

1 Python编程（一级）

1.1 考试标准

1.1.1 了解Python有多种开发环境，熟练使用Python自带的IDLE开发环境，能够进行程序编写、调试和分析，具备使用Python开发环境进行程序设计的能力：

- 1) 了解Python常见的几种编程环境：IDLE、Visual Studio Code、Jupyter Notebook；
- 2) 熟悉IDLE的操作过程，会打开IDLE，会新建文件、保存文件；
- 3) 熟练掌握使用IDLE进行编程，会修改文件、运行文件等操作；
- 4) 熟悉IDLE的两种开发模式，会在不同模式下进行切换；
- 5) 了解Python的版本号和目前最常用的Python版本；

1.1.2 熟悉Python程序编写的基本方法：

- 1) 理解“输入、处理、输出”程序编写方法；
- 2) 掌握Python的基本格式，编写程序时会合理的使用缩进、注释、字符串标识；
- 3) 掌握变量基本概念，会使用变量，并且掌握变量名的命名和保留字等基本语法；
- 4) 理解字符串、数值型变量，会对变量类型进行转换；
- 5) 掌握并熟练编写带有数值类型变量的程序，具备解决数学运算基本问题的能力；
- 6) 理解比较表达式、运算符、逻辑运算的基本概念，掌握Python编程基础的逻辑表达式。

1.1.3 具备基本的计算思维能力，能够完成较简单的Python程序编写：

- 1) 理解顺序结构语句的特点和写法，能够完成简单顺序结构的程序；
- 2) 理解比较表达式、运算符、逻辑运算的基本概念，掌握Python编程基础的逻辑表达式；
- 3) 知道第三方库turtle的功能，会导入该库文件，掌握它的一些简单使用方法：前进，后退，左右转，提落笔，画点，画圆。

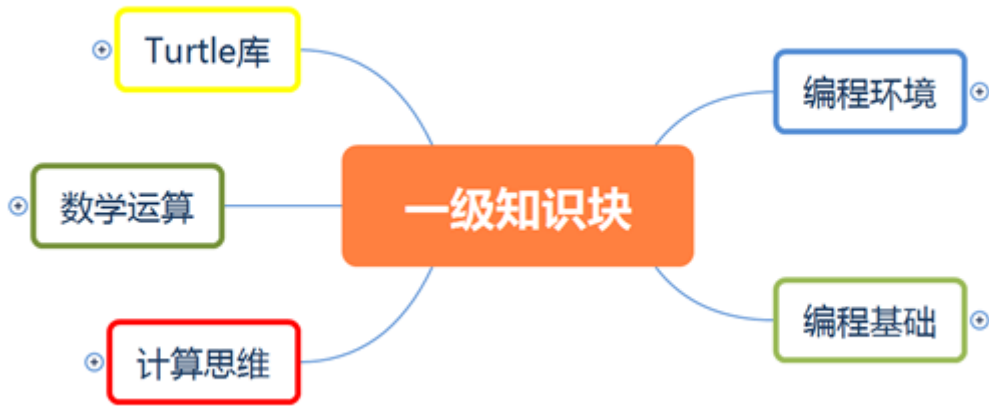
1.2 考核目标

让学生掌握基本的Python编程相关知识和方法，会使用IDLE进行编程，熟悉Python的基本语法规则，会用turtle库完成简单的顺序执行的Python程序，能够解决较为简单的问题。

1.3 能力目标

通过本级考核的学生，对Python编程有了基本的了解，熟悉至少一种Python编程环境的操作，会编写含有变量及库文件的基本程序。具备用计算思维的方式解决简单的问题能力。

1.4 知识块



1.5 知识点描述

序号	知识块	知识点
1	编程环境	Python版本、IDLE操作、其他编程环境、新建文件、文件保存、代码缩进、代码注释、程序运行
2	编程基础	print()语句、双引号和单引号、字符串及数值类型转换、input()语句、变量的命名和使用、保留字
3	Turtle库	导入库文件、画布设置、画笔设置、前进、后退、左转、右转、提笔、落笔、到达指定坐标、画点、画圆等命令
4	数学运算	+、-、*、/运算；赋值运算符；==、<、>、<=、>=、!=运算符；and、or、not运算符；运算符的优先顺序；
5	计算思维	能编写顺序执行的程序、能分析简单逻辑运算和比较运算中的结果并且会使用这些结果



1.6 题型配比及分值

知识体系	单选	判断	编程
编程环境 (10分)	6	4	0
编程基础 (26分)	10	12	4
Turtle库 (24分)	14	4	6
数学运算 (30分)	18	0	12
计算思维 (10分)	2	0	8
分值	50	20	30
题目数量	15	10	2

2 编程环境

2.1 版本

两个版本，分别是什么？

是否兼容？

如何查看Python版本？

我们可以在命令窗口(Windows 使用 win+R 调出 cmd 运行框)使用以下命令查看我们使用的 Python 版本：

```
python -V
```

2.1.1 Python2.x与3.x版本区别 (非考点)

变化1: print 函数 print语句没有了，取而代之的是print()函数；

变化2: 由于 Python3.X 源码文件默认使用utf-8编码，所以可以直接使用中文作为编码名或者进行中文字符的处理；

变化3: 除法运算，在python 2.x中/除法就跟我们熟悉的大多数语言，比如Java啊C啊差不多，整数相除的结果是一个整数，把小数部分完全忽略掉，浮点数除法会保留小数点的部分得到一个浮点数的结果。而在python 3.x中/除法不再这么做了，对于整数之间的相除，结果也会是浮点数。

变化4: 在 Python 3 中处理异常也轻微的改变了；

变化5: 在 Python 3 中，range() 是像 xrange() 那样实现遍历，以至于一个专门的 xrange() 函数都不再存，在（在 Python 3 中 xrange() 会抛出命名异常）。

