

CHAPTER 2

Python语法基础



主讲人：吴春彪

2025年5月11日

01

Python 编程语言



Python 编程语言

- 目前主流的编程语言包括：**Python**（简洁易学，AI/数据分析首选）、**JavaScript**（前端核心，全栈开发）、**Java**（企业级后端/Android开发）、**C/C++**（高性能系统/游戏编程）。

2025 年 4 月编程语言排行榜				
编程语言 TIOBE 4月				
排名	编程语言	流行度	对比上月	年度明星
1	Python	23.08%	▼ -0.77%	2024, 2021
2	C++	10.33%	▼ -0.75%	2022, 2003
3	C	9.94%	▲ 0.41%	2019, 2017
4	Java	9.63%	▼ -0.73%	2015, 2005
5	C#	4.39%	▼ -0.48%	2023
6	JavaScript	3.71%	▲ 0.25%	2014
7	Go	3.02%	▲ 0.24%	2016, 2009
8	Visual Basic	2.94%	▲ 0.42%	-
9	Delphi/Object Pas...	2.53%	▲ 0.38%	-
10	SQL	2.19%	▼ -0.38%	-

11	Fortran	2.04%	▲ 0.34%	-
12	Scratch	1.35%	▼ -0.31%	-
13	PHP	1.31%	▼ -0.17%	2004
14	R	1.19%	▲ 0.25%	-
15	Ada	1.09%	▲ 0.24%	-
16	MATLAB	1.07%	▲ 0.09%	-
17	Assembly language	0.97%	▲ 0.1%	-
18	Rust	0.96%	▼ -0.27%	-
19	Perl	0.91%	▲ 0.21%	-
20	COBOL	0.91%	▲ 0.07%	-

Python 编程语言

- Python语言诞生于1990年，由**吉多·范罗苏姆**（Guido van Rossum）设计并领导开发。Python是一种**跨平台、开源、免费**的解释型、面向对象的高级动态通用脚本语言。

■ Python官网：<https://www.python.org/>

- ① 1994 年 01 月，Python 1.0 正式发布；.
- ② 2000 年 10 月，Python 2.0 正式发布 --- Python 2.x --- 遗产
- ③ 2008 年 12 月，Python 3.0 正式发布 --- Python 3.x --- 现在和未来

最新版本
Python 3.13

Python version	Maintenance status	First released	End of support	Release schedule
3.14	pre-release	2025-10-01 (planned)	2030-10	PEP 745
3.13	bugfix	2024-10-07	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	security	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	end of life, last release was 3.8.20	2019-10-14	2024-10-07	PEP 569

Python 编程语言

■ Python语言的特点：**语法简洁，生态高产**，拥有丰富的标准库和强大的第三方生态，**广泛应用于AI、数据分析、Web开发及自动化脚本**，是入门和工业级项目的热门选择。

- ① Python语言**语法简洁**，易学易懂。
- ② Python语言**跨平台、可移植、可扩展，开源免费**。
- ③ Python语言支持**面向过程**和**面向对象**编程。
- ④ Python语言拥有**功能丰富的第三方库**，改善编程效率和体验。

NumPy 

matplotlib 


OpenCV

 SciPy

StudyAi.com
django
Django Web Framework

 scikits
learn

PyAutoGUI

Python 编程语言

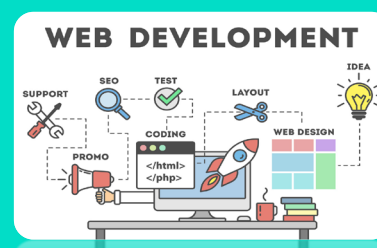
■ Python语言的应用场景



大数据分析
大数据分析工程师



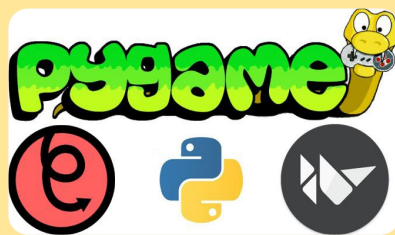
人工智能
人工智能工程师



Web开发
Web开发工程师



Python爬虫
爬虫开发工程师



Python游戏
游戏开发工程师



自动化运维
运维自动化工程师

02

数据类型



Python 程序的格式框架

1	<i># Ch02.1 TempConvert.py</i>	
2	TempStr = input(“请输入带有符号的温度值 :”)	# input() 输入语句
3	if TempStr[-1] in ['F', 'f']:	# if 条件语句
4	C = (eval(TempStr[0: -1]) - 32)/1.8	# 摄氏温度计算表达式
5	print(“转换后的温度是 {:.2f} C”.format(C))	# print() 输出语句
6	elif TempStr[-1] in ['C', 'c']:	# if 条件语句
7	F = 1.8 * eval (TempStr[0: -1]) + 32	# 华氏温度计算表达式
8	print("转换后的温度是{:.2f}F".format(F))	
9	else:	# if 条件语句
10	print("输入格式错误")	

- 代码高亮: 编程的色彩辅助体系, 语法突出。不是语法要求, 不影响代码运行结果。

Python 程序的格式框架

一、缩进

- **缩进:** Python语言采用严格的“缩进”来表明程序的格式框架。缩进指每一行代码开始前的空白区域，用来表示**代码的之间的包含和层次关系**。无需缩进代码顶行编写，不留空白。
- 缩进是Python语法一部分，严格明确，如果缩进不正确程序运行报错。
- Python程序中缩进方式需保持一致，推荐使用**4个空格**或者**一个Tab键**实现缩进。

```
if TempStr[-1] in ['F', 'f']:
    C = (eval(TempStr[0:-1]) - 32)/1.8
    print("转换后的温度是 {:.2f} C".format(C))
elif TempStr[-1] in ['C', 'c']:
    F = 1.8 * eval(TempStr[0:-1]) + 32
    print("转换后的温度是 {:.2f} F".format(F))
else:
    print("输入格式错误")
```

Python 程序的格式框架

二、注释

- **注释**是用来对程序中**关键代码的原理和用途进行解释和说明**，提升代码的可读性和可维护性。
注释是辅助性文字，不被Python执行。

■ Python语言有两种注释方法：

- **单行注释：**
以#开头，其后内容为注释
注释内容
- **多行注释：**
以'''(三个引号) 开头和结尾
'''
这是多行注释第一行
这是多行注释第二行
'''

1	# Ch02.1 TempConvert.py	
2	TempStr = input(“请输入带有符号的温度值:”)	# input() 输入语句
3	if TempStr[-1] in ['F', 'f']:	# if 条件语句
4	C = (eval(TempStr[0: -1]) - 32)/1.8	# 摄氏温度计算表达式
5	print(“转换后的温度是 {:.2f} C”.format(C))	# print() 输出语句
6	elif TempStr[-1] in ['C', 'c']:	# if 条件语句
7	F = 1.8 * eval (TempStr[0: -1]) + 32	# 华氏温度计算表达式
8	print(“转换后的温度是 {:.2f} F”.format(F))	
9	else:	# if 条件语句
10	print(“输入格式错误”)	

变量和赋值

三、变量

■ **变量 (Variable)**: 程序中用来**表示和存储数据的标签**。例如, “TempStr”即一个表示温度值的变量。

■ 变量的命名规则:

- ① 变量名只能包含**字母、数字和下划线**。必须以字母或下划线开头。
 - 例如: `x`, `Y`, `z3`, `TempStr`, `number_of_students`, `Message_1`,
- ② 变量名中不能有**空格或标点符号**。
- ③ 变量名对**字母的大小写**敏感。 “Python” 和 “python” 是两个不同变量。

```
1  # Ch02.1 TempConvert.py
2  TempStr = input("请输入带有符号的温度值:")
```



变量和赋值

编程语言内部定义并保留使用的标记符，是编程语言的基本单词。

- 变量名不能使用Python语言的保留字（关键字）。

Python语言 共有 33个保留字			
False	def	if	raise
None	del	import	return
Ture	elif	in	try
and	elas	is	while
as	except	lambda	with
asssert	finally	nonlocal	yield
break	for	not	
class	from	or	
continue	global	pass	

