PYTHON程序设计

张树兵改编

20230303 修改

春日读书示学子

张树兵改

20230304

今人学问有遗力，
少壮工夫未始成。
纸上得来终觉浅，
绝知此事要躬行。
千锤百炼虽辛苦，
洗尽铅华始见金。
一箫一剑平生意，
负尽狂名十五年。

字符串 专题

•字符串定义

- Python 字符串是一种文本数据，它以**一对双引号或单引号里包裹起来**。
- 例如，
- **"你好"** 是一个字符串，它由双引号里包裹起来的两个字构成。
- Python 字符串也是由很多字符组成的，比如：**"Hello World!"** 中有 11 个字符：H、e、l、l、o、空格、W、o、r、l、d 和 !。

字符串连接 +

- 下面这段代码使用字符串来把两个变量（name和age）组合起来：

```
name = "张三"
age = 12
text = "我叫" + name + "，今年" + str(age) + "岁。"

print(text)
```

- 运行结果：我叫张三，今年12岁。

字符串复制 *

字符串复制是指将一个字符串复制多次，用*实现。比如说，我们有一个字符串"abc"，如果我们想要将它复制3次，就变成"abcabcabc"。这个过程就是字符串复制。

string = "hi"

new_string = string * 5

print(new_string)

```
1 print("CAAC中国民航局:CAUC中国民航大学")
2 print("CAAC中国民航局:CAUC中国民航大学"*8)
3 print("CAAC中国民航局:CAUC中国民航大学\n"*8)
4 print("CAAC中国民航局:CAUC中国民航大学\n"+8)
```

Name

字符串比较

- Python 字符串比较规则就是比较两个字符串中的每一个字符，从左至右依次
- 比较，如果某一位字符较大，那么字符串就较大；如果某一位字符相等，
- 那么继续比较下一位字符；如果比较到最后还是相等，那么两个字符串就是相等的。

- 举个例子：
- 比较两个字符串：**"ABCDE"** 和 **"ABCDF"**
- 首先比较第一个字符："A" 和 "A"，相等，继续比较第二个字符："B" 和 "B"，相等，继续比较第三个字符："C" 和 "C"，相等，继续比较第四个字符："D" 和 "D"，相等，继续比较第五个字符："E" 和 "F",因为E先F后，所以 **"ABCDE" 小于"ABCDF"** 得到结果 **True**

字符串比较 ord()

- Python ord函数可以用来比较字符串。它返回一个字符串中每个字符的Unicode编码值。例如：
- 字符串"apple"
- ord("a") 返回 97
- ord("p") 返回 112
- ord("l") 返回 108
- ord("e") 返回 101
- 所以我们可以比较两个字符串，比较它们中每个字符的Unicode编码值。
- 例如：
- 字符串"apple" 和 字符串"cat"
- 因为 ord("a") < ord("c") 所以 "apple" 比 "cat" 小。

二进制	十进制	十六进制	图形
0010 0000	32	20	（空格）（SP）
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8

二进制	十进制	十六进制	图形
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X

二进制	十进制	十六进制	图形
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o
0111 0000	112	70	p
0111 0001	113	71	q
0111 0010	114	72	r
0111 0011	115	73	s
0111 0100	116	74	t
0111 0101	117	75	u
0111 0110	118	76	v
0111 0111	119	77	w
0111 1000	120	78	x

字符串长度 len()

- Python中的字符串长度是指字符串中字符的数量。
- 例如，字符串“hello”有5个字符，因此它的长度为5。
- 字符串长度就是字符串中字符的数量，可以使用len()函数来计算

```
string = "Python是一个很有用的编程语言"
length = len(string)
print(length)
```

输出结果将是17，因为这个字符串中有17个字符。

字符串 序号概念

- Python字符串序号是一种用数字编号,来表示字符串中每个字符位置的方法。可以将字符串看做是由一个个字符组成的，每个字符都有一个唯一的位置编号。
- 例如，字符串"apple"中，第一个字符"a"的位置编号是0，第二个字符"p"的位置编号是1，以此类推。

那么它的每个字符的位置编号分别是：

a p p l e

0 1 2 3 4

- 作用：可以帮助我们快速**定位**字符串中的每个字符，并对其进行操作。

字符串 序号 正序

Python字符串序号是指字符串每个字符的位置编号，正序从左到右依次为0、1、2、3……


```
s = "apple"
third_char = s[2]  # 获取第三个字符，位置编号为2
print(third_char)  # 输出结果为： p
```

