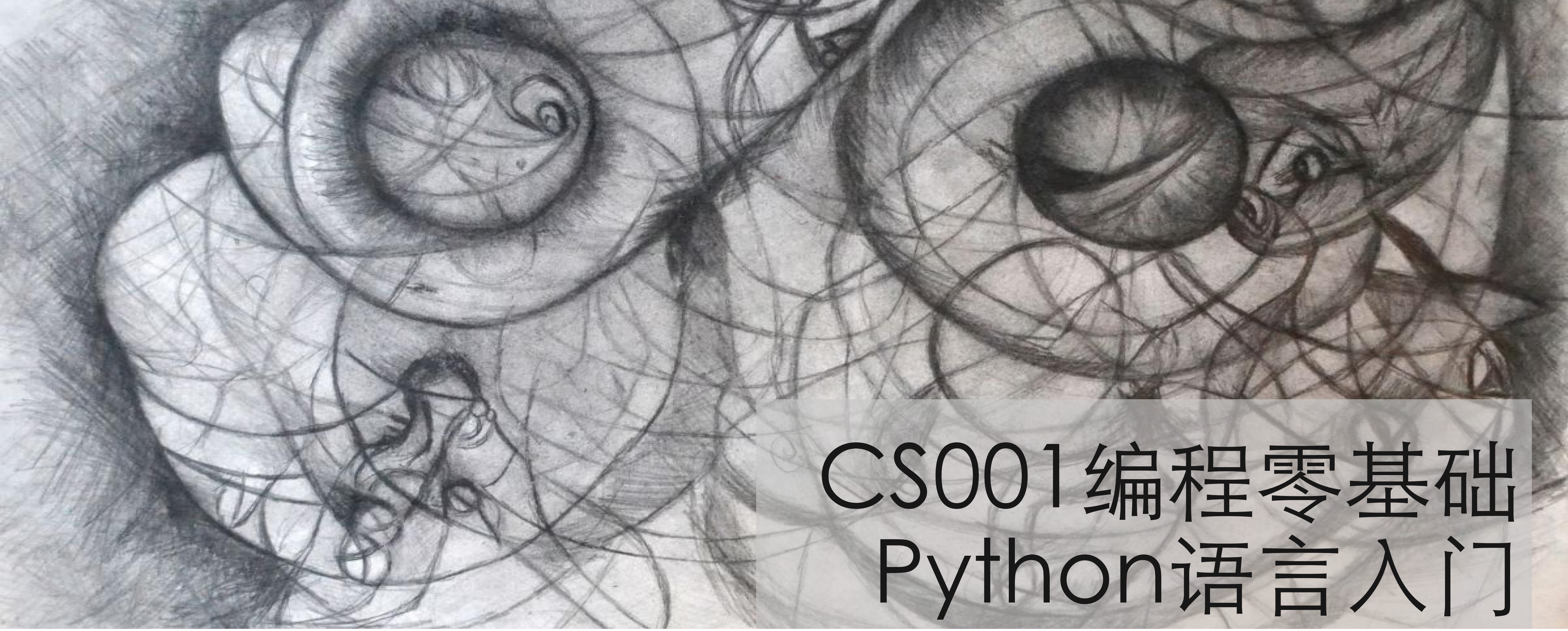
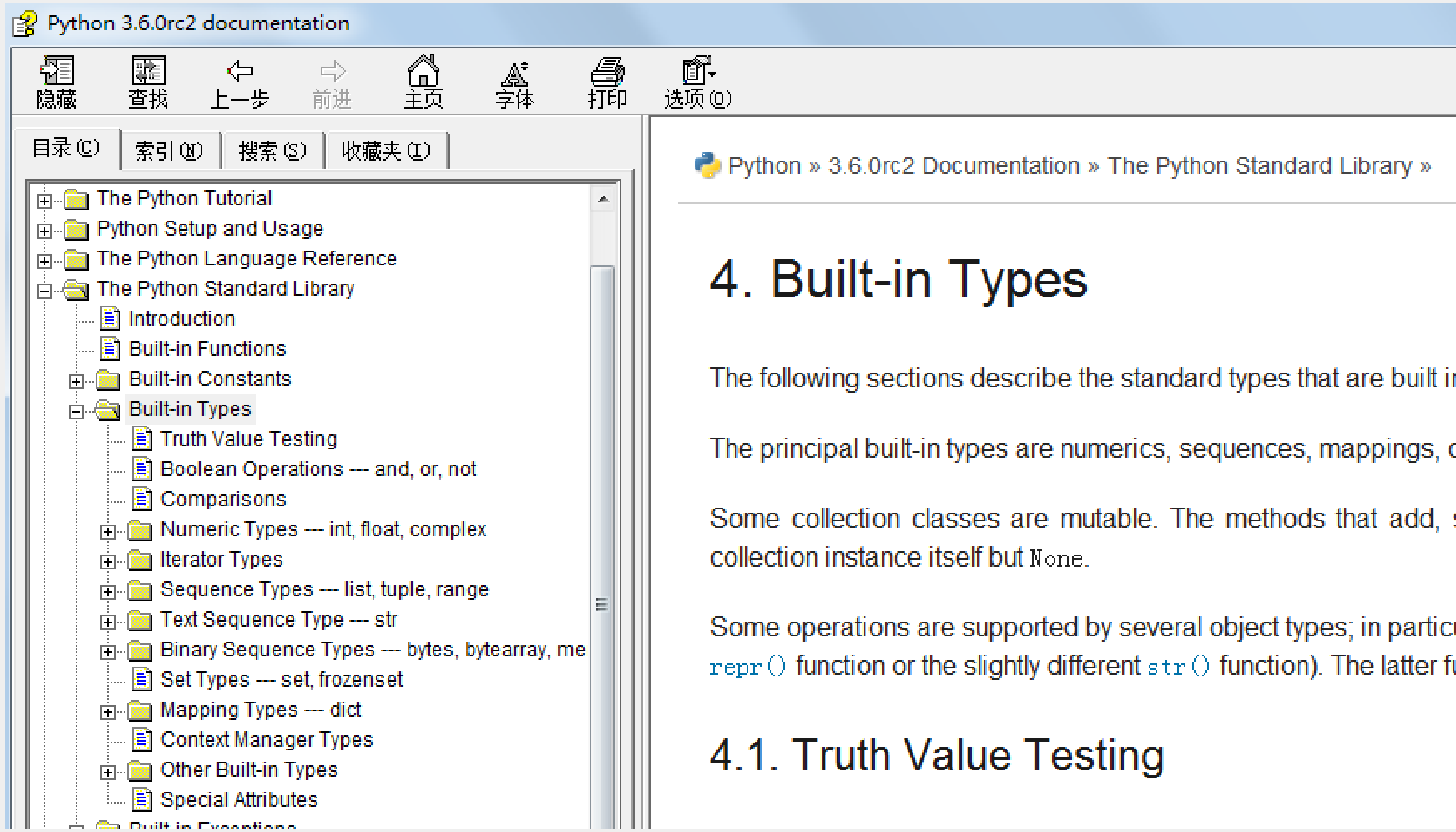


