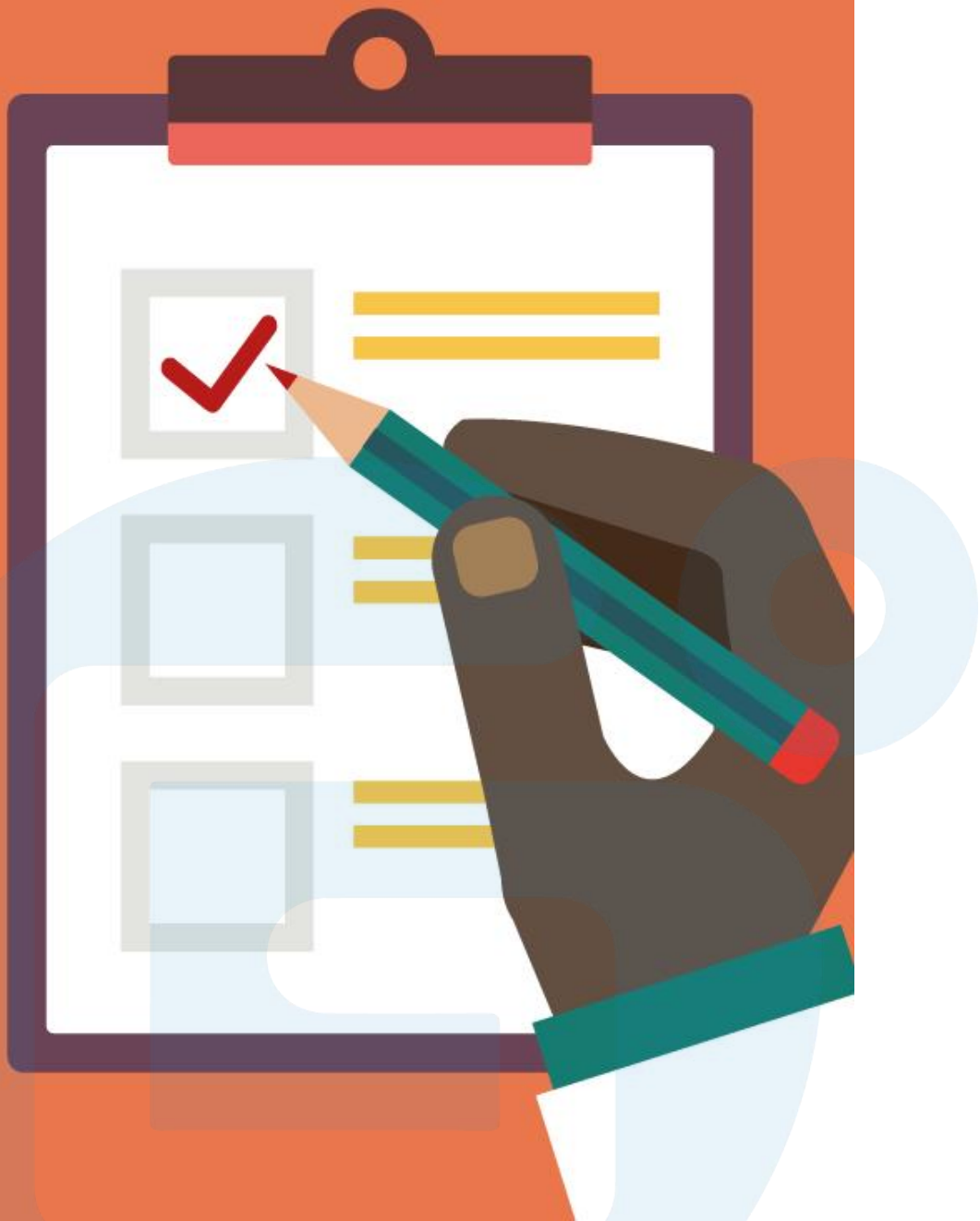


Talk is cheap, show me the code

第四课：Python数据类型（续）及实践

Python初阶入门课程系列



OUTLINE

➤ 字符串型变量

➤ 时间和日期

(`time/datetime`) 变量

➤ 实例（进度条）



一 字符串型变量



Python字符串的概念

由0个或多个字符组成的有序字符序列

- 字符串由一对单引号或一对双引号表示

"请输入带有符号的温度值:" 或者 'C'

- 字符串是字符的有序序列，可以对其中的字符进行索引

例如，“输”是“输入参数列表”这个字符串的第0个字符（再次强调一下python的索引规则，从0开始！）

字符串有2类4种表示方法

- 单行字符串用单引号和双引号标识

“输入序列” 或者 ‘AAA’

- 多行字符串由””和”””标识

和多行注释很类似，不过，多行注释是独立出现的，而字符串不会独立出现。

通常与赋值语句和函数命令联合使用，因此不会混淆。

单引号和双引号的区别？

- 如果希望在字符串中包含双引号或单引号呢？

'这里有个双引号("')' 或者 "这里有个单引号(')"

- 如果希望在字符串中既包括单引号又包括双引号呢？

"" 这里既有单引号(')又有双引号 (") ""

- 1). (单引号、双引号) 表示多行时需要添加换行符\n。
- 2). (三单引号、三双引号) 表示多行时无需使用任何多余字符
- 3). (三单引号、三双引号) 中可直接使用 (单引号、双引号) 而无需使用反斜杠\进行转义

字符串型变量

```
1  ###1
2  print('这是一段话')
3  print('这也是一段话')
4  print('这里有一个双引号""') #单引号里的双引号被视为字符串的一部分
5  print("这里也有一个双引号""")
6  print("这里也有一个双引号\"") #用转义符\标识出来的双引号被视为字符串的一部分
7  print("这里有一个单引号'") #双引号里的单引号也被视为字符串的一部分
8  print(''''这里既有单引号', 也有双引号",
9        但是因为用三引号圈起来,
10       所以都没问题''')
```

控制台 1/A

```
In [10]: runfile('C:/Users/Administrator/Desktop/test/aaaa.py', wdir='C:/Users/Admini:
这是一段话
这也是一段话
这里有一个双引号""
这里也有一个双引号
这里也有一个双引号""
这里有一个单引号'
这里既有单引号', 也有双引号",
    但是因为用三引号圈起来,
    所以都没问题
```

字符串索引

字符串可以被视为是一组有序号索引的符号



字符串索引

字符串可以被视为是一组有序号索引的符号

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]

