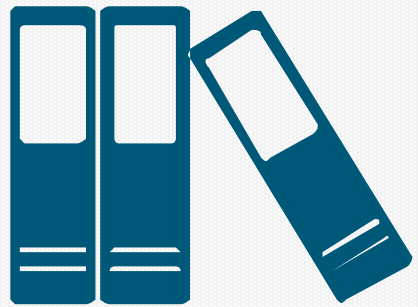
第2讲 Python语法基础

编写简单的程序

万丈高楼平地起

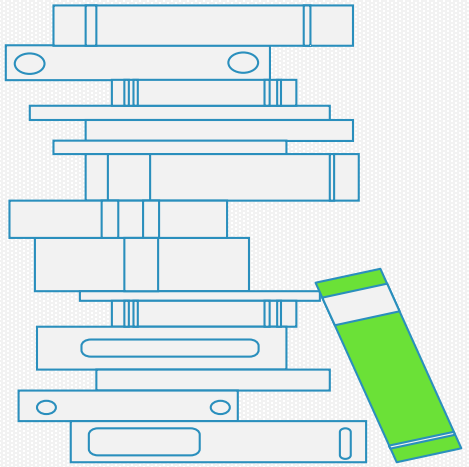


实验过程中问题汇总

- .py 双击运行无法查看结果 – 添加暂停语句或用解释器打开后运行
- = 两侧用不用加空格 - PEP8 建议加，但并不是必须的
- invalid character in identifier – “标识符中的无效字符”,首先检查有没有字符是中文的
- int 和 eval 有什么区别 - int 是单纯类型转换，eval 是尝试把参数作为 python 语句来解释执行

00

一个简单的小程序



引例：从键盘上输入三角形的三条边长，输出其面积。

分析：由用户输入三个数（做为三边构成三角形），求三角形的面积并输出。

Step 1：获取数据

Step 2：计算面积

Step 3：输出结果

Step 1：从键盘分三次输入三个数a、b、c

Step 2：d = (a+b+c)/2

$$s = \sqrt{d*(d-a)*(d-b)*(d-c)}$$

Step 3：输出area

程序：

```
#TriangleArea.py
import math
a = eval(input("请输入三角形的第一条边：  "))
b = eval(input( “请输入三角形的第二条边：  "))
c = eval(input( “请输入三角形的第三条边：  "))
d = (a+b+c)/2
area = math.sqrt(d*(d-a)*(d-b)*(d-c)) # 求面积
print("三角形的面积为：{:.2f}".format(area))
```

试着读一下，能理解吗？

这段程序中涉及到：
数据类型、
常量与变量、
数据的输入与输出、
数值运算等，
本节将对这些基本要素进行学习。

本章主要内容

01 数据类型

02 常量和变量

03 基本输入与输出方法

04 数值运算符与表达式

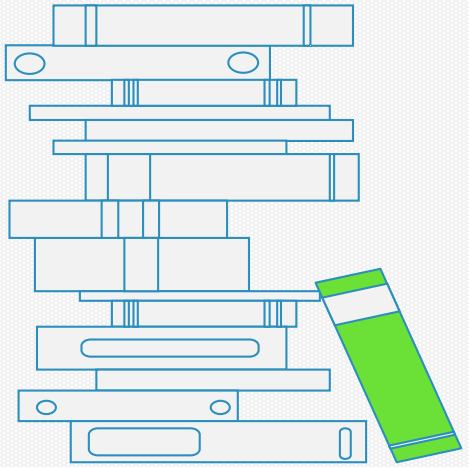
05 字符串操作

01

数据类型

为什么会有数据类型的概念？Python中有哪些数据类型？基本数据类型有哪些？

本小节为了解性内容



数据为什么需要分类？

- **在日常生活中的数据都是区分类型的，不同类型的数据应用场景不同，比如：**
  - 年龄，数量，年，月，日等都是整数
  - 价钱，身高，体重，成绩等都是实数
  - 姓名，地址，性别等都是一些字符串

  计算机能处理的远不止数值，还可以处理文本、图形、音频、视频、网页等各种各样的数据，不同的数据，需要定义不同的数据类型。

- **不同类型的数据有不同的：**
  - 表示形式　123：数字　，　“123”：字符串
  - 可参与的运算种类　123 – 23，　“123” – 23 (错误)

- **Python程序设计语言通过数据的形式表达“数据类型”，比如：**
  - 123表示数字123，　“123”则表示文本字符串123

Python中的数据类型：基本类型 + 组合类型

1. **Number(数值)**
   - **int（整数类型）**
   - **float（浮点数类型）**
   - **complex（复数类型）**
2. **Boolean（布尔）：bool**
3. **String（字符串）： str**
4. List（列表）
5. Tuple(元组)
6. Set（集合）
7. Dictionary（字典）

- type 函数可以返回给定数据的数据类型

```
print(type(2021))
print(type(1.0))
print(type(1+2j))
print(type(True))
print(type('hello'))
d = input("pls input ur date:")
print(type(d))
d = int(d)
print(type(d))
print(type([1,2,3]))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'bool'>
<class 'str'>
pls input ur date: 2021
<class 'str'>
<class 'int'>
<class 'list'>
```