CS001编程零基础 Python语言入门

第二讲

内置数据类型 – Built-in Type

- 在Python IDLE里按F1 查阅帮助文档(或者点击Help - Python Docs)
- 在Python Documentation里点击The Python Standard Library中的Built-in Types
- 你会发现所有内置数据类型，以及使用方法的介绍



The Python Tutorial
Python Setup and Usage
The Python Language Reference
The Python Standard Library
Introduction
Built-in Functions
Built-in Constants
Built-in Types
Truth Value Testing
Boolean Operations — and, or, not
Comparisons
Numeric Types — int, float, complex
Iterator Types
Sequence Types — list, tuple, range
Text Sequence Type — str
Binary Sequence Types — bytes, bytearray, me
Set Types — set, frozenset
Mapping Types — dict
Context Manager Types
Other Built-in Types
Special Attributes

Python » 3.6.0rc2 Documentation » The Python Standard Library »

4. Built-in Types

The following sections describe the standard types that are built i

The principal built-in types are numerics, sequences, mappings, c

Some collection classes are mutable. The methods that add, s
collection instance itself but `None`.

Some operations are supported by several object types; in partic
`repr()` function or the slightly different `str()` function). The latter fu

4.1. Truth Value Testing

整数类型 — int

```
>>> 66+66
132
>>> 6666666*666666666666
4444443999995555556
>>> 10*2
20
>>> 10**2
100
>>> 2**10
1024
```

```
>>> 10**100
1000000000000000000000000
0000000000000000000000000
0000000000000000000000000
0000000000000000000000000
000000000
>>> type(10**100)
<class 'int'>
>>> pow(2,10)
1024
>>> pow(10,2)
100
```

```
>>> 5/2
2.5
>>> 5//2
2
>>> 5%2
1
>>> abs(-10)
10
>>> abs(20)
20
```

浮点类型 – float

```
>>> type(5)
<class 'int'>
>>> type(5.0)
<class 'float'>
>>>5/2
2.5
>>> type(5/2)
<class 'float'>
>>> 5.0//2
2.0
```

```
>>> 0.1+0.2
0.30000000000000004
```

为什么会有这样的结果？

Python在用二进制表示小数时，它的精确度是有限的，存在的这个误差叫做舍入误差（Round-off Error）

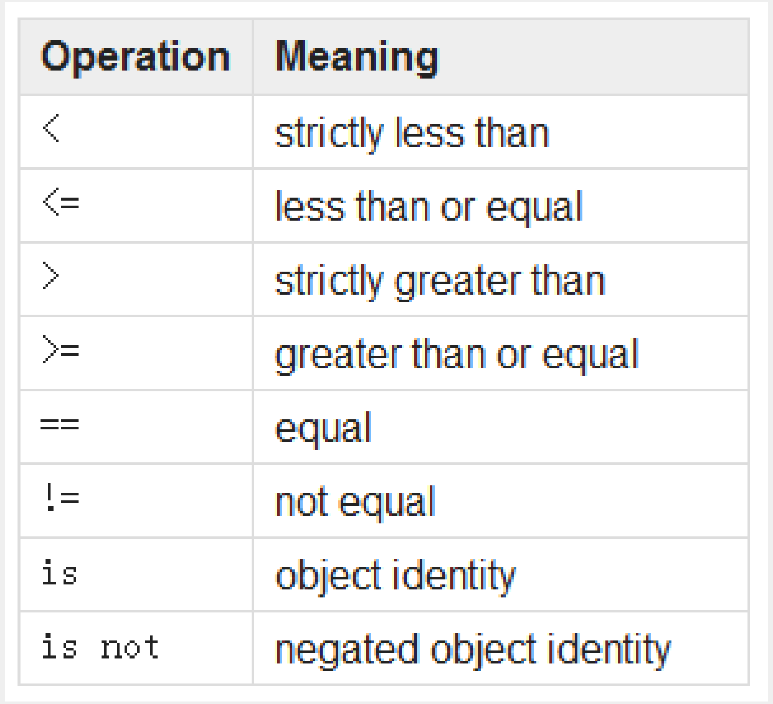
A round-off error, also called rounding error, is the difference between the calculated approximation of a number and its exact mathematical value due to rounding. This is a form of quantization error.

布尔类型 – bool

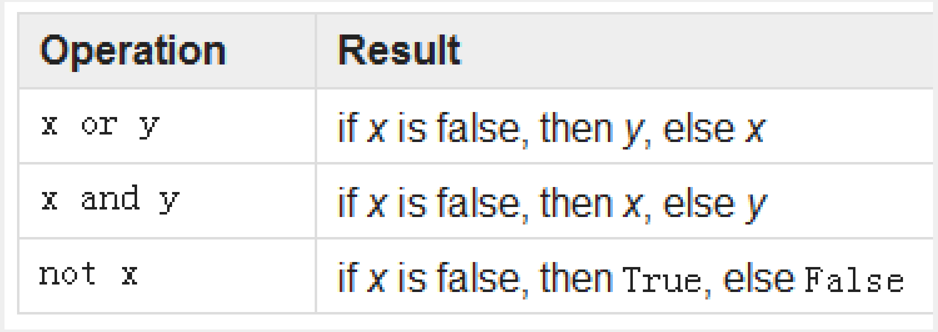
```
>>> 1>2
False
>>> 3>2
True
>>> 3>3
False
>>> 3>=3
True
>>> 3!=2
True
>>> 3-2 == 1
True
>>> 1+1==2==4-2==6/3
True
```

```
>>> not 1==2
True
>>> 1==2 and 3*2>5
False
>>> 1==2 or 3*2>5
True
>>> not 1==2 or 3*2>5
True
>>> 1==1 and 6<5 or 2+2==4
True
>>> 1==1 and 6<5 or not 4==4
False
```

