

python™ 程序设计

3. 选择结构、内置函数2、 内置模块、第三方扩展库



xgsun@fudan.edu.cn

孙晓光

2024-9-17



选择结构

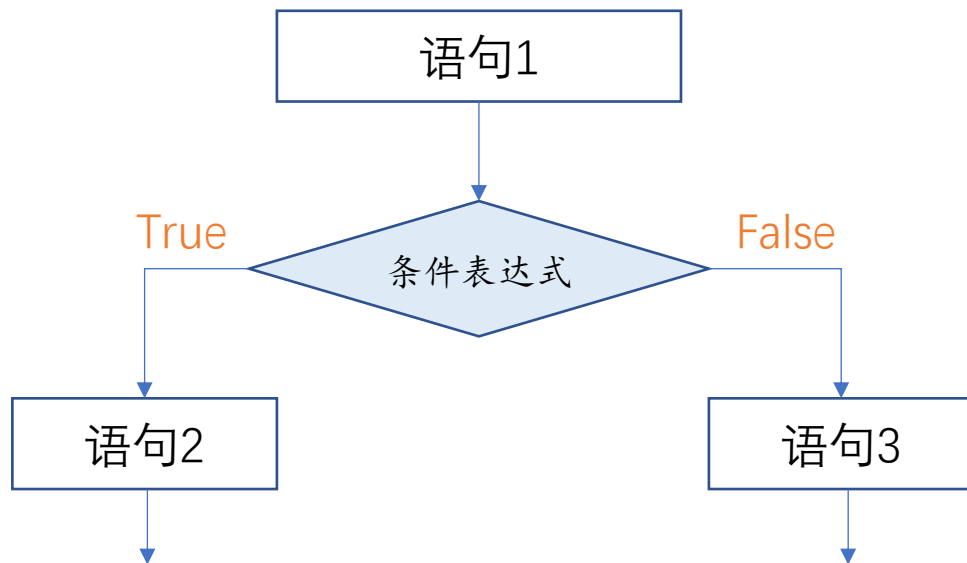
根据特定的**条件**是否满足，决定下一步要执行的代码。

单分支

双分支

多分支

嵌套



True、False?

条件表达式的值只要不是：
False、0（或0.0、0j）、
空值 None、空字符串、
空列表、空元组、空集合、
空字典、空range对象，
均认为 True。

```
>>> bool(0) --> False
```

```
>>> bool(2) --> True
```

关系运算符

数字比较

```
>>> 1 < 3
True
>>> 1 < 3 < 8 # 连用，等价于1 < 3 and 3 < 8
True
>>> 1 > 3 < 8 # 惰性求值
False
```

字符串

```
>>> 'Hello' > 'Python' # 逐个字符比较
False
>>> 'Hello' > 3 # 字符与数字不能比较
TypeError: unorderable types: str() > int()
```

```
>>> [1, 2, 3] < [1, 2, 4] # 列表逐个元素比较
True
>>> {1, 2, 3} == {3, 2, 1} # 两个集合是否相等
True
>>> {1, 2, 3} < {1, 2, 3, 4} # 是否子集
True
```

运算符	说明
>	大于
>=	大于或等于
<	小于
<=	小于或等于
!=	不等于
==	等于

“=” 赋值运算符

逻辑运算

True and True	True
True and False	False
False and True	False
False and False	False

- **and、or、not 连接多个条件表达式**

- **and、or 惰性求值：只在必须用到时才计算求值**

True or True	True
True or False	True
False or True	True
False or False	False

```
>>> 3>5 and a>3      # and不需要计算后者。注意，此时并没有定义变量a
False
```

```
>>> 3>5 or a>3        # 3>5的值为False，需要计算后面的
NameError: name 'a' is not defined
```

```
>>> 3<5 or a>3        # 3<5的值为True，不需要计算后面的
True
```

```
>>> 3 and 3<5          # 3就是True
True
```

```
>>> 3<5 and 3          # 以最后一个计算值(3)为最终结果
3
```

```
>>> not 0              # 只能True或False
True
>>> not 3
False
>>> 3 not in [1, 2, 3]
False
```

成员测试运算符 `in`

测试一个对象是否为另一个对象的元素？

```
>>> 'abc' in 'abcdefg'  
True
```

测试是否为子字符串

```
>>> 3 in [1, 2, 3]  
True
```

测试3是否存在于列表[1, 2, 3]中

```
>>> 5 in range(1, 10)  
True
```

range生成[1~10)之间的整数

```
>>> for i in (3, 5, 7):  
    print(i, end='\\t')
```

循环，元素遍历

```
3    5    7
```

range() 函数

产生一系列数，返回range对象 常当作循环的计数器

```
range ( [start=0,] stop [, step=1] )
```

[start, stop) 左闭右开区间

```
>>> range(5)                                # start默认为0, step默认为1
range(0, 5)

>>> list(_)                                # list(range(5)), 转换为列表
[0, 1, 2, 3, 4]

>>> list(range(1, 10, 2))                    # 指定起始值和步长
[1, 3, 5, 7, 9]

>>> list(range(9, 0, -2))                    # 步长为负数时, start > stop
[9, 7, 5, 3, 1]
```

同一性测试运算符 `is` `None`

`is` 比较两个变量是否为同一个内存地址？

`==` 比较两个变量的值是否相同？

对象有3个基本属性：

类型`type`、地址`id`、值`value`

• 数字型

```
>>> a = 1
```

```
>>> b = 1
```

```
>>> a == b
```

```
True
```

```
>>> a is b
```

```
True
```

```
>>> id(a) 140725969996216
```

```
>>> id(b) 140725969996216
```

a、b的内存地址相同。

• 字符串

```
>>> c = 'Python'
```

```
>>> d = 'Python'
```

```
>>> c == d
```

```
True
```

```
>>> c is d
```

```
True
```

```
>>> id(c) 140725685768992
```

```
>>> id(d) 140725685768992
```

c、d的内存地址相同。

• 列表

```
>>> x = [1, 2, 3]
```

```
>>> y = [1, 2, 3]
```

```
>>> x == y
```

```
True
```

```
>>> x is y
```

```
False
```

```
>>> id(x) 1973062590992
```

```
>>> id(y) 1973062596656
```