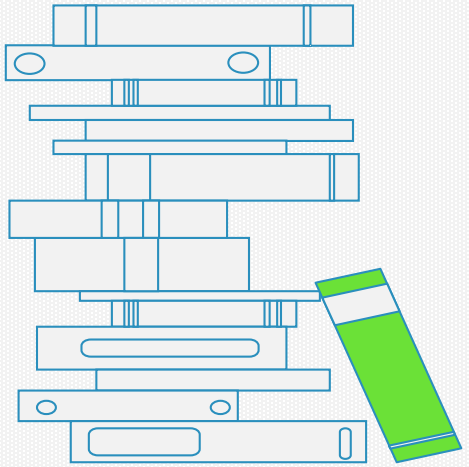
02

常量和变量

数据有哪些表现形式呢？常量和变量

变量的命名与赋值

本小节为了解性内容



数据的表现形式

- 根据数据在程序运行过程中其值是否可变，可以将数据分为常量和变量两种类型

常量 constant

程序运行过程中其值不变的量；也是程序中直接使用的数据。
也称为字面量（literal）

常量

示例	类型
-9，0b1001（二进制），0o17（八进制），0x2f（十六进制）	整数
True False	布尔
2.13, 4.0, 1.2e-6, 3.29E4	浮点数
‘V’，“2022”，'''COVID19'''，'中国\n加油’，''	字符串
[1, 4, 'F', 3.4]	列表
{'a': 3, 'b': 9, 'c': 1}	字典
(1, '23', 'name')	元组
{2, 3, 1, 5}	集合

整数 int

- 无大小限制（只受限制于计算机内存）
- 默认情况下，采用十进制表示，但可以使用其它前导符号表达不同进制

  -9，0b1001（二进制），0o17（八进制），0x2f（十六进制）都是合法的表达形式
- bin() ,oct() 或hex()函数可以将数据转换为其它进制的带有前缀的字符串
- format(x,'o/b/x')函数则可以转换为不带前缀的字符串，例如 format(12,'x') 返回 c
- int() 则可以将字符串转换为十进制整数，其第二个参数可以指定原始数据的进制，例如 int('1010',2) 转换为 10

布尔 bool

- 只有两个值，True 和 False，直接书写，无需定界符，但需注意大小写
- 布尔型数据可以转换为 int 参与算术运算，可以用 int() 函数转换，也可以隐式转换直接参与运算，True 转换为 1，False 转换为 0 ，例如 print(True+1) 返回 2
- bool() 函数可以将整数转换为布尔数据类型，0 为 False，非零值为 True，例如 print(bool(5)) 为 True

浮点数 float

- 即由整数部分和小数部分构成的数，取值范围和精度都有限制（精度与系统相关）。
- 两种表示方法：
  - 小数形式： -34.6、0.346、346.0
  - 科学计数法形式：3.46e-1、1E-6 、3e5
- 浮点数默认保留 17 位有效数字

浮点数计算存在误差

- **浮点数间运算存在不确定尾数，有误差（不是bug），**故不能直接进行等与不等的判定。

```
>>> 0.1 + 0.3
0.4
>>> 0.1 + 0.2
0.30000000000000004
```

不确定尾数

```
>>> round(0.1+0.2,1)
0.3
>>> 0.1 + 0.2 == 0.3
False
>>> fabs(0.1+0.2-0.3) <= 1e-6
True
```

- 不确定尾数一般发生在10^{-16}左右
- 浮点数间运算与比较可用round(x,d)函数辅助
  round(x, d)：对x四舍五入，d是小数截取位数

字符串 string

- 由 0 个或多个字符组成的字符序列，由于变量名也是由字符构成的，为了避免冲突，字符串书写时需要使用定界符括起来，表明这是字符序列而不是某个量的名字。
- 定界符有单引号、双引号、三单引号和三双引号，适用场景不同，注意都是英文符号
  - 如果字符串里有单引号，则可以用双引号做定界符，
  - 同理，如果字符串中包含双引号，则可以用单引号做定界符，
  - 如果字符串包含多行，则可以使用三引号，三引号包含的字符串也可以作为注释出现。
- 定界符连写表明这是一个空字符串，即不包含任何字符的字符串，例如 ''

转义字符 Escape Character

- 指 ASCII 表中无法直接在屏幕显示和键盘输入的字符,或者已经在 Python 中有特定含义和作用的字符，比如换行符、TAB键、单引号等，可以使用字母的转义形态来表达。
- 转义形态以 \ 开始，后跟一个字母或者数字来表示一个字符，例如 \n 表示换行符，\' 表示单引号

常见转义字符

转义字符	意　义	ASCII码值（十进制）
\a	响铃(BEL)	007
\b	退格(BS)	008
\f	换页(FF)	012
\n	换行(LF)	010
\r	回车(CR)	013
\t	水平制表(HT)	009
\v	垂直制表(VT)	011
\\	反斜杠	092
\?	问号字符	063
\'	单引号字符	039
\"	双引号字符	034
\0	空字符(NULL)	000
\ddd	任意字符	三位八进制
\xhh	任意字符	二位十六进制

常见转义字符解释

- \r 回车 return,表明输入输出的结束,光标移动到第一列
- \n 新行 newline,移动光标到下一行。
- \b 回删 backspace ,光标向左移动一列,新的输出会替换原来的输出。
- \t 水平制表 tab ,表现为光标跳跃到下一个制表位(每 8 列为一个制表位),可以实现输出对齐。
- \',\'',\\,\? 用于在字符串中表示 ','',\和?
- \0 ascii 值为 0 的一个空字符，不存在。

字符串示例

```
>>> print("You're the best!")
 You're the best!
>>> print('You\'re the best!')
 You're the best!
>>> print("You're the \"best\"")
 You're the "best"
>>> print('''I sat there with sally.
    We sat there, we two.
    And i said, 'how i wish
    we had something to do!' ''')
 I sat there with sally.
We sat there, we two.
And i said, 'how i wish
we had something to do!'
```

单选题 1分

设置

下列不属于Python中的数据类型是

A 整型

B 货币型

C 列表

D 字符串

提交

变量

程序运行过程中其值可以发生改变的量；

变量是用来保存和表示数据的符号。

注意：在Python中，不仅可以改变变量的值，还可以改变其类型。

```
#TriangleArea.py
import math
a = eval(input("请输入三角形的第一条边："))
b = eval(input( “请输入三角形的第二条边："))
c = eval(input( “请输入三角形的第三条边："))
d = (a+b+c)/2
area = math.sqrt(d*(d-a)*(d-b)*(d-c))
print("三角形的面积为：{:.2f}".format(area))
```