变化6: 不等运算符：Python 2.x中不等于有两种写法 != 和 <>Python 3.x中去掉了<>，只有!=一种写法；

变化7: 去掉了repr表达式``

变化8: 修改了好多命令

综上所述，千万不要再学Python2了。

2.2 编程环境的搭建

2.2.1 Python3 下载

Python 官网: <https://www.python.org/>

你可以在以下链接中下载 Python 的文档, 你可以下载 HTML、PDF 和 PostScript 等格式的文档。

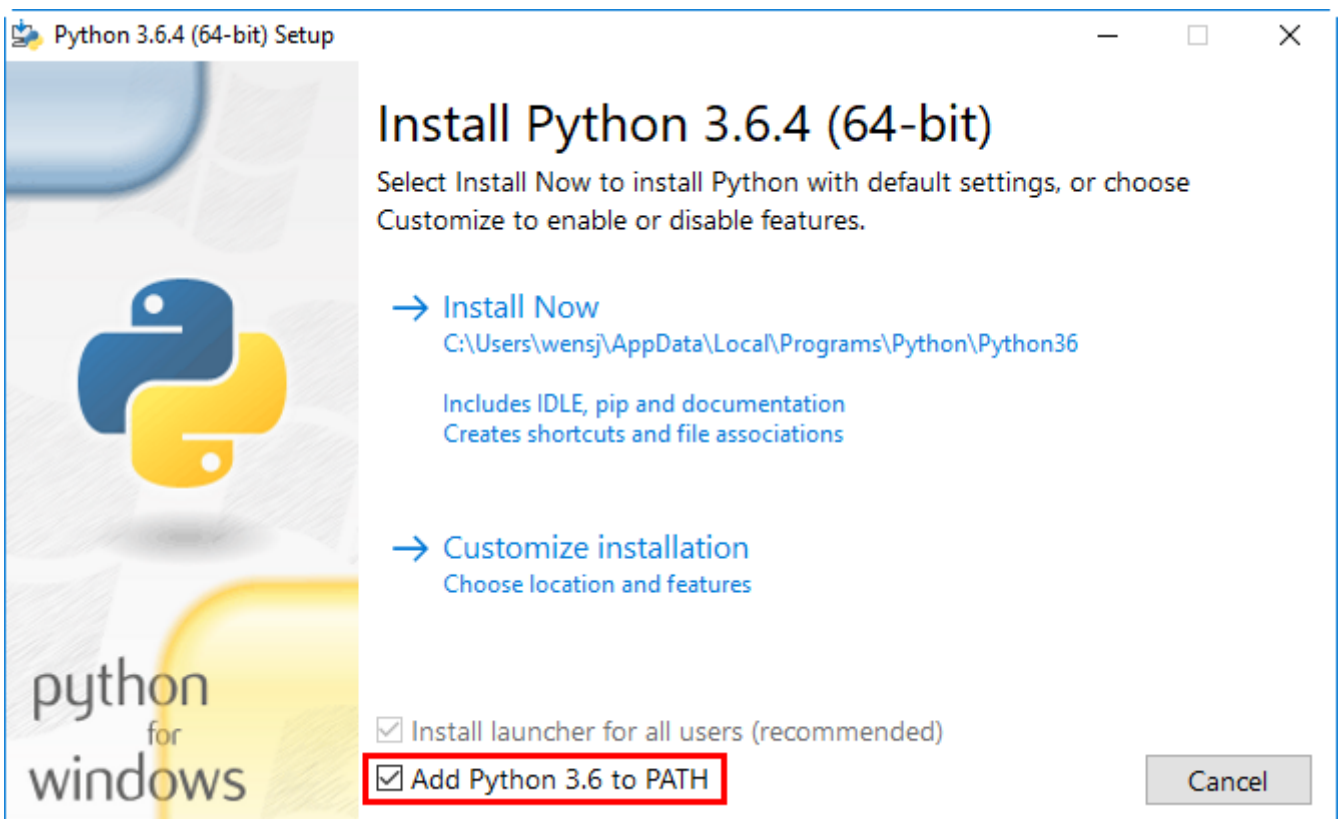
Python文档下载地址: <https://www.python.org/doc/>

2.2.2 Window 平台安装 Python

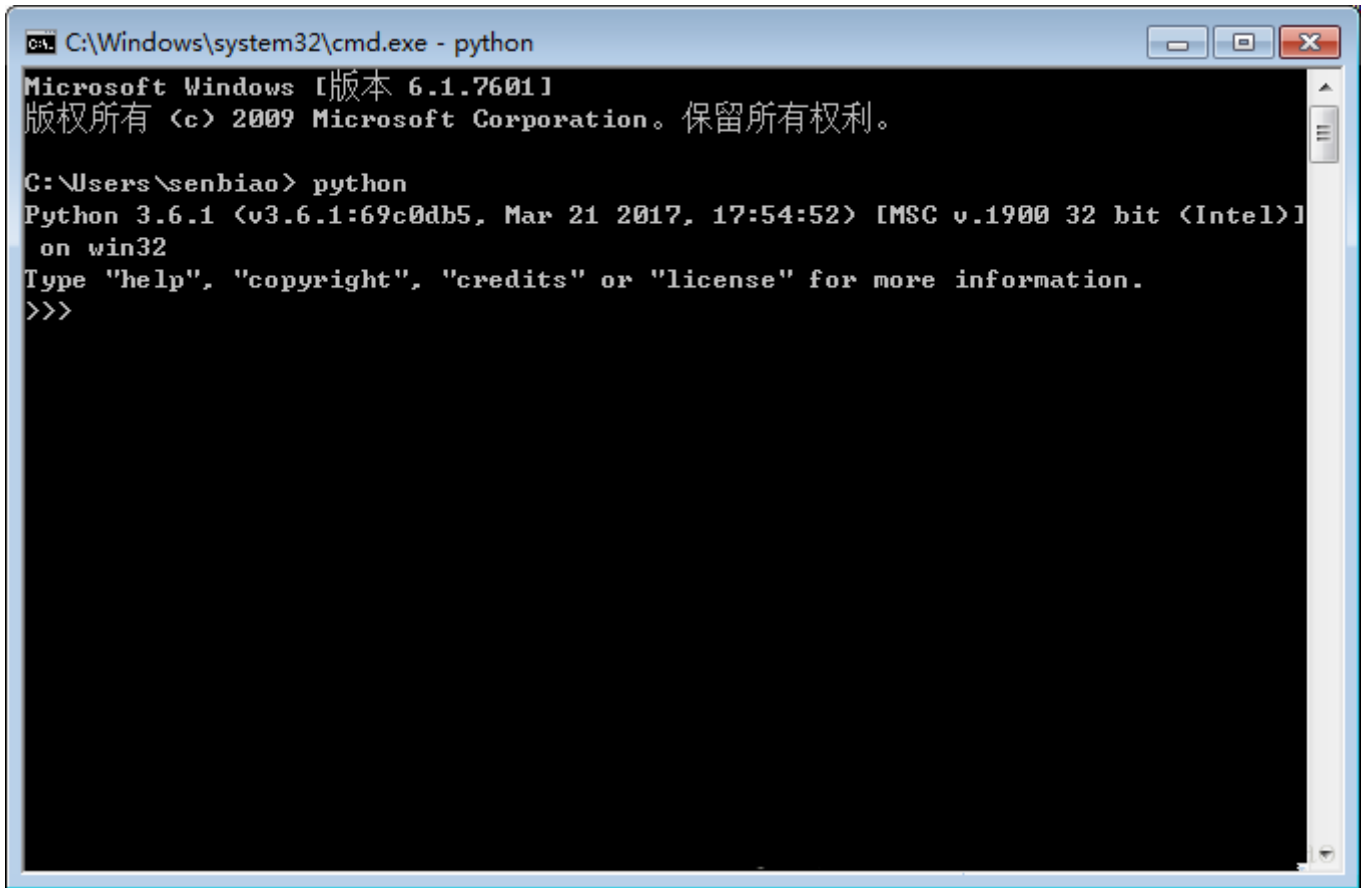
打开 WEB 浏览器访问 <https://www.python.org/downloads/windows/>, 一般就下载 executable installer, x86 表示是 32 位机子的, x86-64 表示 64 位机子的。

Python Releases for Windows

- [Latest Python 3 Release - Python 3.7.0](#)
- [Latest Python 2 Release - Python 2.7.15](#)
- [Python 3.7.0 - 2018-06-27](#)
 - Download [Windows x86 web-based installer](#)
 - Download [Windows x86 executable installer](#)
 - Download [Windows x86 embeddable zip file](#)
 - Download [Windows x86-64 web-based installer](#)
 - Download [Windows x86-64 executable installer](#)
 - Download [Windows x86-64 embeddable zip file](#)
 - Download [Windows help file](#)
- [Python 3.6.6 - 2018-06-27](#)
 - Download [Windows x86 web-based installer](#)