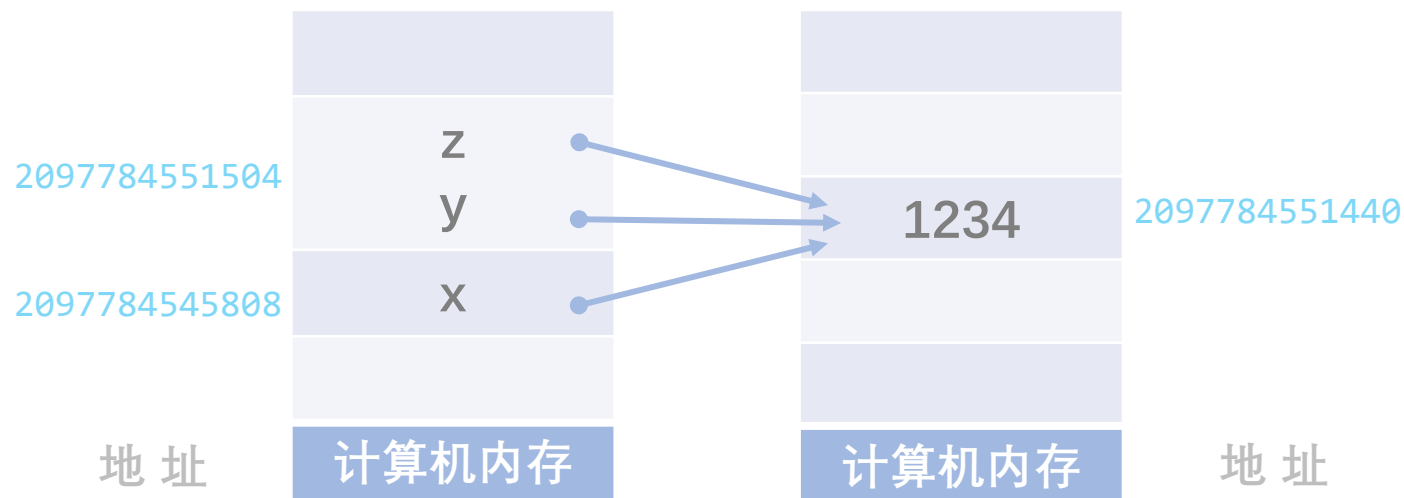
x、y的内存地址不同。

id() 身份标识

返回一个对象的内存地址，唯一地标识了对象在计算机内存中的位置。

`id(object, /)`

```
>>> x = 1234
>>> id(1234)    2097784551440
>>> y = 1234
>>> id(1234)    2097784551440
>>> z = y
>>> id(x)       2097784545808
>>> id(y)       2097784551504
>>> id(z)       2097784551504
```



单分支 选择结构

if 条件表达式:

空格
对齐

语句块

从小到大排序

```
a = int(input('输入第一个整数a='))
```

```
b = int(input('输入第二个整数b='))
```

```
if a > b:
```

```
    a, b = b, a # 交换两个变量的值
```

```
print(a, b)
```

if 条件表达式: 语句

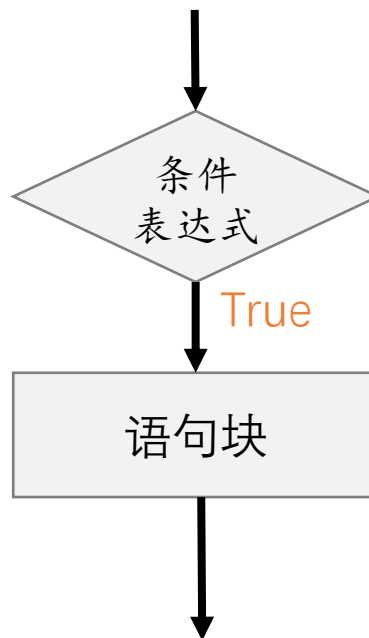
从小到大排序

```
a = int(input('输入第一个整数a='))
```

```
b = int(input('输入第二个整数b='))
```

```
if a > b: a, b = b, a
```

```
print(a, b)
```



运行结果:

```
输入第一个整数a=8
输入第二个整数b=3
3 8
```

双分支 选择结构

if 条件表达式:

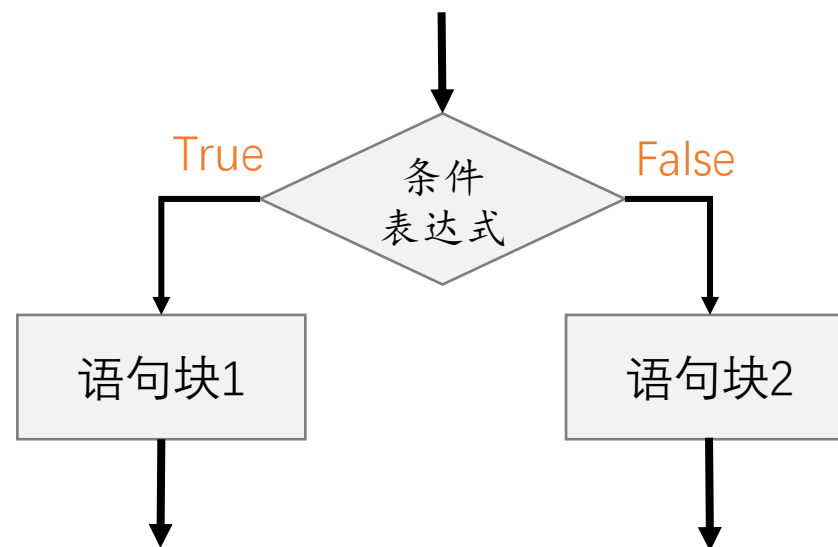
空格
对齐

语句块1

else:

空格
对齐

语句块2



奇偶?

```
print("判断奇数、偶数")
```

```
num = input("请输入任意整值: ")
```

```
rem = int(num) % 2
```

```
if (rem == 0):
```

```
    print("偶数")
```

```
else:
```

```
    print("奇数")
```

运行结果:

```
判断奇数、偶数
请输入任意整值: 5
奇数
```

关键字 `pass`

空语句 / 占位符(表示未完成的代码块), 执行该语句时什么都不做。

没有`pass`时, 结构不完整, 运行会报错。

- 在`if`语句中

```
x = 10
```

```
if x > 5:
```

```
    pass    # 未完成
```

```
else:
```

```
    print('x <= 5')
```

- 在`for`语句中

```
for i in range(5):
```

```
    pass    # 未完成
```

- 在函数中

```
def demo(arg):
```

```
    pass    # 未完成
```

关键字 `assert`

断言语句，确认某个条件必须满足，
不满足会引发 `AssertionError` 错误，帮助调试程序。

- 用`assert`判断输入是否正常？

```
a = int(input('请输入您的年龄：'))
```

```
assert 0 <= a <= 100
```

```
print('您输入的年龄在0~100之间')
```



- 相当于`if`语句

```
a = int(input('请输入您的年龄：'))
```

```
if not (0 <= a <= 100)
```

```
    print('年龄输入错误！')
```

请输入您的年龄： 120

Traceback (most recent call last):

File "C:/2023Sam/03 Python/代码/assert.py", line 3, in <module>

assert 0 <= a <= 100

AssertionError

例4.2 鸡兔同笼

用公式法求解

```
鸡兔 = int(input('请输入鸡兔总数: '))
```

```
腿    = int(input('请输入腿的总数: '))
```

```
兔 = (腿 - 鸡兔 * 2) / 2    # 公式方程1
```

```
if int(兔) == 兔 and 0 <= 兔 <= 鸡兔:
```

```
    鸡 = 鸡兔 - 兔            # 公式方程2
```

```
    print(f'鸡: {int(鸡)}, 兔: {int(兔)}')
```

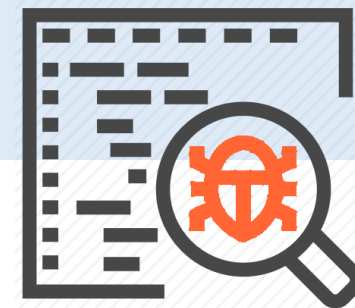
```
else:
```

```
    print('数据不正确, 无解')
```