变量名 variable name

- **变量需要一个名字，遵循标识符(标识某个实体的符号)的命名规则：** a，area
- **标识符的命名规则：**标识符是以字母、数字、下划线和中文**等字符组成**，首字符不能是数字。

  如: my_name，yourAge，x2，计算机课　　(123Python不合法)
- **变量命名尽量做到**“见名知义”、驼峰式，以增加程序的可读性

  如：num，area，myFirstName

注意事项: 大小写敏感、不与保留字、内置函数相同

如: num、Num和NUM是不同变量；　　*if* 是保留字，If 是变量

保留字/关键字 keyword

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

```
>>> import keyword
>>> 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']
>>>
```

变量的创建

- **变量在第一次被赋值时创建**，Python 中使用 = 作为赋值运算符

  变量名 = 表达式

  - m = 2 即创建了值为 2 的变量 m
  - birthyear = input('pls input birthyear:') 也创建了 birthyear 变量
- 变量在使用前，必须先进行赋值创建，否则会提示 NameError
- Python中的**变量类型是由赋值的数据来决定**，而且类型是可变的
- **赋值运算符左侧只能是变量**，而不能是表达式，形如 a-2 = 3 的表达式是错误的

```
>>> m=2
>>> type(m)
<class 'int'>
>>> m=2.6
>>> type(m)
<class 'float'>
>>> m='你好'
>>> type(m)
<class 'str'>
```

m = 2 的赋值过程到底做了什么?

- 与其它高级语言不同，Python 的赋值操作解释为绑定操作更容易理解
- m = 2，Python解释器首先创建了一个整数对象，其值为 2 ，然后创建了一个变量名 m ，并将 m 与 2 进行了绑定操作，或者可以认为 m 指向了数值对象 2 。
- 如果后续 m = 3 ，则可以认为是创建了对象 3，然后让 m 重新绑定(指向) 3 这个对象，而不能简单的认为是将 m 的值改为了 3 。
- Python 的解释器维护着数据和变量名集合的对应关系。
- id() 函数会返回变量名指向的数据的内存地址。

变量名和值对应关系的演示

```
m=2
id(m)
```

```
4557007552
```

```
m=3
id(m)
```

```
4557007584
```

```
n=2
id(n)
```

```
4557007552
```

可以注意到指向的内存发生了变化

可以注意到复用了值 2

单选题 1分

设置

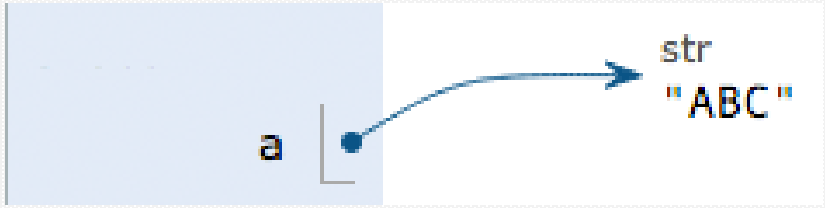
```
a = 'ABC'
b = a
a = 20
print(b)
```

A ABC

B 20

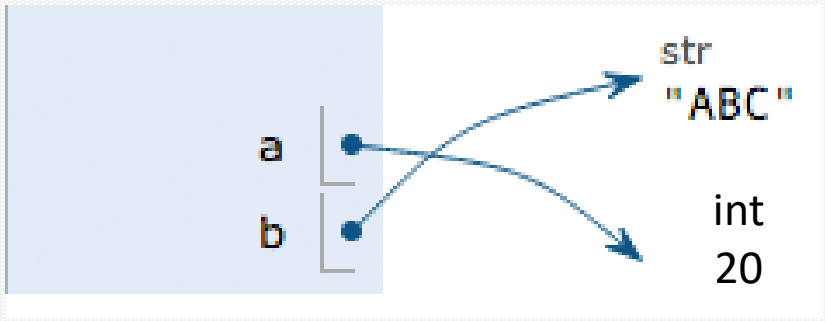
提交

```
a = 'ABC'
b = a
a = 20
print(b)
```



a = 'ABC’

b = a



a = 20

链式赋值 - 同一个值赋给多个变量

变量1 = 变量2 =表达式

```
>>> x=y=z=200
>>> print(x,y,z)
200 200 200
```

解包赋值 - 将序列分解到不同变量

变量1, 变量2 = 表达式1, 表达式2

```
>>> a,b=100,200
>>> print(a,b)
100 200
>>> a,b,c=100,200
Traceback (most recent call last):
  File "<pyshell#24>", line 1, in <module>
    a,b,c=100,200
ValueError: not enough values to unpack (expected 3, got 2)
```

变量的个数必须与序列的元素个数一致

解包赋值应用 - 交换两个数据

其它语言的常规方法

```
a = 100
b = 200
t = a
a = b
b = t
```

Python的同时赋值方法

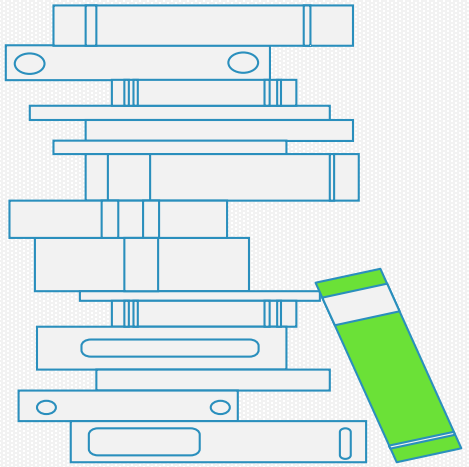
```
a, b = 100, 200
a, b = b, a    #一条语句交换a,b的值
```