“这是字符串序号的示意图：”[0] 或者 Str1[-1]

➤ 切片：返回字符串中一段字符子串 <字符串>[M: N]

“这是字符串序号的示意图：”[1:3] 或者 Str1[0:-1]

```
1  #%%1
2  str1="这是字符串序号的示意图："
3  print("这是字符串序号的示意图："[0])
4  print(str1[-1])
5  print("这是字符串序号的示意图："[0:3])
6  print(str1[-3:-1])
7  print(str1[-3:-1],str1[-1])
```

控制台 1/A

```
In [26]: runfile('C:/Users/Administrator/Desktop/test/aaaa.py')
这
:
这是字
意图
意图 :
```

字符串高级切片

字符串切片作为一种索引方法，在python各种数据结构中是通用的。

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"

字符串型变量

字符串特殊字符

转义符 \

➤ - 转义符表达特定字符的本意

"这里有个双引号(\"")" 结果为

这里有个双引号(")

➤ - 转义符形成一些组合，表达一些不可打印的含义

"\b"回退 (=backspace)

"\n"换行(光标移动到下行首，=Enter)

"\r" 回车(光标移动到本行首)

回车和换行的历史：

机械打字机有回车和换行两个键作用分别是：

换行就是把滚筒卷一格，不改变水平位置。（即移到下一行，但不是行首，而是和上一行水平位置一样）

回车就是把水平位置复位，不卷动滚筒。（即将光标移到行首，但是不会移到下一行，如果继续输入的话会覆盖掉前面的内容）

Enter = 回车+换行(\r\n)

理解：

\n是换行，英文是New line

\r是回车，英文是Carriage return



\r与\n与\r\n的区别图解

字符串特殊字符

```
1  #%%1
2  print("这段话演示\n换行转义符\n的作用")
3  print('~~~~~')
4  print("演示\r回车转义符\r的作用")
5  print('~~~~~')
6  print("这段话演示\b退格转义符\b\b的作用")
```

控制台 1/A

```
In [37]: runfile('C:/Users/Administrator/Desktop/test/aaaa.
这段话演示
换行转义符
的作用
~~~~~
的作用义符
~~~~~
这段话演退格转的作用
```

字符串操作符

类似于数值操作符，字符串也有一些操作符

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
- $x \text{ in } s$ 如果 x 是 s 的子串，返回True，否则返回False

```
1  #%%1
2  x1="1234567"
3  x2="345"
4  print(x1+x2)
5  print(3*x1)
6  print(x2 in x1)
```

控制台 1/A

```
In [42]: runfile('C:/Users/Administ
1234567345
123456712345671234567
True
```

字符串型变量

实例

```
1  #%%1
2  weekStr = "星期一星期二星期三星期四星期五星期六星期日"
3  weekId = eval(input("请输入星期数字(1-7): "))
4  #eval() 函数用来执行一个字符串表达式,并返回表达式的值。
5  pos=(weekId-1)*3
6  print(weekStr[pos: pos+3])
7  #%%2
8  weekStr = "一二三四五六日"
9  weekId = eval(input("请输入星期数字(1-7): "))
10 print("星期" + weekStr[weekId-1])
```

控制台 1/A

In [46]: runfile('C:/Users/Administrator/Desktop/test/aaaa.py', wdir='C:/Us

请输入星期数字(1-7): 5
星期五

请输入星期数字(1-7): 7
星期日

字符串处理函数

一些以函数形式提供的字符串处理功能

- `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"
- `chr(x)` 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

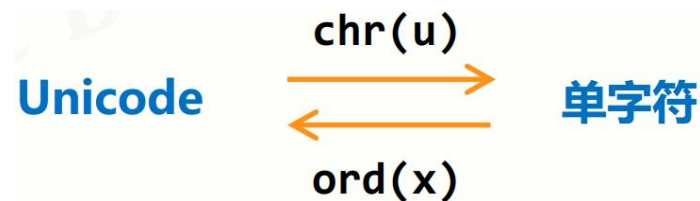
字符串型变量

Unicode编码

python字符串的编码方式

Unicode编码

- 统一字符编码，即覆盖几乎所有字符的编码方式
- 从0到1114111 (0x10FFFF)空间，每个编码对应一个字符
- Python字符串中每个字符都是Unicode编码字符



```
1 #%%1
2 print("1 + 1 = 2 " + chr(10004))
3 print("这个字符𐀀的Unicode值是: " + str(ord("𐀀")))
4 for i in range(12):
5     print(chr(9800 + i), end="")
```