请输入鸡兔总数: 14

请输入腿的总数: 50

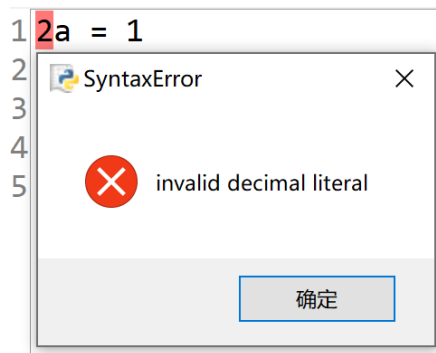
鸡: 3, 兔: 11



编程难免出错，调试就是查找、修改错误的过程。

1

语法错误



2

运行时错误

$a = 1 / 0$

Traceback (most recent call last):

File "C:/2023Sam/03 Python/代码/if-错误类型.py", line 1, in <module>

$a = 1 / 0$

ZeroDivisionError: division by zero

3

逻辑错误

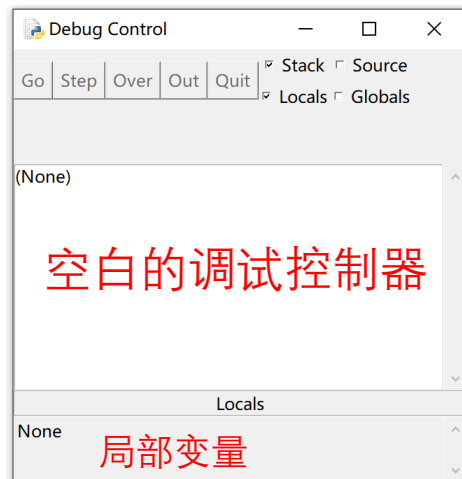
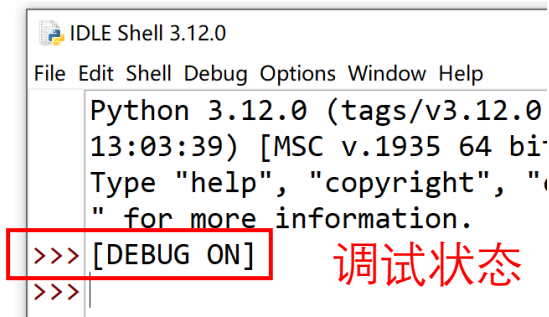
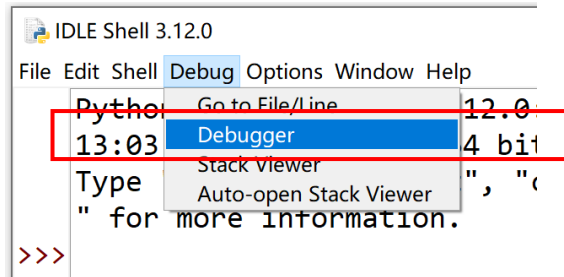
得不到预期的结果。

设置断点、逐句跟踪

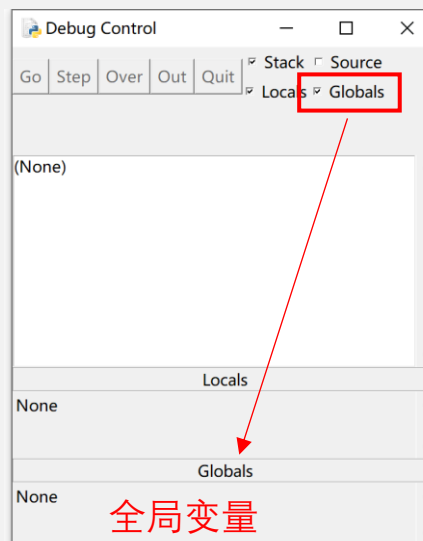
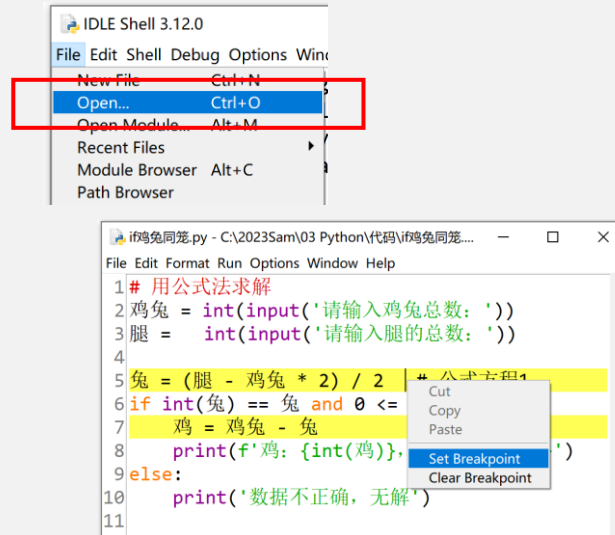
```
1 a = 1
2 b = 2
3 平均值 = a + b / 2 # 应该是 (a + b) / 2
4
```