变量和赋值

四、赋值

- **赋值**：赋值语句用来给**变量**赋予新的数据值。Python中，“=” 表示“赋值”

语句格式：<变量> = <表达式>

<变量1>, ..., <变量N> = <表达式1>, ..., <表达式N>

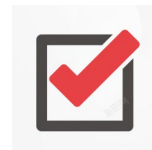
1	<code>a = 10</code>
2	<code>a = "Hello World"</code>
3	<code>a = 3+2</code>
4	<code>x, y = "Hello World", 3+2</code>
5	<code>z = x + y</code>



变量和赋值

■ 输入函数: input() 函数 → 输入数据

- **功能:** input()是从控制台获得输入的函数, 等待用户输入数据后, Python将其赋值给一个变量, 以待后续使用。input()返回为字符串(string)类型数据。
- **格式:** <变量> = input () 或者 <变量> = input (" 提示信息: ")
- **执行:** 运行程序, 程序暂停, 在控制台输入数据, Enter回车程序继续运行



例:

```
TempStr = input("请输入带有符号的温度值: ")
```

```
>>> TempStr=input("请输入带有符号的温度值")
请输入带有符号的温度值
>?
```

```
>>> TempStr=input("请输入带有符号的温度值")
请输入带有符号的温度值>? 200F
```

变量和赋值

■ 输出函数: print() 函数 → 输出结果

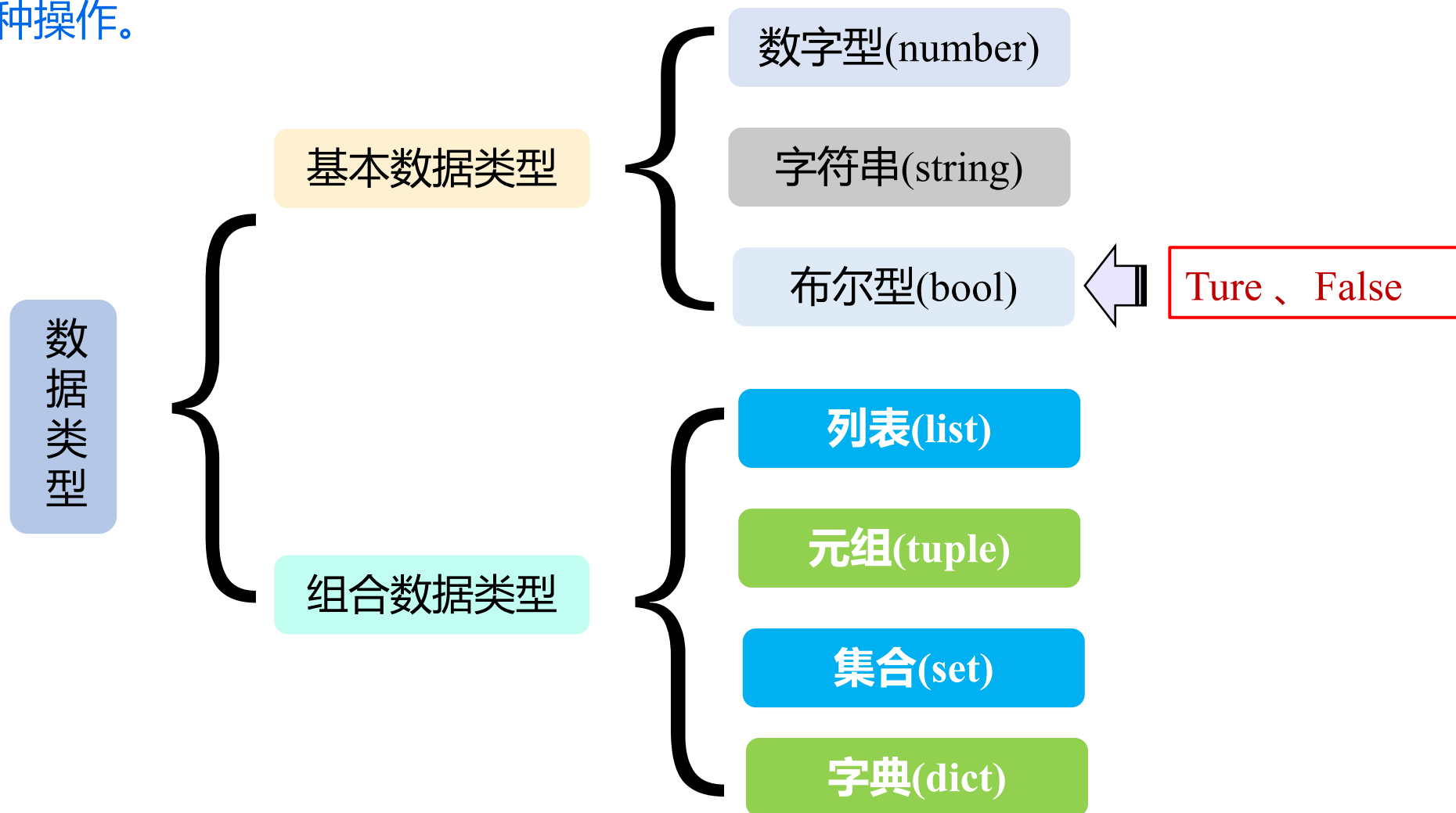
- **功能:** print()是把数据以指定形式输出到控制台。
- **格式:** `print (*objects, sep=' ', end='\n', file = sys.stdout)`

其中:

- ① ***objects:** 表示可以一次输出多个对象。输出多个对象时, 需要用 , (逗号)分隔;
- ② **sep:** 用来间隔多个对象, 默认是空格。
- ③ **end:** 代表输出的结尾符, 默认是换行符**\n**。
- ④ **file:** 代表输出内容存储的文件路径, 默认是系统标准输出路径

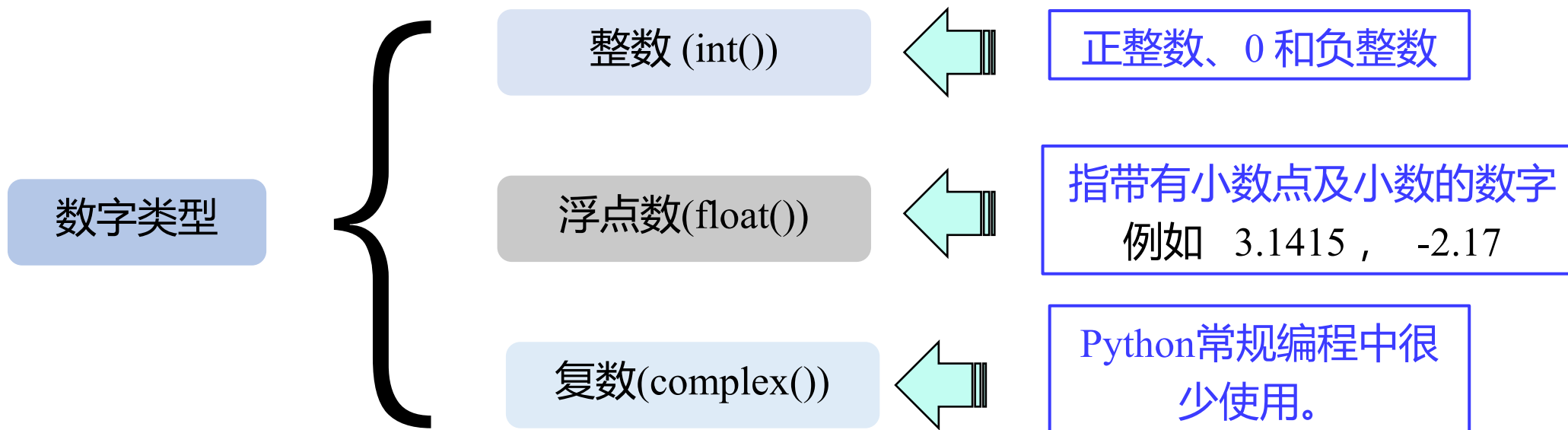
数据类型

- Python语言中，一切皆为对象，而每个**对象(变量)**都拥有某种**数据类型**。不同类型的数据，决定着执行何种操作。



数据类型

(1) 数字类型 (Number) : 表示数字或数值的数据类型



函数	描述	实例
<code>int(x)</code>	将x(字符串或数字)转换为整型	<code>int(3.14)</code> 结果为 3; 舍弃小数部分
<code>float(x)</code>	将x(整数和字符串)转换为浮点型	<code>float(3)</code> 结果为 3.0;