字符串 序号 反序

- 反序是指从后往前访问字符串中的字符。反序序号是从-1开始的.
- 例如，字符串“hello”可以用反序访问每个字符：

h e l l o

-5 -4 -3 -2 -1

- 这里的反序序号是从-1开始的，所以字符串中最后一个字符的反序序号是-1，倒数第二个字符的反序序号是-2，以此类推。

字符串 序号 反序

- 例如，使用 word[-1] 可以访问字符串中的最后一个字符。
- 如果我们有一个字符串 "Python is fun!"，
- 我们可以使用下面的代码来访问其中的某些字符：

```
word = "Python is fun!"
print(word[0])   # 输出 P
print(word[11])  # 输出 f
print(word[-1])  # 输出 !
```

字符串的序号

正向递增序号 和 反向递减序号

字符串　单个字符访问

Python中的字符串索引是指通过下标来访问字符串中的某个字符。比如，我们可以用字符串名和索引号来获取字符串中的一个字符，

语法：　**字符串[索引]**　　**s=”apple”　s[0]**

使用[ ]获取字符串中一个或多个字符

- **索引：返回字符串中单个字符**　　**<字符串>[M]**

"请输入带有符号的温度值:"[0]　**或者**　TempStr[-1]

字符串 单个字符访问

如果我们有一个字符串“Python”，我们可以使用以下代码获取它的第一个和最后一个字符：

```
my_string = "Python"
first_char = my_string[0]  # 获取第一个字符
last_char = my_string[-1]  # 获取最后一个字符
print(first_char)  # 输出 P
print(last_char)  # 输出 n
```

字符串 访问（遍历）

Python字符串遍历指的是逐个访问字符串中的每个字符。

```
s = "你好，欢迎参观CAUC空管学院"
for c in s:
    print(c) #该行有缩进
print(len(s))
```

- 这段代码中，变量`s`存储了字符串 “你好，欢迎参观CAUC空
- 管学院” ，然后使用循环语句`for`遍历字符串中的每个字符，
- 并使用`print`函数将其打印出来。执行这段代码的结果是依次
- 打印出 你好，欢迎参观CAUC空管学院 15个字符。

字符串 切片

- Python字符串切片是一种操作，可以将一个字符串按照一定的规则切分成几个部分。比如，我们可以通过指定起始位置和结束位置，来截取字符串中的一部分。
- 可以类比于将一根香蕉分成几段，每一段就是一个子串。我们可以指定从哪个位置开始切，切到哪个位置结束，就能得到我们想要的子串。
  - **规则： M位置开始，N位置之前结束，不含N位置字符**

- **切片：返回字符串中一段字符子串　<字符串>[M: N]**

```
"请输入带有符号的温度值: "[1:3] 或者  TempStr[0:-1]
```

字符串 切片

```
string = "Hello,World!"
substring = string[6:11]
print(substring)
```

- 这段代码的意思是从字符串中的第6个字符（W）开始，截取到第11个字符（d）之前，也就是截取了"World"这个子字符串。输出结果为"World"。

H e l l o , W o r l d !

0 1 2 3 4 5 6 7 8 9 10 11

字符串 切片

```
str = "Hello,world!"
print(str[0:5])
```

这段代码的意思是，选取字符串str中从第0个字符（即第一个字符）”H”开始，到第5个字符（即第六个字符）结束的部分，也就是"Hello"。我们可以将这个部分打印出来，结果为"Hello"。

H e l l o , W o r l d !

0 1 2 3 4 5 6 7 8 9 10 11

字符串 高级切片

使用[M: N: K]根据步长对字符串切片

- <字符串>[M: N]，M缺失表示至开头，N缺失表示至结尾

  "〇一二三四五六七八九十"[:3] 结果是 "〇一二"

- <字符串>[M: N: K]，根据步长K对字符串切片

  "〇一二三四五六七八九十"[1:8:2] 结果是 "一三五七"

  "〇一二三四五六七八九十"[::-1] 结果是 "十九八七六五四三二一〇"

M表示起始位置，N表示终止位置，K表示步长。

字符串 高级切片

```
str = "Hello,world!"
print(str[6:])
```

这段代码的意思是，选取字符串str中从序号位置6的字符”W”（即第七个字符）开始，一直取到结束的部分，也就是"World!"。我们可以将这个部分打印出来，结果为"World!"。

H e l l o , W o r l d !