用 IDLE 调试 例4.2 鸡兔同笼

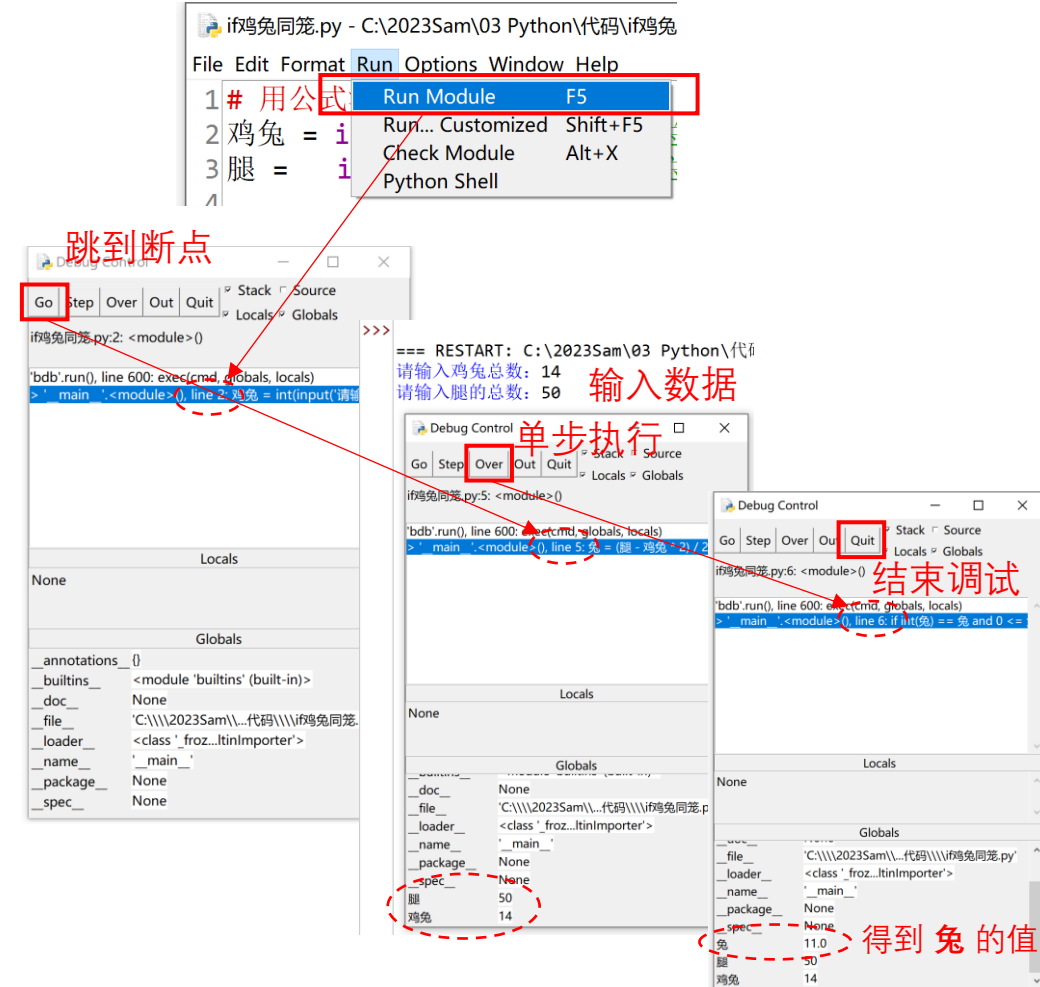
① 打开调试器



② 打开调试文件，设置断点



③ F5 运行代码、调试代码



④ 关闭Debug Control结束调试 15 / 40

三元运算符

双分支结构的一种简化写法

value1 **if** condition **else** value2

True

False

```
>>> 6 if 2 > 1 else 5
```

6

```
>>> a = 6 if 2 > 1 else 5
```

```
>>> a
```

6

多分支 选择结构

if 表达式1:

语句块1

elif 表达式2:

语句块2

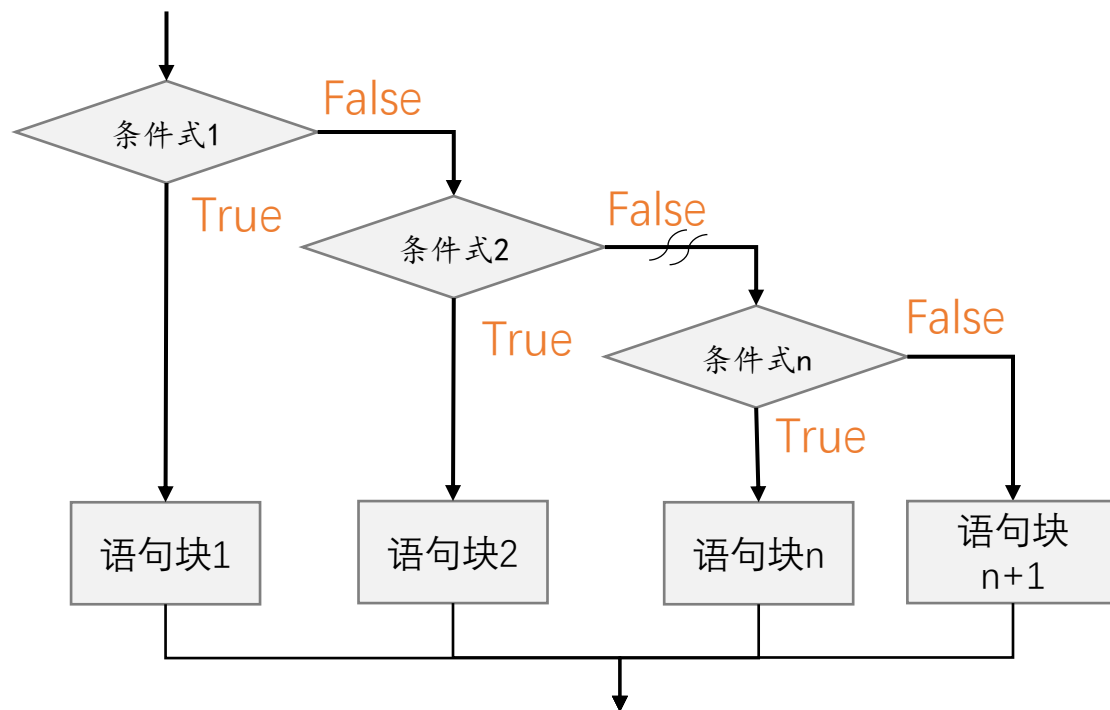
.....

elif 表达式n:

语句块n

else:

语句块n+1



```
>>> match 1+3:
...     case 5:
...         print(1)
...     case 3:
...         print(0)
...     case 4:
...         print(2)
...
2
>>> match 1+4:
...     case 5:
...         print(1)
...     case 3:
...         print(0)
...     case 4:
...         print(2)
...
1
```

Python 3.10 新增

关键字**elif**是else if的缩写

多分支 选择结构

if 表达式1:
 语句块1
elif 表达式2:
 语句块2

elif 表达式n:
 语句块n
else:
 语句块n+1

将百分制成绩变换到等级:

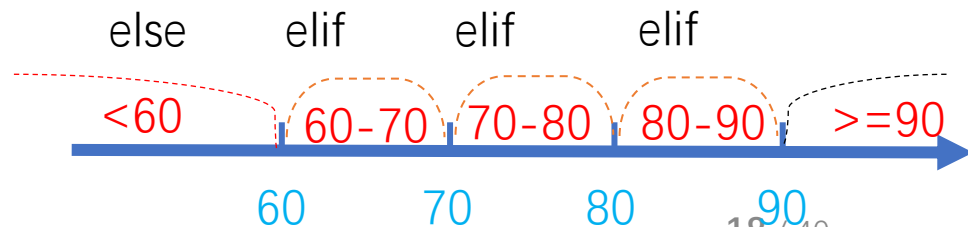
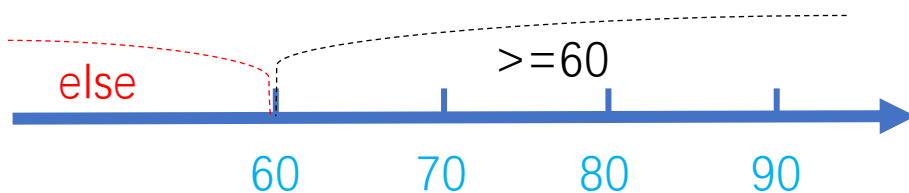
```
score = input("输入成绩[0~100]: ")  
if int(score) >= 60:  
    print('D')  
elif int(score) >= 70:  
    print('C')  
elif int(score) >= 80:  
    print('B')  
elif int(score) >= 90:  
    print('A')  
else:  
    print('F')
```

将百分制成绩变换到等级:

```
score = input("输入成绩[0~100]: ")  
if int(score) >= 90:  
    print('A')  
elif int(score) >= 80:  
    print('B')  
elif int(score) >= 70:  
    print('C')  
elif int(score) >= 60:  
    print('D')  
else:  
    print('F')
```

