

高级语言程序设计

Python

授课教案

•

课程名称：高级语言程序设计 Python

课程性质：通识必修课

授课时间：2023-2024 学年第二学期

授课对象：2023 级非计算机专业本科生

授课教师：刘立群

目 录

Python	1
授课教案	1
第 0 周（2 学时）	3
第一周（4 学时）	5
第二周（4 学时）	11
第三周（4 学时）	18
第四周（4 学时）	25
第五周（4 学时）	32
第六周（4 学时）	39
第七周（4 学时）	44
第八周（4 学时）	50
第九周（4 学时）	60
第十周（4 学时）	66
第十一周（4 学时）	71
第十二周（4 学时）	78
第十三周（4 学时）	82
第十四周（4 学时）	88

第 0 周（2 学时）

教材章节：

课程要求与学习指导

第 2 章 Python 语言概述

2.2 安装与运行

教学目的和要求：

- 1、掌握 Python 环境的下载和安装，熟悉 IDLE 环境使用
- 2、超星学习通平台的注册和使用

教学重点

- 1、Python 环境的下载和安装
- 2、线上及线下学习要求和方法

教学难点

Python 安装中常见问题

教学方法与手段

课前发布任务，学生进行线上学习，线上完成任务

线上教学过程设计

第一部分：学习指导和课程要求

- 1.为什么学
- 2.学什么
- 3.怎么学
- 4.线上线下混合式教学的要求
- 5.课程成绩比例
- 6.线上课堂活动的权重比例



第 1 章 程序和算法

一、程序的演变

编程其实就是把人类的需求用计算机语言来表达，是一场人与计算机的对话。

计算机语言经历了从机器语言、汇编语言，再到高级语言的演变过程。

二、高级语言的运行机制

编译程序对源程序进行解释的方法相当于日常生活中的“整文翻译”。

解释程序对源程序进行翻译的方法相当于日常生活中的“同声传译”。

🔗 线上资源：章节 1.1，1.2

第 2 章 Python 语言概述

2.2 Python 环境安装与运行

一、下载与安装

网址：www.python.org/downloads/，根据操作系统不同选择不同版本，下载相应的 Python

3.0 系列版本程序。

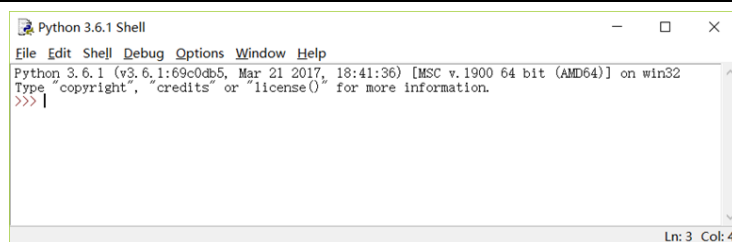
🔗 线上视频资源：2.2.1 下载与安装

🔗 线上帖子：安装常见问题“Python 安装遇到的坑汇总，看这一篇就够了！”

二、Python 的运行

Windows 的“开始” - “程序” - “Python 3.5” - “IDLE (Python 3.5 64bit)”

可以启动内置的解释器（IDLE 集成开发环境）



```
Python 3.6.1 Shell
File Edit Shell Debug Options Window Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 18:41:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

线上资源：课堂实录 2.6.1

线上学习任务单

1. 观看发布的视频：学习指导和课程要求
2. 完成问卷：课程要求及考试形式的问卷
3. 线上任务 1：相关试题
4. 挑战任务 0，自行完成安装，问题及结果截图并发帖

教学后记

1. 开学前一周提前布置了线上学习任务，学生任务完成情况达到 90%以上，同学们能够积极反馈，效果不错。
2. 发布“学习指导和课程要求”的视频，在开课前让学生了解课程的考核方式、线上活动形式、线上成绩权重、课程的学习技巧，同时提出具体的要求，提醒和约束同学们认真对待课程学习，尽快适应。
3. 布置线上问卷，用这种形式检查学生本次任务完成情况，借此让学生熟悉并使用学习通，学会利用线上课程资源，熟悉授课形式，逐渐形成学习习惯，能够适应线上线下的学习形式。

第一周（4 学时）

教材章节：

第 2 章 Python 语言概述

2.1 Python 产生与特性

2.3 基本语法

2.5 turtle 库绘图

教学目的和要求：

1. 了解 Python 语言特性
2. 掌握基本语法规则
3. 命令行、文件执行方式
4. 标准库的导入
5. 学会 Turtle 库的导入，可以绘制基本图形

教学重点

1. 基本语法、程序的执行方式
2. 标准库的导入
3. Turtle 库绘制基本图形

教学难点

1. IDLE 环境下两种程序执行方式
2. 标识符命名规则、缩进、注释
3. Turtle 库的常用函数使用

教学方法与手段

理论课利用学习通开展线上活动与学生的互动

实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

前课回顾：

2.1 Python 产生与特性 ②线上课程视频：章节 2.1

Python 语言的诞生，Guido van Rossum，Python 语言创立者，2002 年 Python 2.x，2008 年 Python 3.x。Python 语言的优势：1. 语法简单 2. 可移植性 3. 粘性扩展 4. 开源理念 5. 面向对象。例：.Hello World，体会 Python 语言的简洁

```
>>> print("Hello World")
Hello World
>>> print("世界，你好")
世界，你好
>>>
```

```
#include <stdio.h>
int main(void)
{
    printf("Hello World\n");
    return 0;
}
```

一般来说，同样功能的程序，Python 语言实现的代码行数仅相当于 C 语言的 1/5 至 1/10，简洁程度取决于程序的复杂度和规模。

Python 的运行方式：PPT18

1. 命令行方式

命令行方式是一种交互式的命令解释方式

输入命令，解释器（Shell）即负责解释并执行命令

2. 文件执行方式

建立程序文件，然后调用并执行这个文件，以.py 为扩展名



演示【例】绘制直线、三角形、正方形（命令行=>循环=>文件方式）

```

Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC
v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inform
ation.
>>> import turtle as t
>>> t.fd(50)
>>> t.left(120)
>>> t.fd(50)
>>> t.left(120)
>>> t.fd(50)
>>> for i in range(3):
>>>     t.fd(50)
>>>     t.left(120)

绘任意多边形.py - C:/Users/Administrator/Desktop...
File Edit Format Run Options Window Help
import turtle as t
n=eval(input("输入多边形的边数:"))
a=180-(180*(n-2)/n)
L=100
for i in range(n):
    t.fd(L)
    t.left(a)
Ln: 8 Col: 0
  
```



线上抢答：

程序文件扩展名？

Python 官网？ <http://www.python.org/downloads>

第一部分：基本语法

2.3 Python 基本语法 P19

1. 注释：#，'''

2. 关键字：保留字是 Python 系统内部定义和使用的特定标识符。Python 3.5.X 中共有 33 个关键字。

```
>>> 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']
```

```
>>>
```

3. 标识符：标识符用来表示常量、变量、函数、对象等程序要素的名字。必须符合命名规则：

- (1) 首字符必须是字母、汉字或下划线。
- (2) 中间可以是字母、汉字、下划线或数字，但不能有空格。
- (3) 字母区分大小写（大写 S 和小写 s 代表了不同的两个名称）。
- (4) 不能使用 Python 的关键字。



线上选人：1 下面正确的标识符： age_18, _year_2021, class, 211school



线上选人：2. 下语句的运行结果：

```
>>>x=100
```

```
>>>y=200
```

```
>>>z=X+y
```

4. 强制缩进：使用缩进来表示代码块，缩进的空格数是可变的，但是同一个代码块的语句必须包含相同的缩进空格数。

5. 一句多行：\

6. 多句一行：；



线上抢答：PPT29

1. 程序错在哪里？怎么修改才正确？

2.缩进的空格数必须为 4 个，是否正确？

第二部分：turtle 绘图

2.5 turtle 库绘图

一、标准库的导入

函数库又被称为模块，它是一个包含所有定义函数和变量的文件，其扩展名是.py。

函数库中的标准库和第三方库都需要先导入再调用。导入模块的语句是 `import`，它有下面的三种形式：

(1) 导入一个或多个模块的全部函数，格式为：

```
import <模块名 1> [<模块名 2>,...<模块名 N>] [as <别名>]
```

(2) 导入某个模块的指定函数，格式为：

```
from <模块名> import <函数名 1> [<函数名 2>,...<函数名 N>]
```

(3) 导入某个模块的全部函数，格式为：

```
from <模块名> import *
```



说明演示

i. 在使用第 1 种形式导入模块后，在调用函数前需要加上模块名做为前缀。

```
>>> import turtle
>>> turtle.forward(15)
```

ii. 使用第 2 种方式和第 3 种方式导入模块后，函数名的前缀则可省略。

```
>>> from turtle import *
>>> forward(15)
```

iii. 为了增加程序的可读性，可以使用模块别名的方式来简化函数名的前缀。

```
>>> import turtle as t
>>> t.forward(15)
```



思考：PPT8

上例中绘制了一个三角形，此例绘制的是一下正方形，那么它的位置是什么？画布中心是？如何具体描述位置？

【例2.10】绘制一个正方形。

```
#example2.10
import turtle as t          #导入turtle, 别名为t
t.setup(300,200)            #设置画布大小
for i in range(4):          #从原点开始绘制一个正方形
    t.forward(50)           #前进50个像素
    t.left(90)              #向左旋转90度
```

哪个位置？



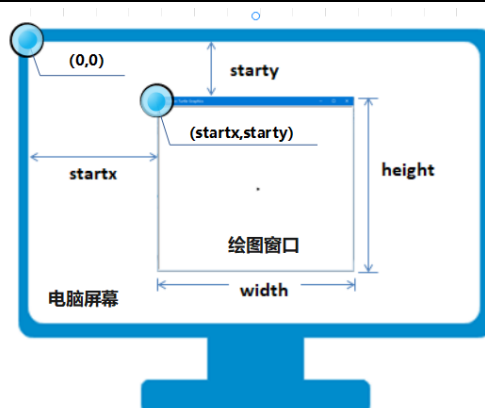
二、窗口与画布

1. 绘图窗口

(1) 设置窗口

```
turtle.setup(width,height,startx,starty)
```

(2) 位置参数



说明演示:

```
>>> import turtle
>>> turtle.setup(200, 200, 0, 0) # 设置窗口大小为 200x200 像素, 初始位置在屏幕的最左上角
>>> turtle.setup(0.75, 0.5, None, None) # 屏幕宽度和高度 75%和 50%, 位置居中
>>> turtle.setup() # 当参数都省略时, 表示设置窗口为默认的初始状态
```

2. 设置画布

画布就是 turtle 的绘图区域。默认情况下, 画布的大小为 400*300, 位于窗口中心。可以使用 screensize 函数设置它的大小和背景颜色。

```
turtle.screensize(canvwidth=None, canvheight=None, bg=None)
```

说明:

canvwidth: 正整数, 表示画布的像素宽度。

canvheight: 正整数, 表示画布的像素高度。

bg: 颜色字符串或颜色元组, 表示画布的背景颜色。

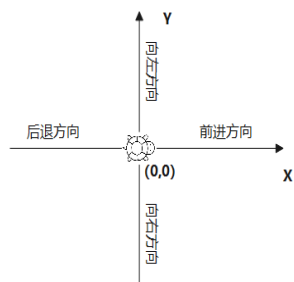
```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
>>> import turtle
>>> turtle.screensize()
(400, 300)
>>> turtle.bgcolor()
'white'
>>> turtle.screensize(800, 600, "blue")
>>> turtle.screensize()
(800, 600)
>>> turtle.bgcolor()
'blue'
>>> |
Ln: 45 Col: 4
```



抢答: 绘图窗口的初始大小是固定的吗? 默认窗口的大小是多少? 默认画布的大小?

默认情况下, 窗口宽度为当前屏幕宽度的 50%, 高度为当前屏幕高度的 75%, 位置在屏幕中心。默认的绘图窗口的大小会根据当前使用的电脑屏幕分辨率而各有不同。

3. 坐标系统



```
>>> import turtle as t
>>> t.pos()           #海龟的初始位置在坐标原点(0,0)
(0.00,0.00)
>>> t.forward(150)    #沿初始的前进方向移动150像素
>>> t.pos()           #海龟的当前位置坐标为(150, 0)
(150.00,0.00)
>>> t.left(90)        #控制海龟向左转动90度
>>> t.forward(150)    #沿当前的前进方向移动150像素
>>> t.pos()           #海龟的当前位置坐标为(150, 150)
(150.00,150.00)
>>> t.home()          #海龟回到初始位置(0, 0)和方向(x轴正方向)
>>> t.pos()           #海龟回到初始位置(0, 0)和方向(x轴正方向)
(0.00,0.00)
>>> t.goto(200,200)   #海龟移动到坐标点(200, 200)
>>> t.pos()           #海龟移动到坐标点(200, 200)
(200.00,200.00)
>>>
```

三、绘图动作与状态

1.绘图状态与控制

`turtle.pendown()` | `turtle.pd()` | `turtle.down()`

`turtle.penup()` | `turtle.pu()` | `turtle.up()`

`turtle.pensize(width=None)` | `turtle.width(width=None)`

2.绘图动作与方向

(1) 相对移动

`turtle.forward(distance)` | `turtle.fd(distance)`

`turtle.back(distance)` | `turtle.bk(distance)` | `turtle.backward(distance)`

(2) 绝对移动

`turtle.goto(x, y=None)`

`turtle.setpos(x, y=None)` | `turtle.setposition(x, y=None)`

(3) 相对方向

`turtle.right(angle)` | `turtle.rt(angle)`

`turtle.left(angle)` | `turtle.lt(angle)`

(4) 绝对方向

`turtle.setheading(to_angle)` | `turtle.seth(to_angle)`

设置海龟方向为一个绝对角度（相对 x 轴正方向的角度），整数或浮点数。

(5) 初始化海龟

`turtle.home()`

初始化海龟的位置和方向，海龟回到位置 (0, 0)，方向指向 x 轴正方向。

实验课教学过程设计：

1、观看视频，课程章节 2.3.1，并完成下面的实验：（作业 1）

实验 2 基本语法：完成实验内容，并将生成的程序文件"e2.1.py"和"e2.2.py"，提交上来。

2、绘制基本图形：绘制正多边形（作业 2）

观看视频 2.5.1：学习使用 Turtle 库绘制正方形、六边形

参照实验中代码，自由绘制一个或一组多边形（要求与实验中不一样的哦！）。

将代码和图片粘贴到，答案处，提交。

3、绘制基本图形：五角星（作业 2）

观看视频 2.5.1

参照实验中代码，绘制五角星。

将代码和图片粘贴到，答案处，提交。

4、绘制不连续图形：（作业 2）

观看章节 2.5.2 视频教程。

绘制数字“2022”。

5、书写文本“一起向未来！”

参考 `turtle` 函数功能说明（主教材表 2.1），使用正确的函数完成。效果参照图示。

挑战任务

挑战 1：用相同图形叠放可以产生不同的效果，如下图。

这只是两个简单的例子（图三是同学们的作品，哇哦！）。

你来试一试，看看还能有哪些奇思妙想！回帖可以获得课程积分，加油呀！



教学后记

0、课前学习通发布第 1 周任务清单。让学生了解本周学习任务要求。

1、第一次课不单纯讲语法，直接引入绘制正方形程序。以兴趣引导为主“先做起来，再学起来”。

2、通过小程序让学生体会 Python 的简洁和高效，语法规则、常见错误。由于这个程序功能简单，学生理解无障碍，语法及语句格式潜移默化地被植入。

3、继续分析 `turtle` 库的使用，循环绘制正方形程序。图形动态的绘制过程，方便学生理解程序的执行方式、缩进、注释。

4、函数库的导入使用，相关的绘图函数，快速讲解基本的函数语句，坐标系统等。课上学生调试程序乐在其中，完成的作品五花八门。学习效果非常好。

5.标准函数库的导入使用，相关的绘图函数，快速讲解基本的函数语句，坐标系统等。课上学生调试程序乐在其中，完成的作品五花八门。学习效果非常好。

6.对绘图兴趣深厚，作业中的书写文本，开始有不会的声音，最后查表后基本可以完成。学生们的自主学习与思考需要逐步培养。

第二周（4 学时）

教材章节：

第 2 章 Python 语言概述

2.5 Turtle 绘图

2.4 程序设计基础

教学目的和要求：

1. 掌握 turtle 库的基本绘图语句（函数）
2. 理解程序设计的基本结构 IPO
3. 掌握基本输入输出语句

教学重点

1. turtle 库的基本绘图语句
2. 问题求解的 IPO 结构
3. 基本输入输出

教学难点

1. 基本输入输出函数 Input()、print()、eval()
2. 使用 IPO 结构进行问题的抽象和求解

教学方法与手段

理论课利用学习通开展线上活动与学生的互动

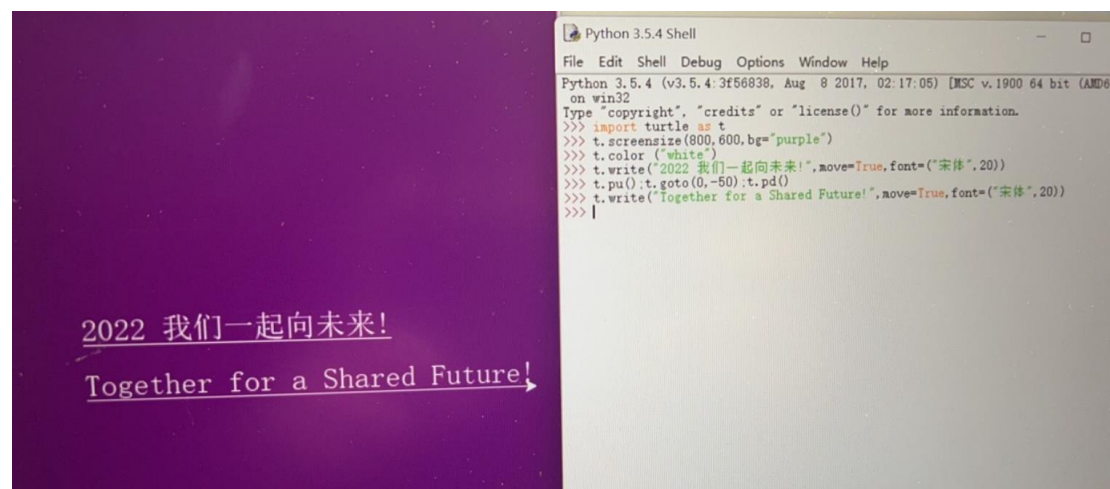
实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

前课回顾：

😊 投屏：挑战任务点评：展示学生作品。已置顶。2 班满怡然的挑战 1，讲一讲。

😊 投屏：作业 1-2，书写文字。1 班张思涵的答案如下图，说一说此函数的用法。



😊 投屏：作业 1-2，2 班徐佳雯

import turtle as t

```
t.bgcolor("blue")
t.pencolor("white")
t.write("我们一起向未来！ \nTogether for a shared future!",font=("楷体",17,"underline"))
```

前课小练:

☺线上投票: 2.X 3.X 兼容吗?

☺线上选人：下面语句的运行结果？ 关键字的命名规则，区别大小写

```
>>>x=100
>>>y=200
>>>z=X+y
```

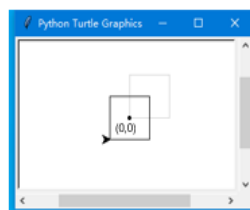
😊线上选人说一说你知道的 turtle 中的颜色？章节 2.5 中：turtle 库中可以使用的颜色单词.py

思考：根据页面的调整五环的大小，和位置，怎么办？

【例2.11】绘制一个以原点为中心，边长为50的正方形。

```
#example2.11
import turtle as t      #导入函数库，并设置别名为t
t.setup(300,200)        #设置窗口大小为300*200
t.pensize(2)            #设置画笔线条宽度为2像素
```

起始位置应是？



②思考：P18，起始位置应该在哪里？补全代码

#example2.11	
import turtle as t	#导入函数库，并设置别名为 t
t.setup(300,200)	#设置窗口大小为 300*200
t.pensize(2)	#设置画笔线条宽度为 2 像素
t.penup()	#设置画笔抬起，非绘图状态
t.goto(-25,-25)	#移动海龟至绝对位置 (-25, -25)
t.pendown()	#设置画笔放下，进入绘图状态
for i in range(4):	#绘制正方形
t.forward(50)	
t.left(90)	

=>>>>>>>>>>>>>>>扩展程序

绘制填充效果的: (color,begin fill())

```
import turtle as t
t.setup(300,200)
t.pensize(2)
t.color("blue","red")
t.penup()
t.goto(-25,-25)
t.pendown()
t.begin_fill()
for i in range(4):
    t.forward(50)
    t.left(90)
t.end_fill()
```

#####另一种方法: `t.circle(50,steps=4)`

思考：输入位置？输入颜色？

本课内容：

2.4 问题求解的程序结构

2.4.1 问题求解的程序结构 IPO

高级语言的程序都是用来求解特定问题的，而问题的求解都可以归结为计算问题。利用计算机解决计算问题时，一般情况下，程序都遵循数据输入（Input）、数据处理（Process）、数据输出（Output）的基本结构。这种程序结构正是反映了实际问题的计算过程。

如同数学问题的求解和证明：

已知条件+现有规则定理→运算推导→得出结论或定理

【例2.5】求一元二次方程 $ax^2+bx+c=0$ 的实根。

```
# example2.5
a=int(input("请输入a: ")) #输入a
b=int(input("请输入b: ")) #输入b
c=int(input("请输入c: ")) #输入c

d=b**2-4*a*c
if d>=0:
    #判断是否有实根
    x1=(-b+d**0.5)/(2*a)
    x2=(-b-d**0.5)/(2*a)
    print('方程的根: x1=%f,x2=%f'%(x1,x2))
else:
    print('input data Error!')
```

数据输入 (Input)

数据处理 (Process)

数据输出 (Output)

输入数据：输入（Input）是一个程序的开始。程序要处理的数据有多种来源，形成了多种输入方式，包括：文件输入、网络输入、控制台输入、交互界面输出、随机数据输入、内部参数输入等。

处理数据：处理（Process）是程序对输入数据进行计算产生输出结果的过程。计算问题的处理方法统称为“算法”，它是程序最重要的组成部分。可以说，算法是一个程序的灵魂。

输出数据：输出（Output）是程序展示运算成果的方式。程序的输出方式包括：控制台输出、图形输出、文件输出、网络输出、操作系统内部变量输出等。

2.4.2 函数是什么

在上面的案例中用到了几个函数：input()、print()、eval()和 int()。那么，程序中的函数是什么呢？处处皆学问：用函数概念应对生活中困境！（两段小视频）



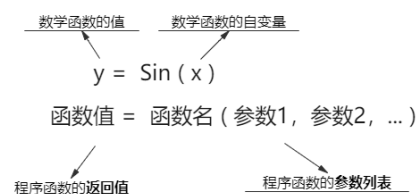
观看小视频，理解封装和复用的概念：

健身包（）无参数直接用，函数()。

书包（教材，作业本，文具盒，水杯，雨伞）

书包（教材，作业本），函数（参数1，参数2），带参数调用。

高级语言中的函数就是一段代码的封装，用来实现某种特定的运算或功能。使用者不用关心这些运算和功能的具体实现细节，只要调用这些函数，就可以进行运算或完成想要的功能。程序中函数的调用与数学函数中方法类似，通过函数名并且给定参数的方式。

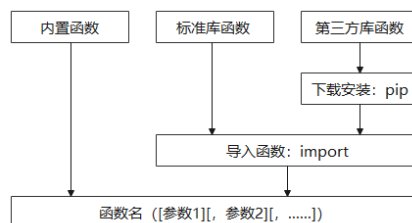


说明：print 函数比较特殊，不用返回值

