Python

极夜酱

目录

Chapter 1 Python 简介

1.1 Python 简介

1.1.1 编程简介

程序（program）是为了让计算机执行某些操作或者解决问题而编写的一系列有序指令的集合。由于计算机只能够识别二进制数字 0 和 1，因此需要使用特殊的编程语言来描述如何解决问题过程和方法。

算法（algorithm）是可完成特定任务的一系列步骤，算法的计算过程定义明确，通过一些值作为输入并产生一些值作为输出。

流程图（flow chart）是算法的一种图形化表示方式，使用一组预定义的符号来说明如何执行特定任务。

- 圆角矩形：开始和结束
- 矩形：数据处理
- 平行四边形：输入/输出
- 菱形：分支判断条件
- 流程线：步骤

1.1.2 Python 简介

在当今的环境下 Python 是一个热门语言，在全世界的范围内，许多大学都开始使用 Python 作为基础语言的教程，同时在数据分析领域以及人工智能领域也取得了显著的成绩。

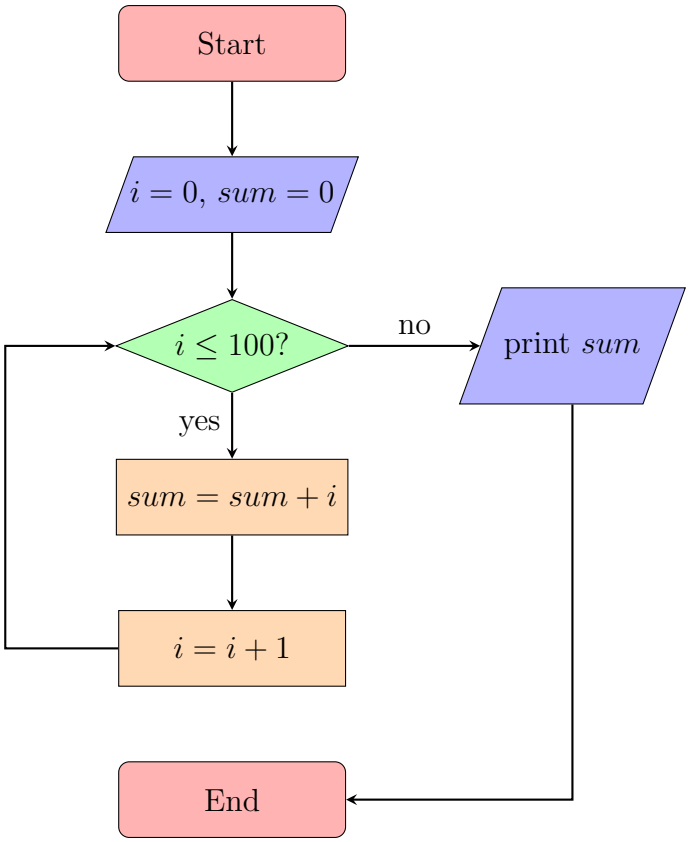


图 1.1: 计算 $\sum_{i=1}^{100} i$ 的流程图

Python 最大的特点就是简单。同样的一个功能，使用 C 语言实现需要 20 行代码，Java 需要 10 行代码，Python 的实现只需要 4 行代码，它从整体的代码量而言是非常简洁的。

- Python 具有很强的可读性，比其他语言更有特色语法结构。
- Python 是解释型语言，这意味着开发过程中没有了编译环节。
- Python 是交互式语言，这意味着可以在提示符后直接执行代码。
- Python 是面向对象语言，这意味着支持面向对象的风格和代码封装。

虽然 Python 提供有交互式的命令模式，但是在很多的情况下，对于程序的编写往往是将其定义在源文件之中，在 Python 里面所有的源文件的后缀名称必须是.py。