控制台 1/A

```
In [53]: runfile('C:/Users/Administrator/Desktop/test/aaaa.py',
1 + 1 = 2 ✓
这个字符𐀀的Unicode值是: 9801
𐀀𐀁𐀂𐀃𐀄𐀅𐀆𐀇𐀈𐀉𐀊𐀋𐀌𐀍𐀎𐀏𐀐𐀑𐀒𐀓𐀔𐀕𐀖𐀗𐀘𐀙𐀚𐀛𐀜𐀝𐀞𐀟𐀠𐀡𐀢𐀣𐀤𐀥𐀦𐀧𐀨𐀩𐀪𐀫𐀬𐀭𐀮𐀯𐀰𐀱𐀲𐀳𐀴𐀵𐀶𐀷𐀸𐀹𐀺𐀻𐀼𐀽𐀾𐀿𐁀𐁁𐁂𐁃𐁄𐁅𐁆𐁇𐁈𐁉𐁊𐁋𐁌𐁍𐁎𐁏𐁐𐁑𐁒𐁓𐁔𐁕𐁖𐁗𐁘𐁙𐁚𐁛𐁜𐁝𐁞𐁟𐁠𐁡𐁢𐁣𐁤𐁥𐁦𐁧𐁨𐁩𐁪𐁫𐁬𐁭𐁮𐁯𐁰𐁱𐁲𐁳𐁴𐁵𐁶𐁷𐁸𐁹𐁺𐁻𐁼𐁽𐁾𐁿𐂀𐂁𐂂𐂃𐂄𐂅𐂆𐂇𐂈𐂉𐂊𐂋𐂌𐂍𐂎𐂏𐂐𐂑𐂒𐂓𐂔𐂕𐂖𐂗𐂘𐂙𐂚𐂛𐂜𐂝𐂞𐂟𐂠𐂡𐂢𐂣𐂤𐂥𐂦𐂧𐂨𐂩𐂪𐂫𐂬𐂭𐂮𐂯𐂰𐂱𐂲𐂳𐂴𐂵𐂶𐂷𐂸𐂹𐂺𐂻𐂼𐂽𐂾𐂿𐃀𐃁𐃂𐃃𐃄𐃅𐃆𐃇𐃈𐃉𐃊𐃋𐃌𐃍𐃎𐃏𐃐𐃑𐃒𐃓𐃔𐃕𐃖𐃗𐃘𐃙𐃚𐃛𐃜𐃝𐃞𐃟𐃠𐃡𐃢𐃣𐃤𐃥𐃦𐃧𐃨𐃩𐃪𐃫𐃬𐃭𐃮𐃯𐃰𐃱𐃲𐃳𐃴𐃵𐃶𐃷𐃸𐃹𐃺𐃻𐃼𐃽𐃾𐃿𐄀𐄁𐄂𐄃𐄄𐄅𐄆𐄇𐄈𐄉𐄊𐄋𐄌𐄍𐄎𐄏𐄐𐄑𐄒𐄓𐄔𐄕𐄖𐄗𐄘𐄙𐄚𐄛𐄜𐄝𐄞𐄟𐄠𐄡𐄢𐄣𐄤𐄥𐄦𐄧𐄨𐄩𐄪𐄫𐄬𐄭𐄮𐄯𐄰𐄱𐄲𐄳𐄴𐄵𐄶𐄷𐄸𐄹𐄺𐄻𐄼𐄽𐄾𐄿𐅀𐅁𐅂𐅃𐅄𐅅𐅆𐅇𐅈𐅉𐅊𐅋𐅌𐅍𐅎𐅏𐅐𐅑𐅒𐅓𐅔𐅕𐅖𐅗𐅘𐅙𐅚𐅛𐅜𐅝𐅞𐅟𐅠𐅡𐅢𐅣𐅤𐅥𐅦𐅧𐅨𐅩𐅪𐅫𐅬𐅭𐅮𐅯𐅰𐅱𐅲𐅳𐅴𐅵𐅶𐅷𐅸𐅹𐅺𐅻𐅼𐅽𐅾𐅿𐆀𐆁𐆂𐆃𐆄𐆅𐆆𐆇𐆈𐆉𐆊𐆋𐆌𐆍𐆎𐆏𐆐𐆑𐆒𐆓𐆔𐆕𐆖𐆗𐆘𐆙𐆚𐆛𐆜𐆝𐆞𐆟𐆠𐆡𐆢𐆣𐆤𐆥𐆦𐆧𐆨𐆩𐆪𐆫𐆬𐆭𐆮𐆯𐆰𐆱𐆲𐆳𐆴𐆵𐆶𐆷𐆸𐆹𐆺𐆻𐆼𐆽𐆾𐆿𐇀𐇁𐇂𐇃𐇄𐇅𐇆𐇇𐇈𐇉𐇊𐇋𐇌𐇍𐇎𐇏𐇐𐇑𐇒𐇓𐇔𐇕𐇖𐇗𐇘𐇙𐇚𐇛𐇜𐇝𐇞𐇟𐇠𐇡𐇢𐇣𐇤𐇥𐇦𐇧𐇨𐇩𐇪𐇫𐇬𐇭𐇮𐇯𐇰𐇱𐇲𐇳𐇴𐇵𐇶𐇷𐇸𐇹𐇺𐇻𐇼𐇽𐇾𐇿𐈀𐈁𐈂𐈃𐈄𐈅𐈆𐈇𐈈𐈉𐈊𐈋𐈌𐈍𐈎𐈏𐈐𐈑𐈒𐈓𐈔𐈕𐈖𐈗𐈘𐈙𐈚𐈛𐈜𐈝𐈞𐈟𐈠𐈡𐈢𐈣𐈤𐈥𐈦𐈧𐈨𐈩𐈪𐈫𐈬𐈭𐈮𐈯𐈰𐈱𐈲𐈳𐈴𐈵𐈶𐈷𐈸𐈹𐈺𐈻𐈼𐈽𐈾𐈿𐉀𐉁𐉂𐉃𐉄𐉅𐉆𐉇𐉈𐉉𐉊𐉋𐉌𐉍𐉎𐉏𐉐𐉑𐉒𐉓𐉔𐉕𐉖𐉗𐉘𐉙𐉚𐉛𐉜𐉝𐉞𐉟𐉠𐉡𐉢𐉣𐉤𐉥𐉦𐉧𐉨𐉩𐉪𐉫𐉬𐉭𐉮𐉯𐉰𐉱𐉲𐉳𐉴𐉵𐉶𐉷𐉸𐉹𐉺𐉻𐉼𐉽𐉾𐉿𐊀𐊁𐊂𐊃𐊄𐊅𐊆𐊇𐊈𐊉𐊊𐊋𐊌𐊍𐊎𐊏𐊐