```
a=int(input("请输入 a: "))
print('方程的根: x1=%f,x2=%f'%(x1,x2))
print('input data Error!')
```

Python 的函数有三种：内置函数、标准库函数、第三方库函数。

内置函数是系统自带的可以直接使用的函数，如案例中的 input、print、eval 等都是内置函数。标准库函数是系统自带的外部函数，需要先用 import 导入函数库后才能使用，如，turtle 是标准函数库。第三方库函数由相关人员或机构开发，需要先进行函数库的安装，再导入后才能使用。如，pyecharts 就是第三方函数库。函数的调用过程如图：



知识扩展：“函”字的演变

“函”字始见于商代甲骨文，它的字形很像一个袋子里装着一支箭的形状，袋子上还有一个便于手拿或挂在腰上的提手或挂钩。“函”的本义即箭袋，泛指包物的东西，又特指包信等物的封套。

函数中的“函”就是取其封装之意，程序中的函数是指一段代码的封装。封装（Encapsulation）是面向对象程序设计方法的重要原则，就是把对象的属性和操作结合为一个独立的整体，并尽可能隐藏对象的内部实现细节。



2.4.3 输入函数 input()

函数的语法格式

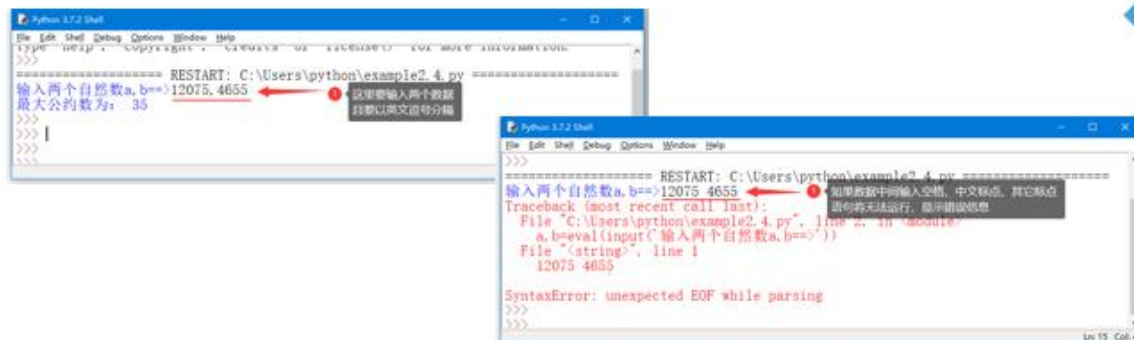
input 是输入函数，用来读取用户输入的数据，并返回一个字符串，具体格式如下：

```
<变量名>=input(["提示信息"])
```

同时为多个变量赋值时

例如: 在【例 2.4】中, 输入两个自然数的语句:

```
a,b=eval(input('输入两个自然数 a, b: '))
```



演示: x+y

使用函数转换数据类型

```
>>> x=input('input x=>')
input x=>5
>>> y=input('input x=>')
input x=>20
>>> print(x+y)
520
>>>
```

```
>>> x=int(input('input x=>'))
input x=>5
>>> y=eval(input('input x=>'))
input x=>10
>>> x+y
15
>>>
```

2.4.5 转换函数 eval()

1. 函数语法格式

eval 函数用来执行一个字符串表达式, 并返回表达式的值。具体格式如下:

```
eval(表达式[,globals[,locals]])
```

【例 2.9】简单公式计算器。

2.4.4 输出函数 print()

1. 函数语法格式

print 是输出函数, 用来输出表达式的值, 具体格式如下:

```
print(value1, value12, ..., sep=' ', end='\n')
```

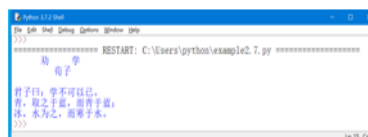
【例 2.6】print 的测试程序

2. 控制数据输出的样式

(1) 分行显示字符串

【例 2.7】两种字符串分行显示的方法示例。

```
#example2.7 荀子之劝学
劝学=" 劝 学\n 荀子"
劝学1=""
君子曰: 学不可以已。
青, 取之于蓝, 而青于蓝;
冰, 水为之, 而寒于水。"
print(劝学)
print(劝学1)
```



思考: 代码与输出的对应关系? \n 的作用?

(2) 设置数据的小数位数

【例 2.8】计算圆的面积并保留 4 位小数。

格式说明符：此处显示一个保留4位小数的浮点数

```
print("圆的面积为: %.4f" % 面积)
```

变量：字符串中要显示的值，存放计算结果

😊 抢答：代码与输出的对应关系？

(3) 在 print() 函数中，用格式字符串实现输出特定样式：

%：格式说明符

格式字符串：格式说明符+普通字符

<格式字符串>% (<值 1>, <值 2>, ..., <值 n>)

```
>>> print("今天是%d 年%d 月%d 日，天气%s!" % (2022,3,16,'晴'))
```

今天是 2022 年 3 月 16 日，天气晴！

```
>>>
```

说白了：是在字符串中嵌入值，%就是占位符

%d 啥意思？ %s 啥意思？ d 代表整数，s 代表字符串，f 代表浮点数

❓ 思考：例 2.8 程序中的 IPO？ 常量、变量，表达式？ 引入概念

实验课自主学习任务：

📎 观看章节 2.4.1 视频，完成递进式任务：实验四：程序设计结构—IPO

作业 3：

- 1、个税计算器：输入给出的代码 P21，运行无误后保存为“实验 4.1.py”并提交。
- 2、复利计算器：将代码补充完整，运行无误。代码保存为“实验 4.2.py”提交。
- 3、根据实训 P25 图 4.7，分析程序结构，新建文件，编写代码。实现“古尺计算器”功能：程序保存为“实验 4.3.py”并提交。

挑战任务：

挑战 2-1：IPO 结构实例。输出一个整数 n，程序自动绘制一个的正 n 边形，且输入颜色可绘制有颜色带填充效果的图形。

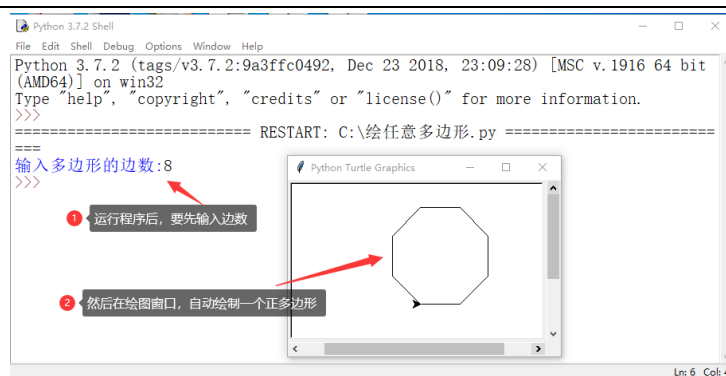
分析程序结构

I: 输入多边形的边数，颜色

P: 导入函数库，调用函数

O: 绘制任意正多边形

编写代码功能参考下图：



挑战 2-2：编程完成如下的输入输出。

```
>>>
===== RESTART: C:/Users/LLQ/Desktop/
=
输入人名：你
输入地名：天涯海角
世界那么大，我想和你去天涯海角看一看！
>>> |
```

```
str1=input("输入人名：")
```

```
str2=input("输入地名：")
```

```
print("世界那么大，我想和%s 去%s 看一看！"%(str1,str2))
```

教学后记

1. 课前学习通发布通知：第 2 周任务清单。让学生了解本周学习任务要求
2. 上一周的挑战任务，有几个作品可以展示一下。同学们很感兴趣，赋积分时引起了赞叹！
3. 通过上周内容的扩展，复习旧知引入新知。讲解绘图的颜色、填充，引入程序的 IPO 结构，函数的概念，常量、变量及表达式。
绘制以原点为中心的正方形=>画布的坐标系=>设置颜色和填充效果=>turtle 函数的参数=>参数的数据类型=>在任意坐标位置绘制正方形？=>输入与输出=>IPO 结构=>输入输出函数=>程序实例
4. 知识点较多，尽可能融入实例展开讲解。
5. 格式化输出理解的不太好，通过指点任务完成情况很好。启发的效果>灌输

第三周（4 学时）

教材章节

第 3 章 数据类型和表达式

- 3.1 基本数据类型
- 3.2 运算符与表达式
- 3.3 常用内置函数

教学目的和要求

理解常量变量、数据类型和表达式
掌握常用内置函数的使用方法

教学重点和难点

- 1、数据类型与表达式
- 2、常用内置函数功能：数值运算函数、数据转换函数、类型转换函数 t

教学方法与手段

理论课利用学习通开展线上活动与学生的互动

实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

前课小练：

😊 随堂练习：输入输出函数与 turtle，课上发布。

【实例 1】 张三的个税计算器(双分支)

个税起征点 = 5000

应发工资 = eval(input("输入应发工资: "))

五险一金 = eval(input("输入五险一金: "))

应纳税所得额 = 应发工资 - 五险一金 - 个税起征点

if 应纳税所得额 <= 3000: ##级数 1

 税率=0.03

 速算扣除数=0

elif 应纳税所得额 > 3000 and 应纳税所得额 <= 12000: ##级数 2

 税率=0.1

 速算扣除数=210

应缴个税 = 应纳税所得额 * 税率 - 速算扣除数

税后工资 = 应发工资 - 五险一金 - 应缴个税

print("您应缴个人所得税为: %f"%(应缴个税))


😊 抢答：实例中如何输出浮点数保留小数的方法？保留两位小数，具体方法。

个税起征点 = 5000
应纳税所得额 = (应发工资 - 五险一金) - 个税起征点
应缴个税 = 应纳税所得额 × 税率 - 速算扣除数
税后工资 = 应发工资 - 五险一金 - 应缴个税

级数	应纳税所得额	税率	速算扣除数
1	不超过3000元的	3%	0
2	超过3000元至12000元的部分	10%	210
3	超过12000元至25000元的部分	20%	1410
4	超过25000元至50000元的部分	25%	2660
5	超过50000元至100000元的部分	30%	4410
6	超过100000元至200000元的部分	35%	7160
7	超过200000元的部分	45%	15160

级数
税率
速算扣除数



 思考：代码中常量和变量有哪些？有哪些数据类型？表达式是什么？赋值？


表达式：数据运算的式子，数据（常量变量函数）和运算符构成。是 IPO 结构中的 Process，也就是数据处理。数据处理，就是分析问题-得出运算模型-构建表达式。

本课内容：


第 3 章 Python 数据类型和表达式

3.1 基本数据类型

1. 整型（int）：十进制，二进制整数 0B100，八进制 0O67，十六进制 0Xff
2. 浮点型（float）：十进制小数表示法 3.14、10.0，科学计数表示法 314.159=>3.14159e2
3. 字符串类型：单引号、双引号、三引号
4. 布尔类型：True, False

 演示：type(), int, float, bool, str

```
>>> a=256                #int
>>> b=0b1010             #int
>>> c=355/113             # float
>>> d=True                #bool
>>> e="0.6180339887"      #str
>>> type(a)
<class 'int'>
>>> type(c)
<class 'float'>
>>> type(d)
<class 'bool'>
>>> type(e)
<class 'str'>
```

 思考：为什么要设置不同的数据类型？解决运算和解决存储

```
>>> import sys
>>> sys.float_info.max
1.7976931348623157e+308
>>> sys.float_info.min
2.2250738585072014e-308
>>>
```

 知识扩展：有意思的浮点数。

1. 圆周率

圆周率的历史: 1500多年前, 南北朝时期的祖冲之计算出圆周率 π 的值在3.1415926和3.1415927之间, 并且得出了两个用分数表示的近似值: 约率为22/7, 密率为355/113。



圆的周长与直径之比是一个常数, 人们称之为圆周率。通常用希腊字母 π 来表示。

我国古代数学家作出了巨大的贡献, 在东汉初年的数学书《周髀(毕)算经》里已经载有“周三径一”, 称之为“古率”, 就是说, 直径是1的圆, 它的周长是3。《周髀算经》算经的十书之一, 是中国最古老的天文学和数学著作, 成书于公元前1世纪的西汉末或东汉初年。

西汉末年, 刘歆(约分元前50年到公元23年)定圆周率为3.1547

东汉时代, 张衡(公元78—139年)求得约等于5/8, 3.1622

三国时, 魏人刘徽(公元263年)用割圆术求得圆周率的前三位数字是 $\pi \approx 3.14$, 称为徽率

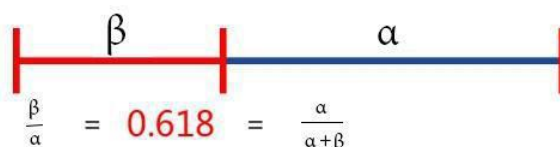
南北朝, 祖冲之(公元480年)已推算出 $3.1415926 < \pi < 3.1415927$, 他是世界上第一个将圆周率精准到7位小数的人。祖冲之又提出了用两个分数表示 π 的近似值, 即22/7及355/113, 分别称为 π 的约率和密度。

在祖冲之发现密率一千多年后, 欧洲的安托尼兹(1625年)才重新发现了这个值。

2019年3月14日, 谷歌宣布圆周率现已到小数点后31.4万亿位

2. 黄金分割

黄金分割是指将整体一分为二, 较大部分与整体部分的比值等于较小部分与较大部分的比值, 其比值是 $(\sqrt{5}-1):2$, 近似值为0.618, 通常用希腊字母 Φ 表示这个值。这个比例被公认为是最能引起美感的比例, 因此被称为黄金分割。



黄金分割具有严格的比例性、艺术性、和谐性, 蕴藏着丰富的美学价值, 这一比值能够引起人们的美感, 被认为是建筑和艺术中最理想的比例。

画家们发现, 按0.618:1来设计的比例, 画出的画最优美, 在达·芬奇的作品《维特鲁威人》、《蒙娜丽莎》还有《最后的晚餐》中都运用了黄金分割。而现今的女性, 腰身以下的长度平均只占身高的0.58, 因此古希腊的著名雕像断臂维纳斯及太阳神阿波罗都通过故意延长双腿, 使之与身高的比值为0.618。建筑师们对数字0.618特别偏爱, 无论是古埃及的金字塔, 还是巴黎的圣母院, 或者是近世纪的法国埃菲尔铁塔, 希腊雅典的巴特农神庙, 都有黄金分割的足迹。埃菲尔铁塔以观景台为分割点, 其中下方与上方高度的比值就是黄金分割的常数。

常量与变量

【实例2】#e4.2 复利计算器0

```
本金=eval(input("您的本金:"))
```

```
年利率=eval(input("年利率:"))
```

```
年期=eval(input("存期(年):"))
```

最终收益=本金*(1+年利率/100)**年期

print("最终收益: %f"%(最终收益))

 思考: 程序结构的 IPO, 哪些是常量变量? 表达式是什么?

变量=表达式(常量+变量+函数+运算符)

常量: 在程序运行过程中, 其值不发生改变的数据对象称为常量。

变量: 在程序运行过程中, 可以随着程序的运行更改的量称之为变量。

变量的声明: Python 变量的赋值操作即是变量的声明和定义的过程。

变量的命名: 变量命名要符合标识符的命名规则。

变量的赋值: 一般形式、增量形式、链式赋值、多重赋值。P22



演示: Python 变量未经赋值就使用, 解释器会提示错误。

3.2 运算符与表达式 P25

1. 算术运算符: +, -, *, /, % 取模, ** 乘方, // 整除

2. 关系运算符: ==, !=, >, <, >=, <=

3. 赋值运算符: =, +=, -=, *=, /=, %=, **=, //=

4. 逻辑运算符: and, or, not

5. 表达式: 运算优先级, 表 2.8, P29

```
>>> not "Abc"=="abc" or 2+3!=5 and "23"<"3"
```

```
True
```

```
>>>
```

**构造表达式注意事项:

1. 与数学表达式不同: 乘号不能省略, 只有小括号。可嵌套。
2. 运算优先级

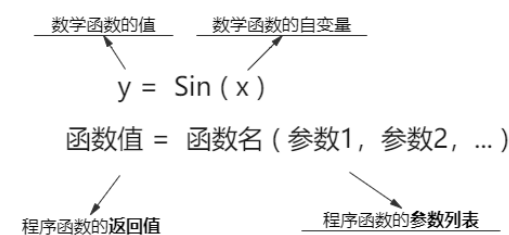


练习题: 数据类型、运算符与表达式? 选择 5T, 填空 1T2T

3.3 常用内置函数 P30

1. 函数功能 (表 2.9 内置函数)

函数的使用: 函数名, 参数个数, 参数类型



2. 函数实例

(1) 数值运算函数 P32

abs(x)函数: 返回一个 x 的绝对值, x 参数可以是整型或浮点型数据。

divmod(a,b): 返回由 a / b 的商和余数构成的一个元组, a、b 可以是整型或浮点型。

pow(x, y[, z]): 返回 x 的 y 次幂, 如果有参数 z 则返回 x 的 y 次幂与 z 的模。

$\text{pow}(x, y, z) \rightarrow \text{pow}(x, y) \% z$

$\text{pow}(x, y) \rightarrow x^{**}y$

$\text{round}(x, n)$: 对 x 进行四舍五入, n 保留的小数位数, 若省略则只保留整数。

```
>>> print(abs(10), abs(-10))
10 10
>>> print(divmod(10, 4), divmod(10.5, 2.5))
(2, 2) (4.0, 0.5)
>>> print(pow(2, 3), pow(2, 3, 4))
8 0
>>> print(round(3.1415926, 3), round(3.1415926))
3.142 3
>>>
```

【实例】构造表达式: 海伦公式, 实训 P29

若三角形的边长 a, b, c 分别为 3, 4, 5。

p 为周长的一半, 即, $p = \frac{a+b+c}{2}$ 。

则, 三角形的面积 S , $S = \sqrt{p(p-a)(p-b)(p-c)}$ 。

(2) 数制转换函数 P33

$\text{int}([x])$: 将十进制数值取整 (截去小数部分)。



```
>>>int(3.14)          # 3
>>>int(2e2)           # 200
>>>int(100, 2)        # 出错, 有 base 时只接收字符串
```

$\text{int}([x, \text{base}])$: 将 “base” 进制的合法整数字符串转换成十进制整数。

```
>>>int('23')          # 23, base 参数省略时默认为十进制数
>>>int('23', 16)      # 35, 将 '23' 做为 16 进制数
>>>int('Pythontab', 8) # 出错, 'Pythontab' 不是个 8 进制数
```

字符串 $0x$ 视作十六进制的符号, $0b$ 视作二进制的符号:

```
>>>int('0x10', 16)    # 16, 0x 是十六进制的符号
>>>int('0b10', 2)     # 2, 0b 是二进制的符号
```

$\text{bin}(x)$: 将十进制整数转换成二进制的数字字符串。

$\text{oct}(x)$: 将十进制整数转换成八进制的数字字符串。

$\text{hex}(x)$: 将十进制整数转换成十六进制的数字字符串。

```
>>> b=255
>>> print(bin(b), hex(b), oct(b))
0b11111111 0xff 0o377
>>> print(int('123', 16), int('123', 8), int('123'))
291 83 123
>>>
```

【实例】进制转换器: 它进制转十进制, 实训 P29

(3) 类型转换函数

`bool([x])`: 返回 `x` 布尔值。

`float([x])`: 返回一个浮点数。

`ord()`: 返回单个字符的 Unicode(ASCII)码, 返回值是一个整数。

`chr(i)`: 返回整数 `i` 对应的字符。与 `ord()` 函数互为反函数。

```
>>> x='A'
>>> y=97
>>> ord(x)
65
>>> chr(y)
'a'
>>> chr(ord(x))
'A'
>>>
```

⑦ 思考：字符的比较？汉字的比较？`ord()` 返回字符的 unicode(ASCII), "0"<"a"<"A"<"安"

```
>>> ord("A")
65
>>> ord("a")
97
>>> ord("男")
30007
>>> ord("女")
22899
>>> "男">"女"
True
>>>
```

`eval(str [,globals[,locals]])`: 将字符串 `str` 当成有效的表达式来求值并返回计算结果。

```
>>> x = 1
>>> eval('x+1')
2
>>> a='5,10'
>>> x,y=eval(a)
>>> print(x,y)
5 10
>>>
```

实验课教学过程设计：

⑧ 观看章节 3.3.1 中的视频“实验五：表达式与内置函数”

完成作业 4:

1. 面积计算器: 新建文件, 运行无误后保存为“实验 5.1.py”并提交。
2. 进制转换器(十转它): 将代码补充完整, 运行无误。代码保存为“实验 5.2.py”提交。

3. 进制转换器（它转十）：将代码补充完整，运行无误。代码保存为“实验 5.3.py”提交。

📎 理论课程内容：直播课程的回放，在课程通知中发布。

挑战任务

挑战任务 3-1：根据下图，分析程序结构，新建文件，编写代码。程序功能为：输入一个任意的三位正整数，计算并输出其各位数字之和。