03

数据的输入与输出

输入函数input()

输出函数print()



输入函数input()

- **input()**是内置函数，接受用户的键盘输入，读入一个字符串
  - **变量 = input ()**：没有提示输入文本的显示
  - **变量 = input (“提示字符串”)**：有提示输入文本的显示

```
>>> course = input("请输入你的计算机课程：")
请输入你的计算机课程：Python
>>> print('Hello',course)
Hello Python
```

- 无论用户输入什么，都一律做为字符串对待
  - 根据需要，可使用内置函数int、float、eval，list等转换到特定类型

input 示例

```
>>>  m=input('请输入一个整数:')
请输入一个整数:3
>>> print(m)
3
>>> print(m*m)
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'str'
```

eval()函数

- 其参数为字符串，作用：去掉参数最外侧的引号，解析为表达式并求值

变量 = eval (input ("提示字符串"))

```
>>> m=eval(input('请输入一个整数:'))
请输入一个整数:3
>>> print(m*m)
9
```

单纯input语句只能得到**字符串类型**，如果希望得到数字，可以用**eval( )**强制转换成**数值类型**。

eval()

- 不同于 int()等特定类型转换函数，eval() 函数试图把传递给他的字符串作为 python 语句来进行解释并执行，如果传递给它一个字符串名字，它就会试图把其看做是变量；如果是一个数值，则尝试转换为对应的数据类型；如果是一个表达式，则会尝试计算。。。
- 它功能广泛，但同时也较为危险

```
[76]: d = eval(input("pls input some data:"))
      print(d)

      pls input some data: birthdate
      ---------------------------------------------------------------------------
      NameError                                 Traceback (most recent call last)
      <ipython-input-76-076262ae0203> in <module>
      ----> 1 d = eval(input("pls input some data:"))
            2 print(d)

      <string> in <module>

      NameError: name 'birthdate' is not defined

[77]: d = eval(input("pls input some data:"))
      print(d)

      pls input some data: 2021
      2021

[78]: d = eval(input("pls input some data:"))
      print(d)

      pls input some data: 1+1
      2

[80]: d = eval(input("pls input some data:"))

      pls input some data: print('hello')
      hello
```

体会eval的强大

问题2-1：获得用户输入的一个字符串，格式如下：

M OP N

其中，M和N是任何数字，OP代表一种操作，表示为如下四种：+, -, *, /（加减乘除），输出M OP N的运算结果。

注意：M和OP、OP和N之间可以存在多个空格，不考虑输入错误情况。

```
1 value = eval(input())
2 print(value)
```

Shell

```
>>> %Run 1.py
1 + 2
3
>>> %Run 1.py
4 - 1
3
>>> %Run 1.py
9 * 5
45
>>> %Run 1.py
8 / 2
4.0
```

输出函数print()

- print()是内置函数，用于显示信息
- 格式：print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
  - value：用逗号分隔的一个或多个输出项
  - sep：多个输出项之间的分隔符，默认为空格
  - end：最后一个输出项后的结束符，默认为换行
  - file：指定输出位置，默认是控制台
  - flush：是否立即刷新输出缓冲区

print 的输出项

- 若为常量，原样输出；print(“hello”)　#hello
- 若为变量名，则输出变量绑定的对象的值　x = 1; print(x)　#1
- 若为表达式，则先计算表达式的值，而后再输出；若有多个表达式，从左到右求值，以从左到右的方式显示.
  - print(2+2, 2-2, 2*2, 2/2) #4 0 4 1.0

print 实例

```
>>> print(3+5)
8
>>> print('a=',100)
a= 100
>>> print()
```

```
print(2021,10,1,sep='/')
```

```
2021/10/1
```

```
print(3)
print(4)
print("the answer is ", end="")
print(3+4)
```

```
3
4
the answer is 7
```

print 实例

```
>>> print(3+5)
8
>>> print('a=',100)
a= 100
>>> print()
```

```
print(2021,10,1,sep='/')
2021/10/1
```

什么参数都没有的，输出一个空行

```
print(3)
print(4)
print("the answer is ", end="")
print(3+4)
```

```
3
4
the answer is 7
```

end= “” ，表示print输出完数据后不加任何字符，包括换行符。

问题2-2 输入两个数，互换位置后输出。

```
#swap.py

a,b = eval(input("请输入两个整数：  "))
print("a=",a," b=",b)
a,b = b,a
print("a=",a," b=",b)
```

用Python的解包赋值实现

```
t = a
a = b
b = t
```

其它语言的常规方法

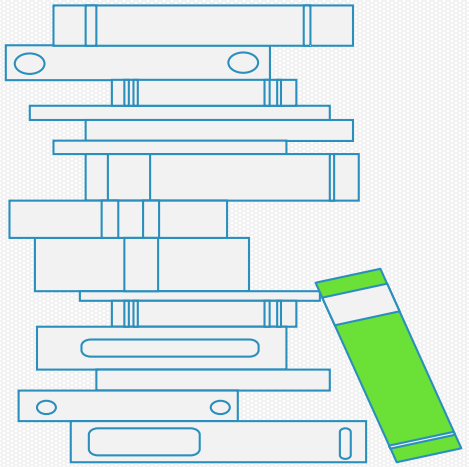
04

数值运算

基本的数值运算符有哪些？表示方法，含义，优先级

数值运算符和赋值运算符结合形成复合的赋值运算符

本节内容为重点内容，包含操作和习题



运算符 operator

- 运算符是进行各类运算的符号
- 常见的运算符有:
  - 算术运算符: + - * / % // **
  - 关系运算符: > >= < <= == !=
  - 逻辑运算符: not and or
  - 赋值运算符: =
  - 复合赋值运算符: += -= *= /= //= **=
  - 位运算符: & | ^ ~
  - 注: 本节只讲授算术运算符

