

a = (1, 2, 3, 4, 5)

b = ('x', 2.2)

c = 'x' , 3

```
In [33]: tup1 = (5)
          tup2 = (5,)
          print(type(tup1), type(tup2))

          <class 'int'> <class 'tuple'>
```

```
In [36]: tup1 = (1, 2, 3, 4, 5)
          tup1[0] = 0
```

```
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-36-9c7896b03f67> in <module>
      1 tup1 = (1, 2, 3, 4, 5)
----> 2 tup1[0] = 0
```

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

# 集合 (set)

无序的不重复元素序列

创建方法: {}或set()

```
In [42]: set1 = set('anaconda')  
set1
```

```
Out[42]: {'a', 'c', 'd', 'n', 'o'}
```

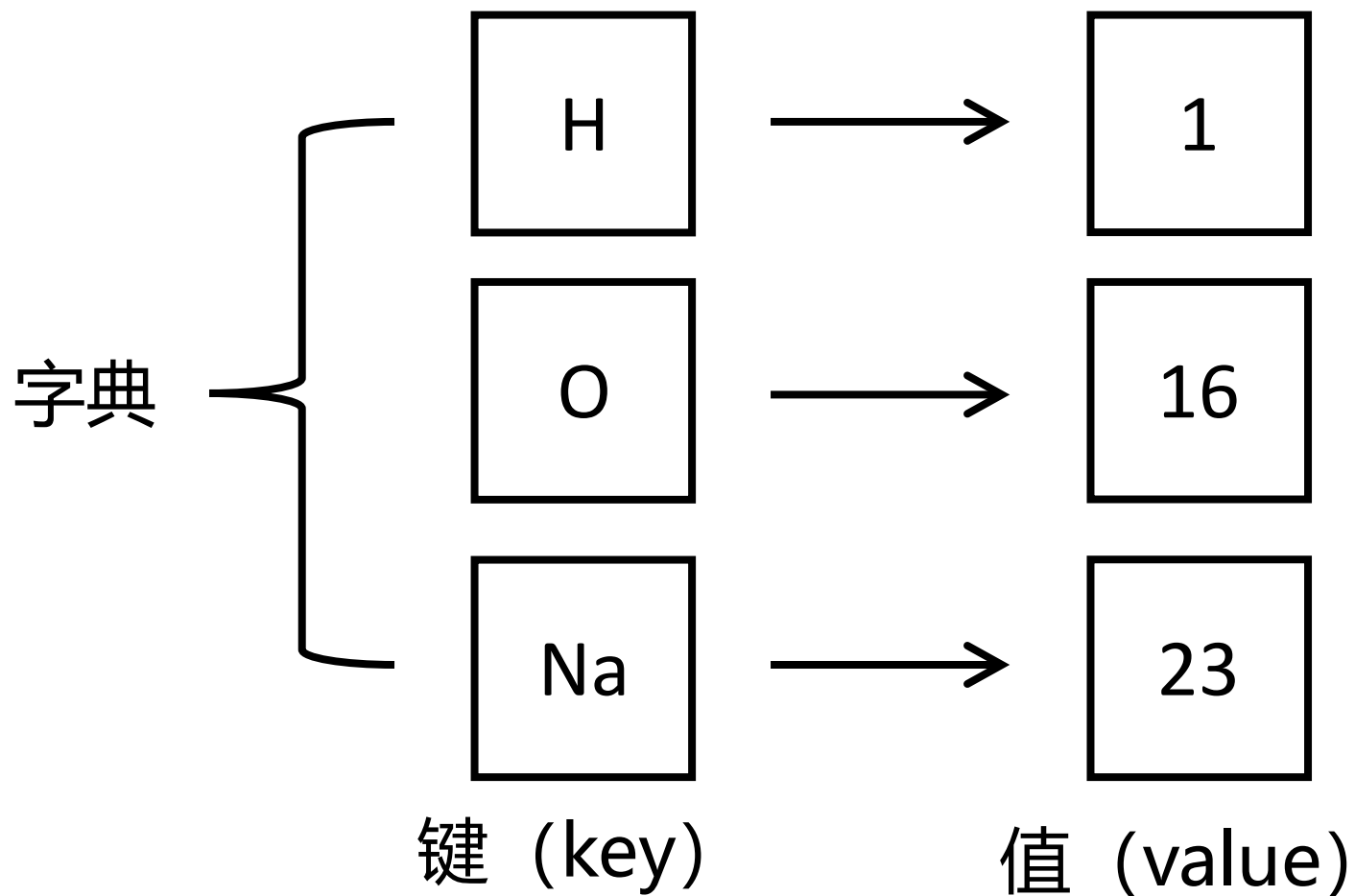
```
In [43]: set2 = set('python')  
set1 & set2
```

```
Out[43]: {'n', 'o'}
```