数据类型

(1) 数字类型 (Number)：数值运算操作符，Python提供了9种基本的数值运算操作符。

操作符及使用	描述
$x + y$	加法
$x - y$	减法
$x * y$	乘法
x / y	除法
$x // y$	整除
$x \% y$	余数
$x ** y$	幂运算
$+x$	x 的本身
$-x$	x 的负数

操作符及使用	描述
$a == b$	等于
$a != b$	不等于
$a > b$	大于
$a < b$	小于
$a >= b$	大于等于
$a <= b$	小于等于

数据类型

(1) 数字类型 (Number) : 数值运算函数, Python提供了6个内置函数与数值运算相关。

函数	描述	实例
<code>abs(x)</code>	x的绝对值	<code>abs(-10.1)</code> 结果为 10.1
<code>divmod(x, y)</code>	计算商与余数的函数	<code>divmod(10, 3)</code> 结果为 (3, 1)
<code>pow(x, y)</code>	幂次方运算函数	<code>pow(10, 2)</code> 结果为 $10^2 = 100$
<code>round(x [, d])</code>	x的四舍五入, [, d] 保留几位小数; 如果省略, 即为整数	<code>round(3.1415)</code> 结果为3; <code>round(3.1415, 2)</code> 结果为3.14;
<code>max()</code>	计算最大值	<code>max(1,5,9,4,3)</code> 结果为9
<code>min()</code>	计算最小值	<code>min(1,5,9,4,3)</code> 结果为1

数据类型

■ 实例：天天向上

一年365天，如果勤奋学习则每天进步千分之一，如果荒废贪玩则每天退步千分之一，问勤奋和贪玩一年后差异有多大？

• 算法设计：

(1) 进步成果：

$$\text{result1} = (1 + 0.001)^{365}$$

(2) 退步结果：

$$\text{result2} = (1 - 0.001)^{365}$$

问题： 5%和1%的力量

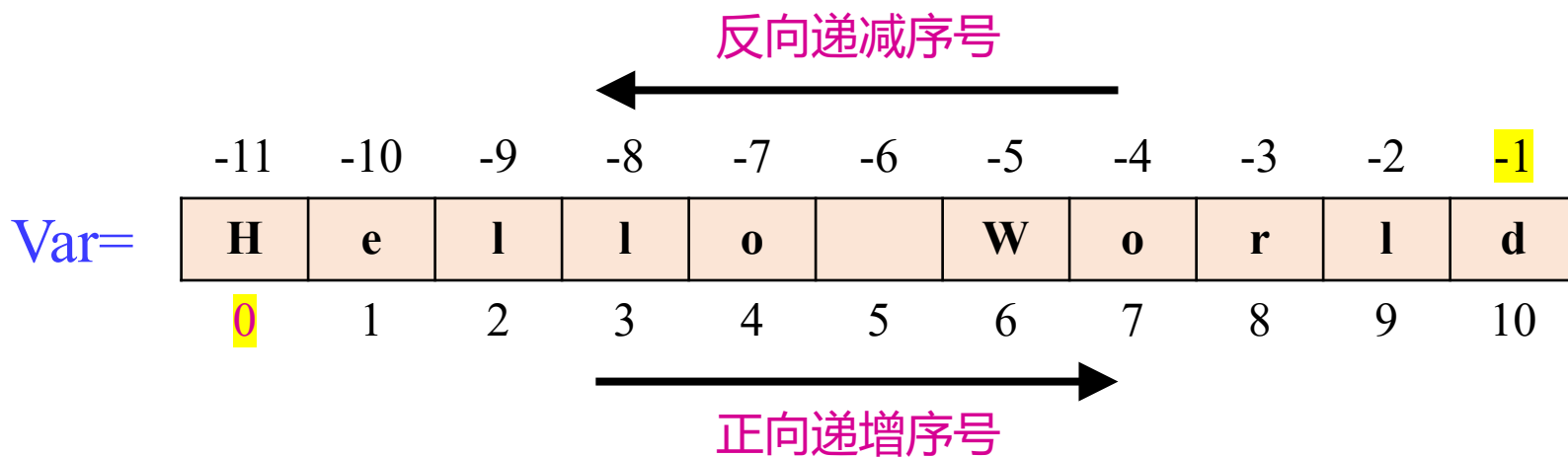


数据类型

(2) **字符串 (string)**：由0或多个字符组成的**有序字符序列**，是 Python 中最常用的数据类型。可以使用引号 (‘ ’ 或 “ ”) 来创建字符串。

```
>>>例： Var= 'Hello World' , y= “学习Python”  
         name = input(“请输入姓名： ”)
```

■ 字符串是字符序列，可按照序号索引





数据类型

(2) 字符串 (string) : Python解释器提供了5个字符串基本操作符。

操作符及使用	描述
$x + y$	连接两个字符串x和y
$x * n$	复制n次字符串x
$x \text{ in } s$	如果x是s的子串, 返回True, 否则返回False
$\text{str}[i]$	索引, 返回第i个字符
$\text{str}[N:M]$	切片, 返回索引第N到第M的子串, 其中不包含M



序号	代码	运行结果
1	<code>>>> "Python语言" + "程序设计"</code>	"Python语言程序设计"
2	<code>>>> "Good!" * 3</code>	"Good! Good! Good!"
3	<code>>>> name="Python语言程序设计"</code> <code>"Python" in name</code>	True

数据类型

(2) 字符串 (string) : 字符串访问 → 方括号[<序号>]来截取字符串

```
>>>实例:  name= 'Python语言程序设计'
```

序号	代码	运行结果
1	>>>print (name[0])	P
2	>>> print (name[0], name[7], name[-1])	P 言计
3	>>>print (name [2:-4])	thon语言
4	>>>print (name [:6])	Python
5	>>>print (name [6:])	语言程序设计
6	>>>print (name [:])	Python语言程序设计

数据类型

(2) 字符串 (string) : Python解释器提供了内置函数与字符串处理相关。

函数及使用	描述
<i>len(x)</i>	返回字符串x的 长度 ，也可返回其他组合数据类型元素个数
<i>str(x)</i>	返回任意类型x所对应的 字符串 形式



序号	代码	运行结果
1	>>> <i>len</i> (“Python语言程序设计”)	12
2	>>> <i>str</i> (3.1415)	“3.1415”

数据类型

(2) 字符串 (string)：字符串**格式化**输出

- str.format() 方法：使用 {} 作为占位符，支持位置、关键字参数和格式控制 (Python 2.6+)
- f-string 方法：在字符串前加 f 或 F，直接嵌入变量和表达式 (Python 3.6+)

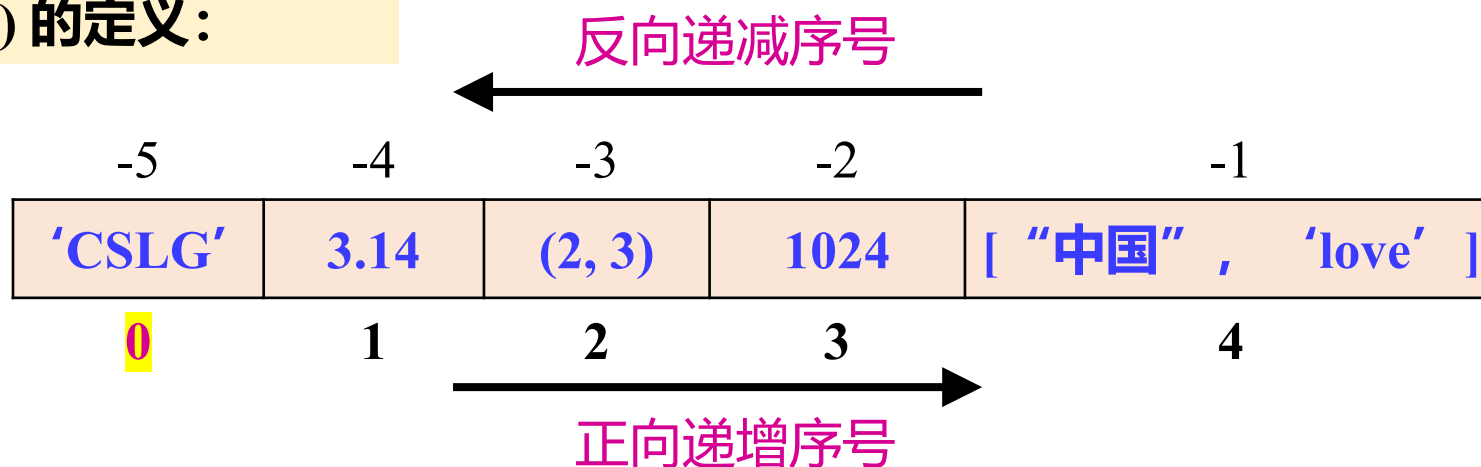
序号	代码
1	>>>name = 'Wu'
2	>>> score = 95
3	>>>print ("姓名: {}, 分数: {}".format(name, score))
4	>>>print (f"姓名: {name}, 分数: {score}")