运行结果:

输入成绩[0~100]: 88
B



例：出租车收费

- 里程数 ≤ 3 公里：车价 = 起步价【14元】
- 3公里 $<$ 里程数 ≤ 15 公里：车价 = 起步价【14元】 + (里程数 - 起步里程数【3公里】) \times 每公里单价【2.7元】
- 里程数 > 15 公里：车价 = 起步价【14元】 + (远程里程数【15公里】 - 起步里程数【3公里】) \times 每公里单价【2.7元】 + (里程数 - 远程里程数【15公里】) \times 远程每公里单价【4元】
- 低速等候时间：4元/4分钟

```
里程数 = int(input("输入里程数(公里): "))
等候时间 = int(input("输入等待时间(分钟): "))

if 里程数 <= 3:
    收费 = 14
elif 3 < 里程数 <= 15:
    收费 = 14 + (里程数 - 3) * 2.7
else:
    收费 = 14 + (15 - 3) * 2.7 + (里程数 - 15) * 4
收费 = 收费 + (等候时间 // 4) * 4
print('收费: ', 收费)
```

输入里程数(公里): 4
输入等待时间(分钟): 5
收费: 20.7

选择结构的嵌套

if 表达式1:

语句块1

if 表达式2:

语句块2

else:

语句块3

else:

语句块4

if 表达式5:

语句块5

缩进必须一致

闰年的条件: 年份能被4整除但不能被100整除(如2004年),

或 能被400整除(如2000年是, 1900年不是)

闰年?

```
print("判断输入年份是否闰年")
```

```
year = int(input("请输入年份: "))
```

```
整除4 = year % 4
```

```
整除100 = year % 100
```

```
整除400 = year % 400
```

```
if 整除4 == 0:
```

```
    if 整除100 != 0 or 整除400 == 0:
```

```
        print("闰年")
```

```
    else:
```

```
        print("不是闰年")
```

```
else:
```

```
    print("不是闰年")
```

```
>>> import calendar
>>> calendar.isleap(2024)
True
```

运行结果:

```
判断输入年份是否闰年
请输入年份: 2024
闰年
```

max()、min()、sum()

最大值

最小值

所有元素之和

```
>>> max([1, 2, 3])           # 列表中的最大值  
3
```

```
>>> min([1, 2, 3])          # 列表中的最小值  
1
```

```
>>> sum([1, 2, 3])          # 列表中元素求和  
6
```

```
>>> sum(range(1, 11))        # 1~10之和  
55
```

```
>>> sum(range(1, 11), 5)     # 1~10之和 + 5  
60
```

字符串

```
>>> max(['2', '111'])        # 字符串最大值  
'2'
```

```
>>> max(['2', '111'], key=len) # 最长的字符串  
'111'
```

type()

判断数据类型

isinstance()

type(object)

```
>>> type(3)                # 查看3的类型
<class 'int'>
>>> type(3.14)
<class 'float'>
```

输入判断

```
n = eval(input('请输入整数: '))
if type(n) != int:
    print('输入的不是整数!')
```

```
>>> isinstance(3, int)    # 3是否为int实例?
True
>>> isinstance(3j, int)
False
>>> isinstance('Hello Python.', str)
True
```

ascii()

转换为ASCII字符串表示形式

ASCII表

(American Standard Code for Information Interchange 美国标准信息交换代码)

ASCII表																											
(American Standard Code for Information Interchange 美国标准信息交换代码)																											
高四位 低四位	ASCII控制字符												ASCII打印字符														
	0000						0001						0010		0011		0100		0101		0100		0111				
	0						1						2		3		4		5		6		7				
	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	Ctrl
0000	0	0		^@	NUL \0	空字符	16	▶	^P	DLE		数据链路转义	32		48	0	64	@	80	P	96	`	112	p			
0001	1	1	☺	^A	SOH	标题开始	17	◀	^Q	DC1		设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q			
0010	2	2	☹	^B	STX	正文开始	18	↕	^R	DC2		设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r			
0011	3	3	♥	^C	ETX	正文结束	19	!!	^S	DC3		设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s			
0100	4	4	♦	^D	EOT	传输结束	20	¶	^T	DC4		设备控制 4	36	\$	52	4	68	D	84	T	100	d	116	t			
0101	5	5	♣	^E	ENQ	查询	21	§	^U	NAK		否定应答	37	%	53	5	69	E	85	U	101	e	117	u			
0110	6	6	♠	^F	ACK	肯定应答	22	—	^V	SYN		同步空闲	38	&	54	6	70	F	86	V	102	f	118	v			
0111	7	7	●	^G	BEL	响铃	23	↕	^W	ETB		传输块结束	39	'	55	7	71	G	87	W	103	g	119	w			
1000	8	8	▣	^H	BS	退格	24	↑	^X	CAN		取消	40	(56	8	72	H	88	X	104	h	120	x			
1001	9	9	○	^I	HT	横向制表	25	↓	^Y	EM		介质结束	41)	57	9	73	I	89	Y	105	i	121	y			
1010	A	10	◐	^J	LF	换行	26	→	^Z	SUB		替代	42	*	58	:	74	J	90	Z	106	j	122	z			
1011	B	11	♂	^K	VT	纵向制表	27	←	^[ESC	le	溢出	43	+	59	;	75	K	91	[107	k	123	{			
1100	C	12	♀	^L	FF	换页	28	└	^\	FS		文件分隔符	44	,	60	<	76	L	92	\	108	l	124				
1101	D	13	♪	^M	CR	回车	29	↔	^]	GS		组分隔符	45	-	61	=	77	M	93]	109	m	125	}			
1110	E	14	🎵	^N	SO	移出	30	▲	^^	RS		记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~			
1111	F	15	🎵	^O	SI	移入	31	▼	^-	US		单元分隔符	47	/	63	?	79	O	95	_	111	o	127	␣			^Backspace 代码: DEL
注：表中的ASCII字符可以用“Alt + 小键盘上的数字键”方法输入。																											
制作: MHL QQ:1208980380 2013/08/08																											

注：表中的ASCII字符可以用“Alt + 小键盘上的数字键”方法输入。

制作：MHL QQ:1208980380 2013/08/08

```
>>> ascii('abc')
```

```
"'abc'"
```

```
# 非ASCII字符转换为转义字符
```

```
>>> ascii('复旦')
```

```
"'\\u590d\\u65e6'"
```

Unicode码：统一码

为每种语言中的每个字符设定了统一的二进制编码，以满足跨语言、跨平台文本转换的要求。英文字母与ASCII相同。

常用 Unicode 编码表

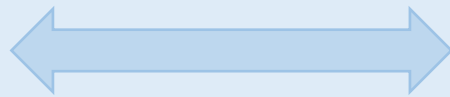
名称	编码范围	举例说明
数字	[0x0030~0x0039]	0 0x0030
大写字母	[0x0041~0x005A]	A 0x0041
小写字母	[0x0061~0x007A]	a 0x0061
箭头	[0x2190~0x21FF]	← 0x2190
数字运算符	[0x2200~0x22FF]	∀ 0x2200
封闭字母数字	[0x2460~0x24FF]	① 0x2460
制表符	[0x2500~0x257F]	≡ 0x2561
方块元素	[0x2580~0x259F]	■ 0x2580