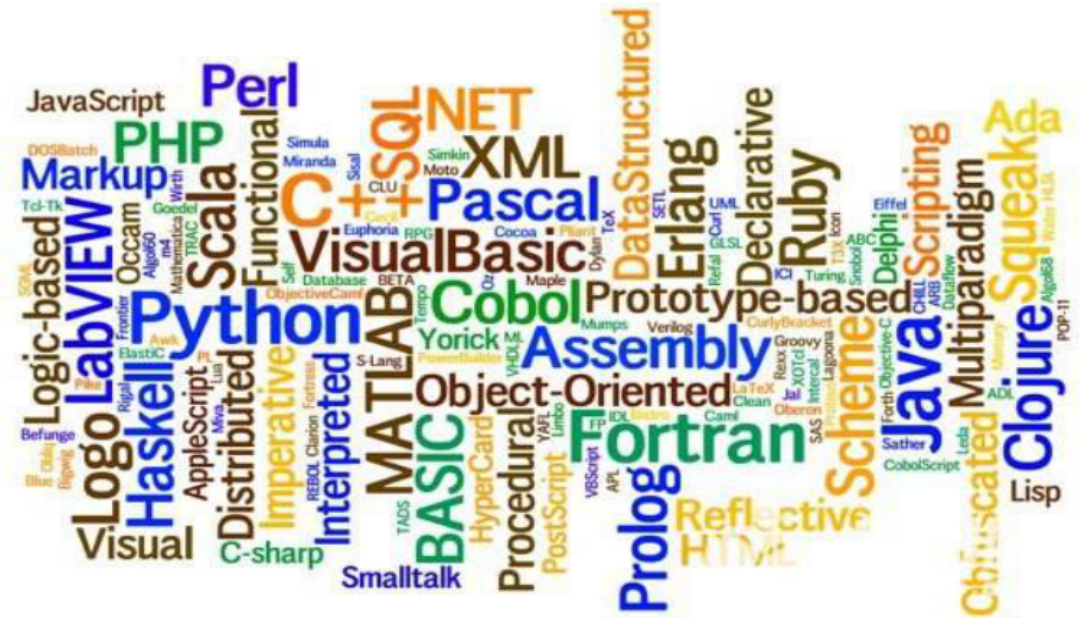


图 1.2: 常见编程语言

1.1.3 Hello World!

输出的时候使用 print() 函数，函数就是一个完成特定功能的代码组织结构。

Hello World!

```
1 print("Hello World!")
```

运行结果

```
Hello World!
```

1.2 注释

1.2.1 注释（Comment）

在进行项目的开发过程中，不可能说一个项目编写完成一次后就在也不进行修改了。所以很多情况下，为了方便下一次的修改，会在一些关键性的代码上进行一些注释信息的定义，开发者根据这些注释的文字信息就可能直到这段代码的主要作用，方便代码的维护。

Python 里面的注释分为两类：

1. 单行注释：将一行内【#】之后的内容视为注释。
2. 多行注释：三引号中间的内容视为注释。

注释

```
"""
这是一条
多行注释
"""
print("Hello World!")       # 输出Hello World!
```

运行结果

```
Hello World!
```

1.3 标识符

1.3.1 标识符（Identifier）

标识符的第一个字符必须是字母表中字母或下划线【_】，标识符的其它部分由字母、数字和下划线组成，标识符对大小写敏感。标识符不可以使用保留字或关键字。标识符应该准确、顾名思义，不要使用汉语拼音。

关键字（key word）也称保留字，关键字是编程语言内置的一些名称，具有特殊的用处和意义。保留字不能用作于标识符名称。Python 的标准库提供了一个 keyword 模块，可以输出当前版本的所有关键字。

关键字

```
1 import keyword
2 print(keyword.kwlist)
```

False	None	True	and	as
assert	break	class	continue	def
del	elif	else	except	finally
for	from	global	if	import
in	is	lambda	nonlocal	not
or	pass	raise	return	try
while	with	yield		

表 1.1: 关键字

1.3.2 变量（Variable）

在程序之中所谓的变量指的是可以被改变的内容，而常量指的是绝对不会被改变的内容。Python 语言最大的特点是变量都是可以被直接定义的，不需要那些复杂的数据类型的声明，直接使用变量名称即可。

所有的变量实际上都会占据内存空间，当一些变量不再使用的时候，可以使用 del 关键字对内存空间进行删除。变量一旦被删除了，那么后续的代码部分将无法继续使用它。

变量

```
1 num = 10
2 print(num)
3 del num
4 print(num)
```

运行结果

```
10
NameError: name 'a' is not defined
```

1.4 数据类型

1.4.1 数据类型

Python 中的变量不需要声明，但是每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建。在 Python 中变量没有类型，我们所说的“类型”是变量所指的内存中对象的类型。使用 type() 函数可以获取变量的数据类型。