𐊑𐊒𐊓𐊔𐊕𐊖𐊗𐊘𐊙𐊚𐊛𐊜𐊝𐊞𐊟𐊠𐊡𐊢𐊣𐊤𐊥𐊦𐊧𐊨𐊩𐊪𐊫𐊬𐊭𐊮𐊯𐊰𐊱𐊲𐊳𐊴𐊵𐊶𐊷𐊸𐊹𐊺𐊻𐊼𐊽𐊾𐊿𐋀𐋁𐋂𐋃𐋄𐋅𐋆𐋇𐋈𐋉𐋊𐋋𐋌𐋍𐋎𐋏𐋐𐋑𐋒𐋓𐋔𐋕𐋖𐋗𐋘𐋙𐋚𐋛𐋜𐋝𐋞𐋟𐋠𐋡𐋢𐋣𐋤𐋥𐋦𐋧𐋨𐋩𐋪𐋫𐋬𐋭𐋮𐋯𐋰𐋱𐋲𐋳𐋴𐋵𐋶𐋷𐋸𐋹𐋺𐋻𐋼𐋽𐋾𐋿𐌀𐌁𐌂𐌃𐌄𐌅𐌆𐌇𐌈𐌉𐌊𐌋𐌌𐌍𐌎𐌏𐌐𐌑𐌒𐌓𐌔𐌕𐌖𐌗𐌘𐌙𐌚𐌛𐌜𐌝𐌞𐌟𐌠𐌡𐌢𐌣𐌤𐌥𐌦𐌧𐌨𐌩𐌪𐌫𐌬𐌭𐌮𐌯𐌰𐌱𐌲𐌳𐌴𐌵𐌶𐌷𐌸𐌹𐌺𐌻𐌼𐌽𐌾𐌿𐍀𐍁𐍂𐍃𐍄𐍅𐍆𐍇𐍈𐍉𐍊𐍋𐍌𐍍𐍎𐍏𐍐𐍑𐍒𐍓𐍔𐍕𐍖𐍗𐍘𐍙𐍚𐍛𐍜𐍝𐍞𐍟𐍠𐍡𐍢𐍣𐍤𐍥𐍦𐍧𐍨𐍩𐍪𐍫𐍬𐍭𐍮𐍯𐍰𐍱𐍲𐍳𐍴𐍵𐍶𐍷𐍸𐍹𐍺𐍻𐍼𐍽𐍾𐍿𐎀𐎁𐎂𐎃𐎄𐎅𐎆𐎇𐎈𐎉𐎊𐎋𐎌𐎍𐎎𐎏𐎐𐎑𐎒𐎓𐎔𐎕𐎖𐎗𐎘𐎙𐎚𐎛𐎜𐎝𐎞𐎟𐎠𐎡𐎢𐎣𐎤𐎥𐎦𐎧𐎨𐎩𐎪𐎫𐎬𐎭𐎮𐎯𐎰𐎱𐎲𐎳𐎴𐎵𐎶𐎷𐎸𐎹𐎺𐎻𐎼𐎽𐎾𐎿𐏀𐏁𐏂𐏃𐏄𐏅𐏆𐏇𐏈𐏉𐏊𐏋𐏌𐏍𐏎𐏏𐏐𐏑𐏒𐏓𐏔𐏕𐏖𐏗𐏘𐏙𐏚𐏛𐏜𐏝𐏞𐏟𐏠𐏡𐏢𐏣𐏤𐏥𐏦𐏧𐏨𐏩𐏪𐏫𐏬𐏭𐏮𐏯𐏰𐏱𐏲𐏳𐏴𐏵𐏶𐏷𐏸𐏹𐏺𐏻𐏼𐏽𐏾𐏿𐐀𐐁𐐂𐐃𐐄𐐅𐐆𐐇𐐈𐐉𐐊𐐋𐐌𐐍𐐎𐐏𐐐𐐑𐐒𐐓𐐔𐐕𐐖𐐗𐐘𐐙𐐚𐐛𐐜𐐝𐐞𐐟𐐠𐐡𐐢𐐣𐐤𐐥𐐦𐐧𐐨𐐩𐐪𐐫𐐬𐐭𐐮𐐯𐐰𐐱𐐲𐐳𐐴𐐵𐐶𐐷𐐸𐐹𐐺𐐻𐐼𐐽𐐾𐐿𐑀𐑁𐑂𐑃𐑄𐑅𐑆𐑇𐑈𐑉𐑊𐑋𐑌𐑍𐑎𐑏𐑐𐑑𐑒𐑓𐑔𐑕𐑖𐑗𐑘𐑙𐑚𐑛𐑜𐑝𐑞𐑟𐑠𐑡𐑢𐑣𐑤𐑥𐑦𐑧𐑨𐑩𐑪𐑫𐑬𐑭𐑮𐑯𐑰𐑱𐑲𐑳𐑴𐑵𐑶𐑷𐑸𐑹𐑺𐑻𐑼𐑽𐑾𐑿𐒀𐒁𐒂𐒃𐒄𐒅𐒆𐒇𐒈𐒉𐒊𐒋𐒌𐒍𐒎𐒏𐒐𐒑𐒒𐒓𐒔𐒕𐒖𐒗𐒘𐒙𐒚𐒛𐒜𐒝𐒞𐒟𐒠𐒡𐒢𐒣𐒤𐒥𐒦𐒧𐒨𐒩𐒪𐒫𐒬𐒭𐒮𐒯𐒰𐒱𐒲𐒳𐒴𐒵𐒶𐒷𐒸𐒹𐒺𐒻𐒼𐒽𐒾𐒿𐓀𐓁𐓂𐓃𐓄𐓅𐓆𐓇𐓈𐓉𐓊𐓋𐓌𐓍𐓎𐓏𐓐𐓑𐓒𐓓𐓔𐓕𐓖𐓗𐓘𐓙𐓚𐓛𐓜𐓝𐓞𐓟𐓠𐓡𐓢𐓣𐓤𐓥𐓦𐓧𐓨𐓩𐓪𐓫𐓬𐓭𐓮𐓯𐓰𐓱𐓲𐓳𐓴𐓵𐓶𐓷𐓸𐓹𐓺𐓻𐓼𐓽𐓾𐓿𐔀𐔁𐔂𐔃𐔄𐔅𐔆𐔇𐔈𐔉𐔊𐔋𐔌𐔍𐔎𐔏𐔐𐔑𐔒𐔓𐔔𐔕𐔖𐔗𐔘𐔙𐔚𐔛𐔜𐔝𐔞𐔟𐔠𐔡𐔢𐔣𐔤𐔥𐔦𐔧𐔨𐔩𐔪𐔫𐔬𐔭𐔮𐔯𐔰𐔱𐔲𐔳𐔴𐔵𐔶𐔷𐔸𐔹𐔺𐔻𐔼𐔽𐔾𐔿𐕀𐕁𐕂𐕃𐕄𐕅𐕆𐕇𐕈𐕉𐕊𐕋𐕌𐕍𐕎𐕏𐕐𐕑𐕒𐕓𐕔𐕕𐕖𐕗𐕘𐕙𐕚𐕛𐕜𐕝𐕞𐕟𐕠𐕡𐕢𐕣𐕤𐕥𐕦𐕧𐕨𐕩𐕪𐕫𐕬𐕭𐕮𐕯𐕰𐕱𐕲𐕳𐕴𐕵𐕶𐕷𐕸𐕹𐕺𐕻𐕼𐕽𐕾𐕿𐖀𐖁𐖂𐖃𐖄𐖅𐖆𐖇𐖈𐖉𐖊𐖋𐖌𐖍𐖎𐖏𐖐