数据类型

(3) **列表 (list)**：是包含0个或多个对象(元素)的有序序列，没有长度限制，元素类型可不同，能自由增删修改元素。使用**方括号 []** 或 **list()** 创建，元素间用**逗号(,)** 分隔。

```
>>> a = [1, 2, 3, 4, 5]
      item = list(1, 2.3, 3, 10, -2)
      ls = [ "CSLG" , 3.14, (2, 3), 1.024, [ "中国" , 'love' ] ]
```

■ 序号 (索引) 的定义：



数据类型

(3) 列表 (list) : 6个序列类型的操作符

操作符及使用	描述
$x + y$	连接两个序列x和y
$x * n$	复制n次序列x
$x \text{ in } s$	如果x是s的元素, 返回True, 否则返回False
$x \text{ not in } s$	如果x不是s的元素, 返回True, 否则返回False
$s[i]$	索引, 返回序列s中第i个元素, i是序号
$s[i:j]$	切片, 返回序列s中第i 到第j 个元素子序列, i、j是序号

数据类型

(3) 列表 (list) : 6个列表类型特有的操作方法

处理函数	描述
$Is[i] = x$	替换列表Is中第i个元素为x
$Is[i:j] = item$	使用列表item替代Is切片后所对应元素子列表
$del\ Is[i]$	删除列表Is中第i个元素
$del\ Is[i:j]$	删除列表Is中第i 到第j 个元素
$Is += item$	更新列表Is, 将列表item元素增加到列表Is中
$Is * = n$	更新列表Is, 其元素复制n次

数据类型

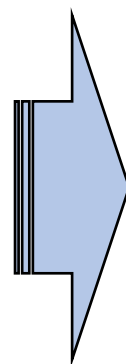
(3) 列表 (list) : 7个列表类型特有的操作函数

处理函数	描述
<code>ls.append(x)</code>	在列表ls最后添加一个元素x
<code>ls.clear()</code>	删除列表ls中所有元素
<code>ls.copy()</code>	生成一个新列表, 赋值ls中所有元素
<code>ls.insert(i, x)</code>	在列表ls中第i位置添加一个元素x
<code>ls.pop(i)</code>	在列表ls中第i位置元素取出, 并删除该元素
<code>ls.remove(x)</code>	在列表ls中出现的第一个x删除
<code>ls.reverse()</code>	将列表ls中元素反转

数据类型

■ 实例：列表操作

- ① 定义空列表item
- ② 向列表item新增5个元素
- ③ 修改列表item第2个元素
- ④ 向列表item第2个位置增加一个元素
- ⑤ 从列表item中第1个位置删除一个元素
- ⑥ 删除列表item第1-3位置元素
- ⑦ 判断列表item中是否包含数字0
- ⑧ 向列表item最后增加一个元素
- ⑨ 列表item的元素个数
- ⑩ 清空列表item



序号	代码
1	<code>item=[]</code>
2	<code>item += [1, 2, 3, 4, 5]</code>
3	<code>item[1]= "学习"</code>
4	<code>item.insert(1, "Python")</code>
5	<code>del item[0]</code>
6	<code>del item[0:3]</code>
7	<code>0 in item</code>
8	<code>item.append(2023)</code>
9	<code>len(item)</code>
10	<code>item.clear()</code>

数据类型

(4) **元组 (tuple)**：一种**只读性**的数据序列，用圆括号`()`表示，其内部元素用逗号隔开。元组中的元素可以是不同类型的数据或结构。

常用的元组操作符有：

拼接	+
重复	*
获取元素	[]
获取子元组	[:]
判断包含	in
判断不包含	not in
迭代（遍历）	for x in (1,2,3): print(x)

常用的元组函数有：

求长度	len((1,2,3))
最大值	max((1,2,3))
最小值	min((1,2,3))
求索引 (匹配第一个)	(1,2,3).index(2)

数据类型

(5) **集合 (set)**：集合是一个无序的不重复元素序列。用**大括号{ }**表示，其内部元素用逗号隔开。构建空集合，必须使用`set()`

- 集合元素之间无序，每个元素唯一、不重复，不存在相同的元素；
- 集合的数据类型只能是固定数据类型，例如整数、浮点数、字符串、元组等
- 集合元素不可更改，可变数据类型(列表、集合等)不能作为元素出现

#创建

```
a = set()
```

```
b = set((1, 2, 3))
```

```
c = {4, 5, 6, 'abc' }
```

#创建空集合时用`set()`，不用`{ }`

数据类型

(6) **字典 (dict)**：是一种键(索引)-值(数据)的对应关系，每个元素是一个键值对，即元素(key, value), 元素之间是无序的。Python中，映射类型主要以字典(dict)体现。使用大括号 { } 或 dict() 创建，键值对用冒号：表示。

◆ 语法结构：

<字典变量> = { <键1> : <值1>, <键2> : <值2>, ... <键n> : <值n> }

```
>>>例： dict1= { '中国': '北京', '美国': '华盛顿', }
```

```
dict2={'姓名':'方舟', '年龄':20, '身高':180, '体重':75}
```

数据类型

(6) 字典 (dict) : 创建

① 创建空字典: `dict1={}`

② 直接赋值创建字典:

例: `dict2={'北京':'010','上海':'021','广州':'020'}`

③ 使用`dict()`函数、`zip()`函数把两个列表创建为字典:

例: `city=['北京','上海','天津','重庆']`

`quhao=['010','021','022','023']`

`dict3 = dict(zip(city,quhao))`

数据类型

(6) 字典 (dict) : 字典的访问

访问字典，就是根据键获得与键关联的值。由于字典是无序序列，推荐使用get()函数获得指定键的值，如果字典中不存在，可返回None或者指定的内容。

<字典变量> [<键>] 或者 <字典变量>.get(<键>)