运算符的一些知识点

- 按照操作数的个数分为单目和双目运算符，例如：加法运算符+两边有两个运算数，所以+为双目运算符
- 优先级 precedence
  - 算术 > 关系 > 逻辑 > 赋值
- 结合性 associativity
  - 左结合：大多数双目（双目中除了赋值外都是左结合性）
  - 右结合：单目、赋值运算符

表达式 expression

- 由上述运算符配合各类数据以及圆括号组成的式子
  - 例如：2021-birthyear+1
- 圆括号可以嵌套，没有大括号和中括号，括号内部表达式为一个整体，优先进行运算，如果有嵌套，则内部括号优先级高。
  - 例如：d = (-b+sqrt(b*b-4*a*c))/(2*a)
- 表达式的运算结果取决于参与运算的数据的数据类型，通常会向精度高的方向转换
  - 例如：a = 1.2/2 的结果会是 float

数值运算符

先乘方，再 乘、除、取余，最后加减

运算符	描述	实例	结果	备注
+	加法	3 + 4.5	7.5	
-	减法	12 – 4.8	7.2	
*	乘法	2 * 5.0	10.0	不同于数学中的写法，“*”不可以省略 例如：m = 4ab，必须写成 m = 4*a*b
/	浮点数除法	10 / 4	2.5	
//	**整数除法**	10 // 4	2	采用向左取整方式，-10 // 4 = -3
%	取余	10 % 3	1	操作数可以为实数，3.5 % 3 = 0.5
**	乘方	2 ** 3	8	操作数可以为实数，4.0 ** 0.5 = 2.0

问题2-3：编写一个程序，需要找钱给用户，现在只有50元，5元和1元的人民币若干张。输入一个整数金额值，给出找钱的方案，假设人民币足够多，且优先使用面额大的钱币。

```
money = eval(input("输入金额: "))
m50 = money // 50      # 计算需要的50元面额的纸币数量
money = money % 50     # 使用50元面额钱币后剩下的金额
m5 = money // 5
money = money % 5
m1 = money
print("50元面额需要的张数: ", m50)
print("5元面额需要的张数: ", m5)
print("1元面额需要的张数: ", m1)
```

```
输入金额: 285
50元面额需要的张数:  5
5元面额需要的张数:  7
1元面额需要的张数:  0
```

示例

- 将645分钟转换为小时和分钟

```
>>> minutes = 645
>>> hours = minutes / 60
>>> hours
10.75
```

```
>>> minutes = 645
>>> hour = minutes // 60
>>> minute = minutes % 60
>>> hour
10
>>> minute
45
```

复合赋值运算符与表达式

- 复合赋值运算符 +=、-=、*=、/=、//=、%=、**=
  - 复合赋值运算符中间不可有空格。
  - 若a和b为操作数，则a += b，等价于a = a + b

    a *= b，等价于a = a * b

```
>>> a,b=10,20
>>> a+=b
>>> print(a,b)
30 20
```

单选题 1分

设置

```
x , y = 5, 3
x *=   (y+2)
y ** = 2
print(x, y)
```

A 17 9

B 25 5

C 25 9

D 17 3

提交

Python内置数值运算函数

函数	描述
abs(x)	求x的绝对值；若x为复数，则为求其模 abs(-10.01) 结果为 10.01
divmod(x,y)	商余，输出(x//y, x%y) divmod(10, 3) 结果为 (3, 1)
pow(x,y[,z])	(x**y)%z, []表示可选参数，当z省略的时候，等价于x**y pow(3, pow(3, 99), 10000) 结果为 4587
round(x[,d])	对x四舍五入，保留d位小数，当d省略时返回x四舍五入后的整数值 round(-10.123, 2) 结果为 -10.12
max(x_1,x_2,……,x_n)	返回x_1,x_2,……,x_n中的最大值 max(1, 9, 5, 4, 3) 结果为 9
min(x_1,x_2,……,x_n)	返回x_1,x_2,……,x_n中的最小值 min(1, 9, 5, 4, 3) 结果为 1

divmod 示例

- 将645分钟转换为小时和分钟

```
>>> minutes = 645
>>> hour, minute = divmod(minutes, 60)
>>> hour, minute
(10, 45)
```

- 将出生年月中的年和月分解

```
birth = 200012
year,month = divmod(birth,100)
print(year,month)
```

```
2000 12
```

```
>>> abs(3 + 4j)
5.0
>>> abs(-2.5)
2.5
>>> max(1, -3, 3.2)
3.2
>>> min(1, -3, 3.2)
-3
>>> pow(4, 0.5)
2.0
>>> pow(2, 3)
8
>>> divmod(28, 12)
(2, 4)
>>> round(2.342, 2)
2.34
>>> round(-1.57, 1)
-1.6
```

问题2-4：从键盘上输入三角形的三条边长，输出其面积。（假设三边长数据合理，能构成三角形）

```
#TriangleArea.py
import math
a = eval(input("请输入三角形的第一条边："))
b = eval(input( “请输入三角形的第二条边："))
c = eval(input( “请输入三角形的第三条边："))
d = (a+b+c)/2
area = math.sqrt(d*(d-a)*(d-b)*(d-c))
print("三角形的面积为：{:.2f}".format(area))
```

数值运算标准函数库 math

- 当内置函数不能够满足要求时，可以使用标准库 math 中提供的常数及数学函数
- 由于 math 库并不是内置模块，故需要先使用 import 关键字导入其中的对象和函数才能够使用
- 提供了4个数学常数和44个函数。
- 不支持复数类型，仅支持整数和浮点数运算。