𐖑𐖒𐖓𐖔𐖕𐖖𐖗𐖘𐖙𐖚𐖛𐖜𐖝𐖞𐖟𐖠𐖡𐖢𐖣𐖤𐖥𐖦𐖧𐖨𐖩𐖪𐖫𐖬𐖭𐖮𐖯𐖰𐖱𐖲𐖳𐖴𐖵𐖶𐖷𐖸𐖹𐖺𐖻𐖼𐖽𐖾𐖿𐗀𐗁𐗂𐗃𐗄𐗅𐗆𐗇𐗈𐗉𐗊𐗋𐗌𐗍𐗎𐗏𐗐𐗑𐗒𐗓𐗔𐗕𐗖𐗗𐗘𐗙𐗚𐗛𐗜𐗝𐗞𐗟𐗠𐗡𐗢𐗣𐗤𐗥𐗦𐗧𐗨𐗩𐗪𐗫𐗬𐗭𐗮𐗯𐗰𐗱𐗲𐗳𐗴𐗵𐗶𐗷𐗸𐗹𐗺𐗻𐗼𐗽𐗾𐗿𐘀𐘁𐘂𐘃𐘄𐘅𐘆𐘇𐘈𐘉𐘊𐘋𐘌𐘍𐘎𐘏𐘐𐘑𐘒𐘓𐘔𐘕𐘖𐘗𐘘𐘙𐘚𐘛𐘜𐘝𐘞𐘟𐘠𐘡𐘢𐘣𐘤𐘥𐘦𐘧𐘨𐘩𐘪𐘫𐘬𐘭𐘮𐘯𐘰𐘱𐘲𐘳𐘴𐘵𐘶𐘷𐘸𐘹𐘺𐘻𐘼𐘽𐘾𐘿𐙀𐙁𐙂𐙃𐙄𐙅𐙆𐙇𐙈𐙉𐙊𐙋𐙌𐙍𐙎𐙏𐙐𐙑𐙒𐙓𐙔𐙕𐙖𐙗𐙘𐙙𐙚𐙛𐙜𐙝𐙞𐙟𐙠𐙡𐙢𐙣𐙤𐙥𐙦𐙧𐙨𐙩𐙪𐙫𐙬𐙭𐙮𐙯𐙰𐙱𐙲𐙳𐙴𐙵𐙶𐙷𐙸𐙹𐙺𐙻𐙼𐙽𐙾𐙿𐚀𐚁𐚂𐚃𐚄𐚅𐚆𐚇𐚈𐚉𐚊𐚋𐚌𐚍𐚎𐚏𐚐𐚑𐚒𐚓𐚔𐚕𐚖𐚗𐚘𐚙𐚚𐚛𐚜𐚝𐚞𐚟𐚠𐚡𐚢𐚣𐚤𐚥𐚦𐚧𐚨𐚩𐚪𐚫𐚬𐚭𐚮𐚯𐚰𐚱𐚲𐚳𐚴𐚵𐚶𐚷𐚸𐚹𐚺𐚻𐚼𐚽𐚾𐚿𐛀𐛁𐛂𐛃𐛄𐛅𐛆𐛇𐛈𐛉𐛊𐛋𐛌𐛍𐛎𐛏𐛐𐛑𐛒𐛓𐛔𐛕𐛖𐛗𐛘𐛙𐛚𐛛𐛜𐛝𐛞𐛟𐛠𐛡𐛢𐛣𐛤𐛥𐛦𐛧𐛨𐛩𐛪𐛫𐛬𐛭𐛮𐛯𐛰𐛱𐛲𐛳𐛴𐛵𐛶𐛷𐛸𐛹𐛺𐛻𐛼𐛽𐛾𐛿𐜀𐜁𐜂𐜃𐜄𐜅𐜆𐜇𐜈𐜉𐜊𐜋𐜌𐜍𐜎𐜏𐜐𐜑𐜒𐜓𐜔𐜕𐜖𐜗𐜘𐜙𐜚𐜛𐜜𐜝𐜞𐜟𐜠𐜡𐜢𐜣𐜤𐜥𐜦𐜧𐜨𐜩𐜪𐜫𐜬𐜭𐜮𐜯𐜰𐜱𐜲𐜳𐜴𐜵𐜶𐜷𐜸𐜹𐜺𐜻𐜼𐜽𐜾𐜿𐝀𐝁𐝂𐝃𐝄𐝅𐝆𐝇𐝈𐝉𐝊𐝋𐝌𐝍𐝎𐝏𐝐𐝑𐝒𐝓𐝔𐝕𐝖𐝗𐝘𐝙𐝚𐝛𐝜𐝝𐝞𐝟𐝠𐝡𐝢𐝣𐝤𐝥𐝦𐝧𐝨𐝩𐝪𐝫𐝬𐝭𐝮𐝯𐝰𐝱𐝲𐝳𐝴𐝵𐝶𐝷𐝸𐝹𐝺𐝻𐝼𐝽𐝾𐝿𐞀𐞁𐞂𐞃𐞄𐞅𐞆𐞇𐞈𐞉𐞊𐞋𐞌𐞍𐞎𐞏𐞐𐞑𐞒𐞓𐞔𐞕𐞖𐞗𐞘𐞙𐞚𐞛𐞜𐞝𐞞𐞟𐞠𐞡𐞢𐞣𐞤𐞥𐞦𐞧𐞨𐞩𐞪𐞫𐞬𐞭𐞮𐞯𐞰𐞱𐞲𐞳𐞴𐞵𐞶𐞷𐞸𐞹𐞺𐞻𐞼𐞽𐞾𐞿𐟀𐟁𐟂𐟃𐟄𐟅𐟆𐟇𐟈𐟉𐟊𐟋𐟌𐟍𐟎𐟏𐟐𐟑𐟒𐟓𐟔𐟕𐟖𐟗𐟘𐟙𐟚𐟛𐟜𐟝𐟞𐟟𐟠𐟡𐟢𐟣𐟤𐟥𐟦𐟧𐟨𐟩𐟪𐟫𐟬𐟭𐟮𐟯𐟰𐟱𐟲𐟳𐟴𐟵𐟶𐟷𐟸𐟹𐟺𐟻𐟼𐟽𐟾𐟿𐠀𐠁𐠂𐠃𐠄𐠅𐠆𐠇𐠈𐠉𐠊𐠋𐠌𐠍𐠎𐠏𐠐𐠑𐠒𐠓𐠔𐠕𐠖𐠗𐠘𐠙𐠚𐠛𐠜𐠝𐠞𐠟𐠠𐠡𐠢𐠣𐠤𐠥𐠦𐠧𐠨𐠩𐠪𐠫𐠬𐠭𐠮𐠯𐠰𐠱𐠲𐠳𐠴𐠵𐠶𐠷𐠸𐠹𐠺𐠻𐠼𐠽𐠾𐠿𐡀𐡁𐡂𐡃𐡄𐡅𐡆𐡇𐡈𐡉𐡊𐡋𐡌𐡍𐡎𐡏𐡐𐡑𐡒𐡓𐡔𐡕𐡖𐡗𐡘𐡙𐡚𐡛𐡜𐡝𐡞𐡟𐡠𐡡𐡢𐡣𐡤𐡥𐡦𐡧𐡨𐡩𐡪𐡫𐡬𐡭𐡮𐡯𐡰𐡱𐡲𐡳𐡴𐡵𐡶𐡷𐡸𐡹𐡺𐡻𐡼𐡽𐡾𐡿𐢀𐢁𐢂𐢃𐢄𐢅𐢆𐢇𐢈𐢉𐢊𐢋𐢌𐢍𐢎𐢏