名称	编码范围	举例说明
几何图形	[0x25A0~0x25FF]	▲ 0x25b2
标点符号	[0x2000~0x206F]	“ 0x201c
韩文	[0xAC00~0xD7A3]	갸 0xac20
货币符号	[0x20A0~0x20CF]	£ 0x20a4
泰文	[0x0E00~0x0E7F]	й 0x040d
中日韩符号	[0x3000~0x303F]	【 0x3010
中日韩括号数字	[0x3200~0x32FF]	(四) 0x3223
基础汉字	[0x4E00~0x9FA5]	复 0x590d

`chr(0x590d)` --> '复'

`hex(ord('复'))` --> 0x590d

ord()



chr()

返回单个字符的Unicode数字编码

ordinal
顺序

ord(c, /)

```
>>> ord('a') # 英文ASCII
97
```

#复在Unicode表中对应的十进制

```
>>> ord('复')
22797
```

```
>>> ord('旦')
26086
```

```
>>> chr(ord('A') + 1)
'B'
```

```
>>> chr(ord('上') + 1)
'下'
```

返回Unicode编码对应的字符

chr(i, /)

character

```
>>> chr(97)
'a'
```

22797在Unicode表中对应字符

```
>>> chr(22797)
'复'
```

```
>>> ord(26086)
'旦'
```

Python不支持字符串和数字之间的加法操作

支持中文

str()

转换为字符串

```
str([object=' '])
```

```
>>> str(123)
'123'
>>> str([1,2,3])
'[1, 2, 3]'
>>> str((1,2,3))
'(1, 2, 3)'
>>> str({1,2,3})
'{1, 2, 3}'
```

强制

bytes()

生成字节串

```
bytes([object=' '])
```

```
>>> bytes()                # 空字节串
b''
>>> bytes(3)               # 长度为3的字节串
b'\x00\x00\x00'
>>> bytes('复旦', 'utf8') # 字符串转为字节串
b'\xe5\x8d\x97\xa6'
>>> str(_, 'utf8')         # 字节串转为字符串
'复旦'
```

上一次正确的输出结果

`list()`、`tuple()`、`dict()`、`set()`

把其他类型的数据转换成为列表、元组、字典、集合

```
>>> list('1234')           # 字符串转为列表
```

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

```
>>> tuple('1234')          # 字符串转为元组
```

```
(1, 2, 3, 4)
```

```
>>> dict([('1', 'a'), ('2', 'b'), ('3', 'c')]) # 创建字典
```

```
{'1': 'a', '2': 'b', '3': 'c'}
```

```
>>> set('1112234')          # 创建集合，自动去除重复，不保证输出顺序
```

```
{'4', '2', '3', '1'}
```

random 随机数模块

产生各种分布的**伪随机数**序列，用于*除商业加密解密算法外*的众多领域。

常用函数	描述
<code>random()</code>	生成一个 $[0.0, 1.0)$ 之间的随机小数
<code>randint(a, b)</code>	生成一个 $[a, b]$ 之间的随机整数
<code>uniform(a, b)</code>	生成一个 $[a, b]$ 之间的随机小数， a, b : 整数或浮点数
<code>randrange(start, stop[, step])</code>	生成一个 $[start, stop)$ 之间以 $step$ 为步长的随机整数
<code>choice(seq)</code>	从序列(如，列表)中随机返回一个元素
<code>shuffle(seq)</code>	将序列(如，列表)中的元素随机排列，即打乱次序
<code>seed(a=None)</code>	初始化随机数种子，默认值为当前系统时间
<code>getrandbits(k)</code>	生成一个 k 比特长度的随机整数
<code>sample(seq, k)</code>	从序列(如，列表)中随机选取 k 个元素，以列表类型返回

random 模块常用函数-1

random()

$0.0 \leq \text{随机数} < 1.0$

```
>>> import random
>>> random.random()
0.8797040450095865
```

randint(a, b)

$a \leq \text{随机数} \leq b$

```
>>> import random
>>> random.randint(0, 100)
9
```

uniform(a, b)

$a \leq \text{随机数} \leq b$

```
>>> from random import *
>>> uniform(10.5, 100)
55.75457143863575
```

randrange(start, stop[, step])

```
>>> from random import *
>>> randrange(10, 100, 5)
95
```

random 模块常用函数-2

choices(序列, k=1)

从序列中随机选择1个

```
>>> import random
>>> lst = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> random.choices(lst, k=2) # 随机选择2个
[2, 5]
```

shuffle(序列)

乱序重新排列

```
>>> import random
>>> num = [1, 2, 3, 4, 5, 6, 7, 8]
>>> random.shuffle(num)
>>> num --> [8, 7, 1, 3, 2, 4, 5, 6]
```

seed(a=None)

初始化随机数种子

设置种子便于随机数复现

```
>>> from random import *
>>> seed(10)
>>> random()
0.5714025946899135
>>> random()
0.4288890546751146

>>> seed(10)
>>> random()
0.5714025946899135
>>> random()
0.4288890546751146
```

36 40

time 时间模块

提供系统级精确计时的内置模块，可用来分析程序性能、暂停程序运行。

- 时间获取：`localtime()`、`time()`、`ctime()`、`gmtime()`
- 程序计时：`perf_counter()`、`sleep(n)` 令Python暂停n秒
- 时间格式化：`mktime()`、`strftime()`、`strptime()`

```
>>> import time
>>> date = time.localtime() # 返回当前本地时间
>>> date
time.struct_time( # 时间对象结构体
    tm_year=2024, tm_mon=3, tm_mday=4, # 年、月、日
    tm_hour=10, tm_min=31, tm_sec=24, # 时、分、秒
    tm_wday=0, # 一周的第几天(0:周一)
    tm_yday=64, # 一年中的第几天
    tm_isdst=0) # 是否为夏令时(1:是, 0:否, -1:不确定)
```

例4-2 判断今天是今年的第几天？

```
import time

date = time.localtime()    # 获取当前日期时间
year  = date[0]            # 索引[0]为year
month = date[1]            # 索引[1]为month
day   = date[2]            # 索引[2]为day
# 每个月份中共有多少天
day_month = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
if year%400==0 or (year%4==0 and year%100!=0):    # 判断是否为闰年
    day_month[1] = 29    # 闰年2月为29天, 这天出生的人怎么过生日呢?
if month == 1:
    print(day)
else:
    print(sum(day_month[0 : month-1]) + day)
    # 除最后1个月外, 前面几个月天数之和 + 最后1个月的天数
```


例：程序计时

```
import time
```

```
start_time = time.time() # 记录开始时间
```

```
# 在这里写需要计时的代码或函数调用
```

```
time.sleep(2)           # 让程序休眠2秒
```

```
end_time = time.time()  # 记录结束时间
```

```
elapsed_time = end_time - start_time    # 计算经过的时间（单位为秒）
```

```
print("程序运行时间：", elapsed_time, '秒')
```