0 1 2 3 4 5 6 7 8 9 10 11

字符串 高级切片

```
s = "hello,world!"
result = s[1:9:2]
print(result)
```

这里的步长为2，表示每隔一个字符进行切片。所以，我们从索引为1的字符“e”开始，每隔一个字符取一个，直到索引为9的字符“l”结束，得到了我们需要的字符串“el,o”。

H	e	l	l	o	,	W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11

字符串 高级切片 反序

```
s = "hello,world!"
result = s[::-1]
print(result)
```

这里的步长为1，表示每隔0个字符进行切片,**负号表示方向是从右往左取，**所以，我们从最后一个的字符“！”开始，依次取，直到第一个字符“h”结束，得到了我们需要的字符串“!dlrow,olleh”,也就是原字符串 "hello，world!" 倒序排列后的结果。换句话说，这个方法可以把一个字符串中的字符顺序颠倒过来。

H e l l o , W o r l d !

0 1 2 3 4 5 6 7 8 9 10 11

字符串 操作符

字符串操作符

由0个或多个字符组成的有序字符序列

操作符及使用	描述
x + y	连接两个字符串x和y
n * x 或 x * n	复制n次字符串x
x in s	如果x是s的子串，返回True，否则返回False

字符串 操作符 in

- Python中的“in”指的是一个特殊的操作符，用于检查一个字符串是否包含
- 另一个字符串。例如，我们可以使用以下代码来检查一个字符串是否包含
- 另一个字符串：

```
if "hello" in "hello,world!":
    print("包含")
else:
    print("不包含")
```

这段代码的意思是，如果字符串“hello”包含在字符串“hello,world!”中，则输出“包含”，否则输出“不包含”。结果是“包含”

字符串处理函数

字符串处理函数

一些以函数形式提供的字符串处理功能

函数及使用	描述
len(x)	长度，返回字符串x的长度 len("一二三456") 结果为 6
str(x)	任意类型x所对应的字符串形式 str(1.23)结果为"1.23" str([1,2])结果为"[1,2]"
hex(x) 或 oct(x)	整数x的十六进制或八进制小写形式字符串 hex(425)结果为"0x1a9" oct(425)结果为"0o651"

字符串函数 str()

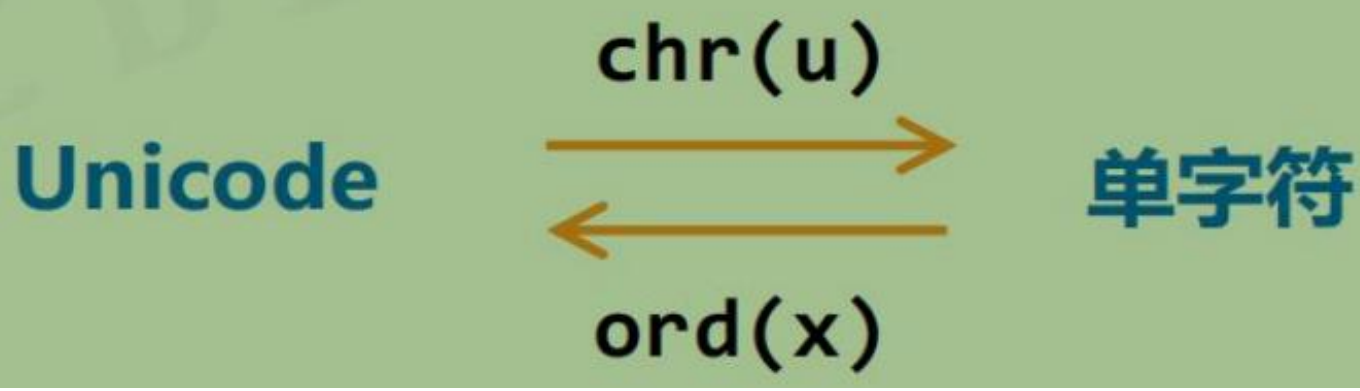
- Python中的字符串str函数是一种用于处理文本字符串的函数。它可以将其他类型的数据转换为字符串类型，并且可以对字符串进行一些操作，如拼接等，经常在两个或多个不同数据类型的进行连接时被使用。
- 拼接

```
x = 10
text = "我的数字是： " + str(x)
print(text)
```