记得勾选 Add Python 3.6 to PATH。



```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\senbiao> python
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

按 Win+R 键，输入 cmd 调出命令提示符，输入 python：

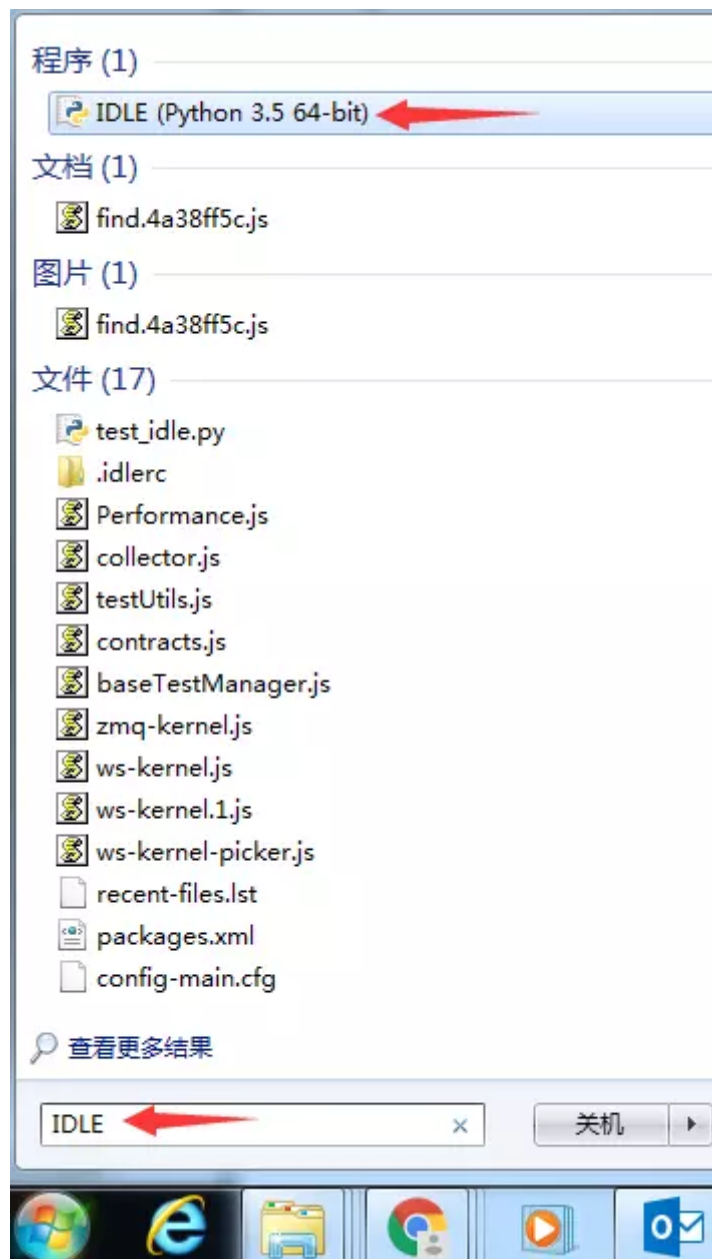