```
=====计算整数各位数字之和=====  
输入一个三位正整数：123 ① 输入一个三位整数  
结果是：6  
>>> ② 计算后，显示出计算结果
```

挑战任务 3-2：

已知，位移 S ，初速度 V_0 ，时间 t ，加速度 a ，计算公式为：

$$S = V_0 t + at^2/2$$

要求：程序运行后，依次输入位移、初速度和时间，程序可以计算并输出加速度。

测试数据：输入：距离 100、初速度 6、时间 10，输出：加速度 0.8。

教学后记

1. 总结函数的用法、封装与复用的概念，期望能够建立基本的编程方法，通过测试和查表自主理解内置函数的功能并能够灵活运用。介绍线上的资源，引导学生自主学习。由“能学会”到“能会学”！
2. 发布的挑战任务，同学们的答案让人挺惊喜的。说明启发后的“会学”效果不错！

第四周（4 学时）

教材章节：

- 第 4 章 Python 控制语句
- 4.1 结构化程序设计
- 4.2 分支结构
- 3.4.1 常用标准函数-random

教学目的和要求：

- 1、了解程序结构的基本概念
- 2、掌握分支结构的基本语句
- 3、掌握随机函数的使用

教学重点和难点

- 1、结构化程序设计方法的基本思想
- 2、逻辑功能设计和数据流关系
- 3、分支结构（选择结构）
- 4、random 模块的常用函数的使用

教学方法与手段

理论课利用学习通开展线上活动与学生的互动

实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

上周回顾：练习题：数据类型、运算符与表达式（第 3 周）

前课小练：国王的债务

传说国际象棋是舍罕王的宰相西萨·班·达依尔发明的。他把这个有趣的娱乐品进贡给国王。舍罕王对于这一奇妙的发明异常喜爱，决定让宰相自己要求得到什么赏赐。

西萨并没有要求任何金银财宝，他只是指着面前的棋盘奏道：“陛下，就请您赏给我一些麦子吧，它们只要这样放在棋盘里就行了：第一个格里放一颗，第二个格里放两颗，第三个格里放四颗，以后每一个格里都比前一个格里的麦粒增加一倍。圣明的王啊，只要把这样摆满棋盘上全部六十四格的麦粒都赏给你的仆人，他就心满意足了”。

国王欣然答应了，很快他发现这是一笔无法兑现的承诺！

```
= RESTART: C:\Users\Administrator\Desktop\2023Python\
务.py =
一粒小麦的重量约为(g):0.03
全世界的小麦年产量为(亿吨): 7.788
国王的债务为: 5534.02322211亿吨!!!
约为全世界710.58336年的产量!
>>> |
```

输入

输出

已知，假设一粒小麦的重量为 0.02g:

步骤 1：确定 IPO，

输入：一粒小麦重量、全世界小麦的年产量；

处理：计算国王的债务；

输出：国王需要多少年还清债务？

定义变量：重量、产量，多少吨，多少年

步骤 2：分析问题的计算部分，构造计算表达式。

步骤 3：编写程序，调试、运行程序。



抢答：

程序代码：

#国王的债务

```
onerice=eval(input("一粒小麦的重量约为(g):"))
```

```
production=eval(input('全世界的小麦年产量为(亿吨): '))
```

```
rice=pow(2,64)-1
```

```
weight=(rice*onerice)/10**6/10**8      #亿吨
```

```
year=weight/production/10**4           #万年
```

```
print('国王的债务为: {:.8f}亿吨!!! \n'.format(weight))
```

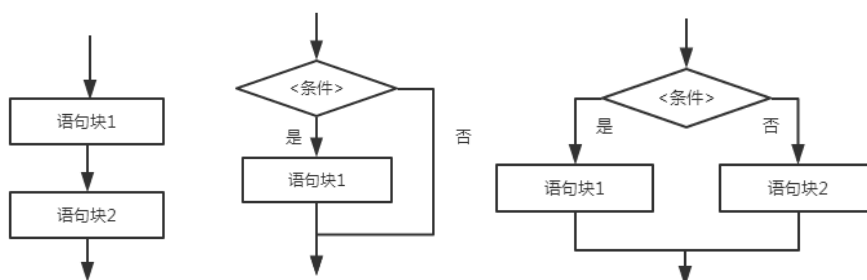
```
print('约为全世界 {:.8}万年的产量! '.format(year))
```

注意：

1. input 函数返回输入的字符串，若进行算术运算，则需要 eval 函数。
2. 程序中的公式，可以采用函数或运算符来设计。

本课内容：

4.1 结构化程序设计



顺序结构：程序从第一行语句开始执行，执行到最后一行语句结束，程序中的每条语句都会被执行一次。

4.2 分支结构：

也称选择结构，表示程序的处理步骤出现了分支，它需要根据某一特定的条件选择其中的一个分支执行。

1. 单分支结构

Python 中单分支 if 语句的语法格式如下：

```
if <条件>:
```

```
<语句块>
```

【例 1】输入两个数字，输出其中较大数字。P46，绘制出流程图。

#example4.1

```
x=eval(input("请输入第一个数字:"))
```

```
y=eval(input("请输入第二个数字:"))
print("输入的两个数字为: ",x,y)
if x>y:
    print("较大的是:",x)
if x<y:
    print("较大的是:",y)
```

2. 双分支结构

是使用比较多的一种程序结构，Python 中双分支 if 语句的语法格式如下：

```
if <条件>:
    <语句块 1>
else:
    <语句块 2>
```

【例 2】用双分支结构改写【上例】。P47

#example4.2

```
x=eval(input("请输入第一个数字:"))
y=eval(input("请输入第二个数字:"))
print("输入的两个数字为: ",x,y)
if x>y:
    print("较大的是:",x)
else:
    print("较大的是:",y)
```

❓ 思考：例 2 与例 1 是否完全相同？绘制此例的流程图。

😊 抢答：如果 $x=y$ ，结果是什么？

3. 多分支结构

多分支结构是对双分支结构的一种补充，当判断的条件有多个，结果也有多种情况的时候，可用多分支 if 语句进行判断。多分支结构语法格式如下：

```
if <条件 1>:
    <语句块 1>
elif <条件 2>:
    <语句块 2>
elif <条件 3>:
    <语句块 3>
...
else:
    <语句块 n>
```

程序执行时，按照条件序列从上向下进行判断，当第一个条件 i 的值为 True，就执行该条件下的语句块，然后整个多分支 if 结构结束。如果没有任何条件为 True，则执行 else 下的语句块。注意：语句中的 else 是可选的。

【例】输入一个分数，判断它应该到的学分绩点。

```
#example4.4.1
score = eval(input("请输入分数: "))
if score >= 90:
    gpa = 4
elif score >= 80:
    gpa = 3
elif score >= 70:
    gpa = 2
elif score >= 60:
    gpa = 1
else:
    gpa = 0
print("应得学分绩点为: ", gpa)

#example4.4.2
score = eval(input("请输入分数: "))
if score < 60:
    gpa = 0
elif score >= 60:
    gpa = 1
elif score >= 70:
    gpa = 2
elif score >= 80:
    gpa = 3
else:
    gpa = 4
print("应得学分绩点为: ", gpa)
```

思考：多分支结构的流程图？抢答：

【例 4.5】分支结构嵌套 P73

多分支按照条件序列从上往下依次判断，只执行第一个为 True 的那个语句块。如果没有任何条件为 True，则执行 else 下的语句块。

3.4.1 random 模块

随机数可以用于数学、游戏以及安全等领域，还经常被嵌入到算法中，用以提高算法效率，并提高程序的安全性。Python 的常用随机数函数如表 3.11 所示：

表 3.11 random 模块常用函数功能说明

函数	描述
seed([a])	初始化随机数生成器的种子，默认为系统时间。需要在生成随机数之前调用此函数。
random()	生成一个在[0.0,1.0)之间的随机浮点数
randint(a,b)	生成一个[a,b]之间的随机整数，其中 a<=b。
randrange ([start,]stop [,step])	生成一个[start,stop)，以 step 为步长的范围内的随机整数，step 默认值为 1，等价于 choice(range(m,n,step))。
uniform(a,b)	生成一个[a,b]之间的随机浮点数，a 可以大于 b
choice(seq)	从非空序列 seq 中随机挑选一个元素
sample(seq,k)	从序列 seq 中随机获取 k 个元素，并返回一个新序列
shuffle(seq)	将序列 seq 的所有元素随机排序，并修改原序列
getrandbits(k)	生成一个 k 比特的随机整数

函数的应用实例：

```
>>> from random import *          #导入 random 模块的所有函数
>>> random()                      #生成一个在[0.0,1.0)之间的随机浮点数
0.14836545714990013
```

```
>>> randint(1,10)           #生成一个[1, 10]之间的随机整数
5
>>> x=[1,2,3,4,5]           #x 赋值为一个序列
>>> sample(x,2)              #从 x 序列中随机挑选 2 个元素生成新序列
[5, 1]
>>> shuffle(x)               #将 x 序列的元素随机排序
>>> x
[2, 5, 4, 3, 1]
>>>
```

`random` 是如何生成随机数的呢？Python 的随机数是用确定性的算法计算出来在[0.0,1.0)范围均匀分布的随机数序列，它并不是真正的随机被称为伪随机数。伪随机数具有类似于随机数的统计特征，如均匀性、独立性等。因此在模拟研究中可以采用伪随机数代替真正的随机数，从而提高模拟效率。

在计算伪随机数时，若使用的初值（种子）不变，那么伪随机数的序列也不变。例如，当随机种子为 2 时，生成的随机数序列如图 3.1 所示。



图 3.1 随机种子为 2 时的随机序列

当种子不变时，`random` 函数将在此序列中按顺序依次返回随机数。如执行下面语句：

```
>>> from random import *
>>> seed(2)                #随机种子为 2
>>> random()
0.9560342718892494         #随机序列中第一个数
>>> random()
0.9478274870593494         #随机序列中第二个数
>>> random()
0.05655136772680869        #随机序列中第三个数
>>> seed(2)                #重新生成随机序列
>>> random()
0.9560342718892494         #新生成的随机序列的第一个数
>>>
```

在默认的情况下，当未设置种子时，`random` 将以系统时间作为种子，这时产生的随机序列就是随时变化的了。那么，什么时候会使用 `seed` 函数呢？当我们的程序希望生成的随机数能够复现的时候，就需要设置 `seed` 函数了，此时生成的随机数就是固定的。

实验课教学过程设计：

🔗 观看章节 4.2.1 中的视频，完成实验七：程序的分支与选择

1. 海伦公式：求任意三角形面积。新建文件，运行无误后保存为“实验 7.1.py”并提交。
2. 水费计算器：新建文件编写代码，运行无误。代码保存为“实验 7.3.py”提交。

🔗 观看章节 3.4.1 中的视频，完成实验 6：标准库函数 random

3. 随机函数测试；
4. 随机绘图：将代码补充完整，运行无误。代码保存为“实验 6.1.py”提交。
5. 五彩万花筒。将代码补充完整，运行无误。代码保存为“实验 6.2.py”提交。

挑战任务

挑战任务 4-1：温度转换器：摄氏与华氏温度转换。

通过完成问题求解的 IPO 设计。.

问题：如何利用 Python 程序进行摄氏度和华氏度之间的转换

步骤 1：分析问题的计算部分：采用公式转换方式解决计算问题

步骤 2：确定功能

输入：华氏或者摄氏温度值、温度标识

处理：温度转化算法

输出：华氏或者摄氏温度值、温度标识

F 表示华氏度，82F 表示华氏 82 度

C 表示摄氏度，28C 表示摄氏 28 度

步骤 3：设计算法，根据华氏和摄氏温度定义，转换公式如下：

$$C = (F - 32) / 1.8$$

$$F = C * 1.8 + 32$$

其中，C 表示摄氏温度，F 表示华氏温度

步骤 4：编写程序。

步骤 5：调试、运行程序。使用 IDLE 打开上述文件，按 F5 运行（推荐），输入数值，

观察输出

```
转换温度.py =====
请输入温度值: 100 ① 输入温度值
请输入温度单位 (C-摄氏度, F-华氏度): C ② 输入字母C 或 F
转换后的温度是212.00F ③ 计算并输出转换后的温度
>>>
请输入温度值: 100
请输入温度单位 (C-摄氏度, F-华氏度): F
转换后的温度是37.78C
>>>
```

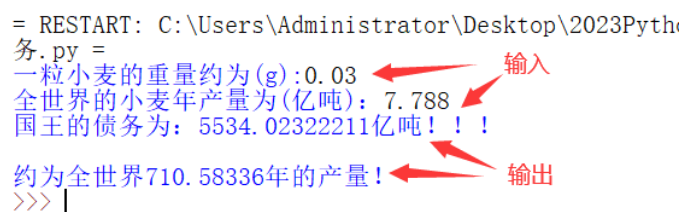
挑战任务 4-2：国王的债务。自行设计 IPO 结构。算一算：国王的债务究竟是多少？需要多少年能偿还？

输入数据 I：一粒小麦重量、全世界小麦的年产量。（数据要有根据，上网自行查阅资料确定数值）

数据运算 P：构造表达式完成计算

输出数据 O：输出结果。

```
= RESTART: C:\Users\Administrator\Desktop\2023Python\
务.py =
一粒小麦的重量约为(g):0.03
全世界的小麦年产量为(亿吨): 7.788
国王的债务为: 5534.02322211亿吨!!!
约为全世界710.58336年的产量!
>>> |
```



教学后记

1. 学生对挑战任务的热情不足，回帖的数量不够。课上可以展示一下学生的精华帖，给积分奖励激励一下。
2. 作业中：五彩万花筒（随机位置与大小），经过多次启发才能完成
3. 随机函数的掌握较好，可以进行标准函数的导入与调用

第五周（4 学时）

教材章节：

- 第 4 章 Python 控制语句
- 4.3 循环结构
- 3.4.2 常用标准函数-time

教学目的和要求：

- 1、了解程序结构的基本概念
- 2、掌握循环结构的基本语句
- 3、掌握时间函数的使用

教学重点和难点

- 1、结构化程序设计方法的基本思想
- 2、循环结构
- 3、time 模块中的常用函数的使用

教学方法与手段


理论课利用学习通开展线上活动与学生的互动

实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

温故知新：【例】随机五彩万花筒（随机函数与循环）##e6.2

```
import random as r
import turtle as t
t.speed(0)
col=["red","yellow","blue","green","purple"]
for i in range(1,100):
    x=r.randint(1,10)
    c=r.choice(col)
    a=r.randint(-100,100)
    b=r.randint(-100,100)
    t.penup()
    t.goto(a,b)
    t.pendown()
    t.pencolor(c)
    t.circle(x,steps=5)
```

 **引入：**通过此例引入遍历 for 循环

4.3 循环结构

Python 提供了两种不同风格的循环结构，包括遍历循环 for 和条件循环 while。

- for 语句循环是在指定遍历范围内进行循环，次数取决于元素个数
- while 语句循环是在条件满足时执行循环，次数取决于条件

4.3.1 for 语句循环

循环变量逐一使用遍历结构中的每个项目,当遍历结构中的所有项目都被取出后,循环结束。遍历结构可以是字符串、列表、文件、range()函数等。for 语句的执行过程是:

for 语句循环语法格式如下:

```
for <循环变量> in <遍历结构>:  
    <语句块>
```

【例 1】求 1-100 之间的奇数之和。数列元素的计算问题

```
#example4.6.1  
s=0  
for i in range(1,101,2):  
    s=s+i  
print("奇数和为",s)
```

程序运行结果如下:

```
>>>  
===== RESTART: C:/Python/Python36/example4.6.1.py =====  
奇数和为 2500  
>>>
```

❓思考: $s=s+i$ 语句的作用? 如果是 $s=s+1$, 求的是什么? (累加与计数的运算式)

for 语句循环还可以使用 else 关键字,语法格式如下:

```
for <循环变量> in <遍历结构>:  
    <语句块 1>  
else:  
    <语句块 2>
```

当 for 循环正常执行完成后,程序会继续执行 else 语句中的内容。如果 for 循环因为某种原因没有正常执行完成,例如遇到了 break 语句,则不会执行 else 语句中的内容。所以通常用 else 来检验 for 循环是否正常结束。可以改为以下格式:

```
#example4.6.2  
s=0  
for i in range(1,101,2):  
    s=s+i  
else:  
    #只有当循环没有被中断时,才会执行此分支  
    print("计算完毕!",end="")  
print("数列和为",s)
```

程序运行结果如下:

```
>>>  
===== RESTART: C:/Python/Python36/example4.6.2.py =====  
计算完毕! 奇数和为 2500
```

>>>

上面的遍历循环结构是用来进行数列计算是常用方法。

- 数列元素的计算问题：

1. 建立遍历结构，for
2. 构造计算式，累加等
3. 显示输出

😊 **抢答：** 如何用上面循环结构求解下面的数列计算问题？主题讨论：发帖

$1*2*3*...*n$ ，阶乘的计算

$1^2+2^2+...+10^2$

$1+1/2+1/3+...+1/n$

$11+22+33+.....+99$

其它自行设计的类似的数列计算问题，给出具体数列和代码

【例 2】 循环结构如下：数列符合条件元素的计算问题

```
#example
s=0
for i in range(1,101):
    if i % 2 ==1:
        s=s+i
    else:
        print("计算完毕! ",end="")
print("数列和为",s)
```

- 数列符合条件的元素的计算问题：

1. 建立遍历结构（全数列），for 结构
2. 设定条件，if 结构
3. 构造计算式，累加等
4. 显示输出

❓ **思考：** 如何用上面循环结构求解下面的数列计算问题？

100 以内能被 3 且不能被 5 整除的整数之和？

找出 1000-3000 以内能被 100 整除且不能被 400 整除的整数？

找出 2000-3000 间的闰年？挑战任务 8。

for 遍历字符串

【例 4.7】 求字符串中字母“o”，出现的次数，不区分大小写字母。

```
#example4.7
n=0
```

```
str="Life is short, YOU need Python!"
for i in str:
    if i=="o" or i=="O":
        n=n+1
    else:
        print("计算完毕,字母 o 的个数为: ",n)
```

代码执行结果为:

```
>>>
===== RESTART: C: /Python/Python36/example4.7.py =====
计算完毕, 字母'o'的个数为:  3
>>>
```

说明: `n` 作为计数器, 初始值为 0 的变量

❓思考: 将字符文本中的 `o`, 替换为 `@`?

游戏时间: 引入当循环

😊选人: 猜价格的游戏, 5 次机会。5 次机会很难猜中价格, 多次机会如何解决呢?

```
from random import randint
n=randint(10,100)
print("商品价格已经产生, 请输入 10 到 100 间的价格。")
for i in range(5):
    guess=eval(input("请输入您猜的价格: "))
    if guess>n:
        print("您输入的价格高于指定价格, 请继续。")
    elif guess<n:
        print("您输入的价格低于指定价格, 请继续。")
    else:
        print("恭喜您猜对了! 价格为",guess)
        break
else:
    print("游戏结束!")
```

4.3.2 while 循环

无法明确遍历结构, 或者不确定循环次数时就要使用 `while` 循环了。

`while` 语句循环语法格式如下:

```
while <条件>:
```

```
    <语句块>
```

`while` 循环和 `for` 一样也可以使用 `else` 关键字, 语法格式如下:

```
while <条件>:
```

```
    <语句块 1>
```

```
else:
```

```
    <语句块 2>
```

循环正常结束后, 执行 `else`, 用来检验循环是否正常结束。

【例 4.8】猜价格。 [10,100]之间的随机整数作为价格。

```
#example4.8
from random import randint
n=randint(10,100)
print("商品价格已经产生，请输入 10 到 100 间的价格。")
bingo=False
while bingo==False:
    guess=eval(input("请输入您猜的价格。"))
    if guess>n:
        print("您输入的价格高于指定价格，请继续。")
    elif guess<n:
        print("您输入的价格低于指定价格，请继续。")
    else:
        print("恭喜您猜对了！价格为",guess)
        bingo=True
else:
    print("游戏结束！")
```


 **思考：**使循环结束的语句在哪里？不能结束的循环是死循环。

while 循环结构：未知次数的循环

1. 构造循环条件，while
2. 构造计算式
3. 结束循环条件，if
4. 显示输出

【实例分析】假设一张可以多次对折的纸张，厚度为 0.2 毫米。珠穆朗玛峰的高度为 8844 米，问纸张对折多少次厚度可以超越珠峰的高度？

```
paper=0.0002 #单位 m
mount=8848.86
n=0 #记录折叠次数
while paper<=mount:
    paper*=2
    n=n+1
    print("折叠%d 次，纸张厚度为%f 米" %(n,paper))
print("折叠第%d 次后，纸张厚度超过珠峰"%(n))
```

 **思考：实验 8.5** 已知 2018 年末中国人口为 142706 万，人口增长为 0.381%；印度人口为 135405 万，人口增长为 1.11%，如果按此数据进行推演，印度将在哪一年人口超过中

国。实验 8.3.py。

【例 3】用 while 循环求 100 以内奇数之和。while 编写已知次数的循环

```
#example4.9
s=0
i=1
while i<=100:
    s=s+i
    i+=2
else:
    print("奇数和为",s)
```

说明：

- (1) `i=1` 作用是定义循环变量初始值。
- (2) 在 `while` 条件处设置 `i<=10` 为循环变量的终止值。
- (3) 在循环中 `i+=1` 语句和 `i=i+1` 等价，设置每次 `i` 增加的步长为 1。

While 循环结构：已知次数的循环

1. `while` 循环设置循环的终止值
2. `while` 之前定义初始值，`i=1`;
3. 循环中指定循环变量的步长值，`i=i+1`;


永真的循环

在编写 `while` 语句循环时，如果条件一直为 `True`，而循环中没有 `break` 来结束循环，也就是没有逻辑出口，则循环将陷入永远执行的状态，称为死循环。例如，以下两行语句为死循环，程序将一直输出数字 1。


```
while True:
    print(1)
```

在【例 4.9】中，如果去掉了 `i=i+1` 语句，`i` 值在每次循环时都是 1，而循环条件是 `i<=10`，每次判断时都为 `True`，循环将始终执行。如果程序陷入死循环，在 IDLE 环境中可以按组合键 **【Ctrl+C】**，开发环境中会显示“KeyboardInterrupt”，程序终止。

实验课教学过程设计：

 观看章节 4.3.1 中的视频：实验 8.3：绘制由 9 行“@”组成的菱形，完成下面任务：

1. 绘制一个由 15 行“*”组成的菱形。新建文件，运行无误后保存为“实验 8.1.py”并提交。
2. 绘制一个 9 行菱形。要求使用一个循环 `for` 来输出，保存为“实验 8.2.py”提交。

 观看章节 4.3.1 中的视频：实验 8.5，完成下面任务：

3. 印度人口：新建并编写代码，运行无误。代码保存为“实验 8.3.py”提交。

4. 选择使用任一种循环结构，完成数列计算： $s=1+1/2+1/4+\dots+1/20$ ，保存为“.py”提交。

挑战任务

挑战 5：闰年问题求解。计算并输出 2000-3000 年间的闰年及其个数。



“闰年分为普通闰年和世纪闰年。闰年的条件是：普通闰年：公历年份是 4 的倍数的，但不是 100 的倍数，为普通闰年（如 2004 年、2020 年就是闰年）。世纪闰年：公历年份是整百数的，必须是 400 的倍数才是世纪闰年（如 1900 年不是世纪闰年，2000 年是世纪闰年）。”

挑战：如何用上面循环结构求解下面的数列计算问题？自拟题目完成一个数列求和计算。

教学后记

1. 单循环绘制菱形的程序，有一定难度。启发后，同学们给出了多种解法，让人惊喜！

#e8.3 单循环绘制菱形