示例：result={'林华':92,'张晴':90,'李小明':85,'刘丽':89}

result['林华'] → 92

result['王明'] → 字典无此键，程序报错

result.get('刘丽') → 89

result.get('张晓薇') → 字典无此键，输出None

result.get('张晓薇', 93) → 字典无此键，输出指定的默认值93

03

控制结构



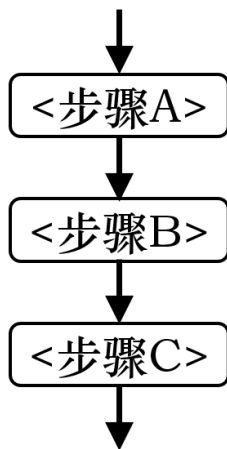
控制结构

- **控制结构**：程序设计主要强调某个功能、需求的算法实现，而算法的实现过程是由一系列数据操作构成的，这些操作之间的执行次序就是程序的控制结构。

- ① Python语言提供了3种基本结构：**顺序结构、分支结构、循环结构**
- ② 任何简单或复杂的算法都可由这三种程序基本结构组合嵌套而成

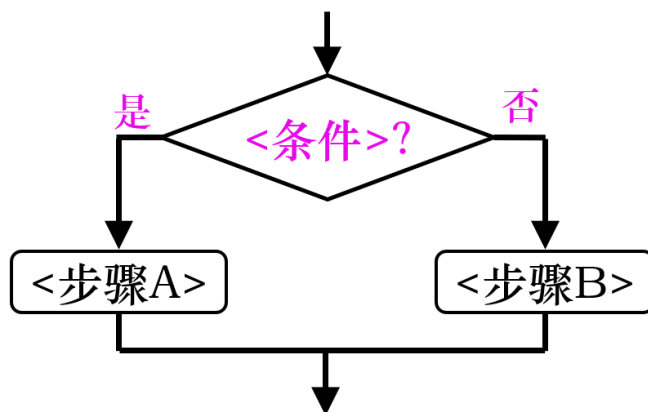
(1) 顺序结构

按照顺序依次执行程序



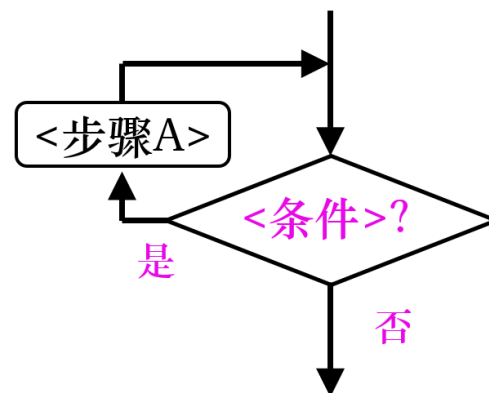
(2) 分支结构

由判断条件决定程序运行方向



(3) 循环语句

由判断条件决定程序是否再次执行



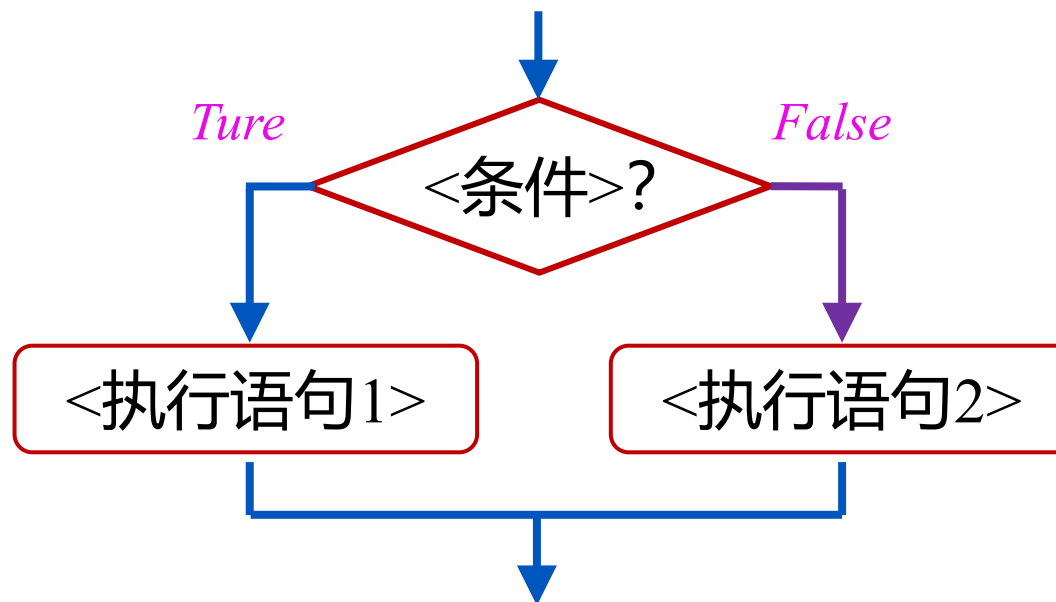
控制结构

一、双分支结构：if-else语句

Python 中 if – else 语句描述双分支结构，语法结构：

◆ 语法结构：

```
if <判断条件> :  
    <执行语句1>  
else :  
    <执行语句2>
```



控制结构

■ 实例：空气质量监测

PM2.5是衡量空气质量重要指标，如果PM2.5低于35，则空气质量为优秀；如果PM2.5高于35而低于75，则空气质量为良好；如果PM2.5是75以上，则空气污染严重。编写Python监测测序。

```
1 PM = eval (input ("请输入PM2.5数值: "))
2 if PM >= 75 :
3     print("空气污染, 请小心! ! ")
4 else :
5     print("空气没有污染, 可开展户外运动! ")
```

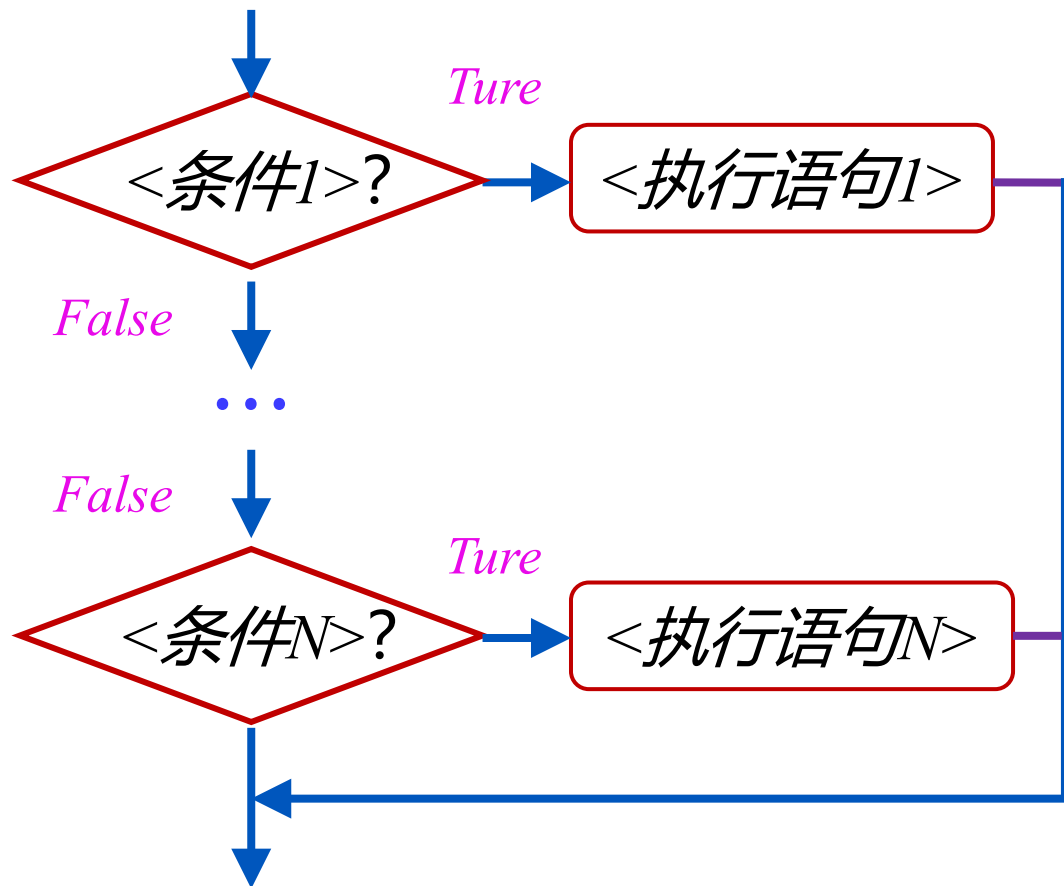
控制结构

二、多分支结构：if-elif-else 语句

Python中if-elif-else语句描述多分支结构，语法结构：

◆ 语法结构：

```
if <判断条件1> :  
    <执行语句1>  
elif <判断条件2> :  
    <执行语句2>  
...  
else :  
    <执行语句N>
```



控制结构

■ 实例：空气质量监测

PM2.5是衡量空气质量重要指标，如果PM2.5低于35，则空气质量为优秀；如果PM2.5高于35而低于75，则空气质量为良好；如果PM2.5是75以上，则空气污染严重。编写Python监测测序。

```
1 PM = eval (input ("请输入PM2.5数值: "))
2 if 0 <= PM < 35 :
3     print("空气优质, 快去户外运动! ")
4 elif 35 <= PM < 75:
5     print("空气良好, 适度户外运动! ")
6 else :
7     print("空气污染, 请小心! ")
```