字符串处理函数

一些以函数形式提供的字符串处理功能

函数及使用	描述
chr(u)	x为Unicode编码，返回其对应的字符
ord(x)	x为字符，返回其对应的Unicode编码



字符串函数 chr()

- Python的chr()函数是一个将整数转换为对应字符的函数。
- 例如，如果你输入数字65，它将返回字符'A'。
- 举个例子，假设你想将数字65转换为字符，你可以这样写：

```
print(chr(65))
```

- 输出结果将会是：A

字符串 ord函数

- Python ord函数可以用来比较字符串。它返回一个字符串中每个字符的Unicode编码值。例如：
- 字符串"apple"
- ord("a") 返回 97
- ord("p") 返回 112
- ord("l") 返回 108
- ord("e") 返回 101
- 所以我们可以比较两个字符串，比较它们中每个字符的Unicode编码值。
- 例如：
- 字符串"apple" 和 字符串"cat"
- **ord("a") < ord("c")** 所以 **"apple" 比 "cat" 小**。

字符串处理方法

一些以方法形式提供的字符串处理功能

方法及使用 1/3	描述
str.lower() 或 str.upper()	返回字符串的副本，全部字符小写/大写 "AbCdEfGh".lower() 结果为 "abcdefgh"
str.split(sep=None)	返回一个列表，由str根据sep被分隔的部分组成 "A,B,C".split(",") 结果为 ['A','B','C']
str.count(sub)	返回子串sub在str中出现的次数 "a apple a day".count("a") 结果为 4

字符串方法 变量名 .lower()

- Python中的lower()是一个字符串方法，它可以将字符串中的
- 所有字母变成小写。比如：

```
word = "HELLO"
print(word.lower())
```

这个程序会输出`hello`。可以看到，lower()方法把大写字母变成了小写字母。这个方法可以让我们方便地在输入字符串时忽略大小写，将输入的单词统一转换成小写字母，这样就可以避免输入错误了。

字符串方法 变量名 .upper()

- Python中的upper()函数是一个字符串方法，它将字符串中的所有字母都变成大写字母。

例如：

```
s = "hello world!"
s_upper = s.upper()
print(s_upper)
```

- 这段代码会输出"HELLO WORLD!"，因为upper()函数将字符串中的所有字母都变成了大写字母。
- 所以，upper()函数，可以让字母变成大写，让我们的字符串看起来更整齐、更有规律。

字符串方法 变量名 .split()

- Python中的split()是一个字符串函数，它将一个字符串分割成多个部
- 分，并返回这些部分的列表。这个函数可以通过指定分隔符来分割
- 字符串，如果不指定分隔符，则默认使用空格作为分隔符。
- 例如，如果有一个字符串"Hello World"，我们可以使用split()函数来
- 将它分割成两个部分，即"Hello"和"World"，代码如下：

```
string = "Hello World"
split_string = string.split()
print(split_string)
```

- 输出结果为：
- ['Hello', 'World']
- 这里我们没有指定分隔符，因此默认使用空格作为分隔符。
- 使用split()函数可以方便地对字符串进行分割并进行处理。

字符串方法 变量名 .split()

- Python中的split()是一个字符串函数，它将一个字符串分割成多个部分，并返回这些部分的列表。这个函数可以通过指定分隔符来分割字符串，如果指定分隔符，使用它作为分隔符。
- 例如，代码如下：

```
s = "你好,欢迎参观,CAUC空管学院"
split_string = s.split(",")
print(split_string)
```

输出结果为：

['你好', '欢迎参观', 'CAUC空管学院']

- 这里是以指定分隔符 ","， 将"你好,欢迎参观,CAUC空管学院"
- 分割成由三个子串组成的一个**列表**。

字符串方法 变量名 .split()

- Python中的split函数可以将一个字符串按照指定的分隔符进行分割。
- 比如说，我们有一个句子“我喜欢吃火锅”，如果我们想把这个句
- 子按照“吃”这个字进行分割，就可以使用split函数。 这样写代码

```
sentence = "我喜欢吃火锅"
words = sentence.split("吃")
print(words)
```