𐢐𐢑𐢒𐢓𐢔𐢕𐢖𐢗𐢘𐢙𐢚𐢛𐢜𐢝𐢞𐢟𐢠𐢡𐢢𐢣𐢤𐢥𐢦𐢧𐢨𐢩𐢪𐢫𐢬𐢭𐢮𐢯𐢰𐢱𐢲𐢳𐢴𐢵𐢶𐢷𐢸𐢹𐢺𐢻𐢼𐢽𐢾𐢿𐣀𐣁𐣂𐣃𐣄𐣅𐣆𐣇𐣈𐣉𐣊𐣋𐣌𐣍𐣎𐣏𐣐𐣑𐣒𐣓𐣔𐣕𐣖𐣗𐣘𐣙𐣚𐣛𐣜𐣝𐣞𐣟𐣠𐣡𐣢𐣣𐣤𐣥𐣦𐣧𐣨𐣩𐣪𐣫𐣬𐣭𐣮𐣯𐣰𐣱𐣲𐣳𐣴𐣵𐣶𐣷𐣸𐣹𐣺𐣻𐣼𐣽𐣾𐣿𐤀𐤁𐤂𐤃𐤄𐤅𐤆𐤇𐤈𐤉𐤊𐤋𐤌𐤍𐤎𐤏𐤐𐤑𐤒𐤓𐤔𐤕𐤖𐤗𐤘𐤙𐤚𐤛𐤜𐤝𐤞𐤟𐤠𐤡𐤢𐤣𐤤𐤥𐤦𐤧𐤨𐤩𐤪𐤫𐤬𐤭𐤮𐤯𐤰𐤱𐤲𐤳𐤴𐤵𐤶𐤷𐤸𐤹𐤺𐤻𐤼𐤽𐤾𐤿𐥀𐥁𐥂𐥃𐥄𐥅𐥆𐥇𐥈𐥉𐥊𐥋𐥌𐥍𐥎𐥏𐥐𐥑𐥒𐥓𐥔𐥕𐥖𐥗𐥘𐥙𐥚𐥛𐥜𐥝𐥞𐥟𐥠𐥡𐥢𐥣𐥤𐥥𐥦𐥧𐥨𐥩𐥪𐥫𐥬𐥭𐥮𐥯𐥰𐥱𐥲𐥳𐥴𐥵𐥶𐥷𐥸𐥹𐥺𐥻𐥼𐥽𐥾𐥿𐦀𐦁𐦂𐦃𐦄𐦅𐦆𐦇𐦈𐦉𐦊𐦋𐦌𐦍𐦎𐦏𐦐𐦑𐦒𐦓𐦔𐦕𐦖𐦗𐦘𐦙𐦚𐦛𐦜𐦝𐦞𐦟𐦠𐦡𐦢𐦣𐦤𐦥𐦦𐦧𐦨𐦩𐦪𐦫𐦬𐦭𐦮𐦯𐦰𐦱𐦲𐦳𐦴𐦵𐦶𐦷𐦸𐦹𐦺𐦻𐦼𐦽𐦾𐦿𐧀𐧁𐧂𐧃𐧄𐧅𐧆𐧇𐧈𐧉𐧊𐧋𐧌𐧍𐧎𐧏𐧐𐧑𐧒𐧓𐧔𐧕𐧖𐧗𐧘𐧙𐧚𐧛𐧜𐧝𐧞𐧟𐧠𐧡𐧢𐧣𐧤𐧥𐧦𐧧𐧨𐧩𐧪𐧫𐧬𐧭𐧮𐧯𐧰𐧱𐧲𐧳𐧴𐧵𐧶𐧷𐧸𐧹𐧺𐧻𐧼𐧽𐧾𐧿𐨀𐨁𐨂𐨃𐨄𐨅𐨆𐨇𐨈𐨉𐨊𐨋𐨌𐨍𐨎𐨏𐨐𐨑𐨒𐨓𐨔𐨕𐨖𐨗𐨘𐨙𐨚𐨛𐨜𐨝𐨞𐨟𐨠𐨡𐨢𐨣𐨤𐨥𐨦𐨧𐨨𐨩𐨪𐨫𐨬𐨭𐨮𐨯𐨰𐨱𐨲𐨳𐨴𐨵𐨶𐨷𐨹𐨺𐨸𐨻𐨼𐨽𐨾𐨿𐩀𐩁𐩂𐩃𐩄𐩅𐩆𐩇𐩈𐩉𐩊𐩋𐩌𐩍𐩎𐩏𐩐𐩑𐩒𐩓𐩔𐩕𐩖𐩗𐩘𐩙𐩚𐩛𐩜𐩝𐩞𐩟𐩠𐩡𐩢𐩣𐩤𐩥𐩦𐩧𐩨𐩩𐩪𐩫𐩬𐩭𐩮𐩯𐩰𐩱𐩲𐩳𐩴𐩵𐩶𐩷𐩸𐩹𐩺𐩻𐩼𐩽𐩾𐩿𐪀𐪁𐪂𐪃𐪄𐪅𐪆𐪇𐪈𐪉𐪊𐪋𐪌𐪍𐪎𐪏𐪐𐪑𐪒𐪓𐪔𐪕𐪖𐪗𐪘𐪙𐪚𐪛𐪜𐪝𐪞𐪟𐪠𐪡𐪢𐪣𐪤𐪥𐪦𐪧𐪨𐪩𐪪𐪫𐪬𐪭𐪮𐪯𐪰𐪱𐪲𐪳𐪴𐪵𐪶𐪷𐪸𐪹𐪺𐪻𐪼𐪽𐪾𐪿𐫀𐫁𐫂𐫃𐫄𐫅𐫆𐫇𐫈𐫉𐫊𐫋𐫌𐫍𐫎𐫏𐫐𐫑𐫒𐫓𐫔𐫕𐫖𐫗𐫘𐫙𐫚𐫛𐫜𐫝𐫞𐫟𐫠𐫡𐫢𐫣𐫤𐫦𐫥𐫧𐫨𐫩𐫪𐫫𐫬𐫭𐫮𐫯𐫰𐫱𐫲𐫳𐫴𐫵𐫶𐫷𐫸𐫹𐫺𐫻𐫼𐫽𐫾𐫿𐬀𐬁𐬂𐬃𐬄𐬅𐬆𐬇𐬈𐬉𐬊𐬋𐬌𐬍𐬎𐬏𐬐𐬑𐬒𐬓𐬔𐬕𐬖𐬗𐬘𐬙𐬚𐬛𐬜𐬝𐬞𐬟𐬠𐬡𐬢𐬣𐬤𐬥𐬦𐬧𐬨𐬩𐬪𐬫𐬬𐬭𐬮𐬯𐬰𐬱𐬲𐬳𐬴𐬵𐬶𐬷𐬸𐬹𐬺𐬻𐬼𐬽𐬾𐬿𐭀𐭁𐭂𐭃𐭄𐭅𐭆𐭇𐭈𐭉𐭊𐭋𐭌𐭍𐭎𐭏𐭐𐭑𐭒𐭓𐭔𐭕𐭖𐭗𐭘𐭙𐭚𐭛𐭜𐭝𐭞𐭟𐭠𐭡𐭢𐭣𐭤𐭥𐭦𐭧𐭨𐭩𐭪𐭫𐭬𐭭𐭮𐭯𐭰𐭱𐭲𐭳𐭴𐭵𐭶𐭷𐭸𐭹𐭺𐭻𐭼𐭽𐭾𐭿𐮀𐮁𐮂𐮃𐮄𐮅𐮆𐮇𐮈
```