控制结构

■ 实例：身体质量指数 BMI

身体质量指数(Body Mass Index, BMI): 衡量身体肥胖程度和是否健康的重要程度。BMI定义如下:

$$[BMI] = [\text{体重}/\text{kg}] \div [\text{身高}^2/\text{m}^2]$$

● BMI指标分类:

分类	国际BMI值 (kg/m^2)	中国BMI值 (kg/m^2)
偏瘦	<18.5	<18.5
正常	18.5~25	18.5~24
偏胖	25~30	24~28
肥胖	≥ 30	≥ 28

控制结构

```
1  """ BMI计算程序
    作者: Wu C.B.
    版本: Python 3.11.2 """
2  height= float(input("请输入身高(米): "))
3  weight= eval(input("请输入体重(公斤): "))
4  bmi = weight / pow(height, 2)          # BMI指数计算公式
5  if bmi < 18.5:
6      index = "偏瘦"
7  elif 18.5<=bmi<25:
8      index = "正常"
9  elif 25 <= bmi < 30:
10     index = "偏胖"
11 else:
12     index = "肥胖"
13 print("BMI 数值为: {:.2f}, 健康状态: {}".format(bmi,index))
```

● 运行结果:

```
请输入身高(米): 1.75
请输入体重(公斤): 80
BMI 数值为: 26.12, 健康状态: 肥胖
```

控制结构

■ 练习题 – 分支结构

1. 成绩评价系统，需要根据输入的具体成绩（百分制），生成对应的评价指标。

输出格式：<成绩分数***，等级是***，综合评价为***>

● 成绩评价指标：

成绩	等级
大于90	等级=A, 优秀
80~90	等级=B, 良好
70~80	等级=C, 中等
60~70	等级=D, 及格
小于60	等级=E, 不及格

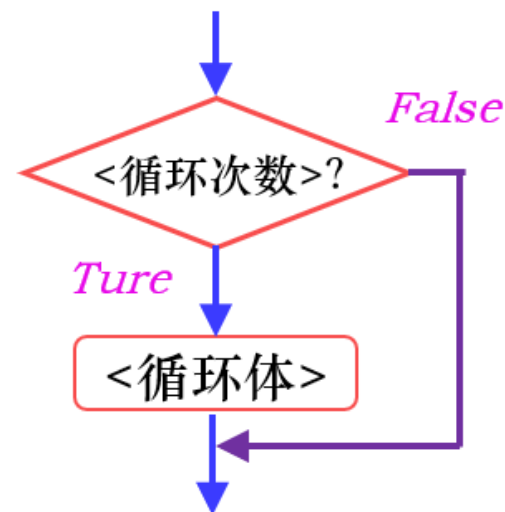
控制结构

三、循环结构

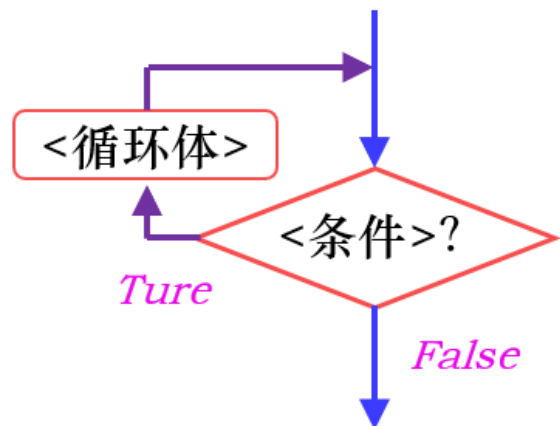
循环结构：指程序中需要反复执行某个子程序的控制结构。根据循环执行次数的确定性，Python提供了两种循环结构：遍历循环、无限循环。

(1) 遍历循环：for语句

⇒ 循环次数确定，由遍历结构中的元素个数来确定。



(2) 无限循环：while语句



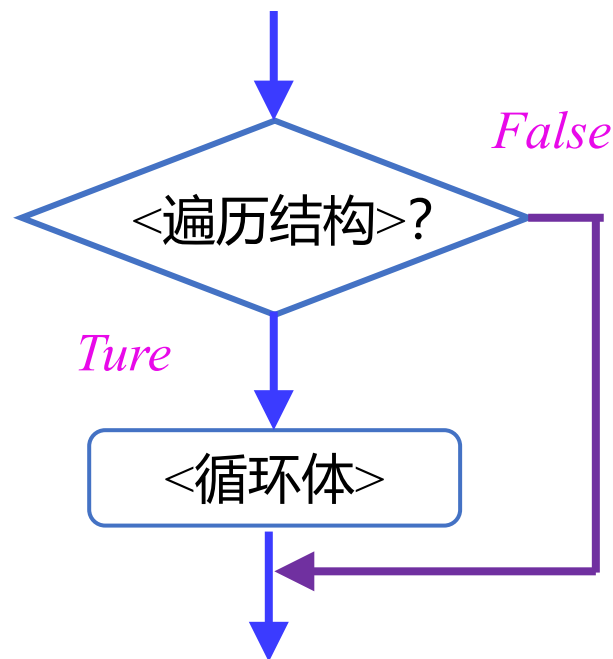
⇒ 循环次数不确定，通过条件判断是否继续执行循环体。

控制结构

(1) 遍历循环：for 语句

■ 语法结构：

```
for <循环变量> in <遍历结构> :  
    <执行子程序>
```



- a) 由保留字for 和 in 组成，由缩进体现层次关系；
- b) 每次循环，从遍历结构中逐一提取元素，赋值循环变量，逐次执行循环体；
- c) 当完整提取遍历结构中所有元素后，遍历循环结束；

控制结构

(a) 基数循环 (N次)

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

<执行子程序>

- <循环变量>：可选择 i 、 j ...
- <循环结构>：由 `range()` 函数产生数字序列

```
>>> range(start, stop [,step])
```

函数创建一个整数列表

序号	代码	运行结果
1	<pre>>>> for i in range(5): print (i)</pre>	0 1 2 3 4
2	<pre>>>> for j in range(1, 6, 2): print (j)</pre>	1 3 5

控制结构

(b) 字符串遍历循环

```
for c in str:
```

<执行子程序>

- <循环变量>：可选择 a 、 c ...
- <循环结构>：str 是字符串，遍历字符串每个字符，从字符串中提取每一个字符，产生循环。字符串看作一个循环结构。

序号	代码
1	>>> for c in “学习Python”: print (c, end=“ - ”)

控制结构

(c) 列表遍历循环

```
for item in Is:
```

<执行子程序>

- <循环变量>：可选择 *item* ... 任意合法变量
- <循环结构>：Is 是列表，遍历列表中每个元素，从列表中提取每一个元素，产生循环。列表可作一个循环结构。

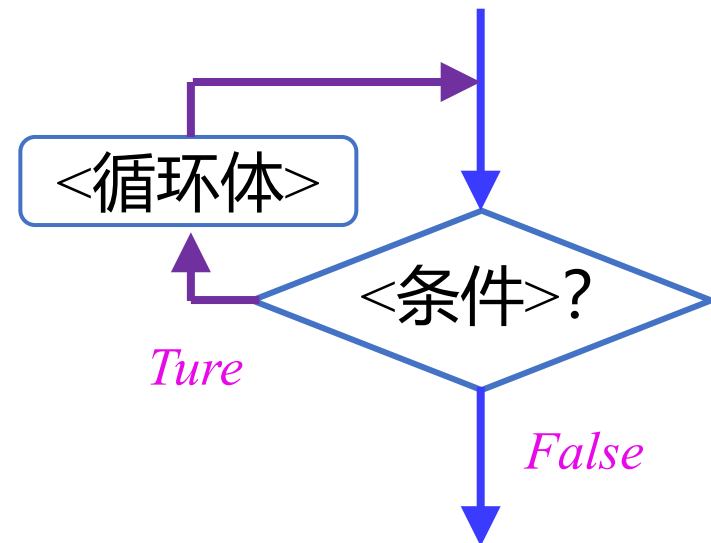
序号	代码	运行结果
1	>>> for item in ["Py", "thon", 123]: print (item, end=" + ")	Py+thon+123+