- 运行结果会输出一个列表，['我喜欢', '火锅']，其中包含两个元素：
- “我喜欢”和“火锅”。
- 在这个例子中，我们使用split函数将句子按照“吃”这个字进行了分割，
- 得到了两个部分。

字符串方法 变量名 .count()

- Python中的字符串计数(count)是一种可以用来计算一个字符串中某个字符或子字符串在其中出现的次数。
- 例如，下面代码展示如何使用count功能来计算字符串中字符“o”的数量：

```
string = "hello world"
count = string.count("o")
print(count)
```

- 在这个例子中，我们定义了一个字符串变量string，它包含了“hello world”这个字符串。然后，我们使用count功能来计算字符“o”在其中出现的次数，并将结果存储在count变量中。
- 最后，我们使用print函数来输出count的值，即字符串中“o”的数量： 2个。

字符串处理方法

一些以方法形式提供的字符串处理功能

方法及使用 2/3	描述
str.replace(old, new)	返回字符串str副本，所有old子串被替换为new "python".replace("n","n123.io") 结果为 "python123.io"
str.center(width[,fillchar])	字符串str根据宽度width居中，fillchar可选 "python".center(20,"=") 结果为 '=======python======='

字符串方法 变量名 .replace()

- Python中的字符串的replace是用来替换字符串中的某些字符为另一些字符。
- 举个例子

```
str = "Hello World!"
new_str = str.replace("l", "p")
print(new_str)
```

这段代码的意思是把字符串"Hello World!"中的所有"l"替换成"p"，然后把新的字符串赋值给变量new_str，并且输出这个新的字符串。

输出的结果就是"Heppo Worpd!"，可以看到所有的"l"都被替换成了"p"。

这样就像我们可以用橡皮擦擦掉铅笔写的字一样，replace函数可以帮我们把字符串中的某些字换成别的字，让我们的程序更加灵活。

字符串方法 变量名 .center()

- Python中的字符串center是一个方法，它可以让你把一个字符串放在中间。
- 例如，如果你有一个字符串"hello"，你可以使用center方法，让它在一个长
- 度为20的字符串中居中显示。代码如下：

```
s = "hello"
c= s.center(20)
print(c)
print("*"*20)
```

这会输出：

```
       hello
********************
```

- 这就是center方法的作用。它会在字符串两边添加空格，让字符串居中显示。

字符串处理方法

一些以方法形式提供的字符串处理功能

方法及使用 3/3	描述
str.strip(chars)	从str中去掉在其左侧和右侧chars中列出的字符 "= python= ".strip(" =np") 结果为 "ytho"
str.join(iter)	在iter变量除最后元素外每个元素后增加一个str ",".join("12345") 结果为 "1,2,3,4,5" #主要用于字符串分隔等

字符串方法 变量名 .strip()

- Python中的strip是一个字符串方法，用于删除字符串开头和结尾的空格或特定字符。
- 例如，假设有一个字符串 "　hello world!　"，我们想要去掉这个字符串前后的空格，可以使用strip方法：

```
s= "　hello world!　"
print(len(s))
n = string.strip()
print(n)
print(len(n))
```

- 输出：

```
22
hello world
11
```

- 这样，我们就得到了一个没有空格的新字符串，长度由22个字符变为11个字符。

字符串方法 变量名 .join()

- Python中的字符串join是将多个字符串连接起来形成一个新的字符串的方法。
- 例如，假设你有两个字符串“hello”和“world”，你可以使用join方法将它
- 们连接起来形成一个新的字符串“helloworld”。 这就像是把两个石头粘在
- 一起形成一个更大的石头一样。示例代码：

```
words = ['hello', 'world']
sentence = ''.join(words)
print(sentence)

s= "欢迎报考中国民航大学空管学院"
sentence = '*'.join(s)
print(sentence)
```

- 输出

```
helloworld
欢*迎*报*考*中*国*民*航*大*学*空*管*学*院
```

- 这样，你就可以把多个字符串连接起来形成一个新的字符串啦！

字符串类型的格式化

格式化是对字符串进行格式表达的方式

- **字符串格式化使用.format()方法，用法如下：**

<模板字符串>.format(<逗号分隔的参数>)