也可以在开始菜单中搜索 IDLE：

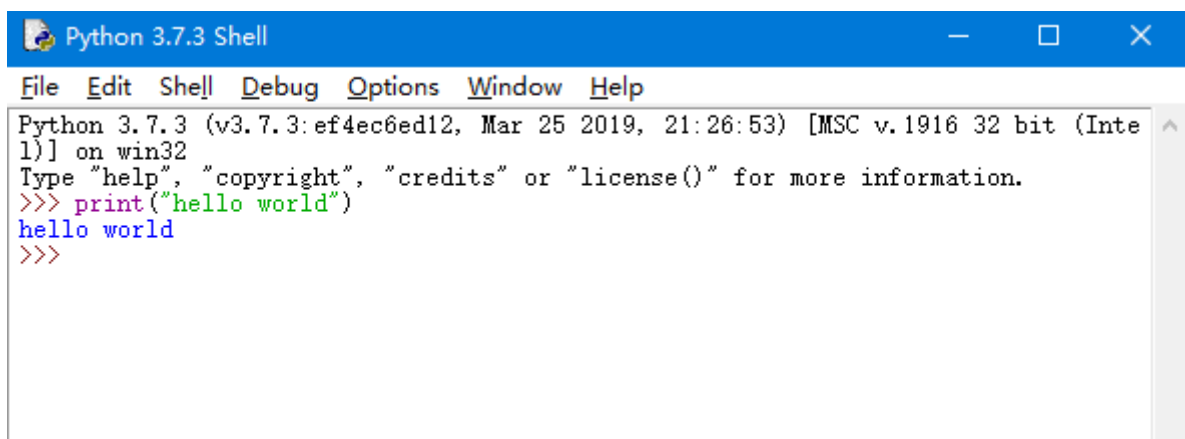
2.3 IDLE的操作

2.3.1 打开方式

找到键盘上的Windows键——按一下，调出应用管理——在输入框里输入关键字python（或者IDLE）

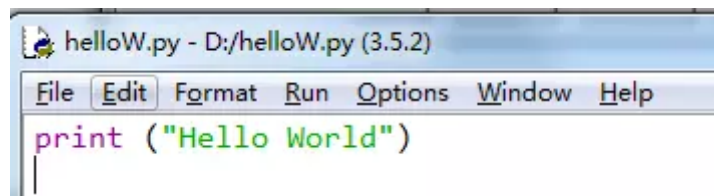
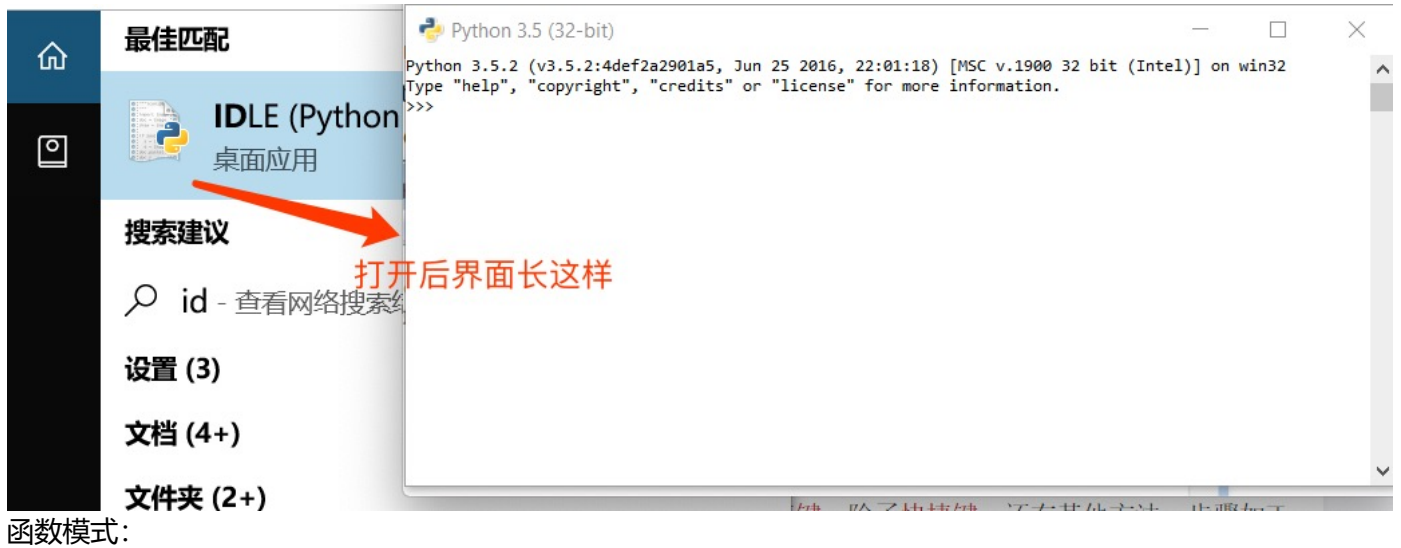