- 第二章2.5.3节

math库中的数学常数

常数	数学形式	描述
pi	π	圆周率，值为3.141592653589793
e	e	自然对数，值为2.718281828459045
inf	∞	正无穷大，负无穷大为-inf
nan		非浮点数标记，Not a Number

math库中的部分数值函数

自学掌握

函数	数学形式	描述
factorial(a)	x!	返回a的阶乘，a为整数
fabs(x)	$\|x\|$	返回x的绝对值，浮点数
sqrt(x)	$\sqrt{x}$	返回x的平方根
fsum([x,y,……])	x+y+……	浮点数精确求和
gcd(a,b)		返回x和y的最大公约数，a和b为整数
trunc(x)		返回x的整数部分
modf(x)		返回x的小数和整数部分
ceil(x)	$\lceil x \rceil$	向上取整，返回不小于x的最小整数
floor(x)	$\lfloor x \rfloor$	向下取整，返回不大于x的最大整数

四种常见导入方式

- import math
  - 用法：math.pi　　math.sqrt(4)
- import math as mt
  - 用法：　mt.pi　mt.sqrt(4)
- from math import *
  - 用法：　pi　　sqrt(4)
- **from math import pi,sqrt**
  - 用法：　pi　　sqrt(4)

05

字符串处理

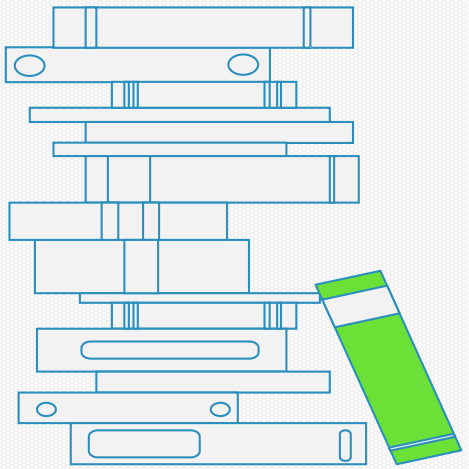
字符串运算符

字符串的索引、切片来获得字符串的一个字符或一段字符串

字符串处理函数和方法

字符串格式化

本节内容为重点内容。



内置的字符串运算符

运算符	描述
+	字符串拼接，例如“AB” + “123”结果为“AB123”
*	字符串复制，例如“Tom” *3，结果为“TomTomTom”
in	判断是否为子串，例如“H” in “Hello”结果为True；“h” in “Hello”结果为False

示例：

```
name = input("姓名：")
country = input("国家：")
s = "世界那么大，" + name + "想去" + country + "看看"
print(s)
```

"+"实现字符串的拼接

```
姓名：  李白
国家：俄罗斯
世界那么大，  李白想去俄罗斯看看
```

字符串的索引与切片

字符串的序号：序号用于标识字符串中的位置, Python中使用两种类型(方向)的编号体系

正向递增序号

0	1	2	3	4	5	6	7	8	9	10	11	12
人	生	苦	短	,	我	学	P	y	t	h	o	n
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

反向递减序号

字符串的索引

- 使用[ ]**运算符配合序号**，获取字符串中**一个**字符
- 利用**索引**值，可以访问字符串的每一个字符。

想一想：
要想获得一个字符串呢？

0	1	2	3	4	5	6	7	8	9
H	e	l	l	o		M	i	k	e
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> s='Hello Mike'
>>> print(s[0],s[-1],s[8],s[-2])
H e k k
```

字符串的切片

1. 用于获取字符串的子串，

2. 区间访问方式，采用**[头下标：尾下标]**的方式

若有字符串s，s[头下标：尾下标]表示在字符串s中取索引值从头下标到尾下标（不包含尾下标）的子字符串，注意这是个半开区间

3. 切片方式中，若头下标缺省，表示从开始取子串；若尾下标缺省，表示取到最后一个字符；若头下标和尾下标均缺省，则取整个字符串。

[:5]——从头到4（第5个字符） [2:]——从2(第3个字符) 到末尾 [:]——整个字符串

切片实例

0	1	2	3	4	5	6	7	8	9
H	e	l	l	o		M	i	k	e
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> s='Hello Mike'
>>> print(s[0:5],s[6:-1])
Hello Mik
>>> print(s[:5],s[6:])
Hello Mike
>>> print(s[:])
Hello Mike
```

字符串的切片

带步长切片

[头下标：尾下标：步长]

牢记：

1. 当步长值大于0的时候，表示从左向右取字符；
2. 当步长值小于0的时候，表示从右向左取字符。
3. 步长：表示每次取字符的间隔

字符串切片

0	1	2	3	4	5	6	7	8	9
H	e	l	l	o		M	i	k	e
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> s='Hello Mike'
>>> print(s[0:5:1])
Hello
>>> print(s[0:6:2])
Hlo
>>> print(s[0:6:-1])

>>> print(s[4:0:-1])
olle
>>> print(s[::-1])
ekiM olleH
>>> print(s[::-3])
eMlH
```

1. s[0:5:1] 正向取
2. s[0:6:2] 正向取，间隔一个字符取
3. s[0:6:-1] 反向取，但是头下标小于尾下标无法反向取，因此输出为空
4. s[4:0:-1] 反向取，索引值为0的字符无法取到
5. s[4::-1] 反向取，从索引值为4的字符依次取到开头字符
6. s[::-1] 反向取整串
7. s[::-3] 反向取，间隔两个字符取

问题2-5：输入一个1-7的整数表示星期几，输出整数对应的星期。

方法一问题分析：可以利用字符串的切片操作来巧妙的解决这个问题

注意：切片起始位置的确定和步长

- 我们的基本思想是将所有的 星期名称存储在一个字符串中。

  weekStr = “星期一星期二星期三星期四星期五星期六星期日”

  这样可以通过切出适当的子字符串来查找星期几，关键是应该在哪里切片呢?
- 由于星期几都由三个字符组成，如果知道给定星期在字符串中开始的位置，就可以很容易地提取星期几：**weekStr[pos ： pos+3]**，这将获得从pos指示位置开始的长度为3的子串。那么如何计算这个pos呢?