程序运行时间： 2.0009498596191406 秒

datetime 内置模块

重新包装了time模块，不但有time类，还提供了**date**类、**datetime**类、timedelta类、timezone类等。

```
>>> from datetime import date
>>> days = date(2024, 1, 1) - date(2023, 1, 1)    # 计算两个日期之间相差多少天
>>> print(days)
365 days, 0:00:00
>>> days.days
365

>>> from datetime import datetime
>>> now = datetime.now()    # 也可以用datetime.today()
>>> now
datetime.datetime(2024, 1, 24, 12, 23, 0, 739602)
>>> print(now)
2024-01-24 12:23:00.739602
>>> now.year
2024
```

第三方库管理工具：pip

Python官方提供的在线第三方库下载、安装、卸载、查找等管理工具。

【注】安装时需要联网。不要在IDLE下运行pip，在Win的命令提示符里运行pip/pip3、Mac终端里运行pip3。

pip install <拟安装库名>

```
命令提示符
Microsoft Windows [版本 10.0.19045.3930]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Sam2023>pip install pypinyin
Collecting pypinyin
  Downloading pypinyin-0.50.0-py2.py3-none-any.whl.metadata (12 kB)
  Downloading pypinyin-0.50.0-py2.py3-none-any.whl (1.4 MB)
----- 1.4/1.4 MB 91.3 kB/s eta 0:00:00
Installing collected packages: pypinyin
Successfully installed pypinyin-0.50.0

C:\Users\Sam2023>_
```

pip uninstall <拟卸载库名>

```
命令提示符
Microsoft Windows [版本 10.0.19045.4046]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Sam2023>pip uninstall pypinyin
Found existing installation: pypinyin 0.50.0
Uninstalling pypinyin-0.50.0:
  Would remove:
    c:\users\sam2023\appdata\local\programs\python\python312\lib\site-packages\pypinyin-0.50.0.dist-info\*
    c:\users\sam2023\appdata\local\programs\python\python312\lib\site-packages\pypinyin\*
    c:\users\sam2023\appdata\local\programs\python\python312\scripts\pypinyin.exe
Proceed (Y/n)? y
Successfully uninstalled pypinyin-0.50.0

C:\Users\Sam2023>_
```

pip -h 列出pip常用子命令

```
C:\Users\Sam2023>pip -h
```

Usage:

pip <command> [options]

Commands:

install
download
uninstall
freeze
inspect
list
show
check
config
search
cache
index
wheel
hash
completion
debug
help

Install packages.

Download packages. 下载第三方库的安装包

Uninstall packages.

Output installed packages in requirements format.

Inspect the python environment.

List installed packages. 列出当前已安装的库名

Show information about installed packages.

Verify installed packages have compatible dependencies.

Manage local and global configuration.

Search PyPI for packages.

Inspect and manage pip's wheel cache.

Inspect information available from package indexes.

Build wheels from your requirements.

Compute hashes of package archives.

A helper command used for command completion.

Show information useful for debugging.

Show help for commands.

国内清华镜像 pip

- 设为默认:

升级 pip 到最新的版本 ($\geq 10.0.0$)

```
>python -m pip install --upgrade pip
```

或

```
>python -m pip install -i https://pypi.tuna.tsinghua.edu.cn/simple --upgrade pip
```

配置到国内清华镜像

```
C:\Users\aa>pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple  
Writing to C:\Users\aa\AppData\Roaming\pip\pip.ini
```

安装 *pypinyin* 模块

```
>pip install pypinyin
```

- 临时使用:

```
>pip install -i https://pypi.tuna.tsinghua.edu.cn/simple pypinyin
```

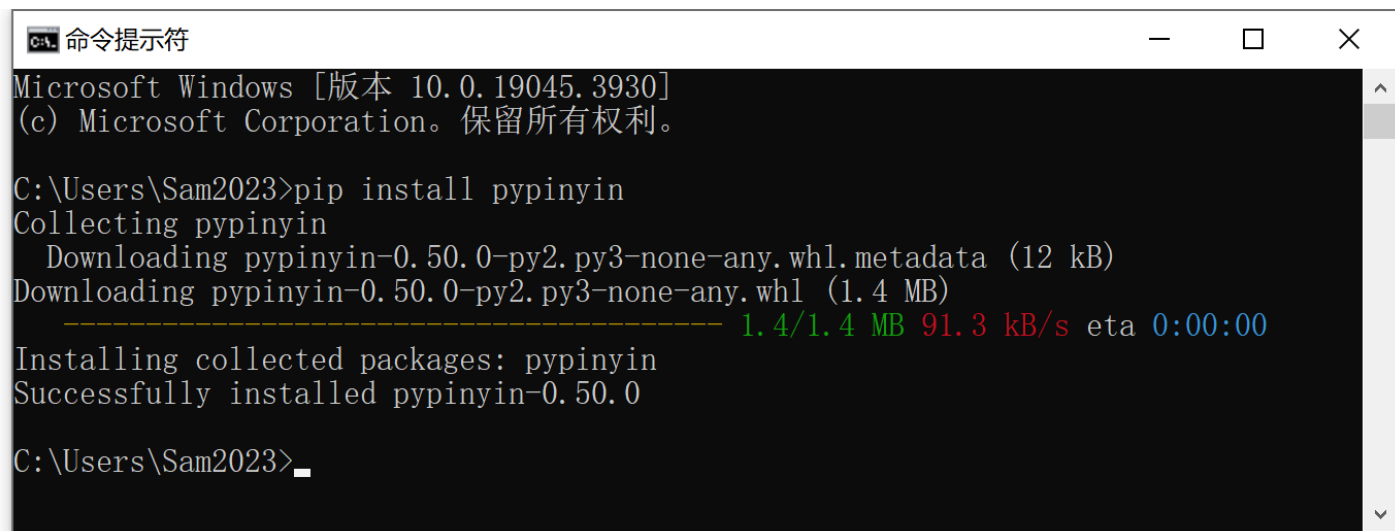
汉字 转 拼音

在命令提示符里安装pypinyin模块:

```
pip install pypinyin
```

```
>>> from pypinyin import pinyin
>>> pinyin('复旦')      # 返回拼音
[['fù'], ['dàn']]
```

```
>>> from pypinyin import lazy_pinyin
>>> lazy_pinyin('复旦')  # 返回拼音
['fu', 'dan']
>>> lazy_pinyin('复旦', 1) # 带声调的拼音
['fù', 'dàn']
>>> lazy_pinyin('复旦', 2) # 另一种拼音形式, 数字表示前面字母的声调
['fu4', 'da4n']
>>> lazy_pinyin('复旦', 3) # 只返回拼音首字母
['f', 'd']
```



```
命令提示符
Microsoft Windows [版本 10.0.19045.3930]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Sam2023>pip install pypinyin
Collecting pypinyin
  Downloading pypinyin-0.50.0-py2.py3-none-any.whl.metadata (12 kB)
  Downloading pypinyin-0.50.0-py2.py3-none-any.whl (1.4 MB)
----- 1.4/1.4 MB 91.3 kB/s eta 0:00:00
Installing collected packages: pypinyin
Successfully installed pypinyin-0.50.0

C:\Users\Sam2023>_
```