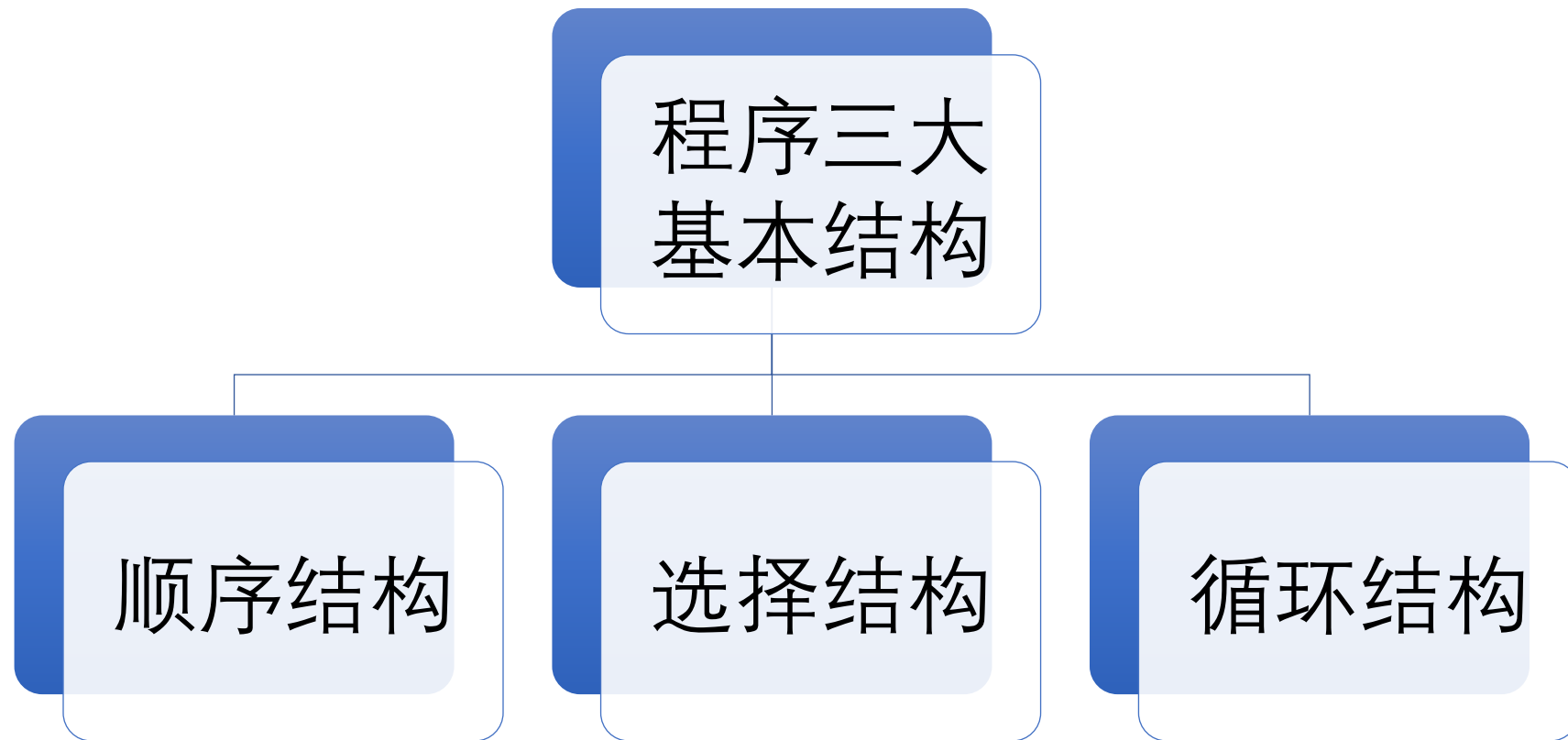
# 字典 (dictionary)