判断顺序： not, and, or



Operation	Meaning
`<`	strictly less than
`<=`	less than or equal
`>`	strictly greater than
`>=`	greater than or equal
`==`	equal
`!=`	not equal
`is`	object identity
`is not`	negated object identity



Operation	Result
`x or y`	if *x* is false, then *y*, else *x*
`x and y`	if *x* is false, then *x*, else *y*
`not x`	if *x* is false, then `True`, else `False`

列表类型 — list

```
>>> a=[0,1,2,3,4,5]
>>> type(a)
<class 'list'>
>>> a[3]
3
>>> a.append(10)
>>> print(a)
[0, 1, 2, 3, 4, 5, 10]
>>> len(a)
7
>>> a[7]
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    a[7]
IndexError: list index out of range
```

```
>>> a.pop()
10
>>> print(a)
[0, 1, 2, 3, 4, 5]
>>> a.insert(2,100)
>>> print(a)
[0, 1, 100, 2, 3, 4, 5]
>>> a.remove(100)
>>> print(a)
[0, 1, 2, 3, 4, 5]
>>> b=[20,21,22,23,24,25]
>>> c=a+b
>>> print(c)
[0, 1, 2, 3, 4, 5, 20, 21, 22, 23, 24, 25]
```

```
>>> max(c)
25
>>> min(c)
0
>>> 22 in c
True
>>> 26 in c
False
>>> [1,2]*3
[1, 2, 1, 2, 1, 2]
>>> c[0]=100
>>> print(c)
[100, 1, 2, 3, 4, 5, 20,
21, 22, 23, 24, 25]
```

列表类型的切片 — slicing

```
>>> a=[0,1,2,3,4,5,6,7,8,9,10]
>>> a[3]
3
>>> a[0]
0
>>> a[3:5]
[3, 4]
>>> a[3:9]
[3, 4, 5, 6, 7, 8]
>>> a[3:9:2]
[3, 5, 7]
```

```
>>> a[3:-1]
[3, 4, 5, 6, 7, 8, 9]
>>> a[3:0]
[]
>>> a[5:1]
[]
>>> a[5:1:-1]
[5, 4, 3, 2]
>>> a[5:-1]
[5, 6, 7, 8, 9]
>>> a[:]
[0, 1, 2, 3, 4, 5, 6, 7,
8, 9, 10]
```

```
>>> a[3:]
[3, 4, 5, 6, 7, 8, 9, 10]
>>> a[:9]
[0, 1, 2, 3, 4, 5, 6, 7, 8]
>>> a[:6:3]
[0, 3]
>>> a[-1]
10
>>> a[-2]
9
>>> a.reverse()
>>> print(a)
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

元组类型 – tuple

```
>>> a=(7,8,9)
>>> type(a)
<class 'tuple'>
>>> a[2]
9
>>> b=tuple([1,2,3,4])
>>> type(b)
<class 'tuple'>
>>> print(b)
(1, 2, 3, 4)
>>> 3 in b
True
```

```
>>> a.append(10)
Traceback (most recent call last):
  File "<pyshell#47>", line 1, in <module>
    a.append(10)
AttributeError: 'tuple' object has no attribute 'append'
>>> a.pop()
Traceback (most recent call last):
  File "<pyshell#52>", line 1, in <module>
    a.pop()
AttributeError: 'tuple' object has no attribute 'pop‘
>>> a[2]=0
Traceback (most recent call last):
  File "<pyshell#48>", line 1, in <module>
    a[2]=0
TypeError: 'tuple' object does not support item assignment
```

范围类型 – range