在 Python 中使用等号【=】用来给变量赋值，赋值运算符左边是一个变量名，赋值运算符右边是存储在变量中的值。

布尔是 19 世纪一位英国数学家的名字，在开发之中布尔主要用于程序的逻辑分支处理，数值包括 True 和 False。

按照各个传统编程语言的做法来讲，整数 / 整数 = 整数，但是 Python 认为这个结果应该包含有小数（整数的结果变为浮点数类型）。

数据类型

```
num = 123       # 整型
PI = 3.1415     # 浮点型
s = "hello"     # 字符串
flag = True     # 布尔

print(num)
print(PI)
print(s)
print(flag)

print(type(num))
print(type(PI))
print(type(s))
print(type(flag))

```

```
16 print(10 / 4)
17 print(type(10 / 4))
```

运行结果

```
123
3.1415
hello
True
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
2.5
<class 'float'>
```

1.4.2 字符串（String）

字符串是开发中最为重要的概念，一个编程语言是否好用，很大程度上也是取决于是否提供有字符串类型。Python 中可以直接使用引号（单引号或双引号）进行字符串的定义。Python 并没有字符的概念，所以对于引号表示的含义是相同的。

使用【+】运算符可以进行字符串的连接处理。

字符串连接

```
1 str = "Hello" + "World"
2 str += "!"
3 print(str)
```

运行结果

```
HelloWorld!
```

1.4.3 转义字符

在一个字符串描述的过程中，有可能会有一些特殊字符的信息。

转义字符	描述
\\	表示一个反斜杠 \
\'	表示一个单引号 '
\"	表示一个双引号 "
\n	换行
\t	横向制表符

表 1.2: 转义字符

转义字符

```
1 print("全球最大同性交友网站\n\"https://github.com\"")
```

运行结果

```
全球最大同性交友网站
"https://github.com"
```

1.5 数据输入

1.5.1 input()

input() 用于接受键盘输入。在所有编程语言里面，输入数据的类型是字符串类型。

数据输入

```
1 data = input("输入数据：")
2 print(data)
3 print(type(data))
```

运行结果

```
输入数据：123
123
<class 'str'>
```

1.5.2 转换函数

很多时候可能需要的是各种类型，例如：整数、浮点数、布尔型，或者将其它类型变为字符串型。如果字符串不是由数字所组成，那么程序的执行就会产生异常。

函数	功能
int(数据)	将指定数据转为整型数据
float(数据)	将指定数据转为浮点型数据
bool(数据)	将指定数据转为布尔型数据
str(数据)	将指定数据转为字符串型数据

表 1.3: 转换函数

加法计算（Bug 版本）

```
num1 = float(input("输入第一个数字: "))
num2 = float(input("输入第二个数字: "))
result = num1 + num2
print(num1 + "+" + num2 + "=" + result)
```

运行结果

```
输入第一个数字: 11.1
输入第二个数字: 22.2
TypeError: unsupported operand type(s) for +: 'float' and 'str'
```

加法计算（正确版本）

```
num1 = float(input("输入第一个数字: "))
num2 = float(input("输入第二个数字: "))
result = num1 + num2
print(str(num1) + "+" + str(num2) + "=" + str(result))
```

运行结果

```
输入第一个数字: 11.1
输入第二个数字: 22.2
11.1+22.2=33.3
```

1.6 格式化输出

1.6.1 格式化输出

为了解决数据的输出问题，Python 提供有格式化的输出操作。

标记	功能
%c	格式化字符
%s	格式化字符串
%d	格式化整型
%f	格式化浮点型，可以设置保留精度
%e	科学计数法，使用小写字母 e
%E	科学计数法，使用大写字母 e
%g	%f 和%e 的简写
%G	%f 和%E 的简写
%u	格式化无符号整型
%o	格式化无符号八进制数
%x	格式化无符号十六进制数
%X	格式化无符号十六进制数（大写字母）

表 1.4: 格式化输出

格式化字符串

```
1 name = "小灰"
2 age = 16
3 height = 175.6
4
5 print("姓名：%s，年龄：%d，身高：%.2f" % (name, age, height))
```

运行结果

```
姓名：小灰，年龄：16，身高：175.60
```

1.6.2 分隔符

在默认情况下，使用 print() 输出数据时，都会使用换行作为分隔符号。如果不希望使用换行，则可以追加一个 end 参数。

分隔符

```
1 sequence = "1 2 4 8 16 32 64"
2 print("序列：", end='')
3 print(sequence, end='...')
```