字符串类型的格式化

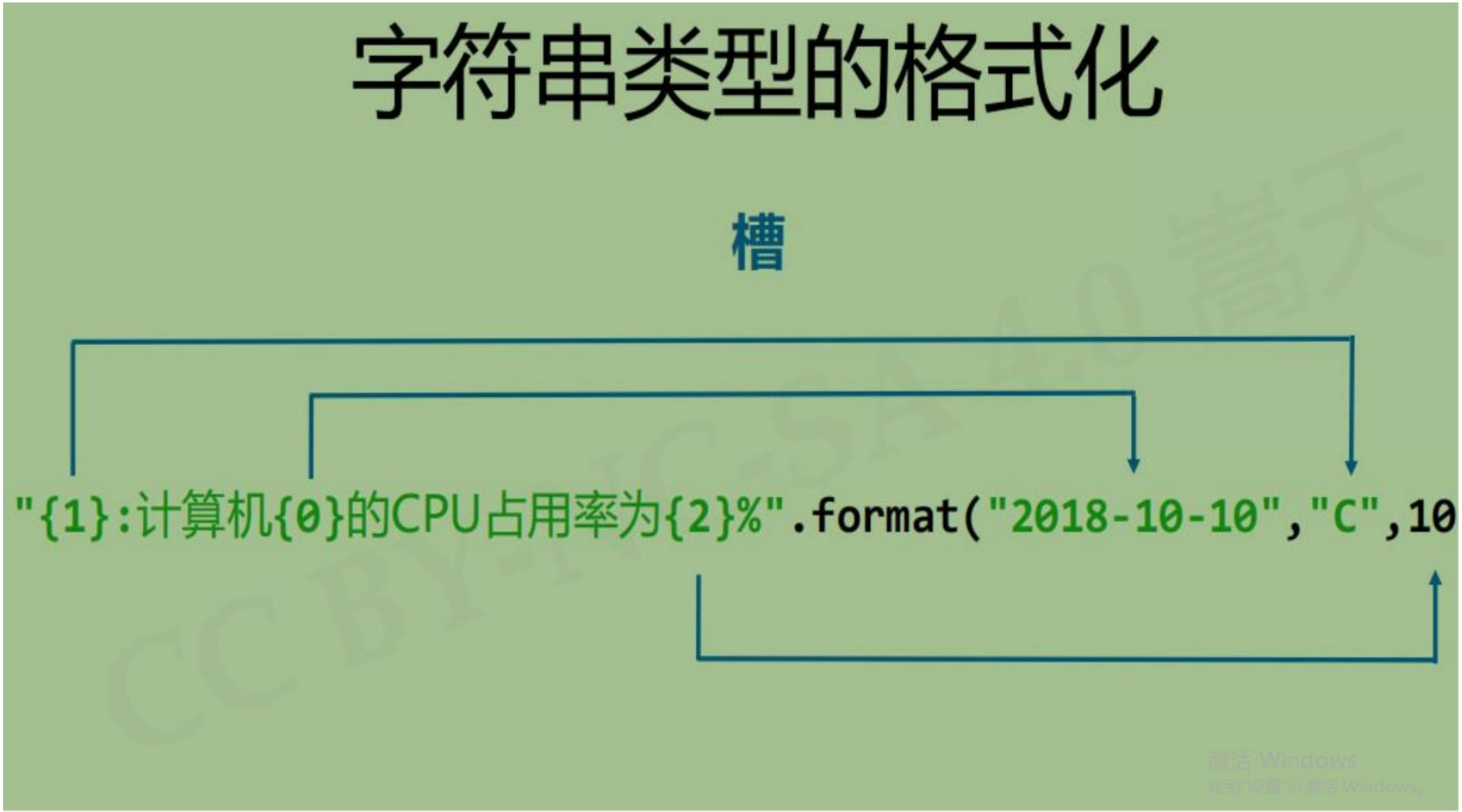
槽

```
"{ }:计算机{ }的CPU占用率为{ }%".format("2018-10-10","C",10)
```