```
>>> x=range(10,0,-2)
>>> type(x)
<class 'range'>
>>> print(x)
range(10, 0, -2)
>>> print(list(x))
[10, 8, 6, 4, 2]
>>> x[3]
4
>>> x[-1]
2
>>> 5 in x
False
```

```
>>> range(3)
range(0, 3)
>>> list(range(3))
[0, 1, 2]
>>> list(range(0,3))
[0, 1, 2]
>>> list(range(3,6))
[3, 4, 5]
>>> list(range(1,10,3))
[1, 4, 7]
>>> list(range(10,0,-2))
[10, 8, 6, 4, 2]
```

字典类型— dict
(别称映射类型/哈希表hash table)

```
>>> a = dict(one=1, two=2, three=3)
>>> b = {'one': 1, 'two': 2, 'three': 3}
>>> c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
>>> d = dict([('two', 2), ('one', 1), ('three', 3)])
>>> e = dict({'three': 3, 'one': 1, 'two': 2})
>>> a == b == c == d == e
True
>>> type(b)
<class 'dict'>
>>> b.keys()
dict_keys(['one', 'two', 'three'])
>>> b.values()
dict_values([1, 2, 3])
```

```
>>> len(b)
3
>>> b['two']
2
>>> 'three' in b
True
>>> b['four']=4
>>> b
{'one': 1, 'two': 2, 'three': 3, 'four': 4}
>>> del b['two']
>>> b
{'one': 1, 'three': 3, 'four': 4}
```

字符串类型 – str

```
>>> 'abc'
'abc'
>>> "abc"
'abc'
>>> '''abc'''
'abc'
```

```
>>> type("abc")
<class 'str'>
>>> x="abc"
>>> x[2]
'c'
```

```
>>> x[2]="f"
Traceback (most recent call last):
  File "<pyshell#120>", line 1, in <module>
    x[2]="f"
TypeError: 'str' object does not support item assignment
```

```
>>> x.join("defg")
'dabceabcfabcg'
>>> "lie" in "believe"
True
>>> "123" + "456"
'123456'
>>> " ".join("xyz")
'x y z'
>>> "".join(["123","abc","xyz"])
'123abcxyz'
>>> "I love coding.".split()
['I', 'love', 'coding.']
```

```
>>> "The sum of 1 + 2 is {0}".format(1+2)
'The sum of 1 + 2 is 3'
>>> "my name is {0}".format("mike")
'my name is mike'
>>> "my name is {xxx}".format(xxx="mike")
'my name is mike'
>>> '   spacious   '.strip()
'spacious'
>>> 'www.example.com'.strip('cmowz.')
'example'
>>> 'www.example.com'.upper()
'WWW.EXAMPLE.COM'
```

强制类型转换

```
>>> int("234")+float("1.5")
235.5
>>> list(range(0,5))
[0, 1, 2, 3, 4]
>>> tuple('abcd')
('a', 'b', 'c', 'd')
>>> list('abcd')
['a', 'b', 'c', 'd']
>>> str(123)
'123‘
```

```
>>> str(["x","y","z"])
"['x', 'y', 'z']"
>>> "".join(["x","y","z"])
'xyz‘
>>> int(111,2)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    int(111,2)
TypeError: int() can't convert non-string with explicit base
>>> int("111",2)
7
>>> int("111",16)
273
```

程序运行顺序I：IPO (input-process-output)

The input–process–output (IPO) model, or input-process-output pattern, is a widely used approach in systems analysis and software engineering for describing the structure of an information processing program or other process. Many introductory programming and systems analysis texts introduce this as the most basic structure for describing a process.

$$\begin{tikzcd}
\boxed{\text{输入 input}} \arrow[r] & \boxed{\text{处理 process}} \arrow[r] & \boxed{\text{输出 output}}
\end{tikzcd}$$

标准输入input, 标准输出print

```
>>> a=input("Please input a number:")
Please input a number:25
>>> b=input("Please input another number:")
Please input another number:35
>>> a+b
'2535‘
>>>int(a)+int(b)
60
>>> int(input())+int(input())
666
888
1554
```

标准输入input, 标准输出print