运行结果

```
序列：1 2 4 8 16 32 64...
```

1.7 运算符

1.7.1 算术运算符

运算符	功能
+	两个数相加
-	得到负数或一个数减去另一个数
*	两个数相乘或是返回一个被重复若干次的字符串
/	两数相除
%	返回除法的余数
**	幂
//	整除（向下取整）

表 1.5: 算术运算符

计算圆的面积

```
1 PI = 3.14159
2 r = float(input("输入半径: "))
3 area = PI * r ** 2
4 print("面积 = %.2f" % area)
```

运行结果

```
输入半径: 5
面积 = 78.54
```

逆序三位数

```
1 num = int(input("输入一个正三位数: "))
2 a = num // 100
3 b = num // 10 % 10
4 c = num % 10
```

```
5 print("逆序：%d" % (c*100 + b*10 + a))
```

运行结果

```
输入一个正三位数：520
逆序：25
```

1.7.2 复合赋值运算符

运算符	功能
+=	a += b 等价于 a = a + b
-=	a -= b 等价于 a = a - b
*=	a *= b 等价于 a = a * b
/=	a /= b 等价于 a = a / b
%=	a %= b 等价于 a = a % b
**=	a **= b 等价于 a = a ** b
//=	a //= b 等价于 a = a // b

表 1.6: 复合赋值运算符

1.7.3 比较运算符

运算符	功能
==	等于
!=	不等于
>	大于
<	小于
>=	大于等于
<=	小于等于

表 1.7: 比较运算符

Chapter 2 判断

2.1 逻辑运算符

2.1.1 逻辑运算符

Python 中逻辑运算符有三种：

1. 逻辑与 and：当多个条件同时为真，结果为真。

条件 1	条件 2	条件 1 and 条件 2
T	T	T
T	F	F
F	T	F
F	F	F

表 2.1: 逻辑与

2. 逻辑或 or：多个条件有一个为真时，结果为真。

条件 1	条件 2	条件 1 or 条件 2
T	T	T
T	F	T
F	T	T
F	F	F

表 2.2: 逻辑或

3. 逻辑非 not：条件为真时，结果为假；条件为假时，结果为真。

条件	not 条件
T	F
F	T

表 2.3: 逻辑非

2.2 if

2.2.1 if

分支结构最大特征就是可以进行指定条件的判断处理，关键字为 if、elif、else。

每一个满足条件之后的语句都可以有多条，并且在 Python 里面是利用缩进来确定语句的关系。使用逻辑运算符可以进行若干个条件的连接。

单分支

```
age = 15
if 0 < age < 18:
    print("未成年")
```

双分支

```
age = 30
if 0 < age < 18:
    print("未成年人")
else:
    print("成年人")
```

多分支

```
score = 76

if 90 <= score <= 100:
    print("优秀")
elif score >= 60:
    print("合格")
else:
    print("不合格")
```

判断整数奇偶

```python
num = int(input("输入一个正整数: "))

if num > 0:
    if num % 2 == 0:
        print("%d是偶数" % num)
    else:
        print("%d是奇数" % num)
```

运行结果

```
输入一个正整数: 66
66是偶数
```

2.3 断言

2.3.1 断言（Assertion）

设置断言表达式后，当满足条件时程序正常执行，当断言失败时，程序会中断执行。在进行断言时可以配置错误的提示信息，否则很难知道那块代码出现了错误。

通过断言可以直接查找出程序的错误，但从另外一个角度来讲，断言由于不受到程序逻辑的控制，可能会造成许多的额外的问题，在实际的开发之中慎用。

断言

```
1  import math
2
3  print("计算三角形面积")
4  a = float(input("第一条边: "))
5  b = float(input("第二条边: "))
6  c = float(input("第三条边: "))
7
8  assert a + b > c, "边长不合法"
9  assert a + c > b, "边长不合法"
10 assert b + c > a, "边长不合法"
11
12 p = (a + b + c) / 2      # 半周长
13 area = math.sqrt(p * (p-a) * (p-b) * (p-c)) # 海伦公式
14 print("面积 = %.2f" % area)
```