找到idle，括号里带的是安装的python版本，点击后打开shell界面



2.3.2 两种编程模式

命令行模式：

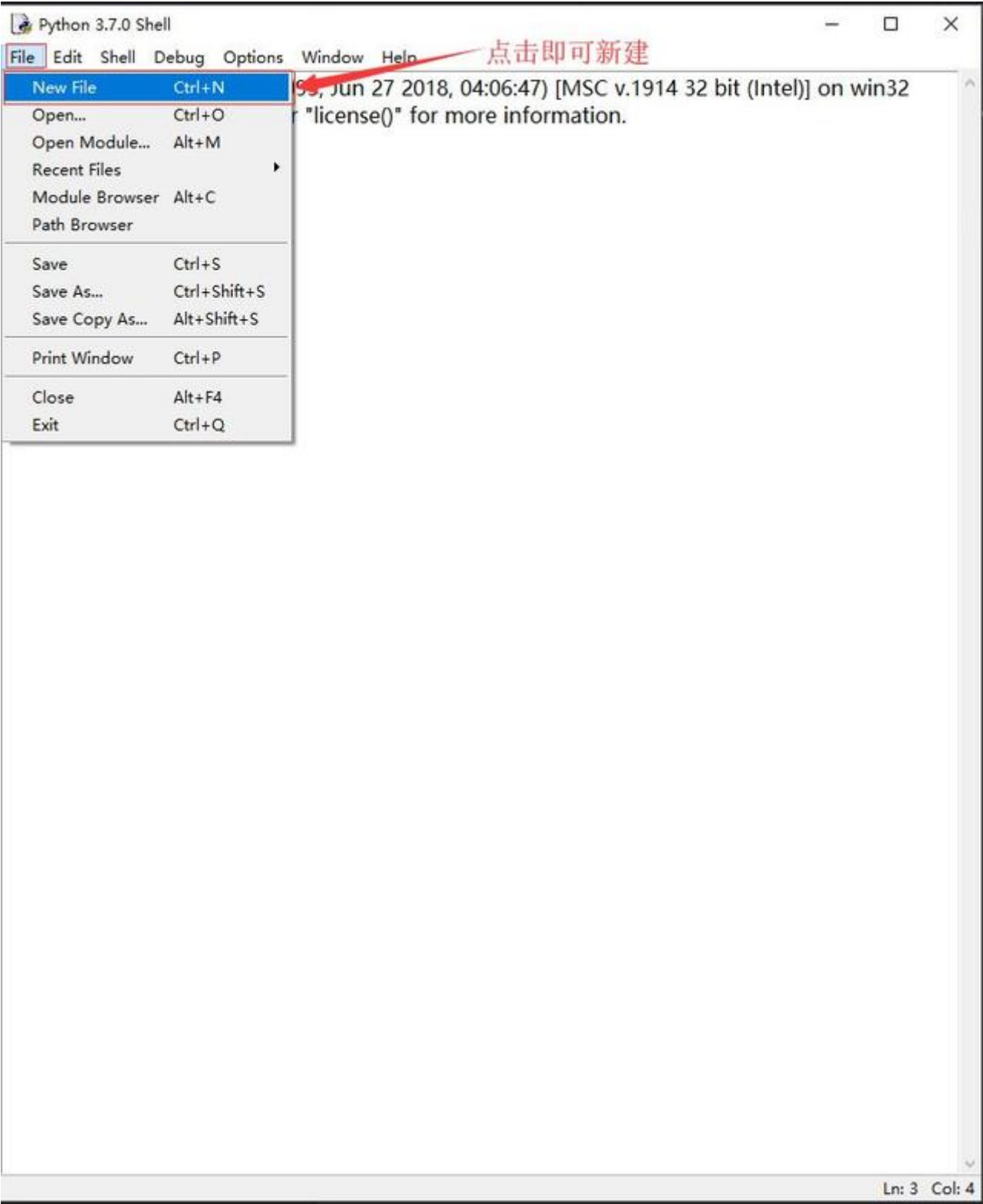


2.3.3 两种编程模式的区别

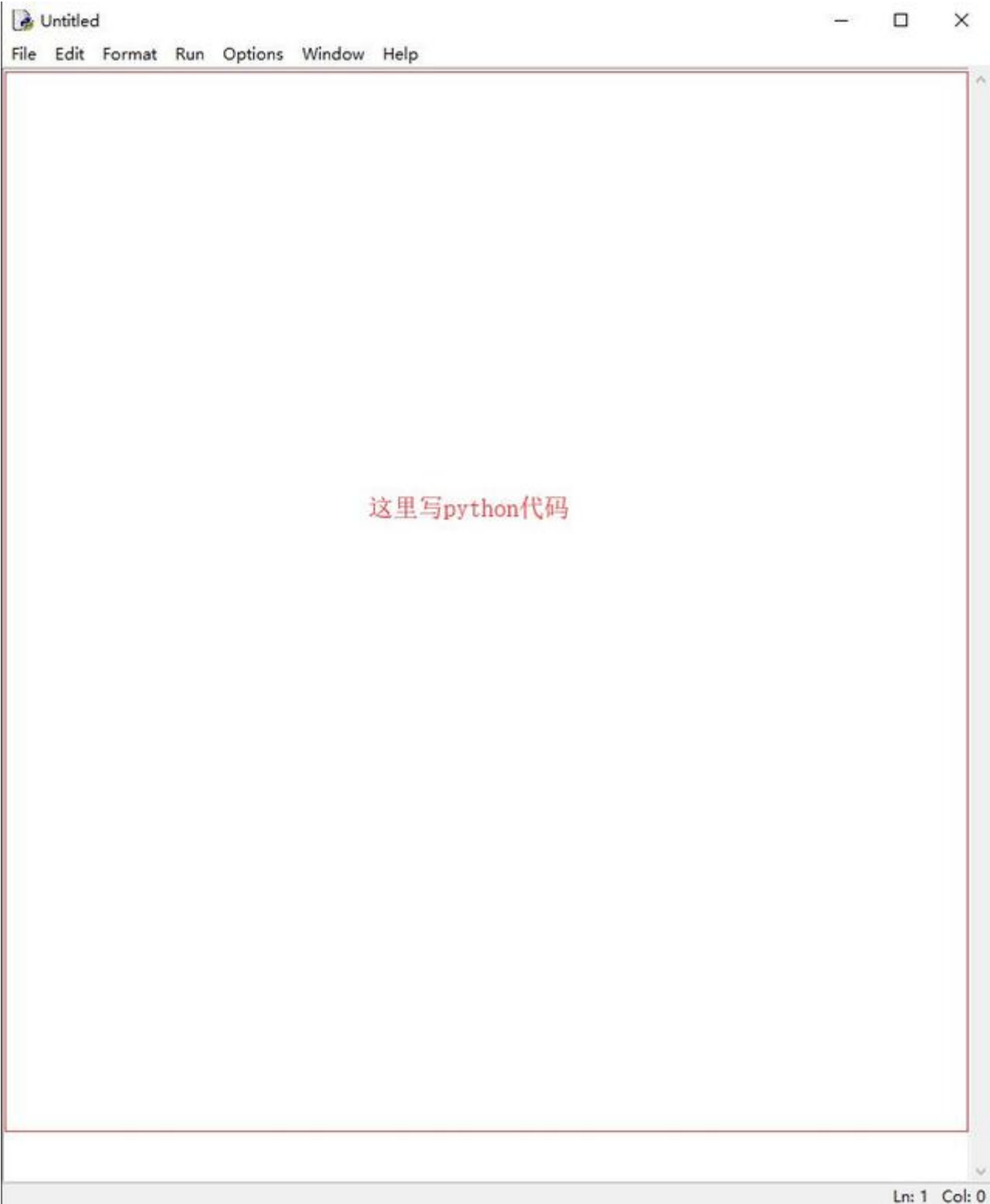
命令行模式一次只能输入一个命名，而函数可以输入很多行一起执行；