中文分词库 jieba “结巴”

在命令提示符里安装jieba模块：

```
pip install jieba
```

将中文文本切分为单独的词，支持简体、繁体。

有3种模式：精确模式、全模式、搜索引擎模式。

```
>>> import jieba
```

```
>>> x = 'Python程序设计'
```

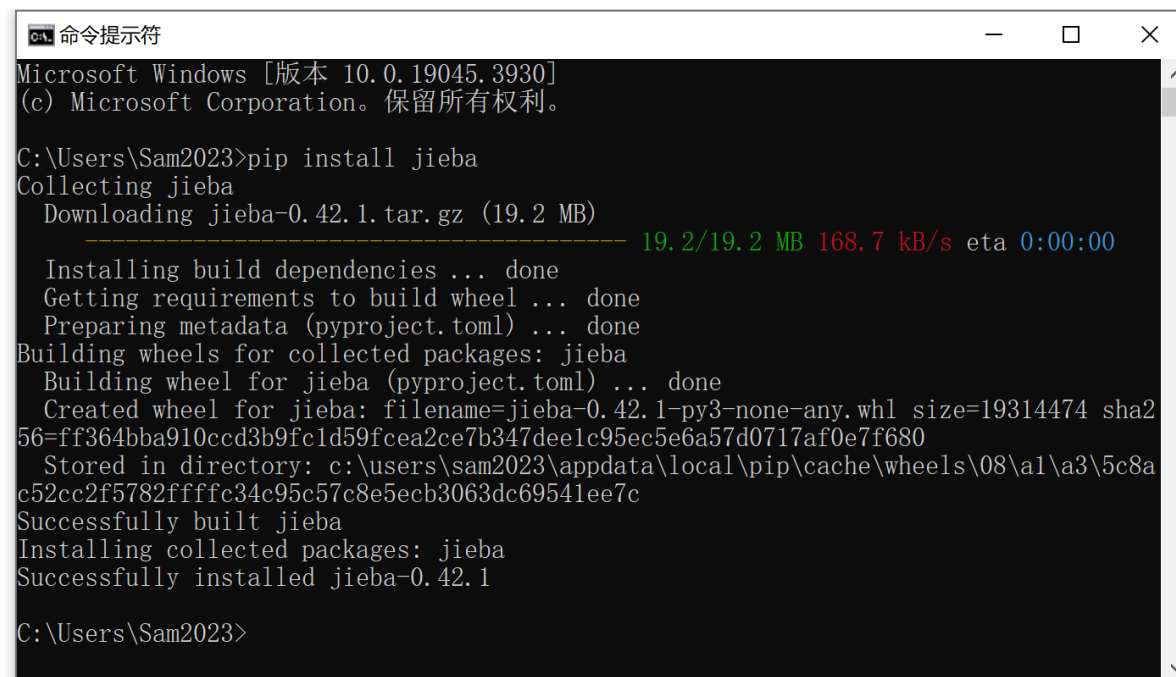
```
>>> jieba.lcut(x) # 精确模式分词，输出为列表 ['Python', '程序设计']
```

```
>>> jieba.lcut(x, True) # 全模式分词 ['Python', '程序', '程序设计', '设计']
```

```
>>> jieba.lcut_for_search(x) # 搜索引擎模式分词 ['Python', '程序', '设计', '程序设计']
```

```
>>> jieba.add_word('太阳花') # 向分词库中增加词条
```

```
>>> jieba.del_word('太阳花') # 从分词库中删除词条
```



```
C:\> 命令提示符
Microsoft Windows [版本 10.0.19045.3930]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Sam2023>pip install jieba
Collecting jieba
  Downloading jieba-0.42.1.tar.gz (19.2 MB)
----- 19.2/19.2 MB 168.7 kB/s eta 0:00:00
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: jieba
  Building wheel for jieba (pyproject.toml) ... done
  Created wheel for jieba: filename=jieba-0.42.1-py3-none-any.whl size=19314474 sha2
56=ff364bba910ccd3b9fc1d59fcea2ce7b347dee1c95ec5e6a57d0717af0e7f680
  Stored in directory: c:\users\sam2023\appdata\local\pip\cache\wheels\08\al\a3\5c8a
c52cc2f5782ffffc34c95c57c8e5ecb3063dc69541ee7c
Successfully built jieba
Installing collected packages: jieba
Successfully installed jieba-0.42.1

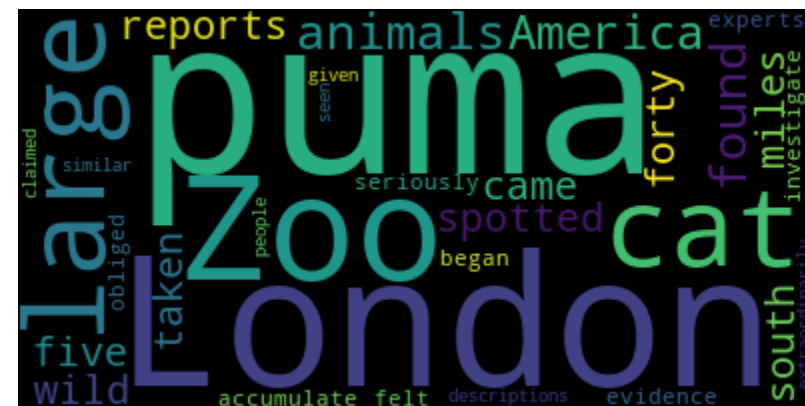
C:\Users\Sam2023>
```

wordcloud 词云库

根据文本生成词云的第三方库。

```
> pip install wordcloud
```

```
>>> from wordcloud import WordCloud # WordCloud类
>>> txt = 'Pumas are large, cat-like animals which are
found in America. When reports came into London Zoo
that a wild puma had been spotted forty-five miles
south of London, they were not taken seriously. However,
as the evidence began to accumulate, experts from the
Zoo felt obliged to investigate, for the descriptions
given by people who claimed to have seen the puma were
extraordinarily similar.'
>>> cloud = WordCloud().generate(txt)
>>> cloud.to_file('Puma.png')
```



相对路径 `C:\Users\Sam2023\AppData\Local\Programs\Python\Python312`

WordCloud类

WordCloud对象创建的常用参数

参 数	功 能
<code>width</code>	生成图片宽度，默认400像素
<code>height</code>	生成图片高度，默认200像素
<code>mask</code>	词云形状，默认None，即方形图
<code>font_path</code>	指定字体文件的完整路径，默认None
<code>font_step</code>	字号步进间隔，默认1
<code>min_font_size</code>	词云中最小的字体字号，默认4号
<code>max_font_size</code>	词云中最大的字体字号，默认None，根据高度自动调整
<code>max_words</code>	词云图中最大词数，默认200
<code>stopwords</code>	被排除词列表，排除词不在词云中显示
<code>background_color</code>	图片背景颜色，默认黑色

WordCloud 类的常用方法

方 法	功 能
<code>generate(text)</code>	由text文本生成词云
<code>to_file(filename)</code>	将词云图保存到文件中