```
>>> name=input("what is your name:")
what is your name:mike
>>> print("How are you doing, "+name+"?")
How are you doing, mike?
>>> print("Hi, {somebody}?".format(somebody=name.upper()))
Hi, MIKE?
>>> age=input("how old are you:")
how old are you:18
>>> print("NAME \tAGE\n"+name+"\t"+age)
NAME    AGE
mike    18
>>> print(name+" was "+str(int(age)-10)+" ten years ago")
mike was 8 ten years ago
>>> print("{0} was {1} ten years ago".format(name,int(age)-10))
mike was 8 ten years ago
```

文件输入, 文件输出

建立一个新程序，能够接受文件输入

输入文件的文件名为wordlist.in
可以用IDLE打开看看里面是什么内容

程序文件的文件名为fileinput.py
可以用IDLE打开并且运行

```
#fileinput.py
with open("wordlist.in","r") as f:
    text=f.readlines()
print(text)
```

建立另一个新程序，能够接受文件输入，并输出到文件

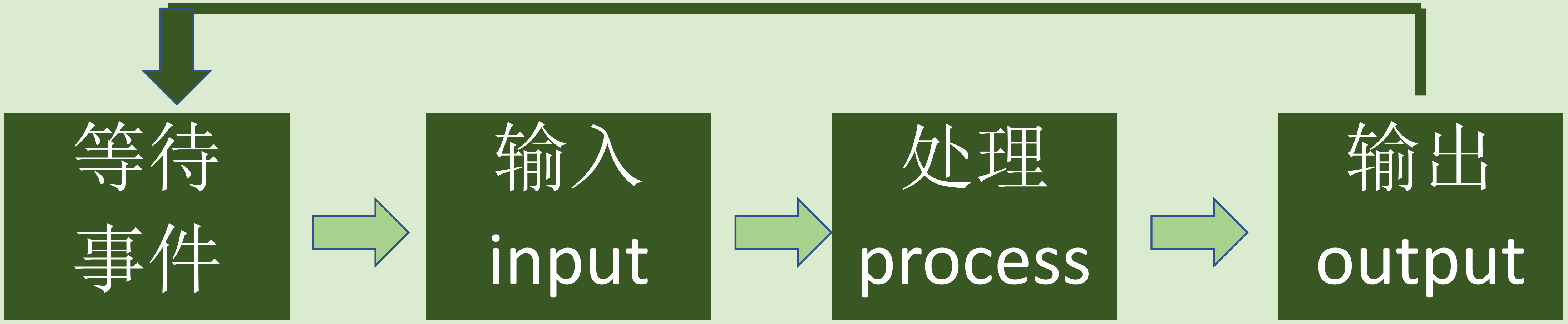
程序文件的文件名为fileio.py
可以用IDLE打开并且运行
输出文件为result.out 可以用IDLE打开看看结果