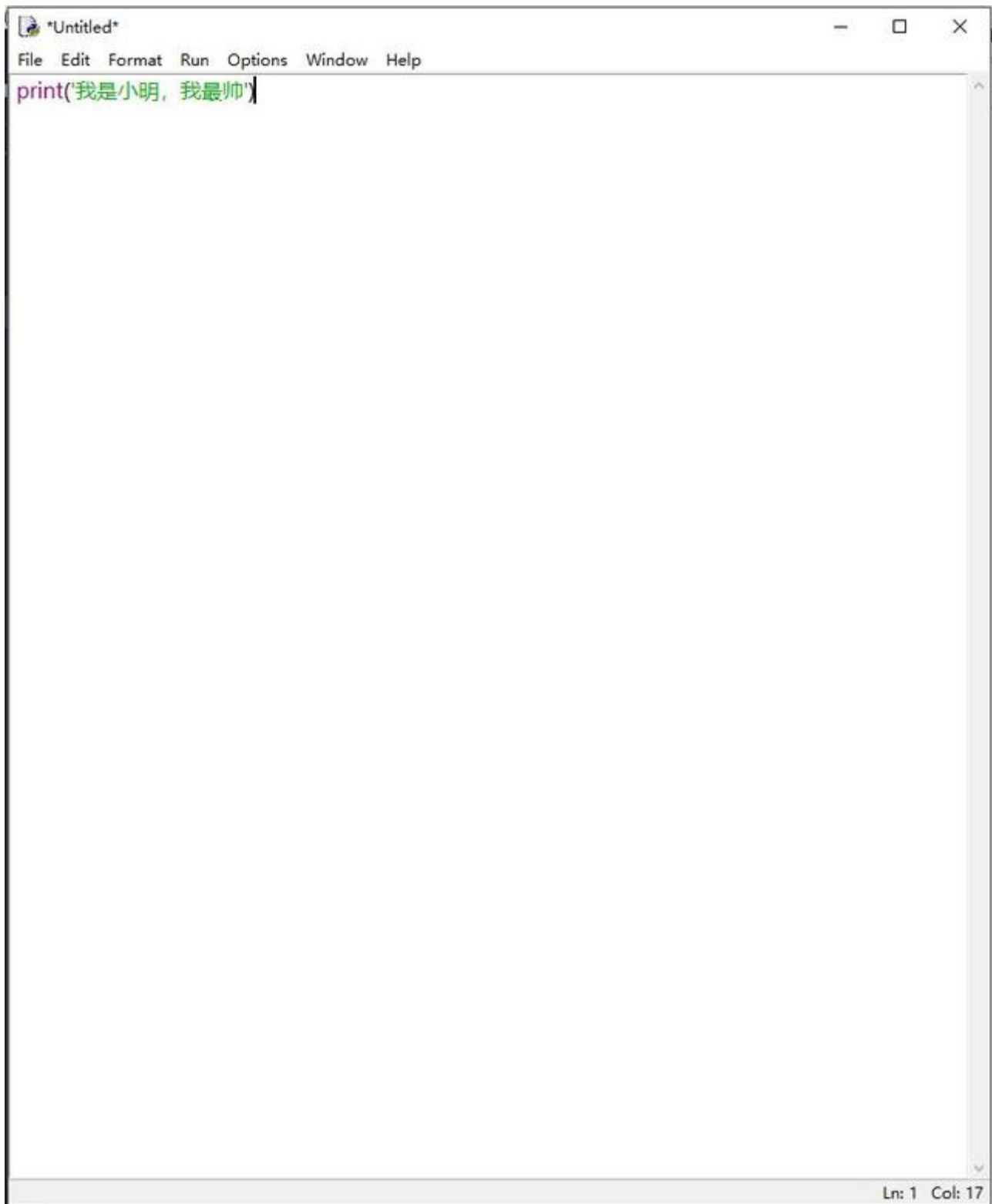
2.3.4 两种方式的转换

打开IDLE后，点击左上角File，然后点击第一项NewFile，即可创建python文件。或者直接使用快捷键Ctrl + N 快速创建。



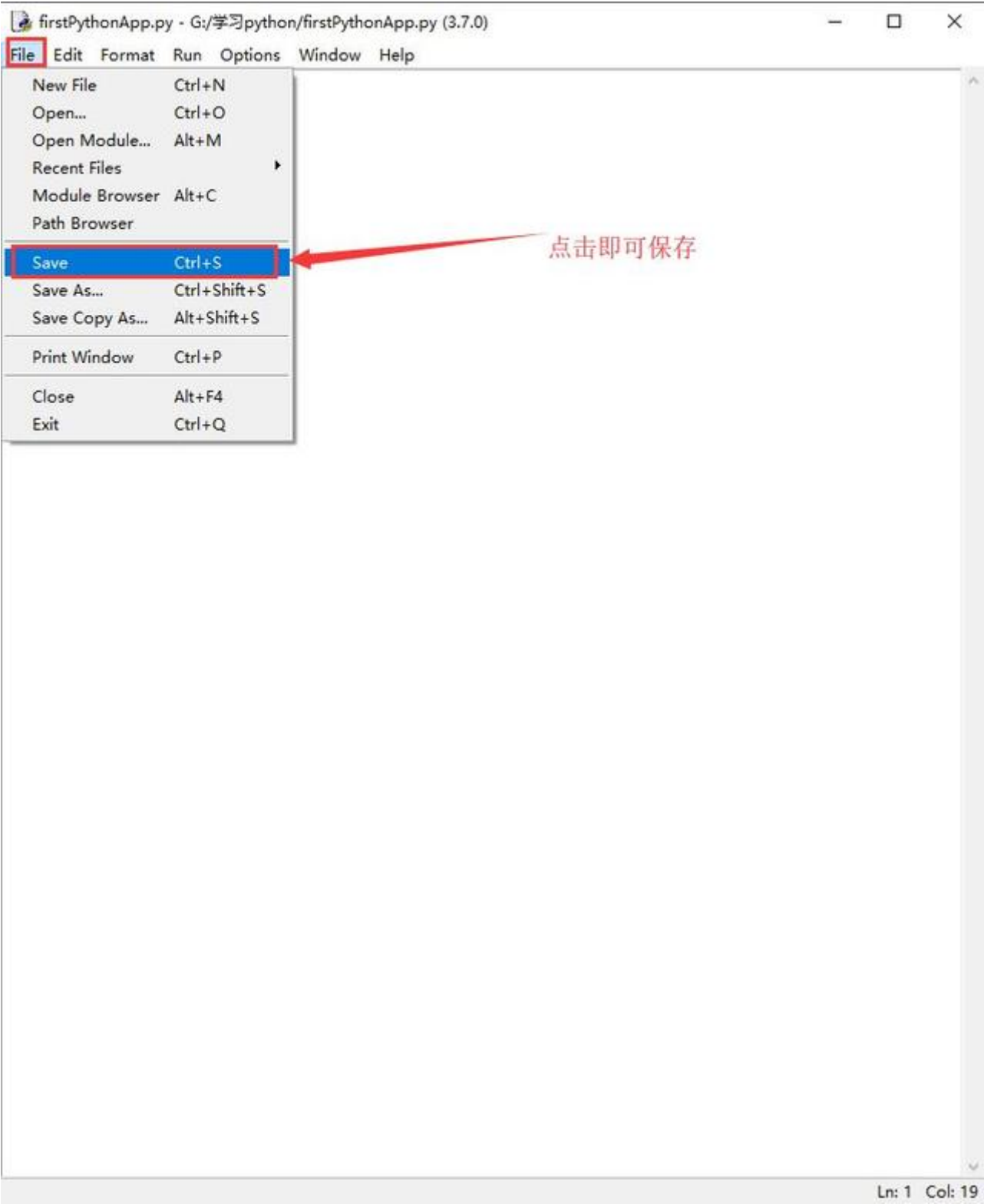
在创建的文件中写 Python 代码



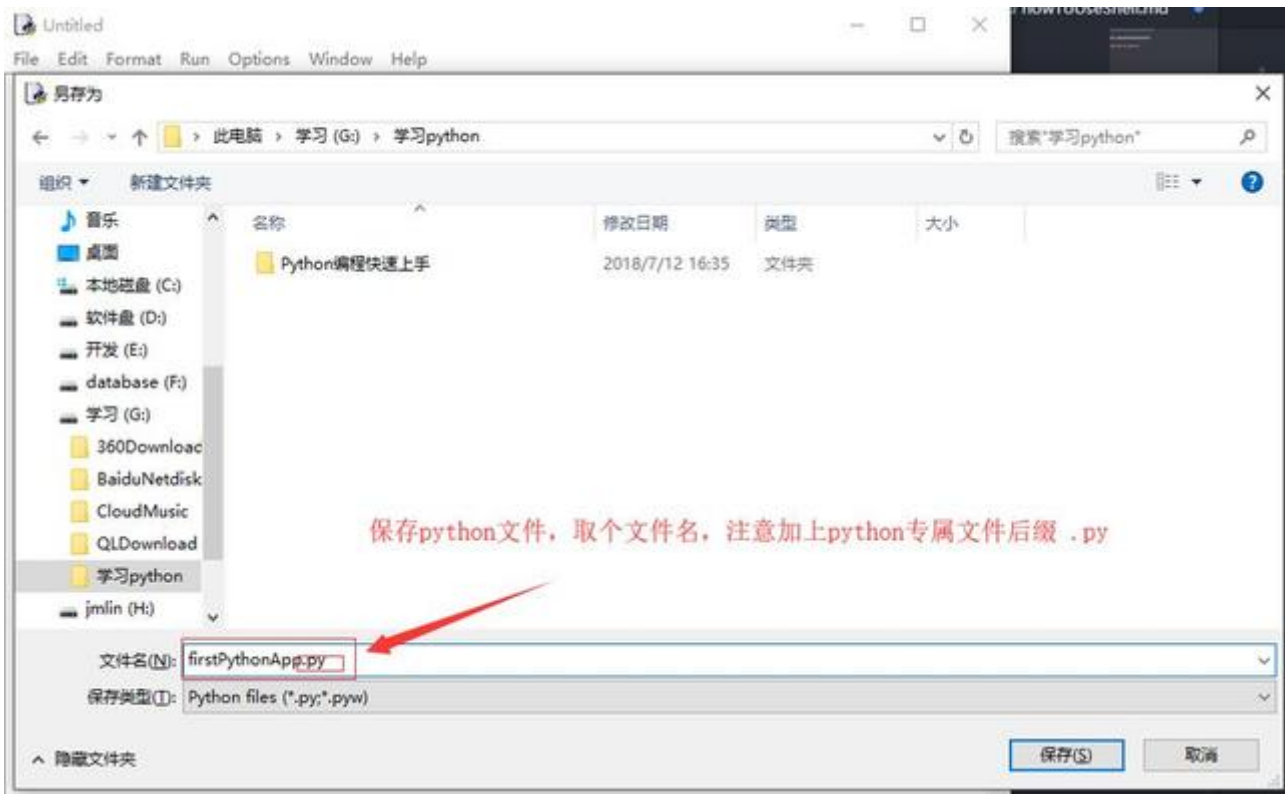