```
weekStr = "星期一星期二星期三星期四星期五星期六星期日"
weekId = eval(input("请输入星期数字(1-7): "))
pos = (weekId - 1 ) * 3
print(weekStr[pos: pos+3])
```

方法二：也可以利用字符串的索引和字符串运算符来解决这个问题

```
weekStr = "一二三四五六日"
weekId = eval(input("请输入星期数字(1-7)："))
print("星期" + weekStr[weekId-1])
```

字符串应用实例

- 输入一个1-12的整数，输出对应的月份名称缩写
- 问题分析：
  - 可以利用字符串的切片操作来巧妙的解决这个问题。我们的基本思想是将所有的月份名称缩写存储在一个大的字符串中。months = “JanFebMarAprMayJunJulAugSepOctNovDec”，这样可以通过切出适当的子字符串来查找特定的月份，关键是应该在哪里切片呢？
  - 由于每个月的名称缩写都由三个字母组成，如果知道一个给定月份在字符串中开始的位置，就可以很容易地提取月份名称缩写，monthAbbrev = months[pos：pos+3]，这将获得从pos指示位置开始的长度为3的子串。那么如何计算这个pos呢？

输入整数输出月份缩写

```
m = int(input("输入0-12的整数: "))
months = "JanFebMarAprMayJunJulAugSepOctNovDec"
pos = ( m - 1 ) * 3
print(months[pos:pos+3])
```

```
输入0-12的整数: 6
Jun
```

内置的字符串处理函数

函数	描述
len(x)	返回字符串x的长度
str(x)	将任意类型x所转换为字符串类型
chr(x)	返回Unicode编码为x的字符
ord(x)	返回字符x的Unicode编码值
hex(x)	将整数x转换为十六进制数，并返回其小写字符串形式
oct(x)	将整数x转换为八进制数，并返回其小写字符串形式

```
>>> x = "好好学习，天天向上"
>>> len(x)
9
>>> str(125)
'125'
>>> str(3+5)
'8'
>>> hex(62)
'0x3e'
>>> oct(62)
'0o76'
```

内置的字符串处理方法

- 方法(method)可以看做是专属于某个对象的函数，这类函数使用时需要使用类似于 obj.fun() 的方式来调用。
- Python对字符串对象提供了大量的内置方法用于字符串的检测、替换和排版等操作。
- 使用时需要注意的是，字符串是不可变对象(immutable)，所以涉及字符串“修改”的方法都是返回修改之后的新字符串，不对原字符串做任何修改。

字符串方法

方法及使用	描述
str.find()、 str.rfind()	查找一个字符串在另一个字符串指定范围（默认是整个字符串）中首次和最后一次出现的位置,**如果不存在则返回-1**
str.index(), str.rindex()	来查找一个字符串在另一个字符串指定范围（默认是整个字符串）中首次和最后一次出现的位置，**如果不存在则抛出异常**

字符串查找

- >>> mystr = "bird,fish,cat,bunny,fish"
- >>> mystr.find("fish") #5
- >>> mystr.rfind("fish") # 20
- >>> mystr.find("dog") #-1
- >>> mystr.count("fish") #2

字符串方法

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，不存在则返回0 "an apple a day".count("a") 结果为 4

字符串方法演示

```
>>> s = "I have two big eyes."
>>> s.lower()          # 返回小写字符串
'i have two big eyes.'
>>> s.upper()          # 返回大写字符串
'I HAVE TWO BIG EYES.'
>>> s.capitalize()     # 首字母大写
'I have two big eyes.'
>>> s.title()          # 每个单词首字母大写
'I Have Two Big Eyes.'
>>> s.swapcase()       # 大小写互换
'i HAVE TWO BIG EYES.'
```

字符串方法

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======='

字符串替换示例

- replace()方法用来替换字符串中指定字符或子字符串，类似于Word，记事本等文本编辑器的查找和替换功能。该方法不修改原字符串，而是返回一个新字符串。

```
>>> s = "你是我的小呀小苹果儿"
>>> s.replace("小", "small")
'你是我的small呀small苹果儿'
```

字符串方法

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

字符串连接示例

- 字符串的join()方法用来将列表中多个字符串进行连接，并在相邻两个字符串之间插入指定字符，返回新字符串。

```
>>> li=['apple','banana','pear','peach']
>>> print(':'.join(li))
apple:banana:pear:peach
>>> print('-'.join(li))
apple-banana-pear-peach
```

字符串删除边界字符示例

- 删除字符串两端，右端或左端连续空白字符和指定字符方法：strip()、rstrip()、lstrip()

```
>>> s = "    abc    "
>>> s.strip()          # 删除两端空白字符
'abc'
>>> s.rstrip()         # 删除右端空白字符
'    abc'
>>> s.lstrip()         # 删除左端空白字符
'abc    '
>>> s = "=====Mike====="
>>> s.strip('=')       # 删除两端指定字符"="
'Mike'
```

字符串判定边界示例

- 判断字符串是否以指定字符开始或结束方法：startswith()、endswith()