```
for i in range(1,10):  
    print(" "*(abs(5-i))+"*"*((10-2*(abs(i-5))-1)))
```

#e8.3 单循环绘制菱形

```
for i in range(1,10):  
    m=abs(i-5)  
    n=abs(-10+2*i)  
    print(" "*m+"*"*((9-n)))
```

#e8.3 单循环绘制菱形

```
s=5  
a=-5  
for i in range(1,11):  
    print(" "*abs(s)+"*"*(((1*abs(a)+5)*2)-1))  
    s=s-1  
    a=a+1
```

第六周（4 学时）

教材章节：

第 4 章 Python 控制语句

4.3.3 循环的嵌套

教学目的和要求：

- 1、掌握循环嵌套结构
- 2、掌握时间函数的使用

教学重点和难点

- 1、循环的嵌套结构的运行机制
- 2、常用的实例算法
- 3、time 常用函数的使用

教学方法与手段

理论课使用学习通投屏，通过线上活动与学生实时互动

实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

😊投票：随堂练习：循环结构（3 题）

课程引例：银行 ATM 密码，暴力破解法。



❓思考：6 位数字多少种密码？6 位数字+字母？8 位？如何提高密码强度？

穷举法：

1. 根据题目的条件确定答案的大致范围，并在此范围内对所有可能的情况逐一验证。
2. 若某个情况验证符合题目的条件，则为本问题的一个解；若全部情况验证后都不符合题目的全部条件，则本题无解。

解题步骤：

1. 确定搜索的范围：循环结构
2. 确定符合问题解的条件：分支结构
3. 缩小搜索范围，减少程序运行时间：控制循环次数

❓思考：刚才习题中哪个程序符合穷举法的要素？单循环+条件

投票 最多可选 1 项

设置

求 1~100 之间既能被 5 整除的数的个数
正确的程序代码是:

```
s = 0
for i in range(1,101,5):
    s = s + 1
print(s)
```

```
s = 0
for i in range(1,101):
    if i%5==0:
        s = s + 1
print(i,s)
```

思考: 闰年问题中的穷举法的要素? 单循环+条件 (上周的挑战任务 8)

百钱百鸡问题: 穷举法求解

“鸡翁一，值钱五，鸡母一，值钱三，鸡雏三，值钱一，百钱买百鸡，问翁、母、雏各几何?”



鸡翁一，值钱五，鸡母一，值钱三，鸡雏三，值钱一，百钱买百鸡，问翁、母、雏各几何?

求解过程如下:

①假设公鸡母鸡小鸡分别为A,B,C.

②满足的条件:

$$A + B + C = 100$$

$$A * 5 + B * 3 + C / 3 = 100$$

③确定取值范围:

A: 1 ~ 20
B: 1 ~ 33
C: 1 ~ 100

思考: 不定方程组求解? 穷举的变量是几个? 用几个循环来实现?

4.3.3 循环的嵌套

在循环语句中使用另一个循环语句称为循环的嵌套，也称为多重循环。for 语句和 while 语句都可以互相嵌套。

```
for i in range(1,4):
    for j in range(1,4):
        print("i 值为",i,";", "j 值为",j)
```

如果把两个循环都加上 else 语句，程序代码如下:

```
for i in range(1,4):
    for j in range(1,4):
        print("i 值为",i,";", "j 值为",j)
    else:
        print("内层循环结束。")
else:
    print("外层循环结束。")
```

程序的运行结果如下:

```
i 值为 1 ; j 值为 1
i 值为 1 ; j 值为 2
i 值为 1 ; j 值为 3
内层循环结束。
i 值为 2 ; j 值为 1
i 值为 2 ; j 值为 2
i 值为 2 ; j 值为 3
内层循环结束。
```

```
i 值为 3 ; j 值为 1
i 值为 3 ; j 值为 2
i 值为 3 ; j 值为 3
内层循环结束。
外层循环结束。
```

可以看出, 外层循环结束一次, 而内层循环一共结束了 3 次。

【解析实例】百钱百鸡问题, 穷举法求解:

编写代码如下:

```
for a in range(1,21):
    for b in range(1,34):
        for c in range(1,101):
            if a+b+c==100 and 5*a+3*b+c/3==100:
                print(a,b,c)
```

思考: 代码优化, 双循环如何实现?

编写代码如下:

```
for a in range(1,21):
    for b in range(1,34):
        c=100-a-b
        if 5*a+3*b+c/3==100:
            print(a,b,c)
```

4.4 break 语句和 continue 语句

4.4.1 break 语句

break 语句, 可以结束本层循环。break 语句要放在一个分支结构中。

```
for i in "abc":
    for j in "xyz":
        if j=="y":break
        print(i+j)
```

说明: 使用 break 的时候, 要注意这条语句属于哪个循环。

【例】#判断是否为素数

```
x=int(input("输入一个正整数: "))
for i in range(2,x):
    if x%i==0:
        print("%d 不是一个素数"%x)
        break
else:
    print("%d 是一个素数"%x)
```

思考: break 后, 循环共执行多少次?

【例】求两个数字的最小公倍数。

4.4.2 continue 语句

`continue` 语句只能结束本次循环的执行，即回到循环语句 `for` 或者 `while`，再次判断是否进行下一次循环。

```
for i in "abc":
    for j in "xyz":
        if j=="y":continue
        print(i+j)
```

【例】输入 10 名同学的分数求及格同学的均值，如果分数低于 60，则不计入计算中。

程序 1:

```
#example4.12.1
s,n=0,0
for i in range(1,11):
    score=eval(input("输入成绩"))
    if score>=60:
        s=s+score
        n=n+1
print("合格人数为:",n)
print("成绩平均值为:",round(s/n,2))
```

程序 2:

```
#example4.12.2
s,n=0,0
for i in range(1,11):
    score=eval(input("输入成绩"))
    if score<60:
        continue
    s=s+score
    n=n+1
print("合格人数为:",n)
print("成绩平均值为:",round(s/n,2))
```



思考： `continue` 后，循环共执行多少次？

注意：

(1) 无论是 `break` 语句还是 `continue` 语句，都只对当前层次的循环有影响，而对上层循环没有影响；

(2) 对于有 `else` 语句的循环：

`break` 语句，不会执行循环结构中 `else` 部分；

`continue` 语句，会执行循环结构中的 `else` 部分。

【解析实例】求三位阿姆斯特朗数（水仙花数）

【解析实例】包装箱的装配方案

【解析实例】输出“九九乘法表”

实验课教学过程设计：

🔗 观看章节 3.3.2 中的视频，完成实验九：循环嵌套与关键字，完成下面任务：

1. 分配包装箱：将代码补充完整，运行无误。代码保存为“实验 9.1.py”提交。
2. 打印乘法九九表（倒三角）：代码保存为“实验 9.2.py”提交。
3. 阿姆斯特朗数（四位数）：新建文件，运行无误后保存为“实验 9.3.py”并提交。
4. 百钱百鸡：穷举法求解，代码保存为“实验 9.4.py”提交。

挑战任务

挑战 6：鸡兔同笼问题：穷举法求解。

今有雉兔同笼，上有三十五头，下有九十四足，问雉兔各几何？

鸡：x 兔：y

解方程：x+y=35

2x+4y=94

穷举法：循环+条件，穷举变量

```
for x in range(1, 36):  
    y = 35 - x  
    if 2 * x + 4 * y == 94:  
        print(x, y)  
        break  
  
for x in range(1, 36):  
    for y in range(1, 36):  
        if x + y == 35 and 2 * x + 4 * y == 94:  
            print(x, y)  
            break
```

68 挑战任务

68 百钱百鸡问题

书中是这样叙述的：
今有雉兔同笼，上有三十五头，下有九十四足，问雉兔各几何？

“鸡兔同笼”是中国古代的数学名题之一。大约在1500年前，《孙子算经》中就记载了这个有趣的问题。



教学后记

1. 程序设计模板
2. break 和 continue 的用法区别，是重要的考点。
3. 百钱百鸡，鸡兔同笼问题均来自我国古代经典数学著作，可以作为思政点展开！
4. 引入案例：半月传视频片段，讲的楚国公子黄歇巧解不定方程的传说故事，同学们看的津津有味！这个案例用来引入“穷举法”算法程序（循环嵌套），效果不错。

第七周（4 学时）

教材章节：

- 第 3 章 Python 控制语句
- 3.5 结构化程序实例
- 2.4.2 常用标准函数-time
- 2.4.3 常用标准函数-math

教学目的和要求：

- 1、掌握几种常见的程序结构
- 2、掌握时间函数的使用
- 3、掌握数学函数的使用

教学重点和难点

- 1、结构化程序设计方法的基本思想
- 2、time 模块中的常用函数的使用
- 3、math 模块中的常用函数的使用

教学方法与手段

理论课利用学习通开展线上活动与学生的互动

实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

穷举法的范围与条件：根据题目的条件确定答案的大致**范围**，并在此范围内对所有可能的情况逐一验证。若某个情况验证符合题目的**条件**，则为本问题的一个解；若全部情况验证后都不符合题目的全部条件，则本题无解。

```
#穷举的程序模板  
for ....循环可嵌套  
for i in range(范围):  
    if 条件:  
        运算或输出  
    [break]
```

求百钱百鸡相的问题？
求阿姆斯特朗数的问题？
求鸡兔同笼的问题？
求箱子分配或奖项设置等等.....

求两个数的最小公倍数？
求两个数的最大公约数？
求判断素数问题？

程序实例：

【挑战 9】鸡兔同笼-单循环穷举

63 挑战任务	02 百钱百鸡问题
<p>书中是这样叙述的： 今有雉兔同笼，上有三十五头，下有九十四足，问雉兔各几何？</p> <p>“鸡兔同笼”是中国古代的数学名题之一。大约在1500年前，《孙子算经》中就记载了这个有趣的问题。</p>	

【例】四叶玫瑰数-多循环穷举（三循环求水仙花数，实训 P47）

```
for x in range(1000, 10000):
    a=x//1000
    b=x//100%10
    c=x//10%10
    d=x%10
    if a**4+b**4+c**4+d**4==x:
        print(x)
```

【例】最小公倍数-穷举法

```
x=eval(input("输入第一个数字: "))
y=eval(input("输入第二个数字: "))
if x<y:
    x, y=y, x
for i in range(x, x*y+1):
    if i%x==0 and i%y==0:
        print("%d和%d的最小公倍数为%d"%(x, y, i))
        break
```

😊 课堂挑战：两个数的最大公约数？

【例】判断一个数是否为素数问题-穷举法

概念 1：只能被 1 和它本身整除的自然数叫做素数，又称为质数。

概念 2：不能被 2 到 N-1 中所有自然数整除的数字 N 叫做素数。

经过证明，一个数字 N，与 [2,N/2]中的数字不发生整除，N 是素数。

#方法 1 判断是否为素数

```
n=int(input("输入一个正整数: "))
for i in range(2,n):
    if n%i==0:
        print("%d 不是一个素数"%n)
        break
```

```
else:
    print("%d 是一个素数"%n)
```

##方法 2

```
n=eval(input("输入一个正整数: "))
is_prime=True
for i in range(2,n//2+1):
    if n % i==0:is_prime=False
if is_prime==True:
    print("%d 是一个素数"%(n))
else:
    print("%d 不是一个素数"%(n))
```

😊 【挑战 10】：100-200 之间的所有素数？

time 模块

1. 时间表示与函数

time 模块提供与时间有关的函数。time 模块有三种时间的表示，时间戳、时间元组和格式化时间字符串。



图 3.2 三种时间表示之间的关系

- 时间戳 (timestamp): 时间戳表示从格林尼治时间 1970 年 1 月 1 日 00: 00: 00 开始到现在所经过的秒数，其值为 float 类型。
- 时间元组 (struct_time): 元组共有 9 个元素，它是三种时间表示的中间过程。
- 格式化时间 (format time): 格式化的结构字符串使时间更具可读性。包括自定义的时间字符串和标准字符串。

通过相应的函数可以实现三种时间表示之间的相互转换，常用的函数和功能描述见表 3.12 所示。

表 3.12 time 常用函数功能说明

函数	描述
<code>time()</code>	返回当前系统时间的时间戳（1970 元年后的浮点秒数）
<code>gmtime([secs])</code>	将一个时间戳转换为 UTC 时区的时间元组，无参数时，默认为当前系统时间
<code>localtime([secs])</code>	将一个时间戳转换为当前时区的时间元组，无参数时，默认为当前系统时间
<code>mktime(tupletime)</code>	将一个时间元组转换为时间戳
<code>strftime(fmt[,tupletime])</code>	将一个时间元组，按指定格式 <code>fmt</code> 转换为时间字符串，即 <code>string format time</code> 。无参数时，默认为当前系统时间
<code>strptime(str[,fmt])</code>	将一个时间字符串解析为时间元组，即 <code>string parse time</code>
<code>asctime([tupletime])</code>	将一个时间元组表示为 24 位的标准格式的时间字符串。无参数时，默认为当前系统时间
<code>ctime([secs])</code>	将一个时间戳表示为 24 位的标准格式的时间字符串。无参数时相当于 <code>asctime()</code>

函数应用实例：

```
>>> import time
>>> time.time()           #返回当前时间的时间戳，是一个浮点数
1611557088.1828556
>>> time.localtime()     #返回当前本地时间的时间元组，元组有 9 个元素
time.struct_time(tm_year=2021, tm_mon=1, tm_mday=25, tm_hour=14, tm_min=45,
```

```

tm_sec=20, tm_wday=0, tm_yday=25, tm_isdst=0)

>>> time.mktime(time.localtime())          #将当前本地时间元组转换为时间戳
1611557166.0

>>> time.asctime()                          #返回当前时间的标准字符串
'Mon Jan 25 14:46:49 2021'

>>> time.strftime('%Y 年%m 月%d 日')        #将当前时间显示为自定义格式的时间字符串
'2021 年 01 月 25 日'

>>>

```

2. 时间元组

时间元组由 9 个元素组成，每个元素有自己的名称和下标，下标从 0 开始。时间元组各元素的构成说明见表 3.13。调用时间元组的元素可以使用元素的下标或者元素的名称。如：

```

>>> import time
>>> t1=time.localtime()                    #返回本地时间的时间元组
>>> t1                                     #t1 为本地时间的时间元组
time.struct_time(tm_year=2021, tm_mon=1, tm_mday=25, tm_hour=15, tm_min=7,
tm_sec=51, tm_wday=0, tm_yday=25, tm_isdst=0)
>>> print(t1[0],t1[1],t1.tm_mday)         #显示输出时间元组中的元素
2021 1 25
>>>

```

表 3.13 时间元组各元素构成说明

下标	元素名称	描述
0	tm_year	年份（0000-9999），如 2020
1	tm_mon	月份（1-12）
2	tm_mday	一个月中的第几天（1-31）
3	tm_hour	小时（0-23）
4	tm_min	分钟（0-59）
5	tm_sec	秒（0-61），60、61 是闰秒
6	tm_wday	一个星期中的第几天（0-6），0 是星期一
7	tm_yday	一年中的第几天（1-366），366 是闰年
8	tm_isdst	1-夏令时,0-非夏令时,-1-不确定，默认值为-1

3. 时间格式符

格式化时间中的标准时间字符串长度固定为 24 位，显示的时间格式也是固定不变，如，'Mon Jan 25 15:14:11 2021'。如果想要自定义时间元组的显示格式，则需要使用带有格式符的时间字符串来完成。时间格式符的功能说明见表 3.14。时间格式符的使用实例如下：

```

>>> import time
>>> time.ctime()
'Mon Jan 25 15:35:02 2021'
>>> t2=time.localtime()                    #获取当前时间的时间元组 t2
>>> t2
time.struct_time(tm_year=2021, tm_mon=1, tm_mday=25, tm_hour=15, tm_min=35,
tm_sec=24, tm_wday=0, tm_yday=25, tm_isdst=0)

```

```
>>> time.strftime('%Y 年%m 月%d 日 %A',t2)      #按时间格式符的样式显示时间 t2
'2021 年 01 月 25 日 Monday'
>>>
```

表 3.14 时间格式符的功能说明

格式符	描述
%a	本地星期名称的简写（如星期四为Thu）
%A	本地星期名称的全称（如星期四为Thursday）
%b	本地月份名称的简写（如八月份为agu）
%B	本地月份名称的全称（如八月份为august）
%c	本地相应的日期和时间的字符串表示
%d	一个月中的第几天（01-31）
%H	一天中的第几个小时（24小时制，00-23）
%I	第几个小时（12小时制，0-11）
%j	一年中的第几天（001-366）
%m	月份（01-12）
%M	分钟数（00-59）
%p	本地am或者pm的相应符
%S	秒（00-61）
%U	一年中的星期数。（00-53星期天是一个星期的开始。）第一个星期天之前的所有天数都放在第0周
%w	一个星期中的第几天（0-6，0是星期天）
%W	和%U基本相同，不同的是%W以星期一为一个星期的开始
%x	本地相应日期字符串（如15/08/01）
%X	本地相应时间字符串（如08:08:10）
%y	去掉世纪的年份（00-99）两个数字表示的年份
%Y	完整的年份（4个数字表示年份）
%z	与UTC时间的间隔（如果是本地时间，返回空字符串）
%Z	时区的名字（如果是本地时间，返回空字符串）
%%	"%" 字符

小贴士

UTC(Coordinated Universal Time，世界协调时)：协调世界时又称世界统一时间、世界标准时间、国际协调时间，在我国为 UTC+8。DST (Daylight Saving Time) 即夏令时。

math 模块

math 模块提供数学相关的运算和函数，其中常用的函数如表 3.15 所示：

函数	返回值
ceil(x)	对x的向上取整，如ceil(4.1) 返回 5。
exp(x)	返回e的x次幂， 例如，exp(1)返回2.718281828459045。
fabs(x)	返回x的绝对值(浮点数)，如fabs(-10) 返回10.0。
floor(x)	对x的向下取整，如floor(4.9)返回4。
fmod (x,y)	返回x/y的余数（浮点数），如fmod(7,4)返回3.0。
log(x[,base])	返回x的自然对数，如log(e)返回1.0，可以用base参数改变对数的底，如，log(100,10)返回2.0。
log10(x)	返回10为基数的x的对数，如log10(100)返回 2.0。
max(x1, x2,...)	返回给定参数的最大值，参数可以为序列。
min(x1, x2,...)	返回给定参数的最小值，参数可以为序列。
pow(x, y)	返回x的y次幂，x**y 运算后的值。
round(x [,n])	返回浮点数x的四舍五入值，如给出n值，则代表舍入到小数点后的位数。
sqrt(x)	返回数字x的平方根，数字可以为负数，返回类型为实数，如sqrt(4)返回2.0。

函数的应用实例：

```
>>> import math
>>> print(math.fabs(-4),math.ceil(4.1),math.floor(4.9))
```

```
4.0 5 4
```

```
>>> print(math.pow(3,4),math.fmod(7,4),math.log(100,10))
```

```
81.0 3.0 2.0
```


实验课教学过程设计：

 观看章节 2.5.2 中的视频，完成实验六：常用标准函数库 `time`

1、测试时间函数： 填空

2、签到打卡机 1.0（自定义）：将代码补充完整，运行无误。代码保存为“实验 6.5.py”提交。

3、生日计算器：将代码补充完整，运行无误。代码保存为“实验 6.6.py”提交。

 观看章节 2.5.3 中的视频，完成实验六：常用标准函数库 `math`

4、测试数学函数

5、计算条幅长度：编写代码，代码保存为“实验 6.7.py”提交。

挑战任务

课堂挑战：求两个数字的最大公约数？

挑战 7：求 100-200 间的所有素数？

教学后记

1. `time` 时间函数，知识点较多而杂，不好理解。以实例引入，效果比较好。按知识章节，不好掌握和理解
2. `math` 函数库，学生自学效果不错，基本通过视频能够独立完成任务。
3. 程序设计实例，总结了几个模板和算法程序，让学生了解一些编程技巧。
4. 周末进行其中考试，自定义了一份试卷与其它班不同。有学生在网上出售其它班的标准答案，我班居然有学生上当的！

第八周（4 学时）

教材章节：

第 4 章 Python 数据结构

4.1 组合数据简介

4.2 列表

4.3 元组

教学目的和要求：

- 1、理解组合数据类型的基本概念
- 2、掌握列表的创建、访问和更新操作
- 3、掌握元组的创建和访问

教学重点和难点

- 1、序列类型的特点
- 2、列表的推导式
- 3、列表的访问与更新
- 4、列表的相关算法实现

教学方法与手段

理论课利用学习通开展线上活动与学生的互动

实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

课前小练：实训书后习题

【例】常用的数列计算

求 $1+1/3+1/5+\dots+1/99$?

求 100 内能被 3 或 5 整除数字之和?

5.1 组合类型简介

基本数据类型：包括数值类型、字符串类型和逻辑型等，只能表示单一的数据。

组合数据类型：是将多个基本数据类型或组合数据类型组织起来，能够更清晰地反应数据之间的关系，也能够更加方便地管理和操作数据。

Python 中的组合数据类型有 3 类：序列类型、映射类型和集合类型。



序列类型是一个元素向量，元素之间存在先后关系，通过索引序号访问。Python 中的序列类型主要有字符串类型（str），列表类型（list）和元组类型（tuple）。

映射类型是一种键值对，一个键只能对应一个值，但是多个键可以对应相同的值。通过键来访问值。Python 中唯一的映射类型是字典类型（dict）。

字典中的元素没有特定的顺序, 每个键都对应一个值。映射类型和序列类型的区别在于存储和访问方式的不同。另外, 序列类型只用整数作为序号, 映射类型则可以用整数、字符串或者其它类型的数据作为键, 而且键和键值有一定关联性, 也就是键可以映射到键值。

集合类型是一种无序不重复的元素集。集合可以进行交、并、差、补等运算, 含义与数学中的相应概念相同。

集合的元素类型只能是固定数据类型, 例如整型、字符串、元组等, 由于列表、字典等是可变数据类型, 因此不能作为集合中的数据元素。

5.2 列表

- 列表是一种可变序列数据类型。
- 数据元素放在一对中括号之间, 逗号分隔。
- 数据元素可以是基本数据类型, 也可以是组合数据类型或自定义数据类型的数据。

例如下面的列表都是合法的列表对象:

```
[1, 2, 3, 4, 5, 6, 7]
[1, "10086", "中国移动", True]
["日期", "中国", [2017, 5, 1]]
[1, [2, 3, 4], (5, 6), {"a": 7, "b": 8, "c": 9}]
```

5.2.1 创建列表

1. 通过赋值创建列表

使用方括号, 并且把方括号内每一个列表元素用逗号分开进行创建

使用 list() 函数进行创建。

```
>>> flist=['吉士','鸡翅','薯条','可乐','甜桶']
>>> list2=[1,2,3,4,5,6]
>>> list3=['Python','c++','Java','VB','Perl']
>>> list5=list(range(1,10,2))
>>> list6=list("沈阳师范大学")
```

2. 通过推导式创建列表

列表推导式使用非常简洁的方式生成满足特定需要的列表。语法格式如下:

[<表达式> for <变量> in <序列>]

```
>>> list1=[x*x for x in range(1,10)]
>>> list2=[i for i in list1 if i%2==0]
>>> list3=[i for i in range(100,1000) if (int(i/100))**3 +
(int(i/10)%10)**3 + (i%10)**3==i]
```

【例】常用的数列计算-使用推导式生成列表求解

求 $1+1/3+1/5+\dots+1/99$?