字符串处理方法

"方法"在python中是一个专有名词

- "方法"特指<a>.()风格中的函数()
- 方法本身也是函数，但与<a>有关，<a>.()风格使用
- 字符串或字符串变量是<a>，存在一些可用方法

1. `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`"
2. `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'`]
3. `str.count(sub)` 返回子串sub在str中出现的次数。"`an apple a`
 `day`".`count("a")` 结果为 4

字符串处理方法

4. `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"`
5. `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====='`
6. `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"`
7. `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>)

（槽）机制

```
2 print("{0}:计算机{1}的CPU占用率为{2}%".format("2018-10-10","C",10))  
3 print("{2}:计算机{0}的CPU占用率为{1}%".format("2018-10-10","C",10))
```

控制台 1/A

```
In [60]: runfile('C:/Users/Administrator/Desktop/test/aaaa.py', wdir='C:/Users:  
2018-10-10:计算机C的CPU占用率为10%  
10:计算机2018-10-10的CPU占用率为C%
```

字符串型变量

字符串格式化

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, %

```
1  #%%1
2  print("{0:=^20}".format("PYTHON"))
3  print("{0:*>20}".format("BIT"))
4  print("{0:=^20}{1:*>20}".format("PYTHON", 'BIT'))
5
6  print("{:10}".format("BIT"))
7  print("{0:,.2f}".format(12345.6789))
8  print("{0:b},{0:c},{0:d},{0:o},{0:x},{0:X}".format(425))
9  print("{0:e},{0:E},{0:f},{0:%}".format(3.14))
```

控制台 1/A

```
In [65]: runfile('C:/Users/Administrator/Desktop/test/aaaa.py', wc
=====PYTHON=====
*****BIT
=====PYTHON=====*****BIT
BIT
12,345.68
110101001,Σ,425,651,1a9,1A9
3.140000e+00,3.140000E+00,3.140000,314.000000%
```

字符串回顾

- 正向递增序号、反向递减序号、<字符串>[M:N:K]
 - +、*、in、len()、str()、hex()、oct()、ord()、chr()
 - .lower()、.upper()、.split()、.count()、.replace()
 - .center()、.strip()、.join()、.format()格式化
-



二 时间和日期 (time/datetime) 变量





时间和日期（time/datetime）变量

time库（python标准库）

- 计算机时间的表达
- 提供获取系统时间并格式化输出功能
- 提供系统级精确计时功能，用于程序性能分析

用法：

```
import time  
time.<b>()
```

三类主要函数：

- 时间获取： time() ctime() gmtime()
- 时间格式化： strftime() strptime()
- 程序计时： sleep() perf_counter()

时间和日期（time/datetime）变量

time库（时间获取）

time()

获取当前时间戳，即计算机内部时间值，浮点数

ctime()

获取当前时间并以易读方式表示，返回字符串

gmtime()

获取当前时间，表示为计算机可处理的时间格式

```
1  #%%1
2  import time
3  print(time.time())
4  print(time.ctime())
5  print(time.gmtime())
```

控制台 1/A

```
In [69]: runfile('C:/Users/Administrator/Desktop/
test/aaaa.py', wdir='C:/Users/Administrator/
Desktop/test')
1631437043.1360924
Sun Sep 12 16:57:23 2021
time.struct_time(tm_year=2021, tm_mon=9,
tm_mday=12, tm_hour=8, tm_min=57, tm_sec=23,
tm_wday=6, tm_yday=255, tm_isdst=0)
```


时间和日期（time/datetime）变量

time库（时间格式化）

将时间以合理的方式展示出来

- 格式化：类似字符串格式化，需要有展示模板
- 展示模板由特定的格式化控制符组成
- strftime()方法

strftime(tpl, ts)

tpl是格式化模板字符串，用来定义输出效果

ts是计算机内部时间类型变量

```
1  #%%1
2  import time
3  t = time.gmtime()
4  print(time.strftime("%Y-%m-%d %H:%M:%S",t))
5
```

控制台 1/A

```
In [72]: runfile('C:/Users/Administrator/Desktop/test/
2021-09-12 09:11:06
```