```
>>> s = "Python程序设计.py"
>>> s.startswith("Python")   # 检测字符串是否以“Python”开始
True
>>> s.endswith("py")          # 检测字符串是否以“py”结束
True
```

字符串判定类型

- 判断字符串类型方法： isupper()、islower()、isdigit、isalnum()、isalpha()

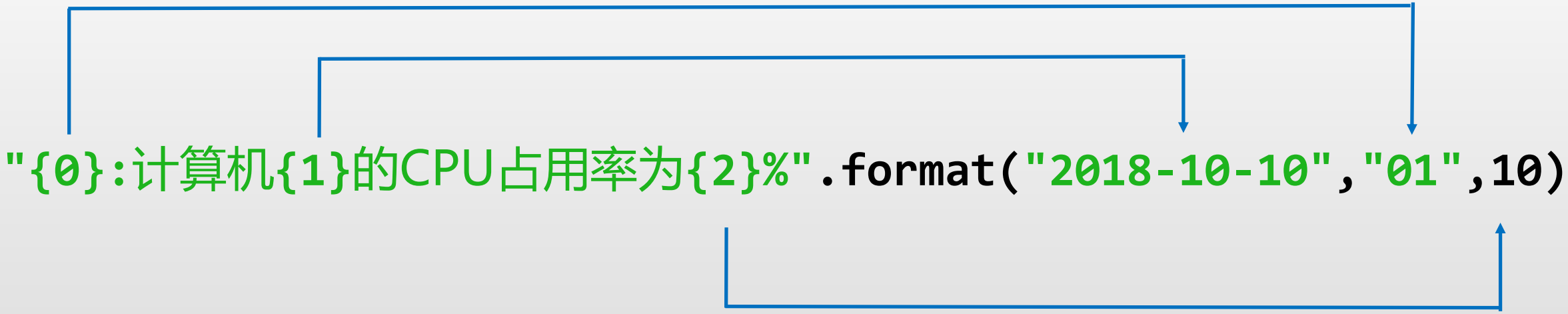
```
>>> s='years'
>>> s.islower()    # 判断字符串是否为全小写
True
>>> s="YEARS"
>>> s.isupper()    # 判断字符串是否为全大写
True
>>> s="20100405"
>>> s.isdigit()    # 判断字符串是否为全数字
True
>>> s="He is 10 years old"      # 字符串s中包含数字字母和空格
>>> s=s.replace(" ", "")        # 去除字符串中的空格
>>> s.isalnum()                 # 判断字符串是否为数字或字母
True
>>> s.isalpha()                 # 判断字符串是否为全字母
False
```

字符串格式化

- 格式化指将字符串转换成固定的样式，适用于规范化或者重复性的输出。
- 可以借由字符串的 format()方法来完成，用法如下
  - **<模板字符串>.format(<逗号分隔的参数>)**

槽 slot

- 指模板字符串中变化的区域，该区域内的内容需要被替换为指定格式。



格式化方式

:	<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, %

格式化数值示例

:	<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%'
```

格式化字符串示例

:	<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("C++")
'****************C++'
>>>"{:10}".format("C++")
'C++       '
```

f-string 格式化

- Python3.6开始提供的一类更加简化的格式化方式
- 这种方式在字符串开头的时候,以 f 标识,然后通过 占位符+变量名 的方式来自动格式化。

```
name = 'Mike'

age = 21

print(f"your name is: {name}, ur age is: {age}")
```

- 变量的输出格式是默认的，如果需要更改，则只需要在变量后添加冒号，然后使用相应的格式字符串即可，同 .format 函数

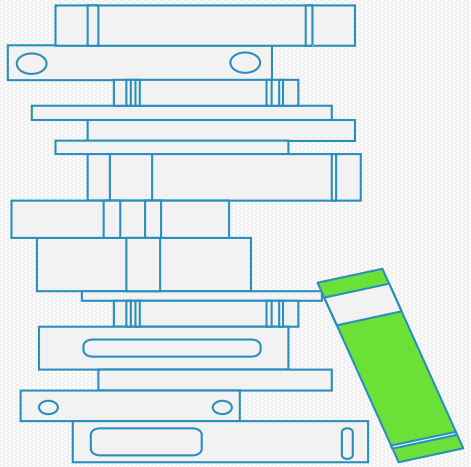
```
print(f'PI={math.pi:.2f}') #3.14
```

自学掌握

07

混合运算和强制类型转换

第二章2.7节

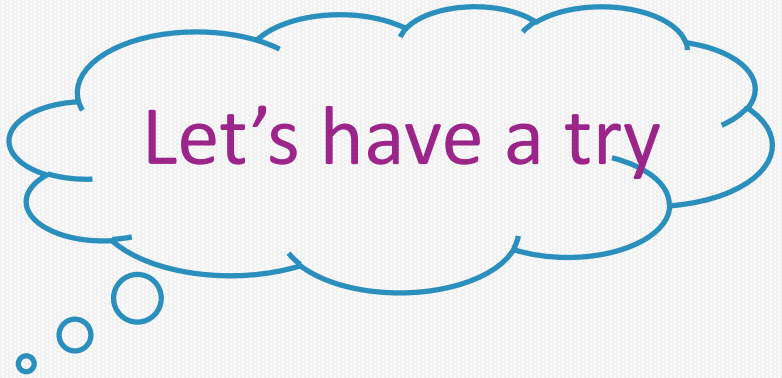


单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

以下语句中不正确的是



A t = int("124")

B t = float("12.58E+2")

C t = int(12.58)

D t = int("12.3E+2")

提交

本章总结

基本数据类型

变量命名、赋值

数据的输入&输出

数值计算

内置数值计算函数:
max, min, abs, round, pow

math库： pi, sqrt, factorial

字符串的运算

字符串的4种表示方法

字符串运算符： +, *,in

字符串的索引与切片

内置字符串函数

字符串处理方法

字符串格式设置

本周课程作业

① 阅读 第二章知识点

掌握数据类型、变量、输入与输出函数、内置函数、

字符串类型的相关运算（预习）

② 复习 课上的实例

可在解释器中输入代码，运行输出结果
可配合课本进行

③ 实操 完成实验书中的习题（上机课进行）

实验1 P27 3.
实验2

④ 提交 python123中的作业（上机课后）

The End