```
#fileio.py
with open("wordlist.in","r") as fin:
    text=fin.read()
with open("result.out","w") as fout:
    fout.write(text)
    fout.write("\n\nThere are {0} letter 'e'".format(text.count("e")))
    fout.write("\n\nThere are {0} letter 'a'".format(text.count("a")))
    fout.write("\n\nThere are {0} letter 'i'".format(text.count("i")))
    fout.write("\n\nThere are {0} letter 'o'".format(text.count("o")))
```

程序运行顺序II: event-driven

Event-driven programming is a paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs. Event-driven programming is the dominant paradigm used in graphical user interfaces and other applications (e.g. web applications) that are centered on performing certain actions in response to user input

In an event-driven application, there is generally a main loop that listens for events, and then triggers a callback function when one of those events is detected.



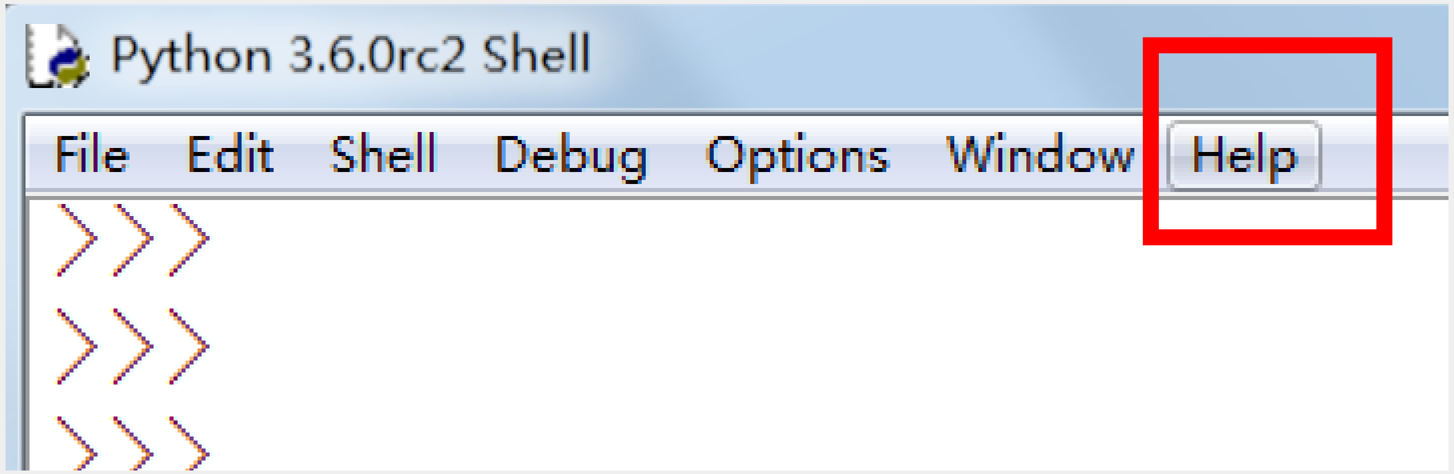
turtle.onscreenclick: 等待鼠标 按键

建立一个新程序，能够等待鼠标 按键

```
#testclick.py
import turtle
turtle.shape("circle")
turtle.onscreenclick(turtle.goto,1)
```

各类小程序Demo

- 在Python IDLE里， 点击帮助栏(Help)
- 在Help下拉菜单里， 点击Turtle Demo
- 在Turtle Demo弹出窗口左上角， 点击Examples进行小程序选择



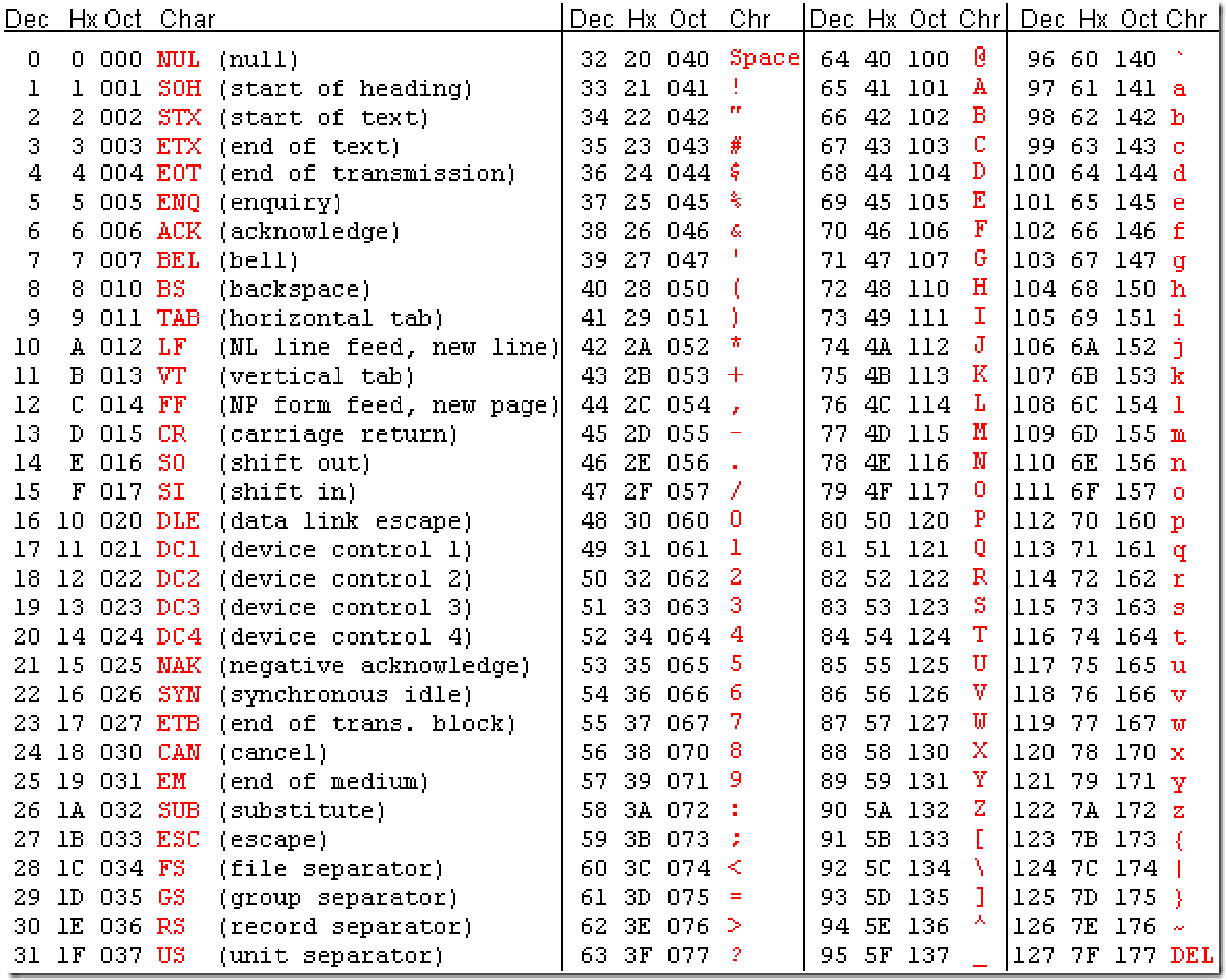
```
>>>
>>>
>>>
```

作业回顾

进制转换