运行结果

```
计算三角形面积
第一条边: 1
第二条边: 1
第三条边: 2
AssertionError: 边长不合法
```

Chapter 3 循环

3.1 while

3.1.1 while

在 while 循环中，当条件满足时重复循环体内的语句。如果条件永远为真，循环会永无止境的进行下去（死循环），因此循环体内要有改变条件的机会。

控制循环次数的方法就是设置循环变量：初值、判断、更新。

while 循环的特点是先判断、再执行，所以循环体有可能会进入一次或多次，也有可能一次也不会进入。

```
while condition:
    # code
```

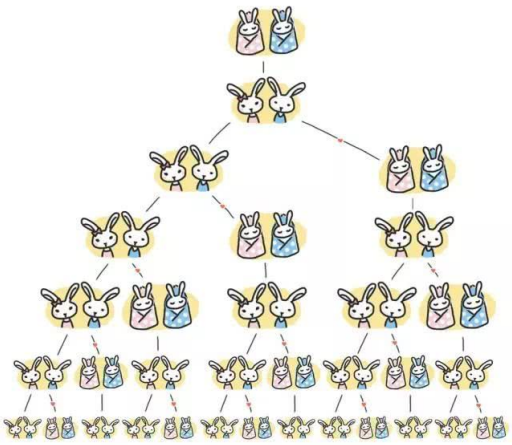
计算 5 个人的平均身高

```
total = 0
i = 1

while i <= 5:
    height = float(input("输入第%d个人的身高： " % i))
    total += height
    i += 1

average = total / 5
print("平均身高：%.2f" % average)
```

运行结果

```
输入第1个人的身高：160.8
输入第2个人的身高：175.2
输入第3个人的身高：171.2
输入第4个人的身高：181.3
输入第5个人的身高：164
平均身高：170.5
```

斐波那契数列



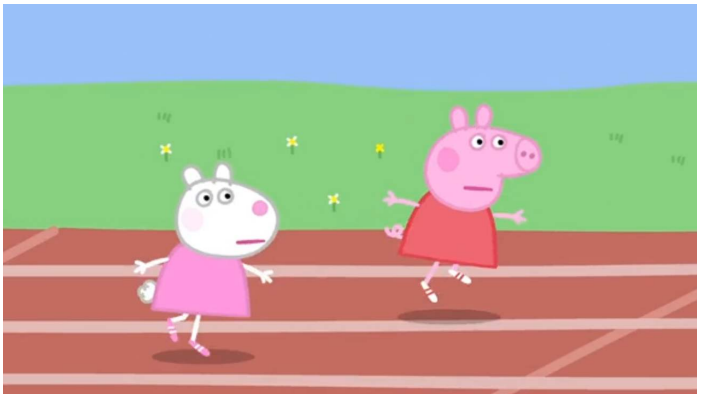
```
num1 = 0
num2 = 1
while num2 < 1000:
    print(num2, end=' ')
    num1, num2 = num2, num1 + num2
```

运行结果

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

3.1.2 死循环

对于循环的操作而言，一定要避免死循环所带来的问题。所谓的死循环指的就是循环结束条件没有得到正常的修改，导致程序无法结束。如果在执行中真的出现了死循环，命令行方式下可以直接按下 Ctrl+C 来中断执行。



猜数字

```
import random           # 随机模块

answer = random.randint(1, 100)     # 产生1-100之间的随机数
cnt = 0         # 猜测次数

while True:
    num = int(input("猜一个1-100之间的数字：  "))
    cnt += 1

    if num > answer:
        print("猜大了")
    elif num < answer:
        print("猜小了")
    else:       # 猜对了就跳出循环
        break

print("猜对了！ 你一共用了%d次猜对！ " % cnt)
```

运行结果

```
猜一个1-100之间的数字：50
猜大了！
猜一个1-100之间的数字：25
猜小了！
猜一个1-100之间的数字：37
猜小了！
猜一个1-100之间的数字：43
猜小了！
猜一个1-100之间的数字：46
猜小了！
猜一个1-100之间的数字：48
猜小了！
猜一个1-100之间的数字：49
猜对了！你一共用了7次猜对！
```