```
*Untitled*
File Edit Format Run Options Window Help
print('我是小明, 我最帅')
Ln: 1 Col: 17
```

保存文件：或者直接使用快捷键Ctrl + S即可快速保存。也可以点击窗口左上角File，然后点击Save完成保存。

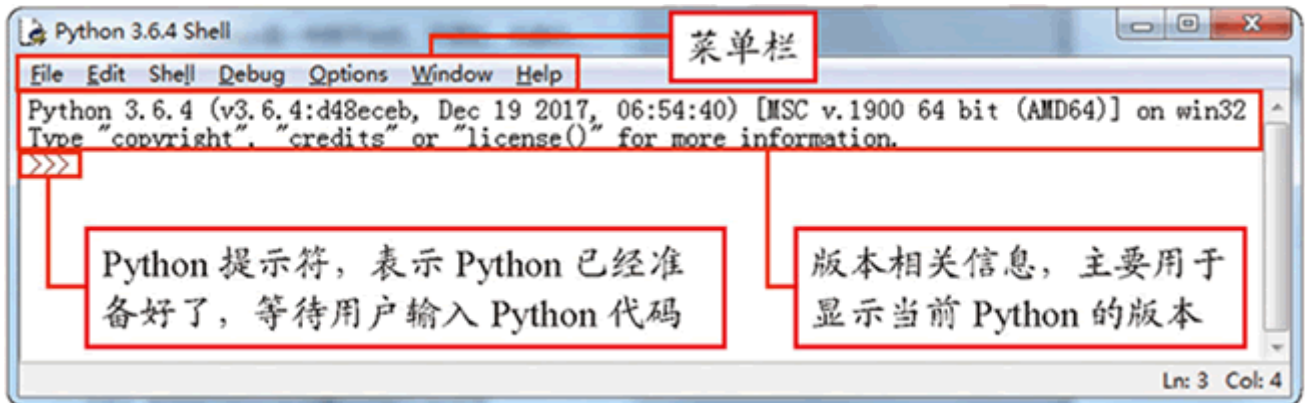


取个文件名，保存成功！

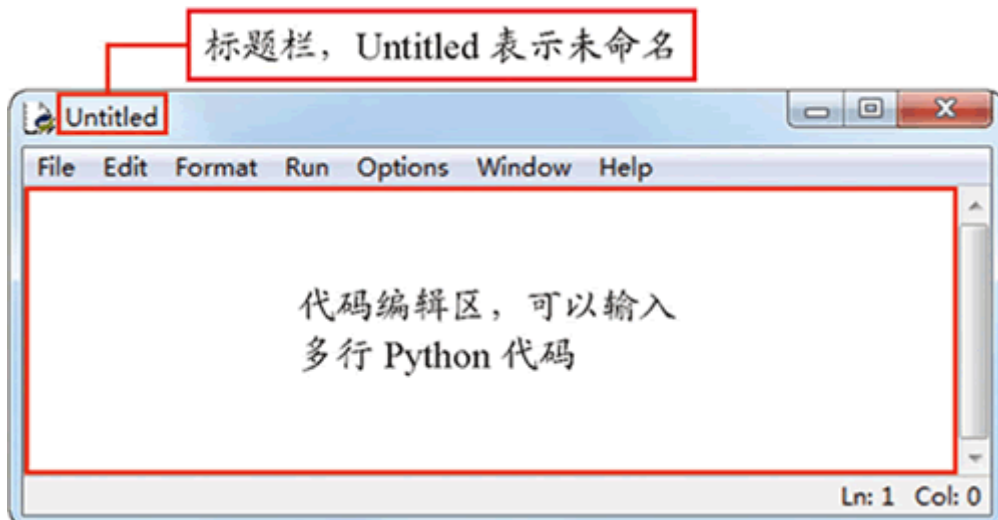


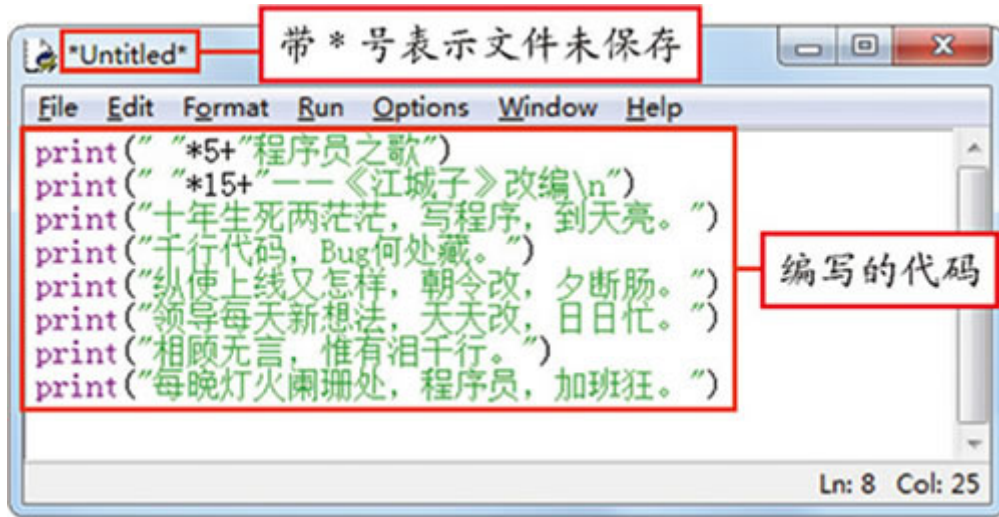
2.3.5 界面的认识

命令行模式



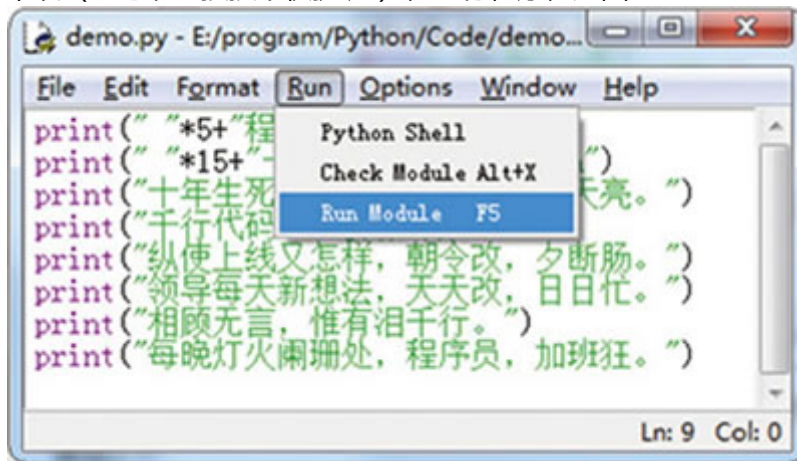
函数模式



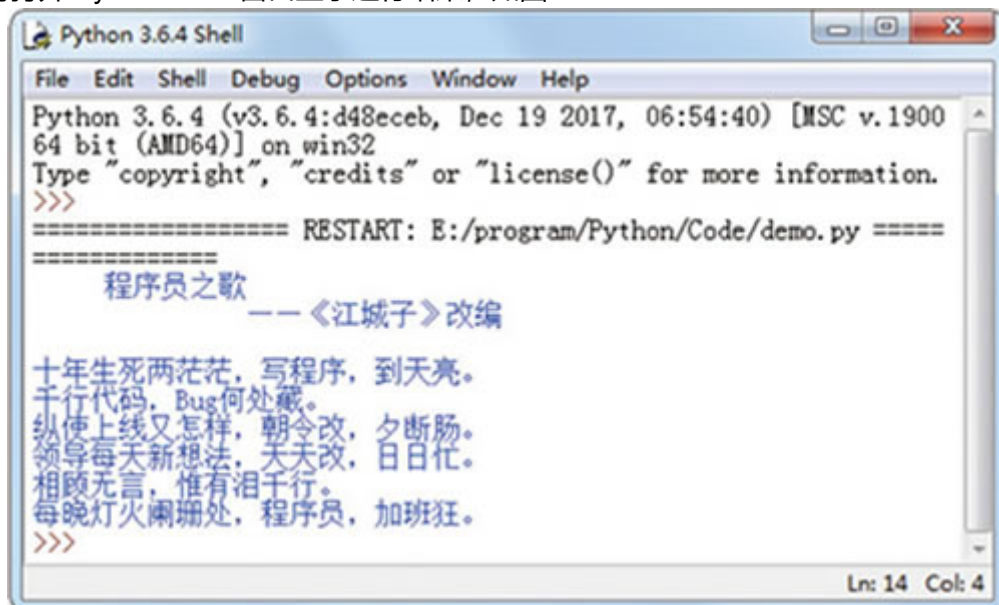


2.3.6 如何运行程序