字符串中槽{}的默认顺序

format()中参数的顺序

字符串类型的格式化



format()方法的格式控制

槽内部对格式化的配置方式

{ <参数序号> ： <格式控制标记>}

：	<填充>	<对齐>	<宽度>	<,>	<.精度>	<类型>
引导符号	用于填充的单个字符	< 左对齐 > 右对齐 ^ 居中对齐	槽设定的输出宽度	数字的千位分隔符	浮点数小数精度 或 字符串最大输出长度	整数类型 b, c, d, o, x, X 浮点数类型 e, E, f, %

format()方法的格式控制

:	<填充>	<对齐>	<宽度>	<,>	<.精度>	<类型>
引导符号	用于填充的单个字符	< 左对齐 > 右对齐 ^ 居中对齐	槽设定的输出宽度			

```
>>>"{0:=^20}".format("PYTHON")
'=======PYTHON======='
>>>"{0:*>20}".format("BIT")
'*****************BIT‘
>>>"{:10}".format("BIT")
'BIT       '
```

format()方法的格式控制

:	<填充>	<对齐>	<宽度>	<,>	<.精度>	<类型>
				数字的千位分隔符	浮点数小数精度 或 字符串最大输出长度	整数类型 b, c, d, o, x, X 浮点数类型 e, E, f, %

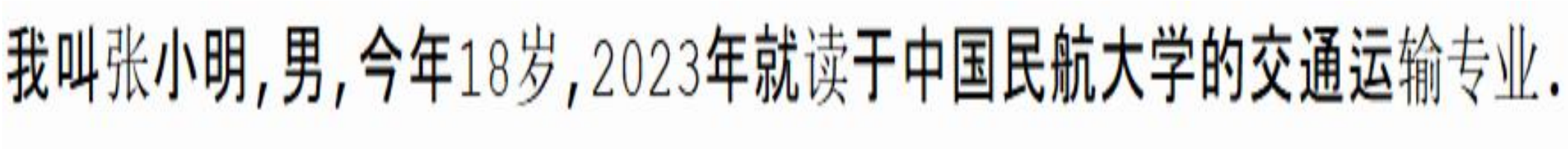
```
>>>"{0:,.2f}".format(12345.6789)
'12,345.68'
>>>"{0:b},{0:c},{0:d},{0:o},{0:x},{0:X}".format(425)
'110101001,Σ,425,651,1a9,1A9'
>>>"{0:e},{0:E},{0:f},{0:%}".format(3.14)
'3.140000e+00,3.140000E+00,3.140000,314.000000%'
```

字符串格式方法“ ”.format()

- Python的字符串格式化是一种将变量插入到字符串中的方法.它可以让你创
- 建几个带有空位的字符串，用变量填充这些空位，从而创建一个完整的字符
- 串。{}表示空位。
- 例如:

```
name = "张小明"
age = 18
school="中国民航大学"
grade ="交通运输"
message = "我叫{},男,今年{}岁,2023年就读于{}的{}专业.".format(name, age,school,grade)
print(message)
```

输出：



解释：

```
message = "我叫{},男,今年{}岁,2023年就读于{}的{}专业".format(name, age,school,grade)
print(message)
```

```
我叫张小明,男,今年18岁,2023年就读于中国民航大学的交通运输专业.
```

在这个例子中，我们使用format方法创建了一个包含四个空位的字符串。我们用变量name、age、school和grade来填充这些空位，然后将结果存储在变量message中。最后，我们使用print函数输出这个字符串。

所以，Python中的字符串format方法可以让你更轻松地创建带有变量的字符串，从而更方便地输出你想要的内容，输出复杂的字符串，可以生成动态的文本

字符串格式方法“ ”.format()

- Python的format函数可以用来控制数字的显示方式。如果要保留一个小数，可以
- 在大括号中使用格式化符号“:.1f”，就是"{:.1f}"表示要输出一个浮点数，并保留
  - 1位小数。下面是一段Python代码，可以进行四则运算并保留1位小数：

```
a = float(input("第一个数"))
b = float(input("第二个数"))
c = a + b
d = a - b
e = a * b
f = a / b
print("a + b = {:.1f}".format(c))
print("a - b = {:.1f}".format(d))
print("a × b = {:.1f}".format(e))
print("a ÷ b = {:.1f}".format(f))
```

```
>>> %Run '四则运算format版.py'
第一个数3.14
第二个数1.23
a + b = 4.4
a - b = 1.9
a × b = 3.9
a ÷ b = 2.6
```

问题：？