```
>>> list1=[1/i for i in range(1,100,2)]
>>> sum(list1)
```

求 100 内能被 3 或 5 整除数字之和?

```
>>> list4=[i for i in range(1,101) if i%2==0 and i%7==0]
>>> sum(list4)
```

😊 抢答: 下面哪些数列可以使用推导式生成? 具体的推导式是什么。

```

求1-100累加和？
求100的阶乘？
求1! +2! +...+100!
求1+1/2+1/3+...+1/100?
求1/2+2/3+3/4+...+99/100?
求1+1/3+1/5+...+1/99?
求11+22+33+...99?
求数列1,2,4,8,16...前10项之和?
求数列1,1,2,3,5,8...前20项之和?

```

```

100内能被2和7同时整除数列之和
100内能被3或5整除数字之和
100内能被3整除不能被5整除之和
100内能被3整除不能被5整除个数
公元1000年-2020年，多少个闰年？
    **闰年的条件？
.....

```

5.2.2 访问列表

1. 一维列表的访问

列表是一个有序序列，通过序号访问。列表的序号（索引）也是一个整数，并且可以由正向或逆向来访问列表。序号从 0 开始。

<列表名>[<索引>]

遍历列表：

```

food='吉士鸡翅薯条可乐甜桶'
flist=['吉士','鸡翅','薯条','可乐','甜桶']
ftup=('吉士','鸡翅','薯条','可乐','甜桶')

```

```

##遍历字符串
for item in food:
    print(item,end=',')

```

```

##遍历列表
for item in flist:
    print(item,end=',')

```

```

##遍历元组
for item in ftup:
    print(item,end=',')

```

2. 二维列表的访问

如果一个列表中的列表元素也是由列表构成的，那就构成了类似矩阵的二维结构。访问二维列表的格式为：

<列表名>[<索引 1>][<索引 2>]

<列表名>[<行号>][<列号>]

其中，索引 1 为二维列表的元素索引号，索引 2 为二维列表中索引 1 指向的列表元素中的元素索引号。

例如：

```

>>> a=[1,2,3,4,5]
>>> b=[6,7,8,9,10]
>>> c=[11,12,13,14,15]
>>> list=[a,b,c]
>>> list[1][3]
9
>>>

```

语句 `list[1][3]` 中，1 代表的是 `list` 列表的索引序号为 1 的列表元素，即列表 `b`；3 代表的是列表 `b` 中索引序号为 3 的列表元素，即数字 9。

【例 5.2】遍历二维列表。

3. 列表的赋值

列表也可以像变量之间的赋值那样，将一个列表的值赋给另一个列表，但是和基本变量赋值不同的是，列表的赋值只是将实际数据的地址引用进行了赋值，而不是将实际数据赋值一份给新的列表。如：

```
>>> list1=[1,2,3,4]
>>> list2=list1           #将 list1 赋值给 list2
>>> list2[0]=5            #更改 list2 中的某些元素值
>>> list2[1]=6
>>> list1,list2
([5, 6, 3, 4], [5, 6, 3, 4]) #list1 和 list2 同时改变。
```

可以看出，`list2` 使用了 `list1` 的存储地址。当改变 `list2` 的元素，`list1` 的元素同时改变。

```
>>> list2=[10,11,12,13]
>>> list1,list2
([5, 6, 3, 4], [10, 11, 12, 13])
```

对 `list2` 进行实际数据的赋值，`list2` 会使用独立的存储地址。不再和 `list1` 关联。

5.2.3 更新列表

列表是一种可变的数据类型，长度和元素值都可以变化。

1. 修改列表元素

修改列表元素值可以用赋值语句，语法格式如下：

<列表名>[<索引>]=<数据值>

其中，列表名为一个已经存在的列表，索引为该列表的正向或逆向索引序号，数据值为需要修改的任意数据值。当索引不在列表的索引范围内时，系统将出错并提示用户“索引超出范围”。例如：

```
>>> # 修改列表中的一个列表元素
>>> fruit=['苹果','香蕉','西瓜','桔子','桃子']
>>> fruit[0]='apple'
>>> fruit[-1]='peach'
>>> fruit
['apple', '香蕉', '西瓜', '桔子', 'peach']
>>> # 修改列表中的多个元素（也具有增加，删除功能）
>>> fruit[1:4]=['banana','watermelon','orange'] #当索引为切片时，值也要是列表
>>> fruit
['apple', 'banana', 'watermelon', 'orange', 'peach']
>>> # 当列表序号范围和赋值列表长度不相等时，就可以增加或删除列表。
>>> fruit[0:2]=['苹果','香蕉','梨','樱桃'] #用 4 个列表元素替换选定的 2 个列表元素
>>> fruit
['苹果', '香蕉', '梨', '樱桃', 'watermelon', 'orange', 'peach']
>>> fruit[-3:]=['柠檬'] #用 1 个列表元素替换选定的 3 个列表元素
>>> fruit
['苹果', '香蕉', '梨', '樱桃', '柠檬']
>>>
```

2. 添加列表元素

对列表赋值的时候可以添加数据元素, 还可以使用专门的函数或方法来对列表进行添加元素操作。

`append()`方法可以向列表的最后添加列表元素;

`insert()`方法可以向列表中指定索引序号处插入列表元素。

<列表名>.append(<数据值>)

<列表名>.insert(<索引>,<数据值>)

```
>>> fruit=['苹果','香蕉','西瓜','桔子','桃子']
>>> fruit.append("樱桃")
>>> fruit
['苹果', '香蕉', '西瓜', '桔子', '桃子', '樱桃']
>>> fruit.insert(2,"柠檬")
>>> fruit
['苹果', '香蕉', '柠檬', '西瓜', '桔子', '桃子', '樱桃']
>>>
```

需要注意的是, `append()`和`insert()`每次只能插入一个列表元素。

3. 删除列表元素

`remove()`用来按值删除列表中的列表元素

`del` 语句可以按索引序号删除列表中的列表元素

<列表名>.remove(<数据值>)

del <列表名> [<索引>] 或者 del <列表名>

其中, 列表名为一个已经存在的列表名称, 数据值为在列表中的列表元素, 索引为列表索引序号。`remove()`方法可以删除列表中第一个与数据值相等的列表元素, 如果要删除的列表中有 1 个以上相同的列表元素, 要多次使用 `remove()` 方法。`del` 语句可以按索引号删除列表中的列表元素, 也可以删除整个列表。

```
>>> fruit=['苹果','香蕉','西瓜','桔子','西瓜','桃子']
>>> fruit.remove('西瓜')          #remove 函数使用一次只能删除一个“西瓜”。
>>> fruit
['苹果', '香蕉', '桔子', '西瓜', '桃子']
>>> del fruit[:2]                  #del 语句删除列表中 0,1 索引序号的列表元素。
>>> fruit
['桔子', '西瓜', '桃子']
>>> del fruit                      #删除列表
>>> fruit                          #列表删除后, 再使用列表, 将出现“未定义”错误。
...
NameError: name 'fruit' is not defined
>>>
```

【例】菲波拉契数列前 20 项（斐波那契）？

1 1 2 3 5 8 13 21 34 55 89 144.....

费氏数列也称神秘数列。各个数字相互间维持着固定的关系与比率, 其顺序的两个数字相加后, 等于后面序列的数字。

2+3=5 3+5=8 8+13=21 21+34=55

费氏数列中前后两个数字相除的结果, 大约会接近 0.618 或 1.618

13/8=1.625 21/13=1.625 89/55=1.618 8/13=0.615
89/144=0.618 144/233=0.618

费氏数列中间隔一个数字相邻的两个数字的比率是 0.382

13/34=0.382 21/55=0.382 34/89=0.382 55/144=0.382

黄金分割是由费氏数列演变而来的, 所以费氏数列也称黄金数列, 将 2. 3. 5. 8. 13. 21. 34. 55. 89....称作黄金数。

列奥纳多·波纳契又名斐波那契，大约 1170 年出生于比萨，是一位富商的儿子。他是一位意大利数学家，被认为是中世纪最有才华的西方数学家。

斐波那契数列是指一组数字，该数字以数字 1 或数字 0 开头，后接另一个数字 1，然后该模式根据以下规则继续：数字（或斐波那契数字）将等于它们前面两个数字的总和（或之前两个数字的总和）。

相信对投资市场了解的股民一定都对斐波那契数列并不陌生，在实战中运用斐波那契数列去预判市场某个重要的阶段变盘时间点发生方向变化的概率。

股市是周期轮回的，时间周期是股价涨跌的奥秘，在周期循环理论中，无论如何寻找变盘点，斐波那契数列都是各种重要分析的基础之一，也被称为“神奇数字”，应用到股市里叫“斐波那契周期线”。

```
#常规解法
n1=1
n2=1
print(n1)
print(n2)
for i in range(3,21):
    n3=n1+n2
    n1,n2=n2,n3
    print(n3)
```

😊抢答：使用列表求解斐波拉契数列前 20 项（斐波那契）？

```
nlist=[1,1]
for i in range(2,20):
    nlist.append(nlist[i-1]+nlist[i-2])
print("数列前 20 项之和为：",sum(nlist))
for j in range(0,20):
    print(nlist[j])
```

5.2.4 列表常用的其他操作

表 5.1 列出了常用的列表操作符和函数方法。

操作	功能
list1+list2	连接两个列表，新列表的列表元素个数为 list1 和 list2 列表元素个数之和。
list1*n 或 n*list1	将 list1 重复 n 次。
x in list1	如果列表 list1 包含 x，则返回 True，否则返回 False。
x not in list1	如果列表 list1 中不包含 x 对象则返回 True，否则返回 False。
len(list1)	函数返回 list1 中列表元素的个数。
max(list1)	函数返回 list1 列表中元素的最大值，要求 list1 中列表元素类型相同。
min(list1)	函数返回 list1 列表中元素的最小值，要求 list1 中列表元素类型相同。
sorted(list1)	函数返回 1 个新列表，将 list1 中的元素进行排序，关键字 reverse=False 或省略为升序排列；关键字 reverse=True 为降序排列。
sum(list1)	如果 list1 中所有列表元素都是数字，函数返回列表元素之和。
list1.append(x)	将数据 x 添加到列表 list1 的最后。
List1.insert(i,x)	在列表 list1 的 i 号位置插入数据 x。

list1.clear()	删除列表 list1 中所有元素，保留空列表。
list1.copy()	产生 1 个与 list1() 相同的列表。
list1.extend(list2)	将列表 list2 中的元素添加到 list1 末尾。
list1.pop(i)	将列表的第 i 个元素退出列表，并返回该元素值。省略参数 i 退出最后一个元素。
list1.remove(x)	删除列表中第一个 x 元素
list1.reverse()	翻转列表
list1.sort()	对列表 list1 中元素进行排序。关键字 reverse=False 或省略为升序排列；关键字 reverse=True 为降序排列。

5.3 元组（观看视频完成）

元组的结构与列表类似，但元组的元素是不可变的。

元组一旦创建，不可以修改其元素，也不能添加或者删除元素。

元组使用一对小括号，在小括号内用逗号分隔开元组元素。

一个元组中的数据元素可以是基本数据类型，也可以是组合数据类型或自定义数据类型的数据。例如下面的元组都是合法的：

```
(1, 2, 3, 4, 5)
("Python", "C#", "Java", "Go", "VB")
()
(5,)
((1, 2, 3), ("a", "b", "c"), (True, False))
```

需要注意的是，只含有一个元素的元组，元素后面一定要有逗号，否则就是一个表达式而不是元组了。两个或者以上元素的元组，最后一个元素后可以有逗号，也可以没有。

5.3.1 创建元组

1. 通过赋值创建元组

创建元组可以通过使用小括号，并将小括号内每一个元素用逗号分隔来进行创建，或者使用 tuple() 函数进行创建。

```
>>> tup1=() #产生一个空元组
>>> tup1
()
>>> tup2=(1, 2, 3, 4, 5,)
>>> tup2
(1, 2, 3, 4, 5)
>>> tup3=('Python', 'c++', 'Java', 'VB', 'Perl')
>>> tup3
('Python', 'c++', 'Java', 'VB', 'Perl')
>>> tup4=tuple() #产生一个空元组，等价于 tup1=()
>>> tup4
()
>>> tup5=tuple(range(1, 10, 2)) #将 range 函数产生的序列变为元组
>>> tup5
(1, 3, 5, 7, 9)
>>> tup6=tuple("沈阳师范大学") #每字符作为元组中的一个数据元素
>>> tup6
('沈', '阳', '师', '范', '大', '学')
>>>
```

使用小括号创建元组的时候，小括号也可以省略，例如：

```
>>> tup=1,2,3,4,5
>>> tup
(1, 2, 3, 4, 5)
>>>
```

2. 通过推导式创建元组

元组也可以使用推导式来生成，不用中括号而用小括号：

(表达式 for 变量 in 序列)

与列表推导式不同的是，这种推导式叫做生成器推导式，它的结果是一个生成器对象，而不是元组。生成器对象可以使用 `next()` 函数依次访问其中的元素，也可以使用 `list()` 函数或者 `tuple()` 函数转化为列表或者元组后使用。例如：

```
>>> g=(x**2 for x in range(1,10))          #利用推导式产生生成器 g
>>> g
<generator object <genexpr> at 0x02D03900>    #访问 g 会提示在某地址有生成器
>>> next(g)                                #使用 next 函数访问生成器中第一个元素
1
>>> next(g)                                #使用 next 函数继续向下访问生成器中的元素
4
>>> next(g)
9
>>> tuple1=tuple(g)                        #使用 tuple() 函数将生成器中没访问的元素生成为元组
>>> tuple1
(16, 25, 36, 49, 64, 81)
>>>
```

列表推导式产生的列表与生成器推导式产生的生成器有着本质的区别：

(1) 列表推导式会直接形成一个新的列表，一次性把列表中的所有数据都放入到内存中，如果列表元素非常多，会占用很大的内存空间。生成器对象只是生成器推导式指定的算法，当访问生成器的时候才产生具体的元素，而不是一次性生成所有元素，所以生成器对象只占用很小的内存空间；

(2) 列表推导式生成的列表中的元素可以多次访问；生成器推导式产生的生成器中的元素，只能从前到后一个元素一个元素的访问，访问后即消失。上述例子中，当访问了生成器 `g` 中的元素 1、4、9 以后，再将生成器 `g` 转化为元组，就只能转化生成器中未访问的元素了，即把 16 到 81 转化为一个元组，生成器使用过 1 次以后就被释放掉，如需再次使用，只能再用生成器推导式重新产生，如：

```
>>> g=(x**2 for x in range(1,10))
>>> for i in g:
        print(i,end=",")
1,4,9,16,25,36,49,64,81,
>>> for i in g:
        print(i,end=",")
>>>
```

注意：第一个 `for` 语句循环遍历生成器 `g` 中的所有元素后，第二个 `for` 语句循环就没有任何结果了，因为生成器中的所有元素都已经访问完，不能回到生成器的第一个元素再次访问了，所以第二个循环没有任何显示。

当需要重复访问一个生成器中的元素时，可以根据需要用 `list()` 函数转为列表或使用 `tuple()` 函数转为元组后使用。

5.3.2 访问元组

元组属于不可变序列，无法为元组增加元素或者删除元素。不可以通过切片访问为元组增加或者删除元素。通过索引号访问元组元素的格式：

<元组名>[<索引>]


如果一个元组中的元素也是由列表、元组等构成的，也会构成二维结构。可以采用和访问二维列表相似的方式访问二维元组。

注意：虽然元组不可改变，但是如果元组中的某元素是列表等可变类型，则该元素是可以改变的。例如：

```
>>> a=[1,2,3]
>>> b=[4,5,6]
>>> c=(7,8,9)
>>> d=(a,b,c)           #元组 d 由 2 个列表和 1 个元组组成。
>>> d
([1, 2, 3], [4, 5, 6], (7, 8, 9))
>>> d[1]=[10,20,30]      #修改元组元素，会产生错误，因为元组是不可变的。
Traceback (most recent call last):
  File "<pysHELL#113>", line 1, in <module>
    d[1]=[10,20,30]
TypeError: 'tuple' object does not support item assignment
>>> d[1][0]=10           #修改 1 号元组元素中 0 号列表元素。
>>> d[1][1]=20           #修改 1 号元组元素中 1 号列表元素。
>>> d[1][2]=30           #修改 1 号元组元素中 2 号列表元素。
>>> d                    #修改成功。
([1, 2, 3], [10, 20, 30], (7, 8, 9))
>>> d[2][0]=70           #修改 2 号元组元素中 0 号元组元组，错误。
Traceback (most recent call last):
  File "<pysHELL#118>", line 1, in <module>
    d[2][0]=70
TypeError: 'tuple' object does not support item assignment
```

元组常用操作和函数方法与列表基本一致，本节不再赘述。

实验课教学过程设计：

 观看章节 4.2.1 中的视频，完成实验十：组合数据类型：列表和元组

- 1、填空题：测试列表相关函数的功能
- 2、求成绩的均值：将代码补充完整，运行无误。代码保存为“实验 10.1.py”提交。
- 3、使用列表：计算并输出“斐波拉契数列”的前 20 项之和。代码保存并提交。
- 4、使用列表：求数列 1, 1/2, 1/4, 1/8.....的前 10 项之和。代码保存并提交。

挑战任务

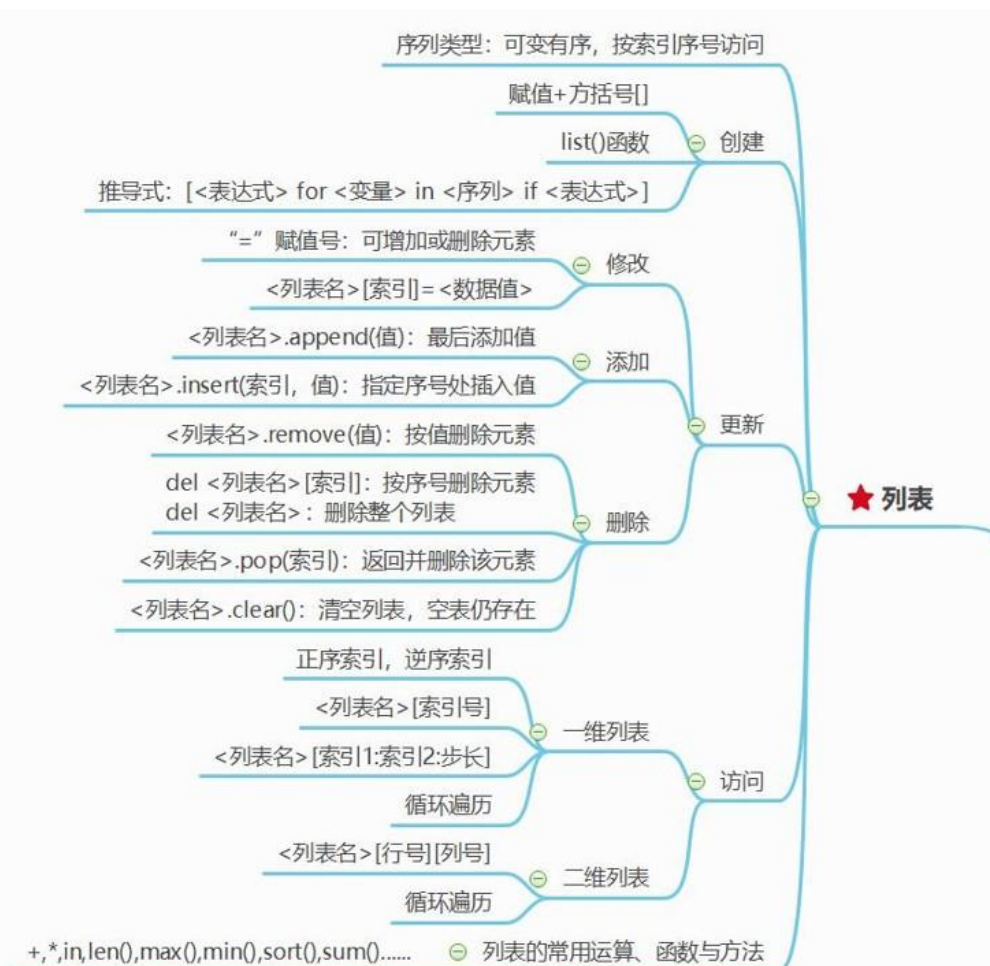
挑战 8：使用推导式，任意生成下面的一个数列，并计算？

求 $1+1/2+1/3+...+1/100$?
求 $1/2+2/3+3/4+...+99/100$?
求 $1+1/3+1/5+...+1/99$?
求 $11+22+33+...99$?
求数列 1,2,4,8,16...前 10 项之和?
求数列 1,1,2,3,5,8...前 20 项之和?

100 内能被 2 和 7 同时整除数列之和
100 内能被 3 或 5 整除数字之和
100 内能被 3 整除不能被 5 整除之和
100 内能被 3 整除不能被 5 整除个数
公元 1000 年-2020 年，多少个闰年?

教学后记

1. 二维列表掌握的不太好
2. 元组学生观看视频，自主学习。
3. 本周录播加学习通实时线上活动的形式进行的，从后台数据情况参与率挺高的。
4. 对列表推导式掌握不太好，两个随堂练习还是存在一些问题。



第九周（4 学时）

教材章节：

- 第 4 章 Python 数据结构
- 4.3 二维列表的访问
- 4.4 字典
- 4.5 集合

教学目的和要求：

- 1、理解二维列表的存储，掌握循环遍历访问的方法
- 2、掌握字典的创建、访问和更新操作
- 3、了解集合的创建、访问和更新

教学重点和难点

- 1、二维列表的遍历
- 2、字典的推导式
- 3、字典的访问，get 方法
- 4、字典的更新，修改、添加与删除，del, pop
- 5、字典的遍历，items(), keys(), values()

教学方法与手段

理论课利用学习通开展线上活动与学生的互动

实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

组合数据类型简述：列表与元组

▶ 组合类型简介

- | | |
|--------------|-----------|
| • 字符串 (str) | “可变有序” |
| • 列表 (list) | [] 可变有序 |
| • 元组 (tuple) | () 不可变有序 |
| • 字典 (dict) | { } 可变键值对 |
| • 集合 (set) | { } 可变无序 |
- food='吉士鸡翅薯条可乐甜桶'
 - flist=['吉士','鸡翅','薯条','可乐','甜桶']
 - ftup=('吉士','鸡翅','薯条','可乐','甜桶')
 - fdict={'吉士':15,'鸡翅':10,'薯条':7,'可乐':5,'甜桶':3}
 - fset={'吉士','鸡翅','薯条','可乐','甜桶'}

列表的小结

- 创建列表：[], list(), 推导式，数列计算
- 访问列表：列表名[<序号>]，列表名[<行号>][<列号>]，二维列表遍历
- 更新列表
 - 修改元素：列表名 [索引]=数据值
 - 添加元素：列表.append(值)，列表.insert(序号,值)
 - 删除元素：列表.remove(值)，del 列表[序号]

- 函数及运算: sum(列表名), len(列表名), sort, max, min, in, P91 表 5.1。