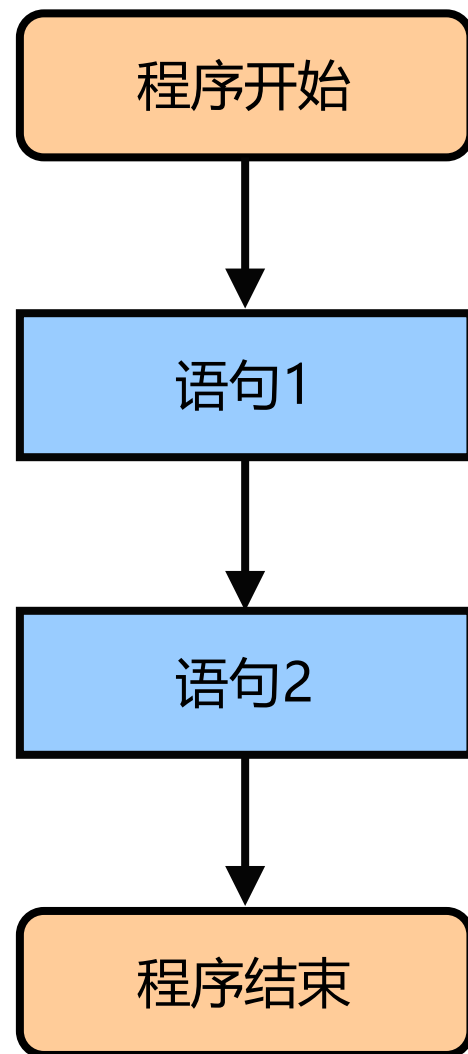
键值对

`{'H' : 1, 'O' : 16, 'Na' : 23}`

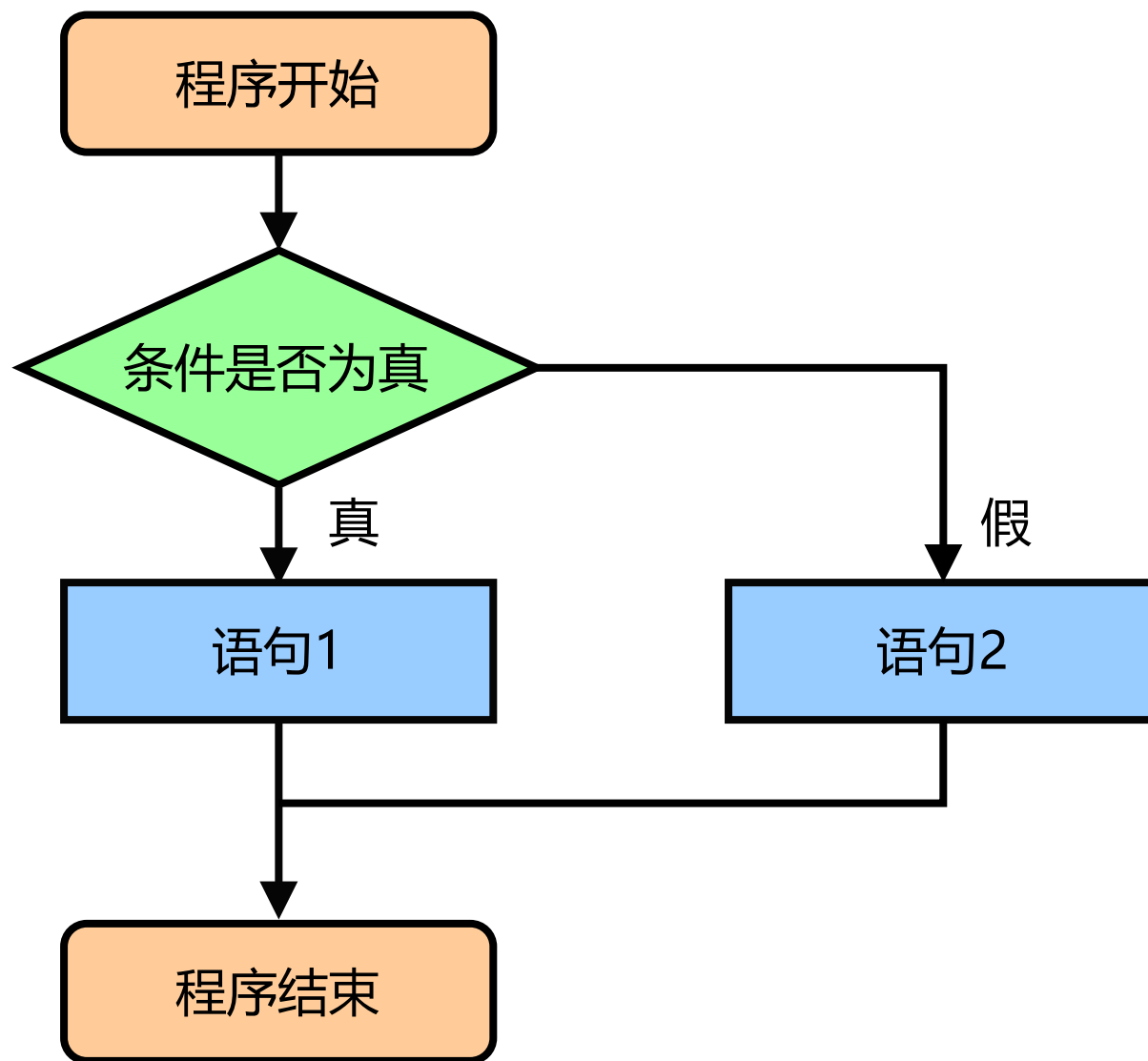


1. 变量与运算符
2. 顺序结构，选择结构与循环结构
3. 函数
4. 案例：反应与压强的关系
5. 文件读写





# 选择结构





# if 语句

if(表达式):  
    语句  
后续语句

若执行语句只有一条可写在同一行中

```
In [5]: a = 5  
        if(a > 0):  
            print('a is positive')  
            print('the square of a is', a * a)
```

```
a is positive  
the square of a is 25
```

```
In [6]: a = 5  
        if(a > 0):print('a is positive')
```

```
a is positive
```

# if ... else ... 语句

if(表达式):

    语句1

else:

    语句2

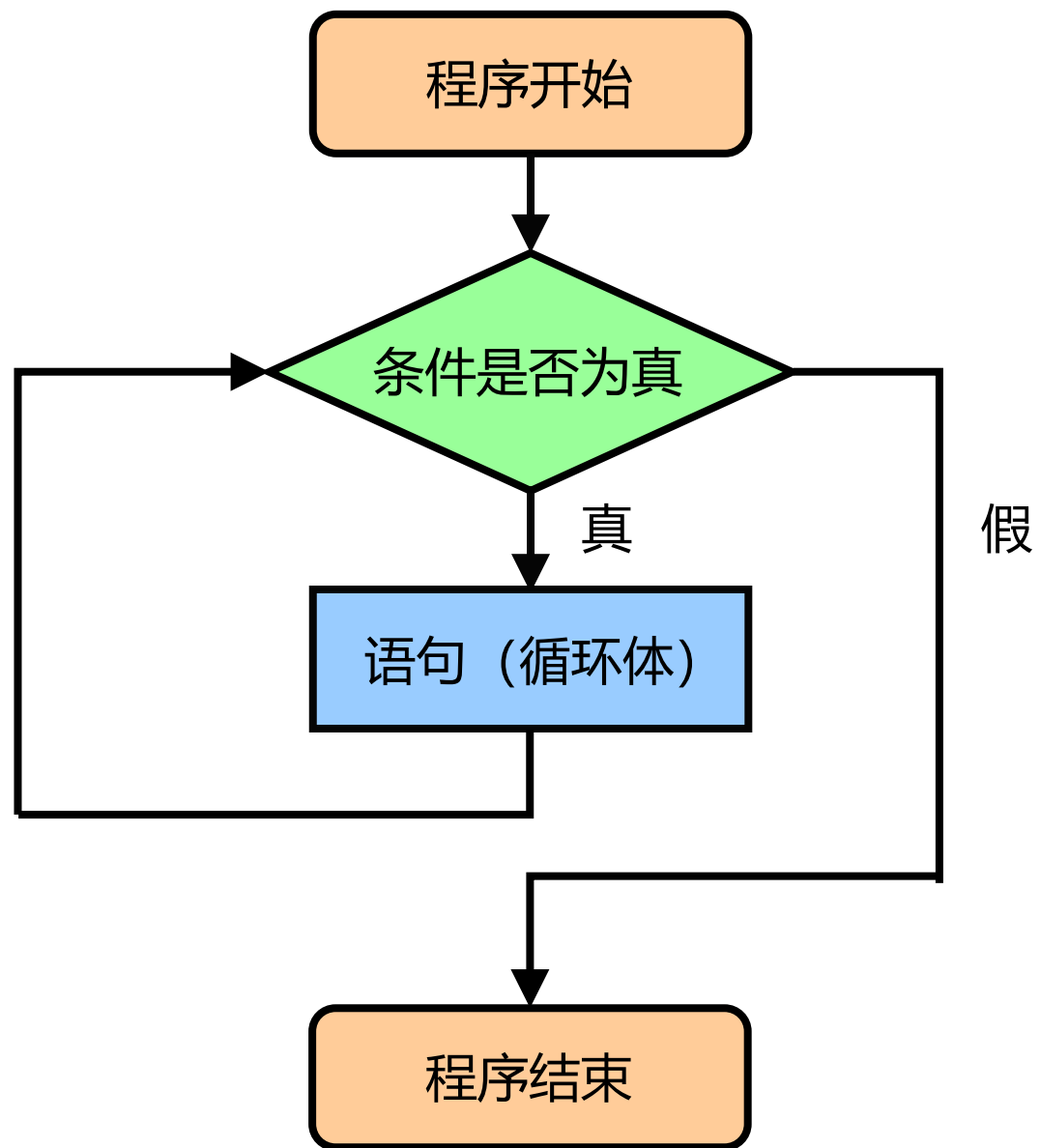
后续语句

In [6]:

```
temp = 25
if(temp > 100):
    print('is gas')
elif(temp > 0):
    print('is liquid')
else:
    print('is solid')
```

is liquid

# 循环结构



# while 语句

输出小于10的  
所有正数:

```
In [7]: a = 1  
while(a < 10):  
    print('a=', a)  
    a += 1
```

a= 1

a= 2

a= 3

a= 4

a= 5

a= 6

a= 7

a= 8

a= 9

# while 语句

判断是否能  
被3整除:

```
In [8]: a = 1
while(a < 20):
    if(a % 3 == 0):
        print('a=', a)
    a += 1
```

a= 3

a= 6

a= 9

a= 12

a= 15

a= 18

# while ... else ... 语句

```
In [9]: a = 1
while(a <= 20):
    if(a % 3 == 0):
        print('a=', a)
    a += 1
else:
    print('a is larger than 20')
```

a= 3

a= 6

a= 9

a= 12

a= 15

a= 18

a is larger than 20

# 死循环

需特别注意，应  
尽可能避免

```
In [10]: a = 1  
while(a < 10):  
    print('a=', a)  
    a -= 1
```