时间和日期（time/datetime）变量

time库（时间格式化）

格式化字符串	日期/时间说明	值范围和实例
%Y	年份	0000~9999, 例如: 1900
%m	月份	01~12, 例如: 10
%B	月份	名称 January~December, 例如: April
%b	月份	名称缩写 Jan~Dec, 例如: Apr
%d	日期	01~31, 例如: 25
%A	星期	Monday~Sunday, 例如: Wednesday
%a	星期	缩写 Mon~Sun, 例如: Wed
%H	小时（24h制）	00~23, 例如: 12
%I	小时（12h制）	01~12, 例如: 7
%p	上/下午	AM, PM, 例如: PM
%M	分钟	00~59, 例如: 26
%S	秒	00~59, 例如: 26

时间和日期（time/datetime）变量

time库（时间计时器）

- 程序计时指测量起止动作所经历时间的过程
- 测量时间：perf_counter()
- 产生时间：sleep()

perf_counter()

返回一个CPU级别的精确时间计数值，单位为秒

由于这个计数值起点不确定，连续调用差值才有意义

```
1  #%%1
2  import time
3  start = time.perf_counter()
4  sum=0
5  for i in range(10000):
6      i=sum+i
7  #%%2
8  end = time.perf_counter()
9  print(end-start)
```

控制台 1/A

```
In [80]: runfile('C:/Users/Administrator/De:
0.0020936999935656786
```

```
In [81]: runcell('1', 'C:/Users/Administrato
```

```
In [82]: runcell('2', 'C:/Users/Administrato
4.54030259999854
```

时间和日期（time/datetime）变量

time库（时间计时器）

sleep(s)

s拟休眠的时间，单位是秒，可以是浮点数

```
1  #%%1
2  import time
3  def wait():
4      time.sleep(3.3)
5  start = time.perf_counter()
6  wait()
7  end = time.perf_counter()
8  print(end-start)
```



控制台 1/A



```
In [85]: runfile('C:/Users/Administrat
3.3103827000013553
```

时间和日期（time/datetime）变量

datetime库

datetime模块定义了5个类，分别是

- 1.datetime.date: 表示日期的类
- 2.datetime.datetime: 表示日期时间的类
- 3.datetime.time: 表示时间的类
- 4.datetime.timedelta: 表示时间间隔，即两个时间点的间隔
- 5.datetime.tzinfo: 时区的相关信息

```
from datetime import date  
from datetime import datetime  
from datetime import time  
from datetime import timedelta  
from datetime import tzinfo
```

时间和日期（time/datetime）变量

datetime库

date类有三个参数,datetime.date(year,month,day), 返回year-month-day

方法:

- 1.datetime.date.ctime(),返回格式如 Sun Apr 16 00:00:00 2017
- 2.datetime.date.fromtimestamp(timestamp),根据给定的时间戳, 返回一个date对象; datetime.date.today()作用相同
- 3.datetime.date.isocalendar():返回格式如(year, month, day)的元组,(2017, 15, 6)
- 4.datetime.date.isoformat(): 返回格式如YYYY-MM-DD
- 5.datetime.date.isoweekday(): 返回给定日期的星期 (0-6) 星期一=0, 星期日=6 这里表明下python3中是从[1-7]表示的 就是本来是星期几现在显示就是星期几
- 6.datetime.date.replace(year,month,day): 替换给定日期, 但不改变原日期
- 7.datetime.date.strftime(format):把日期时间按照给定的format进行格式化。
- 8.datetime.date.timetuple(): 返回日期对应的time.struct_time对象
time.struct_time(tm_year=2017, tm_mon=4, tm_mday=15, tm_hour=0, tm_min=0, tm_sec=0, tm_wday=5, tm_yday=105, tm_isdst=-1)
- 9.datetime.date.weekday(): 返回日期的星期

```
2 from datetime import date  
3 print(date(2021,11,3))
```

控制台 1/A

In [94]: runfile('C:/Users/Administrator/
2021-11-03

时间和日期（time/datetime）变量

datetime库

其它类的使用方法与time库大同小异，可以自行对照参考文档学习。



三 实例



实例（进度条）

```
1  #%%1
2  import time
3  scale = 10
4  print("-----执行开始-----")
5  for i in range(scale+1):
6      a = '*' * i
7      b = '.' * (scale - i)
8      c = (i/scale)*100
9      print("{:^3.0f}%[{}->{}]".format(c,a,b))
10     time.sleep(0.1)
11 print("-----执行结束-----")
```

控制台 1/A

```
In [97]: runfile('C:/Users/Administrator/Desktop/test/aa
-----执行开始-----
 0 %[->.....]
10 %[*->.....]
20 %[*->.....]
30 %[*->.....]
40 %[*->.....]
50 %[*->.....]
60 %[*->.....]
70 %[*->.....]
80 %[*->.....]
90 %[*->.....]
100%[*->.....]
-----执行结束-----
```

```
1  #%%1
2  import time
3  for i in range(101):
4      print("\r{:3}%".format(i), end="")
5      time.sleep(0.1)
```

控制台 1/A

```
In [100]: runfile('C:/Users/Administrator/Desktop
35%
```

实例（进度条）

```
1  ###
2  import time
3  scale = 50
4  print("执行开始".center(scale//2, "-"))
5  start = time.perf_counter()
6  for i in range(scale+1):
7      a = '*' * i
8      b = '.' * (scale - i)
9      c = (i/scale)*100
10     dur = time.perf_counter() - start
11     print("\r{:^3.0f}%[{}->{}][:.2f]s".format(c,a,b,dur),end='')
12     time.sleep(0.1)
13     print("\n"+"执行结束".center(scale//2, '-'))
```

控制台 1/A

```
In [103]: runfile('C:/Users/Administrator/Desktop/test/aaaa.py', wdir='C:/Users/Adm:
-----执行开始-----
46 %[*****->.....]2.41s
```

```
import time
scale = 50
print("执行开始
".center(scale//2, "-"))
start = time.perf_counter()
for i in range(scale+1):
    a = '*' * i
    b = '.' * (scale - i)
    c = (i/scale)*100
    dur = time.perf_counter() -
start
    print("\r{:^3.0f}%[{}-
>{}][:.2f]s".format(c,a,b,dur),en
d='')
    time.sleep(0.1)
print("\n"+"执行结束
".center(scale//2, '-'))
```



感谢参与 下堂课见