```
>>> flist1=[x**2 for x in range(1,11) if x%3==0]
>>> flist1
[9, 36, 81]
>>> flist=['吉士','鸡翅','薯条','可乐','甜桶']
>>> flist[0:2]=['牛奶']
>>> flist
['牛奶', '薯条', '可乐', '甜桶']
>>> flist.append('辣牛堡')
>>> flist.insert(3,'辣牛堡')
>>> flist
['牛奶', '薯条', '可乐', '辣牛堡', '甜桶', '辣牛堡']
>>> flist.remove('辣牛堡')
>>> flist
['牛奶', '薯条', '可乐', '甜桶', '辣牛堡']
>>> del flist[4]
>>> flist
['牛奶', '薯条', '可乐', '甜桶']
>>>
```

😊 随堂练习 1: 列表与元组 (上周的练习解析)

t[::-1]

s[4:]

😊 随堂练习 2: 列表与元组 (上周的练习解析)

3T:list1 = [m+n for m in 'AB' for n in 'CD']

4T:list1+list2

5T:index() 函数用于从列表中找出某个值第一个匹配项的索引位置。list.index(x[, start[, end]])

二维列表:

列表名[<行号>] [<列号>]

操作案例:

```
>>> drinks=[('可口可乐',9),('草莓奶昔',12),('纯牛奶',9),('豆浆',9),('红茶',9.5),('热朱古力',10),('奶茶',10.5)]
```

```
>>> drinks[1]
```

```
('草莓奶昔', 12)
```

```
>>> drinks[1][1]
```

```
12
```

```
>>> drinks[2]=("炒鸡蛋",20)
```

```
>>> drinks
```

```
[('可口可乐', 9), ('草莓奶昔', 12), ('炒鸡蛋', 20), ('豆浆', 9), ('红茶', 9.5), ('热朱古力', 10), ('奶茶', 10.5)]>>>
```

```
>>> drinks.insert(2,("纯牛奶",9))
```

```
>>> drinks
```

```
[('可口可乐', 9), ('草莓奶昔', 12), ('纯牛奶', 9), ('炒鸡蛋', 20), ('豆浆', 9), ('红茶', 9.5), ('热朱古力', 10), ('奶茶', 10.5)]
```

行/列	0	1
0	可口可乐	9
1	草莓奶昔	12
2	纯牛奶	9
3	豆浆	9
4	红茶	9.5
5	热朱古力	10
6	奶茶	10.5

【例】二维列表的遍历

```
drinks=[('可口可乐',9),('草莓奶昔',12),('纯牛奶',9),('豆浆',9),('红茶',9.5),('热朱古力',10),('奶茶',10.50)]
for i in range(len( ?? )):
    for j in range(len( ?? )):
        print(drinks[i][j],end=" ")
    print()
```

☺抢答: 请选择: A. 7 B. 2 C. drinks D. drinks[i] E. drinks[j]

字典: 映射类型, 可变无序键值对

字典是一种映射型, 由键值对组成。一个键对应一个值, 多个键可以对应相同的值。

字典类型和序列类型的区别在于存储和访问方式都不同。序列类型通常通过索引序号来访问, 而且只采用整数作为索引序号; 字典通过关键字访问值, “键”可以是任意不可变类型如数字、字符串、元组等。

字典将键值对放在一对大括号之间, 并使用逗号作为分隔, 每个键值对之间用冒号分隔。例如:

```
{ }
{1:"Python",2:"C++",3:"Java",4:"VB"}
{"No1":"Python","No2":"C++","No3":"Java","No4":"VB"}
{("三班",15):["张晓红","女"],("四班",22):["董强","男"],("五班",7):["张晓红","女"]}
```

创建字典:**1. 通过赋值语句创建字典**

```
>>> drinks1={'可口可乐':9,'草莓奶昔':12,'纯牛奶':9,'豆浆':9,'红茶':9.5,'热朱古力':10,'奶茶':10.50}
```

2. 通过 dict()函数创建字典

```
>>> drinks1=dict([('可口可乐',9),('草莓奶昔',12),('纯牛奶',9),('豆浆',9),('红茶',9.5),('热朱古力',10),('奶茶',10.50)])
```

3. 通过 fromkeys()创建字典

创建值都相同的字典: 将一个序列作为键, 然后指定统一的值。formkeys()函数也可以不指定值, 创建的字典默认为 None 空值。

```
>>> dict6={}.fromkeys(['优', '良', '中', '及'], "大于 60 分")
>>> dict6
{'优': '大于 60 分', '良': '大于 60 分', '中': '大于 60 分', '及': '大于 60 分'}
>>> dict7={}.fromkeys(['优', '良', '中', '及'])
>>> dict7
{'优': None, '良': None, '中': None, '及': None}
```

4. 通过推导式创建字典

推导式语法格式如下: {键:值 for 变量 in 序列}

```
>>> dict8 = {n: n**2 for n in range(1,5)}
>>> dict8
{1: 1, 2: 4, 3: 9, 4: 16}
```

☺抢答: **【实例】遍历字典**

```
drinks={'可口可乐':9,'奶昔':12,'纯牛奶':9,'红茶':9.5,'热朱古力':10,'奶茶':10.50}
for i in drinks.items():
```

```
print( ? )
```

显示如下, 请选择:

- A. "%s 的价格: %.2f"%(i[0],i[1]) B. "%s 的价格: %f"%(i[0],i[1])
C. "%s 的价格: %.2f"%(items[1],items[2]) D. "%s 的价格: %.2f"%(drinks[0],drinks[1])

```
===== RESTART: C:/Users/Administrator/Desktop/遍历drinks.py =====  
可口可乐的价格: 9.00  
奶昔的价格: 12.00  
纯牛奶的价格: 9.00  
红茶的价格: 9.50  
热朱古力的价格: 10.00  
奶茶的价格: 10.50  
>>> |
```

```
>>> drinks={'可口可乐':9,'奶昔':12,'纯牛奶':9,'红茶':9.5,'热朱古力':10,'奶茶':10.50}  
>>> drinks.items()  
dict_items([('可口可乐', 9), ('奶昔', 12), ('纯牛奶', 9), ('红茶', 9.5), ('热朱古力', 10), ('奶茶', 10.5)])  
>>> drinks.keys()  
dict_keys(['可口可乐', '奶昔', '纯牛奶', '红茶', '热朱古力', '奶茶'])  
>>> drinks.values()  
dict_values([9, 12, 9, 9.5, 10, 10.5])>>>
```

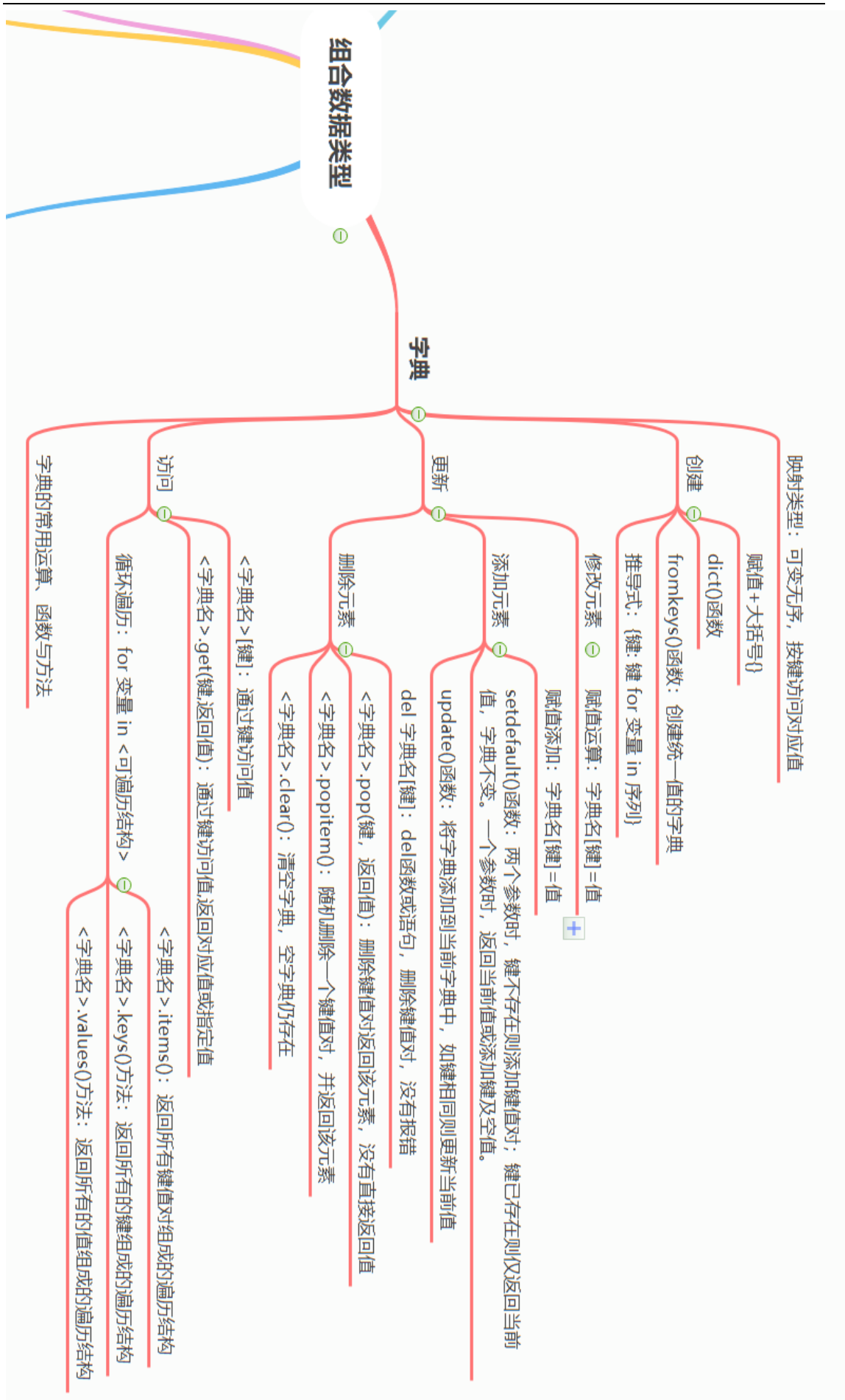


随堂练习 1: 字典的创建与访问


字典更新: 观看视频, 阅读教材, 完成知识导图。




随堂练习 2: 字典的更新



实验课教学过程设计:

 观看视频: 章节 5.4.1, 完成实验 11

1. 填空: 按要求完成字典相关语句的测试。
2. 程序填空: 遍历字典, 在环境下运行无误后, 将代码补充完整。
3. 程序填空: 输入关键字, 删除字典元素, 当输入 “n” 时结束, 显示现存元素。
4. 编程: 建立字典并遍历该字典, 运行无误后, 代码提交。

 观看视频: 章节 5.4.1, 完成实验 11:

5. 填空: 按要求完成集合的测试语句

挑战任务

挑战 9: 编程建立此二维列表, 遍历显示此列表? 遍历并计算平均成绩?

```
L1=["20200001","赵明",80]
L2=["20200002","钱小红",95]
L3=["20200003","孙强",98]
L4=["20200004","李丽",76]
L5=["20200005","陈艳庆",56]
L6=["20200006","刘雨农",88]
```

20200001	赵明	80
20200002	钱小红	95
20200003	孙强	98
20200004	李丽	76
20200005	陈艳庆	56
20200006	刘雨农	88

1. 建立此二维列表, 遍历显示此列表?
2. 遍历并计算平均成绩?

```
stlist=[L1,L2,L3,L4,L5,L6]
sum=0
for i in range(len(stlist)):
    for j in range(len(stlist[i])):
        print(stlist[i][j],'\t',end='')
    print()
    sum=sum+stlist[i][2]
print("平均成绩为: ",sum/len(stlist))
```

教学后记

1. 开始对列表进行了总结, 就上周的随堂练习展开, 强调: 几种组合类型区别, 列表的切片, 函数与方法用法上的区别
2. 二维列表的建立和访问: 二维序号、双循环遍历
3. 二维列表的课堂挑战, 同学们的回复令人惊喜, 谢峰的尤其惊艳, 代码巧妙地使用了推导式, 使得代码非常简洁, 很赞!
4. 字典和列表的知识导图, 发给学生进行知识梳理

第十周（4 学时）

教材章节：

第 6 章 字符串与正则表达式

6.1 字符串的格式化

6.2 字符串的基本操作

6.3 字符串函数与方法

教学目的和要求：

1. 熟悉字符串的格式化、索引和分片的具体方法
2. 掌握 Python 中字符串的基本运算符
3. 掌握 Python 中的字符串运算函数
4. 掌握 Python 中的字符串运算方法

教学重点和难点

字符串的格式化 format 方法

字符串的基本运算

字符串的函数与方法

字频与词频统计

教学方法与手段

理论课利用学习通开展线上活动与学生的互动

实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

😊抢答：字符串属于下面哪种数据类型？ A 序列 B 映射 C 集合

1.转义字符串：（Escape String）

指经过定义使得某些特定字符解释为另外一种含义，而不再是该字符本来的含义。

Python 加 “\” 就构成了转义字符。转义字符的作用有两个方面：

（1）用来将普通字符转为特殊用途。某些不可打印的字符，回车、换行等。表 6.1 转义字符及功能

（2）用来将特殊字符转换为本来的含义。将某些标识性的特定符号，如：单引号、双引号等，说明为普通字符。

```
>>> s='Let's do it'          #直接使用时报错
SyntaxError: invalid syntax

>>> s='Let\'s do it'         #使用转义字符\'时，可以得到正确字符串
>>> print(s)
Let's do it
>>>
```

2.原始字符串

字符串前加字母 “r” 或 “R” 时，表示字符串为原始字符串，此时解释器不对字符串中的转义字符进行转义，也就是说，原始字符串中的所有字符都具有原始含义。通常在表示文件路径、URL 和正则表达式中使用原始字符串。例如：

```
>>> path='C:\Windows\notepad.exe'    #path 字符串中的\n 会被转义为换行符
```

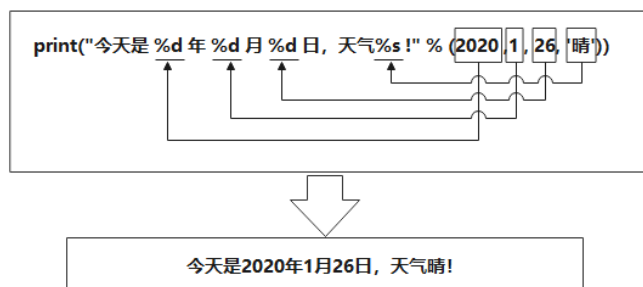
```
>>> path='C:\Windows\notepad.exe'      #字符串前加 r, 表示字符串为原字符
```

3. 字符串的格式化- %

格式说明符以 “%” 开头, 它与普通字符一起构成了 “格式字符串”。当使用 print 函数输出该格式字符串时, 对输出数据的完成显示样式的设置。格式字符串的格式:

表 6.2 常用的格式说明符

<格式字符串>% (<值 1>, <值 2>, ..., <值 n>)



4 字符串的格式化- format

format()方法进行字符串的格式化, 其语法格式如下:

<模板字符串>.format (<值 1>, <值 2>, ..., <值 n>)

说明:

此方法中, “模板字符串” 用于指定字符串的显示样式, “参数表” 用于指定要进行格式转换的项, 多项之间用逗号分隔。例如:

```
>>> print("今天是{}年{}月{}日, 天气{}!".format(2021, 1, 26, '晴'))
今天是 2021 年 1 月 26 日, 天气晴!
>>>
```

模板字符串中的 “{}” 是占位符。默认情况下, 占位符要与 format() 参数表中的数据项一一对应, 其中, 参数表中的每个数据项都按其排列顺序具有一个默认序号 (参数表可视为元组序号从 0 开始), 也可以在 “{}” 中指定序号, 来引用指定的数据项。例如:

```
>>> print("今天是{1}月{2}日{0}年, 天气{3}!".format(2021, 1, 26, '晴'))
今天是 1 月 26 日 2021 年, 天气晴!
>>>
```

模板字符串需要使用 “{}” 和 “:” 来指定占位符, 其完整的语法格式为:

{[序号] [: ([填充] 对齐) [符号] [宽度] [,] [精度] [类型]] }

序号: 引导符号	<填充> 用于填充的 单个字符	<对齐> <左对齐 >右对齐 ^居中对齐	<宽度> 槽的设定 输出宽度	<,> 千分位符 仅对整数 和浮点数	<精度> 浮点数小数 点位数或 字符串宽度	<类型> 整数类型 B, C, d, o, x, X 浮点数类型 E, E, f, %
-------------	-----------------------	-------------------------------	----------------------	-----------------------------	--------------------------------	---

【例 6.2】计算并输出斐波拉契数列前 N 项。

5. 字符串的基本操作-切片与索引

【例 6.4】查询月份英文缩写。

```
#example6.4 查询英文月份
st='''一月 Jan 二月 Feb 三月 Mar 四月 Apr 五月 May 六月 Jun
七月 Jul 八月 Aug 九月 Sep 十月 Oct 十一 Nov 十二 Dec'''
```

```
mon=int(input('input a month:'))
n=(mon-1)*5
print('%s 的英文简称为: %s'%(st[n:n+2],st[n+2:n+5]))
```

程序的运行结果如下：

```
>>>
===== RESTART:C:\Users\Desktop\example6.4.py =====
input a month:6
六月的英文简称为: Jun
>>>
```

6.字符串的基本运算:

字符串的基本运算符如表 6.4 所示。(+, *, in, >, <, ==)

7.字符串运算函数

常用的几个字符串的运算函数如表 6.5 所示

函数名	功能
len(s)	返回集合长度
chr(x)	返回整数 x 对应的 ASCII 字符
ord(s)	返回一个字符的 ASCII 码
str(x)	将数字转换为字符串

【例 6.5】密码的隐藏加密。给定一个由 26 个特殊符号组成的密文字符集 st0，密码加密规则是，将原密码中的字母转换为密文字符集中相应的符号，字母不区分大小写，除字母外的其他字符保持不变。例如，“A”=>“♠”、“b”=>“♡”。

```
#example6.5 密码隐藏加密
st0='♠♡♣♤♥♦♧♨♩♪♫♬♭♮♯♰♱♲♳♴♵♶♷♸♹♺♻♼♽♾♿'
print("这里是密文字符集: ",st0)
st1=input("输入原密码: ")
snew=""
for s in st1:
    if 97<=ord(s)<=122:
        snew=snew+st0[ord(s)-97]
    elif 65<=ord(s)<=90:
        snew=snew+st0[ord(s)-65]
    else:
        snew=snew+s
print("加密后的密码为: {:<}".format(snew))
```

8.字符串运算方法

字符串对象具有一些常用方法可以完成相应的运算。调用字符串方法的格式如下：

<字符串名>.<方法名> (<参数>)

字符串常用方法如表 6.6 所示。

字符串大小写转换方法的实例：

```
>>> s='this is a test!'
>>> s.upper()
'THIS IS A TEST!'
>>> s.capitalize ()
```

```
'This is a test!'
>>> s.title()
'This Is A Test!'
>>>
```

字符串拆分方法的实例:

```
>>> s='this is a test!'
>>> s.split()                                #将字符串以空格为分隔符拆分
['this', 'is', 'a', 'test!']
>>> s.split(sep=' ',maxsplit=1)             #将字符串以空格为分隔符拆分 1 次
['this', 'is a test!']
>>>
```

字符串替换和查找方法的实例:

```
>>> s='this is a test!'
>>> s.find('t')                             #查找字母 t 在字符串 s 中第一次出现的位置
0
>>> s.count('t')                             #查找字母 t 在字符串 s 中出现的次数
3
>>> s.replace('t','T',2)                     #将字符串的 t 替换 T, 替换 2 次
'This is a Test!'
>>>
```

【例 6.6】替换首字母为大写, 其它字母不变。



随堂练习: 字符串的运算函数与方法

【例 6.7】英文词频统计。统计给定原文中的每个单词出现的次数。

```
#example6.7
passage='Do not trouble trouble till trouble troubles you.'
print("统计原文: {:<}".format(passage))
pass1=passage.lower()           #字符串预处理
wlist=pass1.split()             #字符串拆分
counts={}                       #词频统计存入字典
for w in wlist:
    counts[w]=wlist.count(w)
print("统计结果: \n",counts)     #显示结果
```

程序的运行结果如下:

```
>>>
===== RESTART: C:\Users\Desktop\example6.7.py =====
统计原文: Do not trouble trouble till trouble troubles you.
统计结果:
{'do': 1, 'not': 1, 'trouble': 3, 'till': 1, 'troubles': 1, 'you.': 1}
>>>
```

说明: 词频统计的一般步骤分为:

1. 获取字符串: 通过赋值或文件读取的方式获取原字符串。
2. 字符串预处理: 将可能影响统计结果的内容进行预处理, 如标点、大小写状态等。
3. 字符串拆分: 用运算方法将字符串拆分为单词的列表。
4. 词频统计: 遍历单词列表, 并利用字典进行统计。

5.显示结果：将字典中的统计结果，整理后显示输出。

实验课教学过程设计：

 观看章节 6.1.1、6.2.1 中的视频，完成下面实验：

1. 单选：字符串格式化，函数与方法。
2. 填空：转义字符，字符切片，函数。
3. 阅读程序：密码加密。
4. 编程：国王的债务，输出数据的格式化，运行无误代码保存提交。
5. 程序填空：逆序输出字符串。

教学后记

1. 重点讲 format 应用实例，理解模式字符串的用法。
2. 重点讲密码加密的实例，理解字符串的遍历与查找算法。
3. 随堂练习相关函数与方法。
4. 词频统计未讲
5. 函数方法比较多，知识点零碎，还是要用实例串讲起来更好理解。不必精读语句，以实例引导学生自这为好，此处需要一个好的案例！

第十一周（4 学时）

教材章节：

第 8 章 Python 文件处理

8.1 文件的概念

8.2 文件的打开与关闭

8.3 文件的读写

8.4 文件的定位

教学目的和要求：

- 1、掌握文件概念与分类
- 2、掌握文件打开和关闭
- 3、掌握文件的读写与定位

教学重点和难点

文件的打开方式：open,close

文件的读：read,readline,readlines

文件的写：write,writelines

文件的定位：seek,tell

教学方法与手段

理论课利用学习通开展线上活动与学生的互动

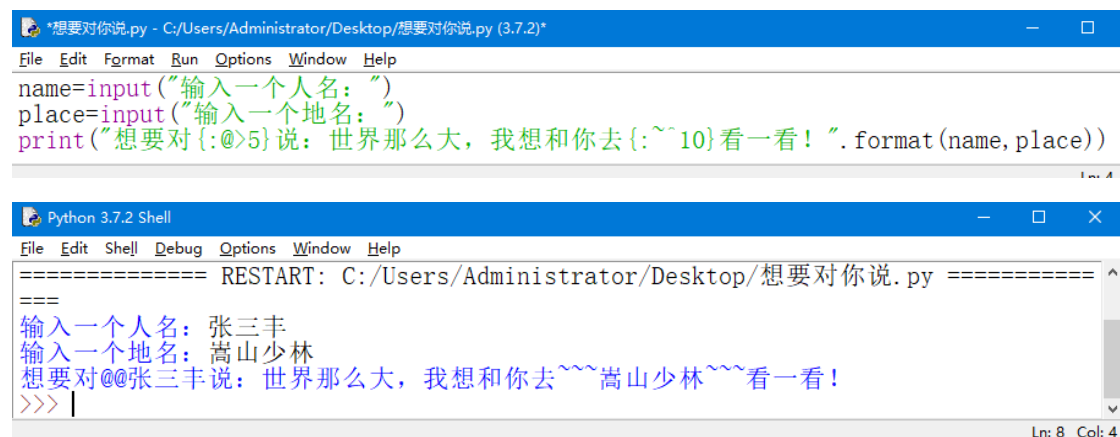
实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

1.字符串格式化输出：format 模板字符串

序号：	<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	槽的设置 输出宽度	千分位符 仅对整数 和浮点数	浮点数小数 点位数或 字符串宽度	整数类型 B,c,d,o,x,X 浮点数类型 E,e,f,%

【例】想要对你说。



```

*想要对你说.py - C:/Users/Administrator/Desktop/想要对你说.py (3.7.2)*
File Edit Format Run Options Window Help
name=input("输入一个人名：")
place=input("输入一个地名：")
print("想要对{:>5}说：世界那么大，我想和你去{:~10}看一看!".format(name,place))

Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/Administrator/Desktop/想要对你说.py =====
===
输入一个人名：张三丰
输入一个地名：嵩山少林
想要对@@张三丰说：世界那么大，我想和你去~~~嵩山少林~~~看一看！
>>>
Ln: 8 Col: 4
  
```

【例】计算并输出斐波拉契数列前 N 项。

2. 字符串的索引与查询

【例】查询月份英文缩写。字符串的-切片与索引

```
#example6.4 查询英文月份
st='''一月 Jan 二月 Feb 三月 Mar 四月 Apr 五月 May 六月 Jun
七月 Jul 八月 Aug 九月 Sep 十月 Oct 十一 Nov 十二 Dec'''
mon=int(input('input a month:'))
n=(mon-1)*5
print('%s 的英文简称为: %s'%(st[n:n+2],st[n+2:n+5]))
```

程序的运行结果如下:

```
>>>
===== RESTART:C:\Users\Desktop\example6.4.py =====
input a month:6
六月的英文简称为: Jun
>>>
```

3. 字符串逆序

有这样一句诗文: 春晚落花余碧草

此句诗文的逆序: 草碧余花落晚春

此句出自苏轼的《题织锦图》, 这种形式称为“回文诗”“回环诗”, 回文诗是我国古典诗歌中一种较为独特的体裁, 它是汉语特有的一种使用词序回环往复的修辞方法, 文体上称之为“回文体”。

```
>>>
===== RESTART: C:/Users
=
输入一句诗文: 春晚落花余碧草
回文: 草碧余花落晚春
>>>
```

【例】单句回文。😊选人

```
行=input("输入一句诗文: ")
print("回文: {}".format(行[-1::-1]))
```

4. 文件的古诗回文

唐宋以来, 以回文体入诗之风很盛。宋代文学家苏轼有《题织锦图回文》诗云:

春晚落花余碧草, 夜凉低月半梧桐。

人随雁远边城暮, 雨映疏帘绣阁空。

倒读则诗曰:

空阁绣帘疏映雨, 暮城边远雁随人。

桐梧半月低凉夜, 草碧余花落晚春。

回文诗是中国文化中心一个特殊形式, 也是世界上为中国仅有的一种艺术形式。

【例】整句回文。全诗以文件形式存储，文件读入后进行回文，并写入新文件。

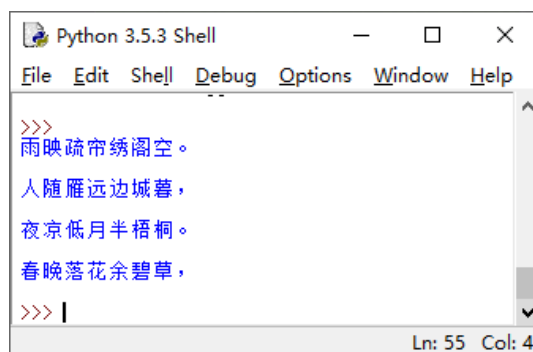
```
#0 整句回文
f=open("题织锦图.txt","r")
f1=open("题织锦图回文.txt","w")
整诗=f.readlines()
for 行 in 整诗[-1::-1]:
    print(行)
    f1.write(行)
f.close()
f1.close()
```

打开文件f，读的方式，“r”
打开文件f1，写的方式，“w”
在f中读取全文，形成列表=>整诗

遍历列表：整诗的逆序
显示输出
写入文件f1中
关闭文件f
关闭文件f1

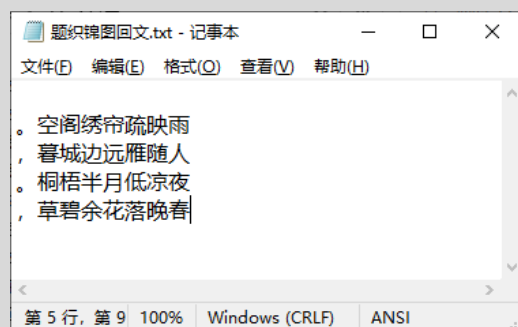
说明：f.readlines()以文件中的行为元素，生成列表。

```
>>> f=open("d:\题织锦图.txt","r",encoding='utf-8')
>>> f.readlines()
['春晚落花余碧草，\n', '夜凉低月半梧桐。 \n', '人随雁远边城暮，\n', '雨映疏帘绣阁空。 \n']
>>>
```



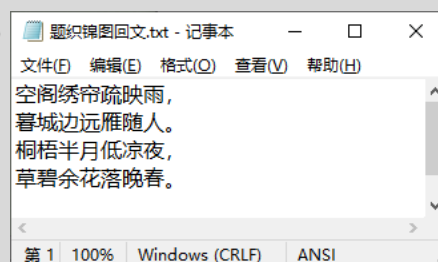
【例】整篇回文。😊程序填空，作业中任务完成如下，即可。

```
#2 整篇回文
f=open("题织锦图.txt","r")
f1=open("题织锦图回文.txt","w")
整诗=f.readlines()
for 行 in 整诗[-1::-1]:
    print(行[-1::-1])
    f1.write(行[-1::-1])
f.close()
f1.close()
```



【程序挑战】回文中的标点如何处理？😊

```
#3 整篇回文
f=open("题织锦图.txt","r",encoding='UTF-8')
f1=open("题织锦图回文.txt","w")
整诗=f.readlines()
i=1
for s in 整诗[-1::-1]:
    if i%2==0:
        s1=s[-3::-1]+"。 \n"
```



```
else:
    s1=s[-3::-1]+", \n"
    print(s1,end='')
    f1.write(s1)
    i+=1
f.close()
f1.close()
```

5.文件的操作

Python 对文件采用统一的操作步骤：“打开-操作-关闭”



文件的打开

<变量名> = open(<文件名>, <打开模式>)

模式	描述
r	以只读方式打开文件。文件指针置于文件的开头。这是默认模式
rb	以二进制只读方式打开文件。文件指针置于文件的开头。这是默认模式
r+	以可读可写方式打开文件。文件指针置于该文件的开头
w	以只写方式打开文件。文件存在，覆盖该文件。文件不存在，则创建新文件
w+	以可读可写方式打开文件。文件已存在，覆盖该文件。文件不存在，则创建新文件
a	以追加写方式打开文件。文件指针指向该文件末尾。该文件不存在，则创建新文件
a+	以追加写且可读方式打开文件。文件指针指向该文件末尾。文件不存在，则创建新文件

文件内容的读取

方法	含义
<file>.read(size)	从文件中读入整个文件内容，如果给出参数，读入前size长度的字符串或字节流
<file>.readline(size)	从文件中读入一行内容，如果给出参数，读入该行前size长度的字符串或字节流
<file>.readlines()	从文件中读入当前指针后的所有行，以每行为元素形成一个列表

文件内容的写入

方法	含义
<file>.write(<字符串>)	将字符串写入文件，文件不能“r”方式打开
<file>.writelines(<列表>)	将一个列表写入文件中，参数为字符串列表

指针的定位

方法	含义
<file>.seek(<偏移>,<起始>)	移动指针到指定位置，f.seek(0)，f.seek(0,2)，f.seek(5)
<file>.tell()	返回指针当前位置

文件的关闭

<文件对象名>.close()

【例】阅读程序

```
#example8.7
f=open("myfile.txt",'w+')
strlist=['abc','def','ghi']
f.writelines(strlist)
f.seek(0)
fc1=f.read(1)
f.seek(0,1)
fc2=f.read(1)
f.seek(5)
fc3=f.read(1)
f.seek(0,2)
f.write('xyz')
f.seek(0)
fc=f.read()
f.close()
print(fc1,fc2,fc3,fc)
f.close()
```

a	b	c	d	e	f	g	h	i	
0	1	2	3	4	5	6	7	8	9
文									文
件									件
头									尾



文件内容的遍历方式

【例】短歌行

```
f = open("短歌行.txt", "r")
for line in f:
    print(line)
f.close()
```

☺思考：找出其中含“月”的诗句？

☺思考：输入上句，对出下句？

实验课教学过程设计：

🔗观看章节 8.1.1 中的视频，完成实验二十：文本文件的操作

1. 读取遍历文件：将短歌行读入，并遍历显示。保存并提交。

2. 诗文对名：输入上句在短歌行中找出下句。保存并提交。
3. 回文诗：将文件中的古诗全文回文，写入新文件，代码保存提交。
4. 裴数列：format 格式输出数列，要求等宽整齐。

挑战任务

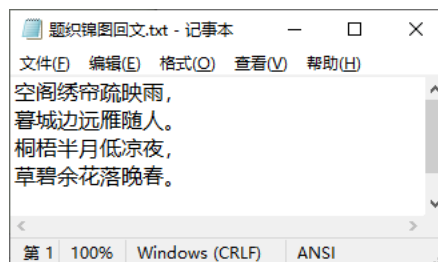
挑战 11：回文中的标点如何处理？

《题织锦图回文》

春晚落花余碧草，夜凉低月半梧桐。

人随雁远边城暮，雨映疏帘绣阁空。

完成效果如下图：



教学后记

1. 下面是案例中的用到的字符串

st0""0 零 1 壹 2 贰 3 参 4 肆 5 伍 6 陆 7 柒 8 捌 9 玖""

st1=""2020 年 5 月 26 日""

st0=""〇一二三四五六七八九""

2. 引入“回文诗”的案例，讲文件操作，学生很感兴趣！回文诗是中国文化中的特殊形式，也是中国仅有的一种艺术形式，可做为一个思政点。

3. 具体教学过程如下：

字符串逆序 => 苏轼的回文诗=> 单句回文（输入输出 IPO）=>回文诗的存储（文件的相关概念）=> 整句回文（文件按行读取）=>整诗回文（文件按字符读取）=>挑战任务：回文诗的标点如何处理？（设计数据结构）

宋昌泽的挑战任务回帖:



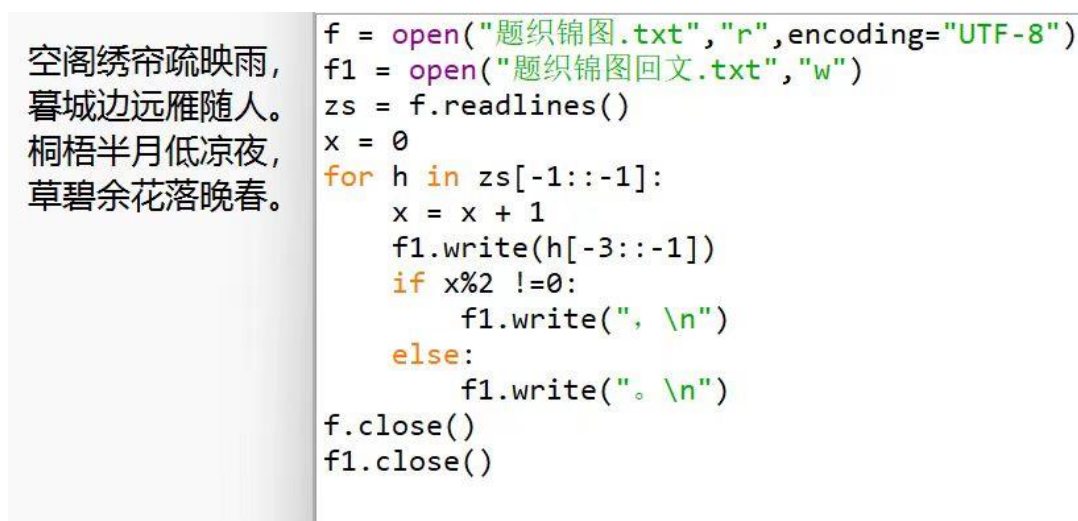
```
挑战11 2.py - C:/Users/2405656126/Desktop/挑战11 2.py (3.10.8)
File Edit Format Run Options Window Help
"""挑战11"""
f = open(r"C:\Users\2405656126\Desktop\题织锦图.txt", "r", encoding="UTF-8")
f1 = open("题织锦图回文.txt", "w")
整诗 = f.readlines()
n = 0
for 行 in 整诗[-1::-1]:
    f1.write(行[-3::-1])
    list1 = 整诗[n]
    f1.write(list1[7])
    f1.write(行[8::1])
    n = n + 1
f.close()
f1.close()
```

题织锦图回文

文件 编辑 查看

空阁绣帘疏映雨,
暮城边远雁随人。
桐梧半月低凉夜,
草碧余花落晚春。

赵文静的挑战任务回帖:



```
f = open("题织锦图.txt", "r", encoding="UTF-8")
f1 = open("题织锦图回文.txt", "w")
zs = f.readlines()
x = 0
for h in zs[-1::-1]:
    x = x + 1
    f1.write(h[-3::-1])
    if x%2 !=0:
        f1.write(", \n")
    else:
        f1.write("。 \n")
f.close()
f1.close()
```

空阁绣帘疏映雨,
暮城边远雁随人。
桐梧半月低凉夜,
草碧余花落晚春。

第十二周（4 学时）

教材章节：

第 4 章 Python 数据结构
列表与字典知识小结
综合应用实例：菲数列、

教学目的和要求：

- 1、理解列表和字典的区别
- 2、熟练列表与字典的访问与遍历
- 3、掌握列表与字典的更新

教学重点和难点

- 1、 列表的访问与遍历
- 2、 字典的访问与遍历
- 3、 使用数据结构进行运算

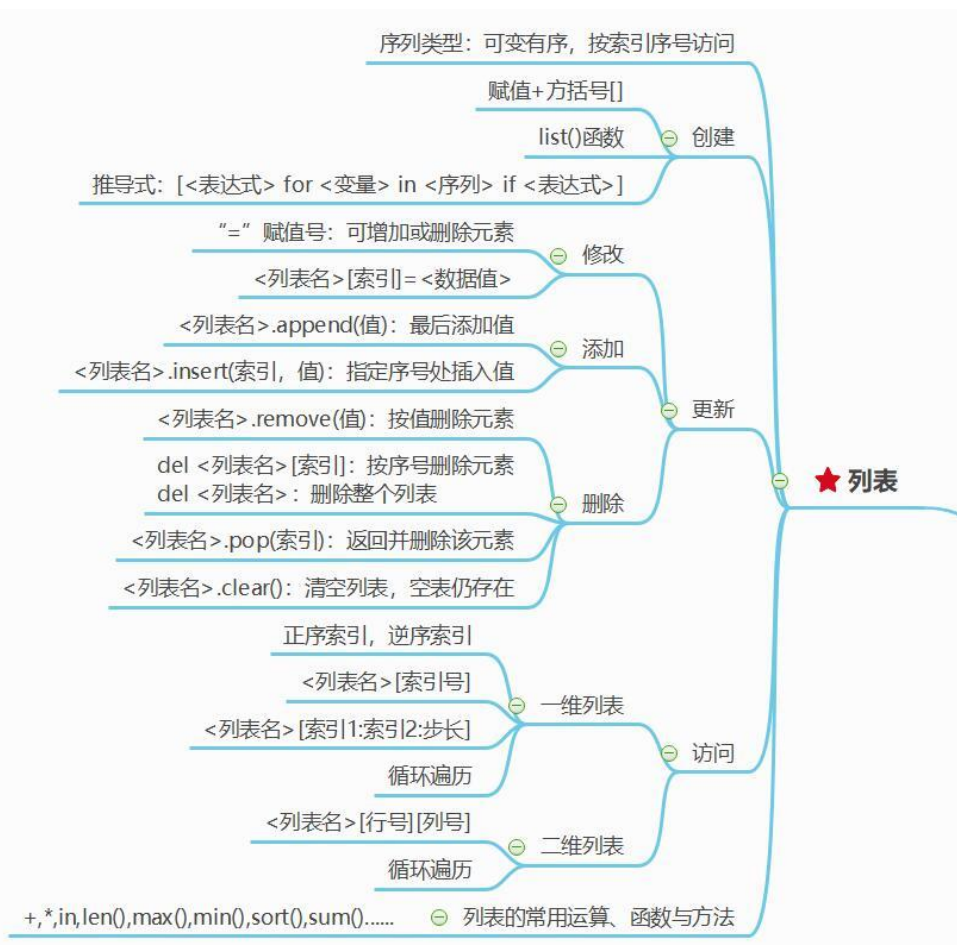
教学方法与手段

理论课利用学习通开展线上活动与学生的互动

实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

列表的知识图谱：



字典的知识图谱：



前课：列表与字典（挑战 12：二维列表计算）

本课内容：列表字典的应用实例

☺应用实例 1：数列计算问题

☺应用实例 2：自幂数

☺应用实例 3：菲波拉契数列（斐波那契）前 20 项

#常规解法：菲波拉契数列前 20 项

```

n1=1
n2=1
print(n1)
print(n2)
for i in range(3,21):
    n3=n1+n2
    n1,n2=n2,n3
    print(n3)
#列表求解：菲波拉契数列前 20 项
nlist=[1,1]
for i in range(2,20):
    nlist.append(nlist[i-1]+nlist[i-2])
print("数列前 20 项之和为：",sum(nlist))
for j in range(0,20):

```

```
print(nlist[j])
#使用字典求解菲波拉契数列前 20 项
ndict={1:1,2:1}
for i in range(3,21):
    ndict[i]=ndict[i-1]+ndict[i-2]
print("数列前 20 项之和为: ",sum(list(ndict.values())))
for x in ndict.items():
    print(x[0],x[1])
```

☺应用实例 4：素数计算问题

```
#方法 1：判断是否为素数
x=int(input("输入一个正整数: "))
for i in range(2,x):
    if x%i==0:
        print("%d 不是一个素数"%x)
        break
else:
    print("%d 是一个素数"%x)

#方法 2：判断是否为素数
n=eval(input("输入一个正整数: "))
is_prime=True
for i in range(2,n//2+1):
    if n % i==0:is_prime=False
if is_prime==True:
    print("%d 是一个素数"%(n))
else:
    print("%d 不是一个素数"%(n))
#使用列表求解


#使用字典求解
```

☺应用实例 5：随机套餐组合：实训 P62

```
import random as r
staple={'巨无霸':17,'双吉士':15.5,'麦辣鸡腿':15,'麦香鸡':9,'麦香鱼':8}
drinks={'可口可乐':9,'奶昔':12,'纯牛奶':9,'红茶':9.5,'热朱古力':10,'奶茶':10.50}
side={'麦辣鸡翅':9,'中薯条':9,'大薯条':11,'麦乐鸡':9,'玉米杯':7.5}
list1=list(staple.keys())
list2=list(drinks.keys())
list3=list(side.keys())
print("下面是为您随机生成的三种套餐组合: \n")
for i in range(3):
    x1=r.choice(list1)
    x2=r.choice(____?____)
    x3=r.choice(list3)
    y1=____?____
```

```
y2=____?____[x2]
y3=side[x3]
stem="{:*<6}{:*<6}{:*<6}\t 套餐金额为: {____?____}元"
print(stem.format(x1,x2,x3,y1+y2+y3))
```

实验课教学过程设计:

 观看视频：超星学习课程“第 5 章组合数据类型”中各章节视频，完成学习通中发布的练习：**作业 10-1~作业 10-4**：组合数据类型

挑战 12：遍历此字典，显示其中 10 元以上的所有单品及其价格？