```
>>> int(111,2)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    int(111,2)
TypeError: int() can't convert non-string with explicit base
>>> int("111",2)
7
>>> int('FF',16)
255
```

```
>>> bin(111)
'0b1101111'
>>> bin(256)
'0b100000000'
>>> hex(255)
'0xff'
>>> bin(256)[2:]
'100000000'
>>> hex(255)[2:].upper()
'FF'
```



Dec	Hx	Oct	Char	Dec	Hx	Oct	Chr	Dec	Hx	Oct	Chr	Dec	Hx	Oct	Chr
0	0	000	NUL (null)	32	20	040	Space	64	40	100	@	96	60	140	`
1	1	001	SOH (start of heading)	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX (start of text)	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX (end of text)	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT (end of transmission)	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ (enquiry)	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK (acknowledge)	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL (bell)	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS (backspace)	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB (horizontal tab)	41	29	051	)	73	49	111	I	105	69	151	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	B	013	VT (vertical tab)	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	D	015	CR (carriage return)	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	E	016	SO (shift out)	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	F	017	SI (shift in)	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE (data link escape)	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1 (device control 1)	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2 (device control 2)	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3 (device control 3)	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4 (device control 4)	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN (synchronous idle)	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB (end of trans. block)	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN (cancel)	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM (end of medium)	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133	[	123	7B	173	{
28	1C	034	FS (file separator)	60	3C	074	<	92	5C	134	\	124	7C	174	|
29	1D	035	GS (group separator)	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS (record separator)	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US (unit separator)	63	3F	077	?	95	5F	137	_	127	7F	177	DEL

ASCII码破译

```
>>> chr(101)
'e'
>>> ord("A")
65
```

建立一个新程序，能够接受输入的ASCII码密文，并自动输出译文

```
#decodeASCII.py


#basic method
words=text.split(" ")
for word in words:
    print(chr(int(word)),end="")


#pythonic method
print("".join(list( map(chr,list(map(int,text.split(" ")))))))
```

作业

本次作业提交要求：

#请将每大题答案写在一个独立txt文本文件里，每行依次是各个小题的答案

#该文件名称格式为 学生姓名拼音+大题号.txt

#例如：Huangxiaoming+1.txt

#作业提交请发送附件到stem888@qq.com 邮件主题为学生姓名拼音。例如Huangxiaoming

#截止日期：2017年3月3日23:59