手动终止程序

```
a= 1  
a= 0  
a= -1  
a= -2  
a= -3  
.
```

## 演示：输出Fibonacci数列

```
In [30]: a = 1  
b = 1  
while a < 100:  
    print(a, end = ' ')  
    a, b = b, a + b
```

1 1 2 3 5 8 13 21 34 55 89



# while 语句与 list

可用于整个列表  
的复制与值的删  
除等

```
In [23]: a = ['red', 'purple', 'blue']  
while a:  
    print(a)  
    a.pop()
```

```
['red', 'purple', 'blue']  
['red', 'purple']  
['red']
```

```
In [24]: a = ['red', 'purple', 'blue']  
while 'blue' in a:  
    print(a)  
    a.remove('blue')
```

```
['red', 'purple', 'blue']
```

# for 语句

擅长逐个访问

```
In [28]: molecule = ['H', 'C', 'N']  
for ele in molecule:  
    print(ele)
```

H

C

N

# for 语句与 range()

range(a, b, c) 在[a, b)范围内生成间隔为c的一组数

```
In [40]: for i in range(0, 21, 3):  
          print(i, end = ' ')
```

0 3 6 9 12 15 18

# range()常见用法

```
In [45]: for i in range(5):  
         print(i, end = ' ')
```

0 1 2 3 4

```
In [46]: for i in range(2, 5):  
         print(i, end = ' ')
```

2 3 4

```
In [47]: for i in range(1, 5, 2):  
         print(i, end = ' ')
```

1 3

# range()常见用法

```
In [55]: a = ['red', 'purple', 'blue']  
         for i in range(len(a)):  
             print(i, a[i])
```

```
0 red  
1 purple  
2 blue
```

```
In [56]: a = list(range(5))  
         a
```

```
Out[56]: [0, 1, 2, 3, 4]
```

**len(a):**  
返回列表a的元素个数

# 循环练习

编写程序，使其输出如下：

? ? ? ?



Python程序

```
*  
**  
***  
****  
*****  
*****  
*****  
****  
***  
**  
*
```



输出

1. 变量与运算符
2. 顺序结构，选择结构与循环结构
3. 函数
4. 案例：反应与压强的关系
5. 文件读写

函数是组织好的、可重复使用的代码段

```
def 函数名(函数参数):
```

```
    函数语句
```

```
    函数语句
```

```
    ...
```

```
    return 返回值
```



# 函数

```
In [6]: temp = 25
        if(temp > 100):
            print('is gas')
        elif(temp > 0):
            print('is liquid')
        else:
            print('is solid')
```

is liquid

```
In [23]: def phase(t):
        if(t > 100):
            print('is gas')
        elif(t > 0):
            print('is liquid')
        else:
            print('is solid')
```

```
phase(-5)
temp = 25
phase(temp)
```

is solid  
is liquid

## 逗号分隔值（Comma-Separated Values, CSV）文件格式



文件以纯文本形式存储表格数据（数字和文本），并使用分隔字符（通常是逗号）进行分隔

lz.csv - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
3.00E-10,2.27E-12,2.24E-12
3.00E-10,2.06E-12,1.90E-12
3.00E-10,2.79E-12,1.56E-12
2.99E-10,2.60E-12,1.93E-12
2.99E-10,1.74E-12,9.89E-13
2.99E-10,2.33E-12,1.98E-12
2.99E-10,2.52E-12,1.69E-12
2.99E-10,2.42E-12,2.28E-12
2.98E-10,2.52E-12,1.76E-12
2.98E-10,2.28E-12,1.96E-12
2.98E-10,2.01E-12,2.68E-12
2.98E-10,2.10E-12,1.76E-12
```

A1				0.000000
	A	B	C	D
1	3.00E-10	2.27E-12	2.24E-12	
2	3.00E-10	2.06E-12	1.90E-12	
3	3.00E-10	2.79E-12	1.56E-12	
4	2.99E-10	2.60E-12	1.93E-12	
5	2.99E-10	1.74E-12	9.89E-13	
6	2.99E-10	2.33E-12	1.98E-12	
7	2.99E-10	2.52E-12	1.69E-12	
8	2.99E-10	2.42E-12	2.28E-12	
9	2.98E-10	2.52E-12	1.76E-12	
10	2.98E-10	2.28E-12	1.96E-12	
11	2.98E-10	2.01E-12	2.68E-12	
12	2.98E-10	2.10E-12	1.76E-12	
13	2.98E-10	1.86E-12	9.97E-13	
14	2.97E-10	2.47E-12	2.11E-12	
15	2.97E-10	1.35E-12	2.12E-12	
16	2.97E-10	1.89E-12	1.64E-12	

```
In [25]: def fun(a):  
          print(a)  
          a += 1  
          print(a)  
  
          a = 1  
          print(a)  
          fun(a)  
          print(a)
```

四次输出结果  
分别是什么？

1  
1  
2  
1

## 可更改(mutable)与不可更改(immutable)对象

不可更改对象： 如整数、字符串，fun(a)生成一个新的a，不会修改原来a的值，类似于C++中的值传递，

可更改对象： 如列表，fun(a)将真正的a传入函数中，类似于C++中的址传递。