控制结构

(2) 无限循环：while 语句

■ 语法结构：

```
while <条件>:  
    <执行子程序>
```



- a) 由保留字while 组成，由缩进体现层次关系；
- b) 每次循环，先判断条件，如果条件满足（True），循环体重复执行子程序；
- c) 当条件判断为False，条件不满足时，无限循环结束；

控制结构

■ 案例 1:

序号	代码
1	>>> a=3
2	>>> while a>0:
3	a =a - 1
4	print (a)

运行结果:

```
>>> 2
    1
    0
```

■ 案例 2:

序号	代码
1	>>> a=3
2	>>> while a>0:
3	a =a + 1
4	print (a)

运行结果:

```
>>> 4
    5
    ⋮
```

无限循环下去

控制结构

(3) break 和 continue

- break: 跳出并结束当前整个循环，执行循环之后的语句。
- continue: 结束当次循环，继续执行后续次数循环。

■ 案例 1:

序号	代码
1	>>> for c in "PYTHON":
2	if c == "T":
3	continue
4	print(c, end="*")

运行结果:

```
>>> P*Y*H*O*N*
```

■ 案例 2:

序号	代码
1	>>> for c in "PYTHON":
2	if c == "T":
3	break
4	print(c, end="*")

运行结果:

```
>>> P*Y*
```

控制结构

■ 实例：打印九九乘法表

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

```
1 for i in range(1,10):
2     for j in range (1,i+1):
3         num = i*j
4         print(f "{i} * {j} = {num:2} ", end=" ")
5     print()
```

● 运行结果:

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

04

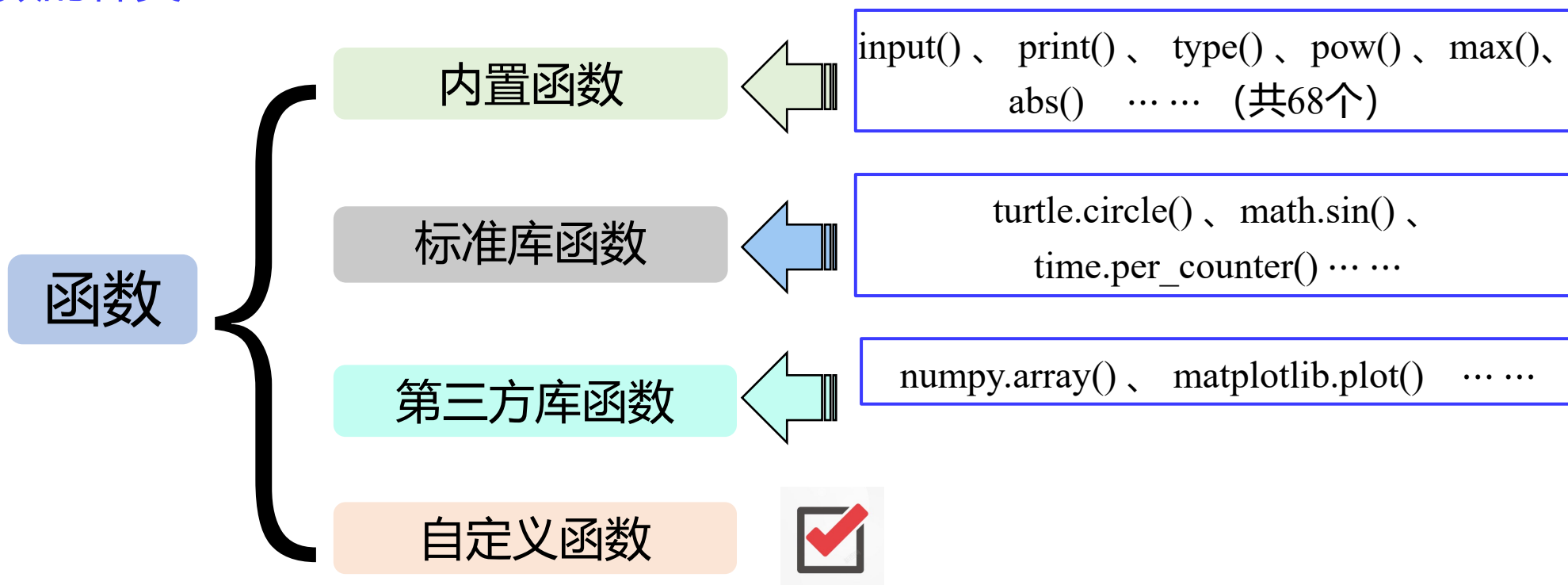
函数与模块



函数与模块

- **函数**：Python函数是指已组织好的、可重复使用的、用来实现单一或相关功能的代码段。
每一个Python函数都可以实现某种特定功能，对函数使用无需了解函数内部算法的原理，通过“<函数名(参数)>”的格式直接调用即可。降低编程难度，调高程序可重用性。

- **函数的种类：**



函数与模块

二、函数的定义

■ 语法结构：

```
def <函数名> (参数1, 参数2 ...):
```

```
<函数体>
```

```
return <返回值>
```

：(冒号)一定不能丢掉，
def 的必须元素

- a) 由保留字 **def** 自定义函数，由**缩进**体现层次关系；
- b) <函数名>可以是任意有效的Python标记符，例如 result、Net_My、Sum_1；**函数名**不可与内置函数重名
- c) **函数的参数**可以由**零个、一个或多个**，根据函数体中运算需要，选择参数；
- d) **return** 返回函数值，**结束函数**，返回一个值给调用方；如果省略，则返回 None

函数与模块

■ 案例：累加函数

$$y = 1 + 2 + \cdots + N = \sum_{i=1}^N i$$

>>> 自定义累加函数：MyFunc()

程序	代码
1	<code>def MyFunc (x):</code>
2	<code> s=0</code>
3	<code> for i in range (1, x+1):</code>
4	<code> s+=i</code>
5	<code> return s</code>

- a) 函数定义时，指定的参数是一种占位符
- b) 函数定义后，若不经过调用，不会被执行
- c) 函数定义时，参数是输入、函数体是处理、返回值是输出（IPO）

函数与模块

■ 案例：打印 “Hello World” 函数

>>> 自定义MyFunc函数: MyPrint()

函数名

程序	代码
1	def MyPrint():
2	print (“Hello World”)

- a) 自定义函数的参数个数是0;
- b) 自定义函数没有return返回值，仅执行函数体

函数与模块

三、函数的调用

调用：是运行函数代码的方式。直接通过函数名调用函数，执行函数的功能。

■ 函数调用方法： **<函数名>** (**<参数列表>**)

程序	代码
1	<code>def MyFunc (x):</code>
2	<code> s=0</code>
3	<code> for i in range (1, x+1):</code>
4	<code> s=s+i</code>
5	<code> return s</code>

```
>>> MyFunc (100)  #100的累加运算
```

```
>>> 运行结果: 5050
```

函数与模块

三、函数的调用

程序	代码
1	<code>def MyFunc (x):</code>
2	<code> s=0</code>
3	<code> for i in range (1, x+1):</code>
4	<code> s=s+i</code>
5	<code> return s</code>