```
drinks={'可口可乐':9,'奶昔':12,'纯牛奶':9,'红茶':9.5,'热朱古力':10,'奶茶':10.50}
```

教学后记

1. 列表和字典是重要的两个数据类型，本节用应用实例进行了总结和复习。
2. 课后布置了相关的习题进行强化。
3. 对编程的几个重要程序实例，使用常规方法、列表方法、字典方法分别编程求解，一是更行列表字典实例练习，另外让学生开拓思路，体会编程中的不同思维方法，学会使用组合数据结构提高编码效率。

第十三周（4 学时）

教材章节：

第 6 章 Python 函数和模块

6.1 函数的定义

6.2 函数的调用

6.3 函数的参数和返回值

教学目的和要求：

- 1、掌握函数的定义和调用
- 2、理解参数传递过程和函数的返回值
- 3、掌握位置参数、关键字参数、默认值参数

教学重点和难点

函数的定义格式

函数的调用方法

几种参数的传递方式

教学方法与手段

理论课利用学习通开展线上活动与学生的互动

实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

1.函数的定义

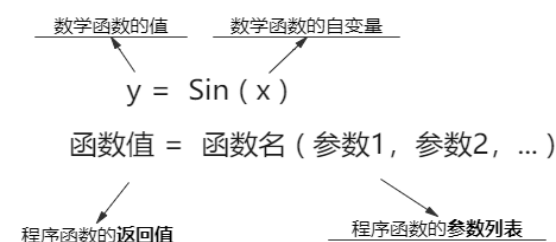
函数是将可以被反复使用的、用来实现单一或相关联功能的代码段封装、组织在一起，形成一个独立的程序单位，并将该程序单位定义一个相应的名称，这样的程序单位被称为函数，该名称被称为函数名。

- 系统函数：<函数名>([参数列表])，range([start],stop[,step])
 - 内置函数
 - 标准库函数
 - 第三方库函数
- 自定义函数
 - def <函数名>([参数列表]):
 - <函数体>

😊选人：说出几个内置函数，标准函数，第三方库函数？

2. 函数的调用

函数是具有特定功能的代码的封装，调用函数会执行函数中的代码，大多数函数都有返回值，有的函数没有返回值。函数的核心思想：**代码封装和复用**。



【例】生日歌

```
#example1:
print("Happy birthday to you!")
print("Happy birthday to you!")
print("Happy birthday to dear Jenny!")
print("Happy birthday to you!")
```

定义函数后，调用函数实现封装与复用：

```
#example2
def happy():
    print("Happy birthday to you!")
def happyto():
    print("Happy birthday to dear Jenny!")
##下面调用函数
happy()
happy()
happyto()
happy()
```

3. 函数的定义和参数

在 Python 中，定义函数的语法如下：

```
def <函数名> ([参数列表]):
    <函数体>
```

说明：

- 函数名：由用户定义的任何有效的标识符
- 圆括号：必须有无论是否有参数，参数之间用“，”分隔
- 形参：函数体内的参数被称为形式参数，简称形参
- 实参：调用该函数时，括号内的参数被称为实际参数，简称实参
- 冒号：括号后面的冒号必不可少
- 缩进：函数体至少有一条语句，且必须保持缩进

【例】生日歌，唱给 tommy 怎么办？

```
#example3
def happy():
    print("Happy birthday to you!")
def happyto(somebody):
    print("Happy birthday to dear {:}!".format(somebody))
##下面是调用函数
happy()
happy()
happyto("Tommy")
happy()
```

#example3	
1 def happy():	#定义函数
2 print("Happy birthday to you!")	
3 def happyto(somebody):	#形参数
4 print("Happy birthday to dear {:}!".format(somebody))	
happy()	#调用函数 (无参数) : 转去执行函数语句1和2后返回
happyto("Tommy")	#调用函数 (实参数) : 传入参数, 执行函数语句3和4后返回
happy()	#调用函数 (无参数) : 转去执行函数语句1和2后返回
happy()	#调用函数 (无参数) : 转去执行函数语句1和2后返回

😊思考：代码中的形参和实参是什么？各自的意义。

定义函数时形参无实际确定的值，在调用函数时实参要有确定的值，传入给形参后，函数以此值进行运算。这样，实参决定了生日歌想要唱给谁！

4. 函数的调用和返回值

函数调用格式如下：

<函数名>(<参数列表>)

调用函数的三种方式：

- (1) 直接调用函数名：函数语句完整执行一遍，往往不需要函数值；
- (2) 出现在表达式中：主程序使用函数返回值，参与表达式的计算；
- (3) 嵌套在另一函数中：函数的返回值被当作另外函数的实参使用。

【例】调用为方式（1），函数没有返回值。

```
happy()           #调用函数（无参数）
happyto("Tommy") #调用函数（实参数）
happy()
happy()
-----
```

函数的返回值：

```
return<表达式列表>
```

【例】调用方式（2），函数有返回值。

```
#example
def factroial(x):
    s=1
    for i in range(1,x+1):
        s=s*i
    return s

#主程序
y=int(input("请输入一个正整数: "))
if y<0:
    print("抱歉, 输入错误!")
else:
    print(y,"的阶乘是: ",factroial(y))
```

【例】已知阶乘函数，还能算什么？求组合数 $c(5,3)$?

组合数公式

$$c(n, m) = p(n, m) / m! = n! / ((n - m)! * m!)$$

公式描述：组合数公式是指从 n 个不同元素中，任取 $m(m \leq n)$ 个元素并成一组，叫做从 n 个不同元素中取出 m 个元素的一个组合。

深入了解：[公式](#) [性质](#) [递推公式](#) [算法举例](#)

来自百度百科

```
def factroial(x):
    s=1
    for i in range(1,x+1):
        s=s*i
    return s
```

```
#main
print('*****计算组合数 c(n,m),m<=n*****')
n=int(input("请输入正整数 n: "))
m=int(input("请输入正整数 m: "))
if m>n:
    print("输入错误 m<=n!")
else:
    result= 【factroial(n)/(factroial(m)*factroial(m-n))】
    print("组合数 c({0},{1})的值为:{2:3.0f}".format(【n,m,result】))
```

注意：程序填空时，要去掉【】。

5.几种参数形式

位置参数

默认情况下，Python 要求调用函数时参数的个数、位置和顺序要与函数定义中形参一致，这种参数也被称为位置参数。

关键字参数

在调用函数的时候，可以明确指定参数值传递给哪个形参，这样的参数被称为关键字参数。使用了关键字参数，可以不考虑形参与实参的位置和顺序一一对应。

默认值参数

Python 允许创建函数时为形参指定默认值，也可以通过显式赋值方式改变默认值。指定了默认值的形参也被称为可选参数，没有指定默认值的形参被称为必选参数。（单默认值，多默认值）

【例】绘制一个五角星

```
import turtle
turtle.color('red','yellow')
turtle.begin_fill()
for i in range(5):
    turtle.forward(200)
    turtle.left(144)    # 角度为正，前进方向逆时针转
turtle.end_fill()
```

☺思考：以此程序为原型，DIY 函数 star：大小、颜色、位置？

【例】def star(L)，定制大小

```
import turtle
def star(L):
    turtle.color('red','yellow')
    turtle.begin_fill()
    for i in range(5):
        turtle.forward(L)
        turtle.left(144)
    turtle.end_fill()
star(50)
```

☺思考：函数有返回值吗？调用方式？

【例】定制大小和颜色，def star(C1,C2,L)

```
import turtle
def star(C1,C2,L):
```

```

    turtle.color(C1,C2)
    turtle.begin_fill()
    for i in range(5):
        turtle.forward(L)
        turtle.left(144)
    turtle.end_fill()
x1=input("Line Color:")
x2=input("Fill Color:")
y=eval(input("Length of the Line:"))
star(x1,x2,y)

```

😊思考: 定制随机位置如何实现? `def star(c1,c2,x,y,l)` 挑战任务

位置参数: 实参与形参数量和顺序一致

```

def star(C1,C2,L)
    star(x1,x2,y)

```

关键字参数: 数量一致, 顺序可不同

```

def star(C1,C2,L)
star(C2=x2,C1=x1,L=y)

```

默认值参数: 指定了默认值的形参也被称为可选参数, 没有指定默认值的形参被称为必选参数。

注意: 第一个参数必须是必选参数

```

def star(L,C1="red",C2="yellow")
star(y,C2=x2)
star(y)
star(y,x1)
.....

```

😊思考: 不同的参数, 在调用时绘制的是什么颜色的星星? `star(y,x1)`

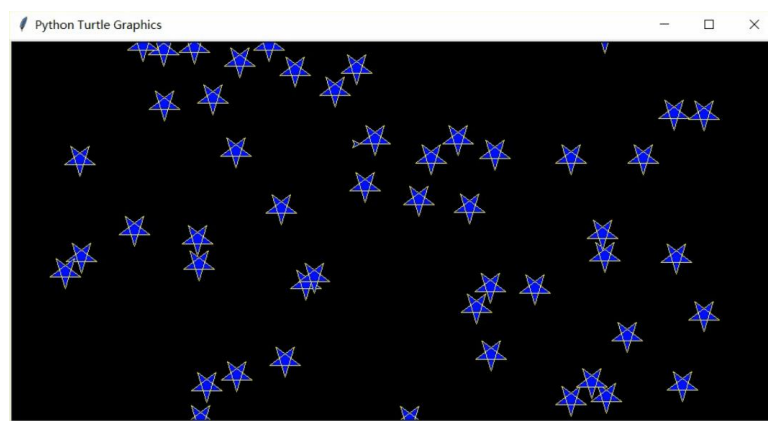
实验课教学过程设计:

📎观看章节 6.2.1 中的视频, 完成实验:

1. 单选题: 函数的定义与调用, 几种参数形式
2. 计算组合数: 自定义阶乘函数计算组合数, 代码保存并提交。
3. 绘制五角星: 定义函数, 将代码补充完整, 运行无误提交。
4. 海伦公式: 使用已给的函数, 完成计算任意三边的三角形面积。

挑战任务

挑战 14: 修改 `star` 函数, 使之可以绘制指定大小、颜色和位置的五角星。调用此函数, 在一块黑色画布上, 随机位置绘制 ≥ 100 颗星星, 让我们一起点亮星空吧!



教学后记

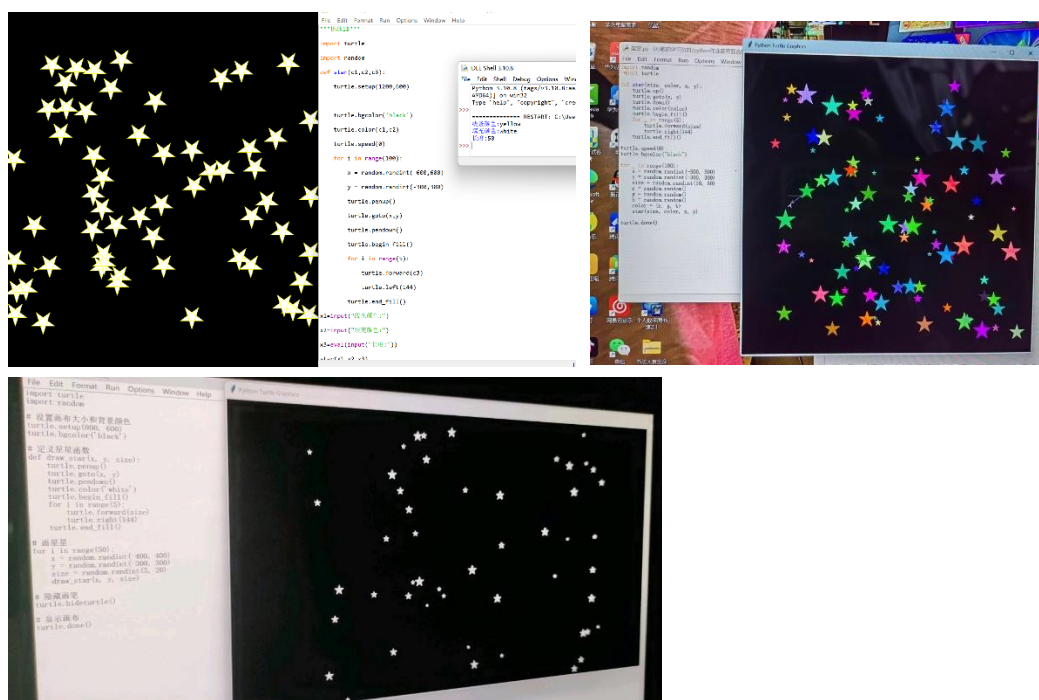
1. 使用递进式的教学案例，将自定义函数的相关概念联系起来，用了同学们最喜欢的绘图形式，最后的挑战任务是用 star 函数点亮星空，点亮了想象力！
2. 递进的过程如下：

绘制一个五角星：回顾前面程序

定制大小，def star(L): 自定义函数的定义，调用，参数

定制大小和颜色，def star(C1,C2,L): 位置参数、关键字参数、默认值参数

挑战任务：def star(c1,c2,x,y,l)：循环调用，随机位置，画布颜色等



第十四周（4 学时）

教材章节：

- 7.3.1 参数的传递的方式
- 7.6 lambda 函数（sorted,list.sort()）
- 7.4 变量的作用域

教学目的和要求：

- 1、了解程序结构的基本概念
- 2、掌握循环结构的基本语句
- 3、掌握时间函数的使用

教学重点和难点

- 结构化程序设计方法的基本思想
- 循环结构
- time 模块中的常用函数的使用

教学方法与手段

- 理论课利用学习通开展线上活动与学生的互动
- 实验课利用超星线上资源，开展线上自主学习，教师进行具体的辅助指导

理论课教学过程设计

7.3.1 参数传递的方式

主程序调用函数时，实参的值传递给形参。

参数传递的方式

传递不可变对象，在函数体中形参值的变化不影响到实参。

传递是可变对象，在函数中形参值的变化会影响到实参。

不可变对象：数值、字符、元组等

可变对象：列表、字典等

【例 7.4】传递不可变对象，形参的变化不会影响到实参。

```
#example7.4
def add(x):
    print("形参 x 的初始值是：",x)
    x+=1
    print("形参 x 的最终值是：",x)
y=4
print("实参 y 的初始值是：",y)
add(y)
print("实参 y 的最终值是：",y)
```

程序运行结果如下：

```
>>>
=====RESTART:C:\Python\example7.4.py=====
```

```

实参 y 的初始值是: 4
形参 x 的初始值是: 4
形参 x 的最终值是: 5
实参 y 的最终值是: 4
>>>

```

说明:

- (1) 主程序中定义了一个变量 `y`, 初值为 4, 即 `y` 指向数值 4;
- (2) 主程序中调用 `add()` 函数, 变量 `y` 作为实参传递给形参 `x`, 此时 `x` 也指向数值 4;
- (3) 在函数体内, 给 `x` 赋值为数值 5, 因为数值是不可变对象, 所以 `x` 指向了新的对象数值 5; 此时, 作为实参的变量 `y`, 仍然指向数值 4, 即 `y` 的值不变, 仍然为数值 4。

在参数传递过程中, 变量与数据的映射关系如图 7.4 所示:

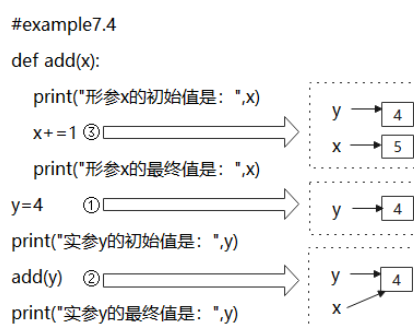


图 7.4 例 7.4 中参数传递的过程

【例 7.5】可变对象列表作为形参, 形参的变化会影响到实参。

```

#example7.5
def change(n):
    n.append(3)
    print('函数中 n 值: ',n)
m=[1]
print('调用函数前 m 的值: ',m)
change(m)
print('调用函数后 m 的值: ',m)

```

程序运行结果如下:

```

>>>
=====RESTART:C:\Python\example7.5.py=====
调用函数前 m 的值:  [1]
函数中 n 值:  [1, 3]
调用函数后 m 的值:  [1, 3]
>>>

```

说明:

- (1) 主程序中定义变量 `m` 为一个列表, 即, `m` 指向列表[1];
- (2) 主程序中调用 `change()` 函数, 变量 `m` 作为实参传递给形参 `n`, 此时 `m` 和 `n` 都指向列表[1];

(3) 在函数体内，给形参 `n` 执行 `append` 操作，因为列表是可变对象，所以该操作直接作用在原来的列表上，原列表变为 `[1,3]`，此时并不会生成新的对象，`m` 和 `n` 都指向列表 `[1,3]`。

【例 7.6】可变对象字典作为形参，形参的变化会影响到实参。

```
#example7.6
def change(x):
    x['院系']='法学院'
    x['专业']='法学'
a={'学号':'20024011','姓名':'杨晓霏','院系':'管理学院','专业':'行政管理'}
print(a)
change(a)
print(a)
```

程序运行结果如下：

```
>>>
=====RESTART:C:\Python\example7.6.py=====
{'学号': '20024011', '姓名': '杨晓霏', '院系': '管理学院', '专业': '行政管理'}
{'学号': '20024011', '姓名': '杨晓霏', '院系': '法学院', '专业': '法学'}
>>>
```

说明：

(1) 调用函数 `change()` 时，字典变量 `a` 作为实参传递给形参 `d`，因为字典变量是可变对象，形参的变化会直接影响到实参。

(2) 调用函数 `change()` 后，院系、专业被重新赋值，字典变量 `a` 中相应的值也发生同时改变，与形参 `x` 指向同一个值。

7.6 lambda 函数

- `lambda` 函数中的表达式只能为单个表达式，不允许包含复杂语句
- 表达式中可以包含函数，并支持默认值参数和关键字参数
- 表达式的计算结果相当于普通函数的返回值

```
>>> f=lambda x,y,z:x+y+z
>>> f(10,20,30)
60
>>> lm=lambda x,y=3,z=5:x+y+z
>>> lm(4)
12
>>> lm(6,z=5,y=4)
15
```

- `list.sort()`：仅对列表排序，且修改该列表

```
>>> list1=[1,13,89,237,100]
>>> list1.sort()
>>> list1
[1, 13, 89, 100, 237]
>>> list1.sort(reverse=True)
```

```
>>> list1
[237, 100, 89, 13, 1]
>>>
```

- **sorted():**

- 对序列和字典排序, 生成一个新的列表
- 对 dict 默认按 key 值排序, 返回按 key 值排序的 list
- `sorted(对象[,key=lambda...][,reverse=True|False])`
- 升序: `sorted(对象)`
- 降序: `sorted(对象,reverse=True)`
- `sorted(对象[,key=lambda...][,reverse=True|False])`

```
>>> b = ['aa', 'BB', 'YY', 'zz', 'CC']
>>> sorted(b)
['BB', 'CC', 'YY', 'aa', 'zz']
>>> sorted(b, key=lambda x: x.lower())
['aa', 'BB', 'CC', 'YY', 'zz']
>>>
```

key 参数为一个函数, 此函数返回一个值用来进行比较

多维列表排序:

```
>>> st = [('john', 'A', 15), ('jane', 'B', 12), ('dave', 'B', 10)]
>>> sorted(st, key=lambda x: x[2])
[('dave', 'B', 10), ('jane', 'B', 12), ('john', 'A', 15)]
>>> sorted(st, key=lambda x: x[2], reverse=True)
[('john', 'A', 15), ('jane', 'B', 12), ('dave', 'B', 10)]
```

字典排序:

```
>>> d = {"沈阳":100, "上海":155, "北京":145}
>>> sorted(d)           #默认按 key 排序
['上海', '北京', '沈阳']
>>> d.items()           #此方法可将字典转换为二维列表
dict_items([('沈阳', 100), ('上海', 155), ('北京', 145)])
>>> sorted(d.items(), key=lambda x: x[1], reverse=True)
[('上海', 155), ('北京', 145), ('沈阳', 100)]
```

7.4 变量的作用域

- 变量的作用域指的是变量的作用范围
- 全局变量 (Global variables) 和局部变量 (Local variables)。

1. 全局变量

(1) 在主程序中定义的变量是全局变量, 确切地说, 在主程序中出现在赋值语句等号左侧的变量是全局变量。

(2) 用 `global` 声明的变量是全局变量, 多个变量中间用逗号隔开。

(3) 没有用 `global` 语句声明, 但出现在函数内赋值号右侧且不是形参数的变量是全局变量, 或者说, 如果在函数体内只是引用了某个变量的值而没有为其赋新值, 则该变量为全局变量。

- 全局变量作用域是整个程序, 都可被引用。

2. 局部变量

(1) 函数的形式参数是局部变量；

(2) 函数体内部赋值号左侧出现的变量是局部变量，或者说，只要在函数体内有为变量赋值的操作，该变量即为局部变量。

- 局部变量的作用域在仅该函数体内部
- 一旦离开函数，变量将失效，不可引用

【例】变量的作用域实例 1。

```
#example6.20
a,b=3,6 #a、b 定义在主程序中，为全局变量
def func_scope():
    c=a*b      #c 在函数体内赋值语句等号左侧，是局部变量
    d=c*2      #d 也是局部变量
    print(a,b,c,d) #在函数体内部，输出局部变量 c、d 的值
func_scope()   #在主程序中调用函数
print(a,b)     #在主程序中输出全局变量 a、b 的值
```

```
>>>
=== RESTART: C:/Us
10 20 200
3 6
>>>
```

【例】变量的作用域实例 3。

```
#example6.22
a,b=3,6          #a、b 定义在主程序中，为全局变量
def func_scope():
    global a      #声明变量 a 为全局变量
    a=10          #这里的变量 a 与主程序中的变量 a 为同一变量
    b=20          #b 定义在函数体内部的赋值语句等号左侧，为局部变量
    c=a*b         #c 出现在函数体内赋值语句等号左侧，是局部变量
    print(a,b,c)
func_scope()
print(a,b)        #输出全局变量 a、b 的值
```

```
>>>
=== RESTART: C:\Users
10 20 200
10 6
>>>
```



随堂练习：

Try Now!

x=10	x=10	x=10
def fun(x):	def fun():	def fun(x):
x+=2	global x	x+=2
fun(x)	fun()	return x
print(x)	print(x)	print(fun(x))

实验课教学过程设计：



本周实验任务：

1. 熟悉模拟考试系统环境
2. 了解机考试卷方案相关说明
3. 辅导与答疑

教学后记

1. 发布平时成绩
2. 提醒学生平时成绩相关事项