```
In [27]: def fun(a):  
         a.append('cm-1')
```

```
b = [525, 566, 573]
```

```
print(b)
```

```
fun(b)
```

```
print(b)
```

```
[525, 566, 573]
```

```
[525, 566, 573, 'cm-1']
```

1. 变量与运算符
2. 顺序结构，选择结构与循环结构
3. 函数
4. 案例：反应与压强的关系
5. 文件读写

对于 $N_2O_4$ 分解反应



在一定温度及压力下， $N_2O_4$ 分解有0.50（摩尔分数）分解为 $NO_2$ ，若压力变为原来的2倍、4倍、8倍， $N_2O_4$ 的解离分数分别为多少？

摩尔分数平衡常数

$$K_x = \frac{(p_{NO_2}/p)^2}{p_{N_2O_4}/p}$$

$$\left(\frac{\partial \ln K_x}{\partial p}\right)_T = -\frac{1}{p}$$



$$1 - x \qquad 2x$$

$$K_x = \frac{(2x/(1+x))^2}{(1-x)/(1+x)} = \frac{4x^2}{1-x^2}$$

初始压力 $p_0$ 下,  $x$ 值为0.5, 有

$$K_x(p_0) = \frac{4}{3}$$



$$\left(\frac{\partial \ln K_x}{\partial p}\right)_T = -\frac{1}{p}$$

$$\ln K_x = -\ln p + c$$

$$\ln\left[\frac{K_x(p_1)}{K_x(p_0)}\right] = \ln(p_0/p_1)$$

$$\text{令 } p_1 = np_0,$$

$$K_x(p_1) = \frac{1}{n} K_x(p_0) = \frac{4}{3n}$$

$$\frac{4x^2}{1-x^2} = \frac{4}{3n}$$

$$3nx^2 = 1 - x^2$$

$$x = \sqrt{\frac{1}{3n+1}}$$

# 实操：Python处理数据

```
In [1]: def res(n):  
        return (1 / (3 * n + 1)) ** 0.5
```

```
In [2]: res(1)
```

```
Out[2]: 0.5
```

```
In [3]: res(2)
```

```
Out[3]: 0.3779644730092272
```

```
In [4]: res(4)
```

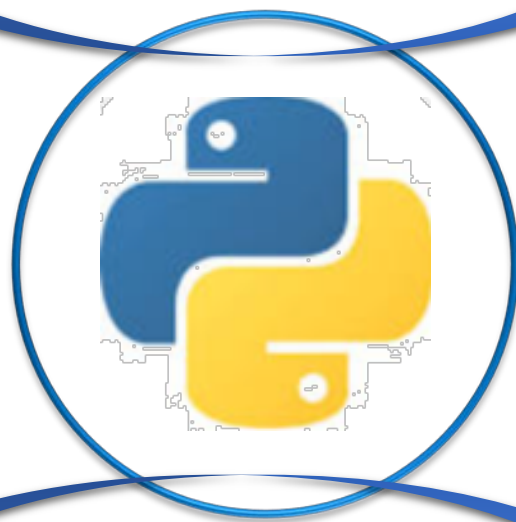
```
Out[4]: 0.2773500981126146
```

```
In [5]: res(8)
```

```
Out[5]: 0.2
```

1. 变量与运算符
2. 顺序结构，选择结构与循环结构
3. 函数
4. 案例：反应与压强的关系
5. 文件读写

# 文件读写



读文件： 获取输入



写文件： 结果输出



有利于文件的保存、与第三方软件结合使用等

## 文本文件格式

存储文本信息，大多数软件可以直接打开，如记事本、浏览器等等



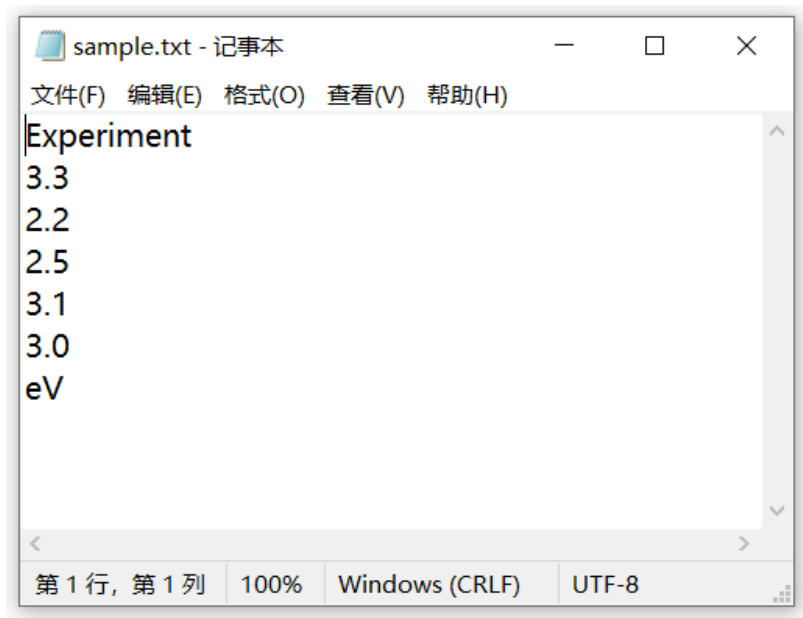
常常同时包含数据、仪器  
测量时条件等信息