3.2 for

3.2.1 for

for 循环的功能是在输出指定范围内的数据操作。在每一次 for 循环里面都会自动的获取给定范围中的每一个元素，并且在 for 语句里面进行相关的处理操作。

range() 生成一个从 0 开始的数据范围，并且是按照线性的方式增长的。range() 也可以设置数据的开始值，默认从 0 开始。range() 的步长默认为 1。

for

```
for i in range(5):
    print(i, end=' ')
print()

for i in range(10, 20):
    print(i, end=' ')
print()

for i in range(1, 10, 2):
    print(i, end=' ')
print()
```

运行结果

```
0 1 2 3 4
10 11 12 13 14 15 16 17 18 19
1 3 5 7 9
```

计算 1 ～ 100 的累加和

```
sum = 0
```

```
2 for i in range(1, 101):
3     sum += i
4 print("累加: %d" % sum)
```

运行结果

```
累加: 5050
```

计算 $1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}$

```
1 sum = 0
2 n = int(input("输入n: "))
3
4 for i in range(1, n+1):
5     sum += 1 / i
6 print(sum)
```

运行结果

```
输入n: 10
2.9289682539682538
```

for 循环主要是通过序列的形式完成的范围设置，而字符串也属于序列。

单个的字符是一种特殊的类型，是用单引号表示字符字面量。每一个字符都有其对应的码值。

ASCII 全称 American Standard Code for Information Interchange（美国信息交换标准代码），一共定义了 128 个字符。

ASCII	字符	ASCII	字符	ASCII	字符	ASCII	字符
0	NUT	32	(space)	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b

3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	’	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	/	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

表 3.1: ASCII 码表

小写字母转大写

```
str = "Hello World!"

for s in str:
    # 小写字母
    if 97 <= ord(s) <= 122:      # ord(): 字符转ASCII码
        s = chr(ord(s) - 32)     # chr(): ASCII码转字符
    print(s, end='')
```

运行结果

```
HELLO WORLD!
```

3.2.2 嵌套循环

循环也可以进行嵌套使用。

九九乘法表

1*1=1	1*2=2	1*3=3	1*4=4	1*5=5	1*6=6	1*7=7	1*8=8	1*9=9
2*1=2	2*2=4	2*3=6	2*4=8	2*5=10	2*6=12	2*7=14	2*8=16	2*9=18
3*1=3	3*2=6	3*3=9	3*4=12	3*5=15	3*6=18	3*7=21	3*8=24	3*9=27
4*1=4	4*2=8	4*3=12	4*4=16	4*5=20	4*6=24	4*7=28	4*8=32	4*9=36
5*1=5	5*2=10	5*3=15	5*4=20	5*5=25	5*6=30	5*7=35	5*8=40	5*9=45
6*1=6	6*2=12	6*3=18	6*4=24	6*5=30	6*6=36	6*7=42	6*8=48	6*9=54
7*1=7	7*2=14	7*3=21	7*4=28	7*5=35	7*6=42	7*7=49	7*8=56	7*9=63
8*1=8	8*2=16	8*3=24	8*4=32	8*5=40	8*6=48	8*7=56	8*8=64	8*9=72
9*1=9	9*2=18	9*3=27	9*4=36	9*5=45	9*6=54	9*7=63	9*8=72	9*9=81

表 3.2: 九九乘法表

```
for i in range(1, 10):
    for j in range(1, 10):
        print("%d*%d=%d" % (i, j, i*j), end='\t')
    print()
```

输出图案

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

```
for i in range(5):
    for j in range(i+1):
        print("*", end='')
    print()
```

3.3 break or continue?

3.3.1 循环控制

循环控制语句的作用是控制当前的循环结构是否继续向下执行，如果不进行控制，那么会根据既定的结构重复执行。如果有一些特殊的情况导致循环的执行中断，就称为循环的控制语句。循环控制语句的关键字有 break 和 continue。

break 的作用是跳出当前循环，执行当前循环之后的语句。break 只能跳出一层循环，如果是嵌套循环，那么需要按照嵌套的层次，逐步使用 break 来跳出。break 语句只能在循环体内和 switch 语句内使用。

continue 的作用是跳过本轮循环，开始下一轮循环的条件判断。continue 终止当前轮的循环过程，但它并不跳出循环。

break

```
1 for i in range(10):
2     if i == 5:
3         break
4     print(i, end=' ')
```

运行结果

```
1 2 3 4
```

continue

```
1 for i in range(10):
2     if i == 5:
3         continue
4     print(i, end=' ')
```

运行结果

```
1 2 3 4 6 7 8 9 10
```