形参

实参

`>>> MyFunc (100)`

■ <形参> vs <实参>

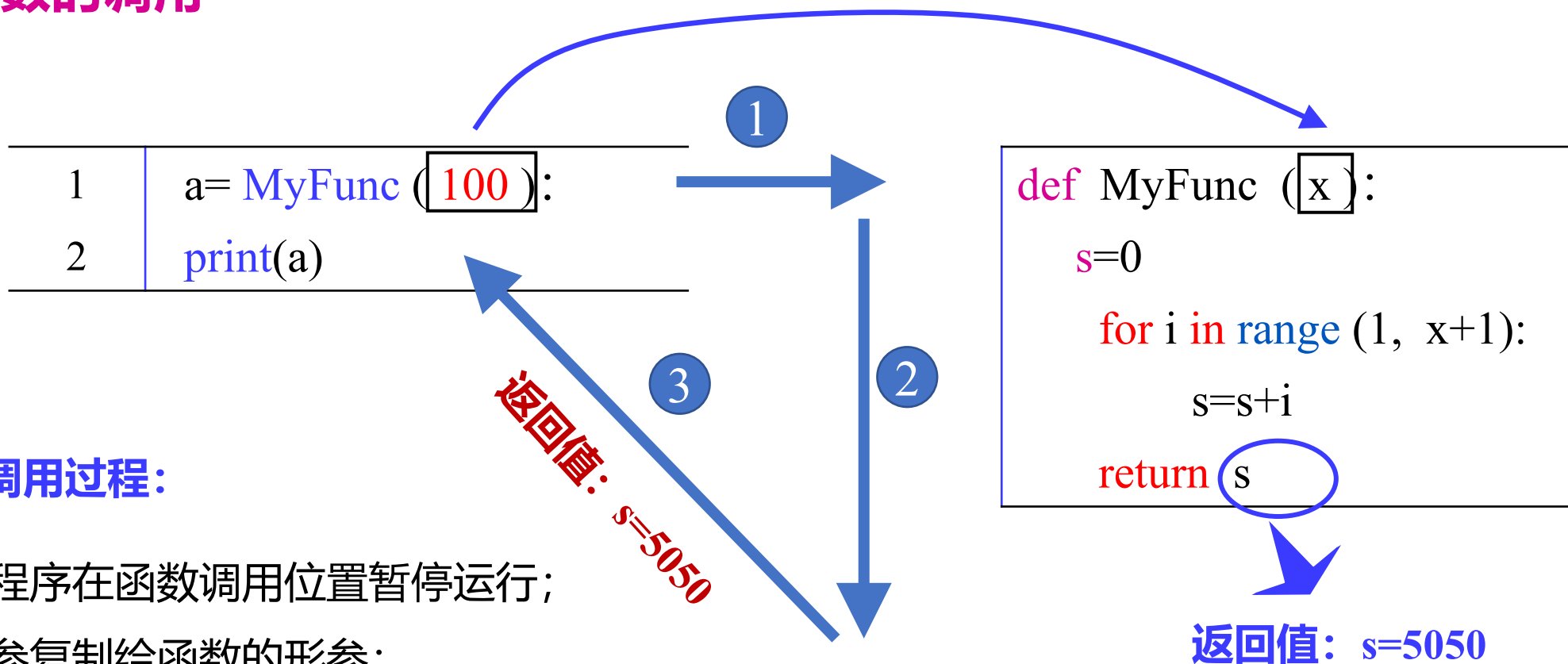
- (1) 形参: 是指形式上的参数, 在未赋值时, 没有实际意义。形参的作用是以变量的形式来传递当前未知的值 → 占位符
- (2) 实参: 是指有具体值的参数

定义函数时x是形参, 调用时x是实参。

函数与模块

三、函数的调用

<实参> 传递 <形参>



■ 函数调用过程:

- ① 执行程序在函数调用位置暂停运行;
- ② 将实参复制给函数的形参;
- ③ 执行函数体程序;
- ④ 函数调用结束给出返回值, 程序回到调用前的暂停位置继续运行;

函数与模块

■ 练习题

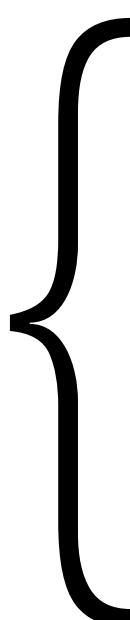
编写函数: 求 $1 \times 2 \times 3 \times \cdots n$ 的阶乘函数, 分别计算 $n=50$ 和 $n=100$ 结果。

$$y = 1 \times 2 \times \cdots \times n = n !$$

调用函数, 输出结果: <输入值: ***, 阶乘结果: ***>

函数与模块

- 模块 (module) :具有相关功能模块的集合。功能丰富的Python模块/库是其特色之一，即具有强大的标准库、第三方库以及自定义模块。

- 
- ① 标准库 (standard library): 标准库随着Python解释器一起安装，是Python自带模块，无需单独安装。
 - ② 第三方库: 第三方机构发布的具有特定功能的模块，Python安装后需单独下载安装的库。全球工程师和机构共享了超过20万个第三方库，极大丰富Python生态。
 - ③ 自定义模块: 用户根据自己需求自行编写的模块。

函数与模块

■ Python 常用库:

数据科学

- Numpy
- Pandas
- SciPy
- Matplotlib
- Sci-Kit Learn

游戏开发

- Pygame
- Pyglet
- Panda3D
- Pykyra
- Ursina

自动化&爬虫

- PyAutoGUI
- Pywinauto
- Selenium
- Requests
- BeautifulSoup

桌面开发

- PyQt5
- Tkinter
- Kivy
- Dear PyGui
- PySide2

Web开发

- Django
- CherryPy
- Flask
- Bottle
- Web2Py

网络自动化

- Netmiko
- NAPALM
- Genie
- NCCClient
- Paramiko

NumPy

SciPy

matplotlib

scikits
learn

PyTorch

TensorFlow

OpenCV

Pandas

StudyAi.com
django
Django Web Framework

PyAutoGUI

函数与模块

■ **模块调用**：先导入程序中，才能正常调用。Python库一般有3种导入方式：

(1) 方式一：

- `import <库名>` `>>> import matplotlib`
- `import <库名> as <缩写>` `>>> import matplotlib as plt`

(2) 方式二：

- `from <库名> import <函数名>` `>>> from math import sin`
- `from <库名> import <函数名> as <缩写>` `>>> from math import sin as f`

(3) 方式三：

- `from <库名> import *` # 其中，*是通配符，表示所有函数

`>>> from math import *`

函数与模块

- math库：是Python中内置的数学计算的标准库，共由4个数学常数和44个函数组成。仅支持整数和浮点数运算，不支持复数类型。

- import math
- from math import <函数名>

函数	描述
math.pi	圆周率，值为3.1415926
math.e	自然对数，值为2.7182818
math.fabs(x)	x的绝对值
math.ceil(x)	不小于x的最小整数
math.floor(x)	不大于x的最大整数
math.factorial(x)	x的阶乘

函数	描述
math.pow(x, y)	x^y
math.exp(x)	e^x
math.sqrt(x)	\sqrt{x}
math.log(x [,base])	$\log_{base} x$
math.sin(x)	sin(x)
math.cos(x)	cos(x)
math.tan(x)	tan(x)

函数与模块

■ 练习题

Python编程：定义分段函数

$$f(x) = \begin{cases} x^2 + \sqrt{x} + 1 & 0 \leq x < 5 \\ e^x + \cos(x) & 5 \leq x < 10 \\ \sin(\pi x) & 10 \leq x < 20 \end{cases}$$

调用函数，输出结果： <输入值： ***, 函数计算结果: ***>

函数与模块

- random 库：Python中随机运算的标准库。主要功能: 采用梅森旋转算法产生各种分布的伪随机数系列。共有9个常用随机数生成函数。

- import random
- from random import <函数名>

函数	描述
random.seed()	初始化随机数种子，默认为当前系统时间。
random.random()	生成一个[0 , 1)之间的随机小数。 >>> random.random() 运行结果: "0.292208911"
random.uniform(a,b)	生成一个[a , b]之间的随机小数。
random.randint(a,b)	生成一个[a , b]之间的随机整数。

函数与模块

■ 练习题

编写程序, 实现如下功能: 猜数游戏

电脑随机生成区间 $[1, 100]$ 的整数, 让用户去猜, 用户每猜一次程序自动判断:

- 如果用户猜的数字小于电脑随机生成的数字则提示 “你猜小了! ” ;
- 如果大于, 则提示 “你猜大了! ” ;
- 如果等于, 则提示 “恭喜你, 猜对了! ” ;

同时统计猜对数字共花费的次数。



函数与模块

■ 练习题

编写程序, 实现如下功能: 摇骰子

编写一个函数, 实现摇骰子的功能, 记录每次骰子点数, 最后计算N次摇骰子的点数和。
输出N=6时, 每次骰子点数和N次点数总和