```
lzsample.dat - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
Experiment      Z spectroscopy
Saved Date      02.04.2019 17:19:27
User
X (m)           65.9001E-9
Y (m)           -57.4945E-9
Z (m)           -45.4765E-9
Z offset (m)     300E-12
Z sweep distance (m) 200E-12
Settling time (s) 200E-6
Integration time (s) 100E-6
Final Z (m)      N/A
Start time       02.04.2019 17:19:16
Filter type      None
Order
Cutoff frq

[DATA]
Z rel (m) Current (A) Current [bwd] (A)
300.00000E-12 2.2669475E-12 2.2399903E-12
299.79980E-12 2.0589192E-12 1.8966675E-12
299.59960E-12 2.7949015E-12 1.5604654E-12
299.39939E-12 2.6036580E-12 1.9261678E-12
```

# open()方法

open() 打开一个文件，并返回文件对象



```
1 f = open('sample.txt', mode = 'r', encoding = 'UTF-8')
2 res = f.read()
3 f.close()
4 print(res)
```

Experiment  
3.3  
2.2  
2.5  
3.1  
3.0  
eV

# open()方法

open(文件名, mode = 'r', encoding = 'UTF-8')

## 可选：文件打开方式

r, 只读模式打开（默认）

x, 新建一个文件，如果文件已存在则报错

w, 打开用于写入，如果文件不存在则创建新文件，  
如果文件已存在则将其覆盖

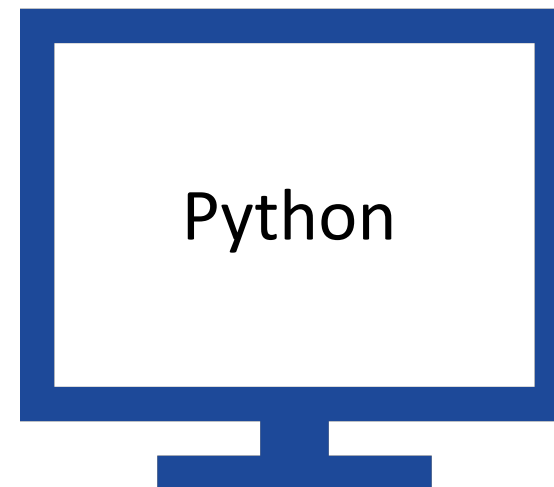
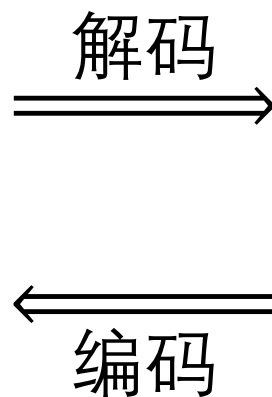
a, 追加模式打开，文件指针放在文件的末尾，新写入  
内容不会覆盖原来内容

rb, 以二进制格式打开

wb, 以二进制格式写入

```
1010000 1111001  
1110100 1101000  
1101111 1101110
```

计算机存储



我们可理解的存储



file对象用open()方法创建后，可用下列函数进行操作

```
f = open('sample.txt')
```

**close()** 关闭文件，关闭后不再能进行读写 f.close()

read([size]) 从文件读取指定的字节数，默认读取全部 f.read()

readline([size]) 读取整行，包括'\n'字符 f.readline()

readlines() 读取整个文件，返回一个列表 f.readlines()

write() 写入文件 f.write('xxx')

```
In [2]: 1 f = open('sample.txt', mode = 'r', encoding = 'UTF-8')
        2 res = f.readline()
        3 f.close()
        4 print(res)
```

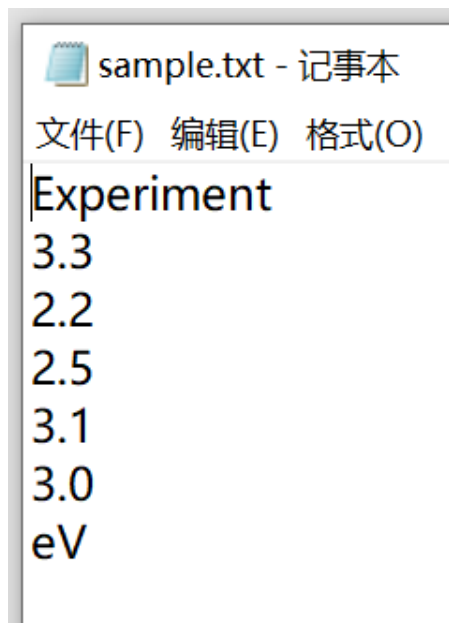
Experiment

```
In [3]: 1 f = open('sample.txt', mode = 'r', encoding = 'UTF-8')
        2 res = f.readlines()
        3 f.close()
        4 print(res)
```

```
['Experiment\n', '3.3\n', '2.2\n', '2.5\n', '3.1\n', '3.0\n', 'eV']
```

# 实操：读取实验数据

将sample.txt中数据读取，并求出平均值



```
sample.txt - 记事本
文件(F) 编辑(E) 格式(O)
Experiment
3.3
2.2
2.5
3.1
3.0
eV
```



# 实操：读取实验数据

```
In [4]: 1 f = open('sample.txt', mode = 'r', encoding = 'UTF-8')
        2 f.readline()
        3 data = []
        4 for i in range(5):
        5     data.append(float(f.readline()))
        6 f.close()
        7 print(data)
```

[3.3, 2.2, 2.5, 3.1, 3.0]

```
In [5]: 1 f = open('sample.txt', mode = 'r', encoding = 'UTF-8')
        2 data = f.readlines()
        3 f.close()
        4 data = data[1:-1]
        5 for i in range(len(data)):
        6     data[i] = float(data[i])
        7 print(data)
```

[3.3, 2.2, 2.5, 3.1, 3.0]

# 实操：读取实验数据

In [6]:

```
1 def avg_list(a):  
2     sum = 0  
3     for ele in a:  
4         sum += ele  
5     return sum / len(a)
```

In [7]:

```
1 avg_list(data)
```

Out[7]: 2.82

avg\_list()函数：求list  
中所有元素的平均数

使用mode = 'w'或mode = 'a'

```
In [8]: 1 f = open('res.txt', mode = 'w')
        2 f.write(str(avg_list(data))+' \n')
        3 f.write(str(avg_list(data)))
        4 f.close()
```

write()后接字符串

## 实操：with open ... as ...的使用

为了避免忘记调用close()函数，Python提供了另一种打开文件的函数

```
In [9]: 1 with open('sample.txt', mode = 'r') as f:
        2     f.readline()
        3     data = []
        4     for i in range(5):
        5         data.append(float(f.readline()))
        6
        7 print(data)
```

```
[3.3, 2.2, 2.5, 3.1, 3.0]
```

## 逗号分隔值（Comma-Separated Values, CSV）文件格式



文件以纯文本形式存储表格数据（数字和文本），并使用分隔字符（通常是逗号）进行分隔

lz.csv - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
3.00E-10,2.27E-12,2.24E-12
3.00E-10,2.06E-12,1.90E-12
3.00E-10,2.79E-12,1.56E-12
2.99E-10,2.60E-12,1.93E-12
2.99E-10,1.74E-12,9.89E-13
2.99E-10,2.33E-12,1.98E-12
2.99E-10,2.52E-12,1.69E-12
2.99E-10,2.42E-12,2.28E-12
2.98E-10,2.52E-12,1.76E-12
2.98E-10,2.28E-12,1.96E-12
2.98E-10,2.01E-12,2.68E-12
2.98E-10,2.10E-12,1.76E-12
```

A1				0.000000
	A	B	C	D
1	3.00E-10	2.27E-12	2.24E-12	
2	3.00E-10	2.06E-12	1.90E-12	
3	3.00E-10	2.79E-12	1.56E-12	
4	2.99E-10	2.60E-12	1.93E-12	
5	2.99E-10	1.74E-12	9.89E-13	
6	2.99E-10	2.33E-12	1.98E-12	
7	2.99E-10	2.52E-12	1.69E-12	
8	2.99E-10	2.42E-12	2.28E-12	
9	2.98E-10	2.52E-12	1.76E-12	
10	2.98E-10	2.28E-12	1.96E-12	
11	2.98E-10	2.01E-12	2.68E-12	
12	2.98E-10	2.10E-12	1.76E-12	
13	2.98E-10	1.86E-12	9.97E-13	
14	2.97E-10	2.47E-12	2.11E-12	
15	2.97E-10	1.35E-12	2.12E-12	
16	2.97E-10	1.89E-12	1.64E-12	



# csv文件读取

```
In [16]: 1 with open('didz.csv', mode = 'r', encoding = 'UTF-8') as f:
          2     res = f.readline()
          3     I_max = res.strip('\n').split(',')
          4     I_min = f.readline().strip('\n').split(',')
          5
          6     z_max = I_max[0]
          7     z_min = I_min[0]
          8     I_max = I_max[1:]
          9     I_min = I_min[1:]
         10     print(I_max)
         11     print(I_min)
```

```
['2.27E-12', '2.24E-12', '2.30E-12', '2.25E-12', '2.22E-12', '2.25E-12', '2.24E-12']
['2.81E-12', '1.84E-12', '2.99E-12', '2.58E-12', '2.31E-12', '2.80E-12', '2.50E-12']
```