按下快捷键 保存文件，这里将文件名称设置为 demo.py。其中，.py 是 Python 文件的扩展名。在菜单栏中选择“Run -> Run Module”菜单项（也可以直接按下快捷键），运行程序，如图：



运行程序后，将打开 Python Shell 窗口显示运行结果，如图：



2.4 其他编程环境

知道几种常用的编程环境即可：jupyter VSCode
要明白这些编程环境与Scratch的区别

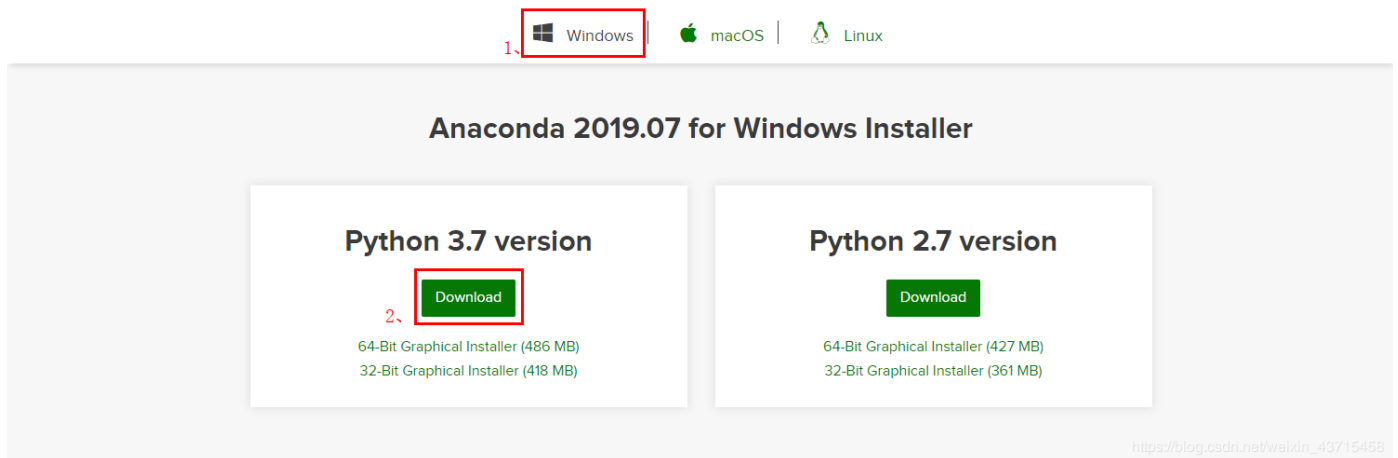
2.4.1 anaconda 的安装:

Anaconda指的是一个开源的Python发行版本，其包含了conda、Python等180多个科学包及其依赖项。

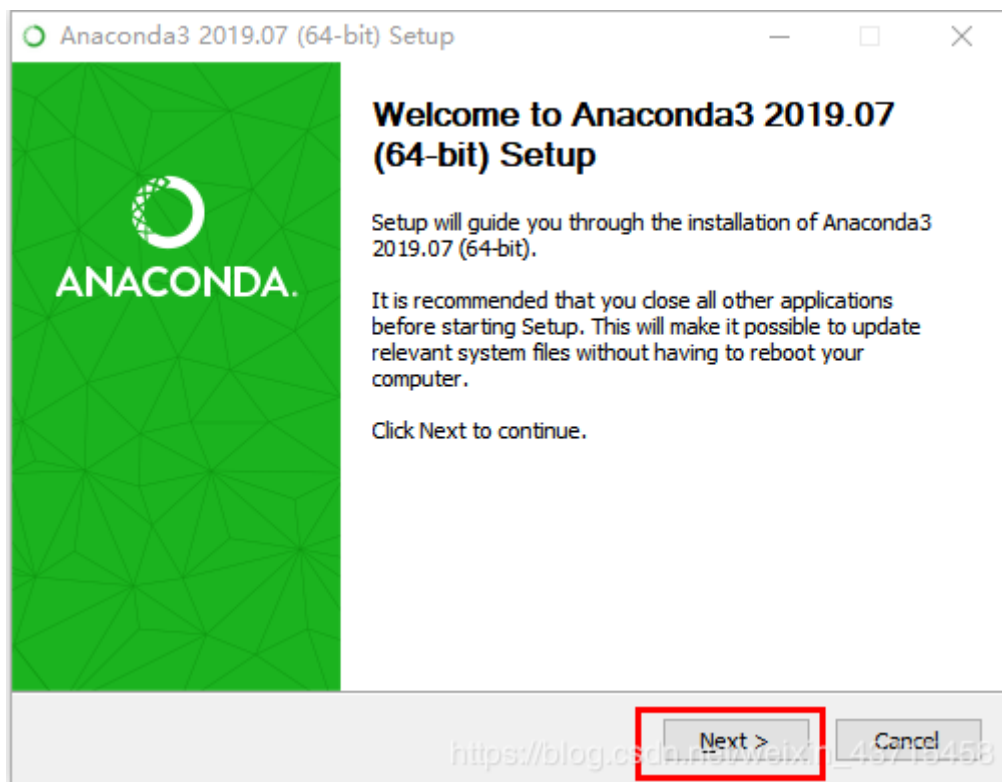
因为包含了大量的科学包，Anaconda 的下载文件比较大（约 531 MB）

Anaconda下载

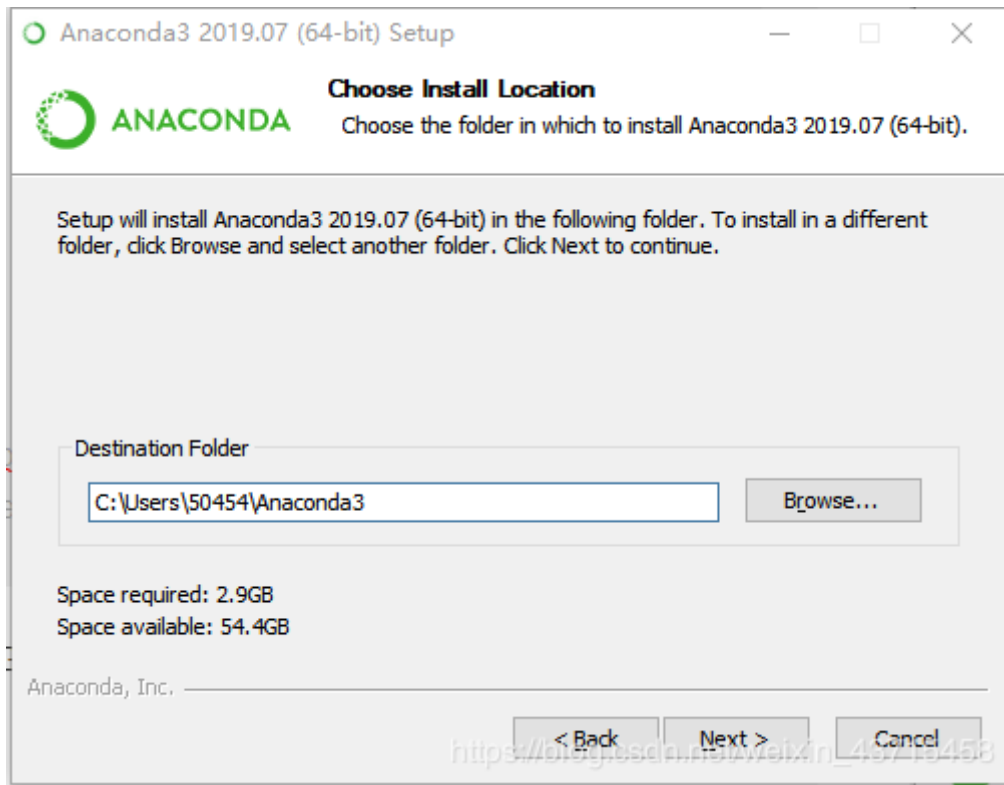
下载地址：<https://www.anaconda.com/download/>



下载完成后直接双击进行安装



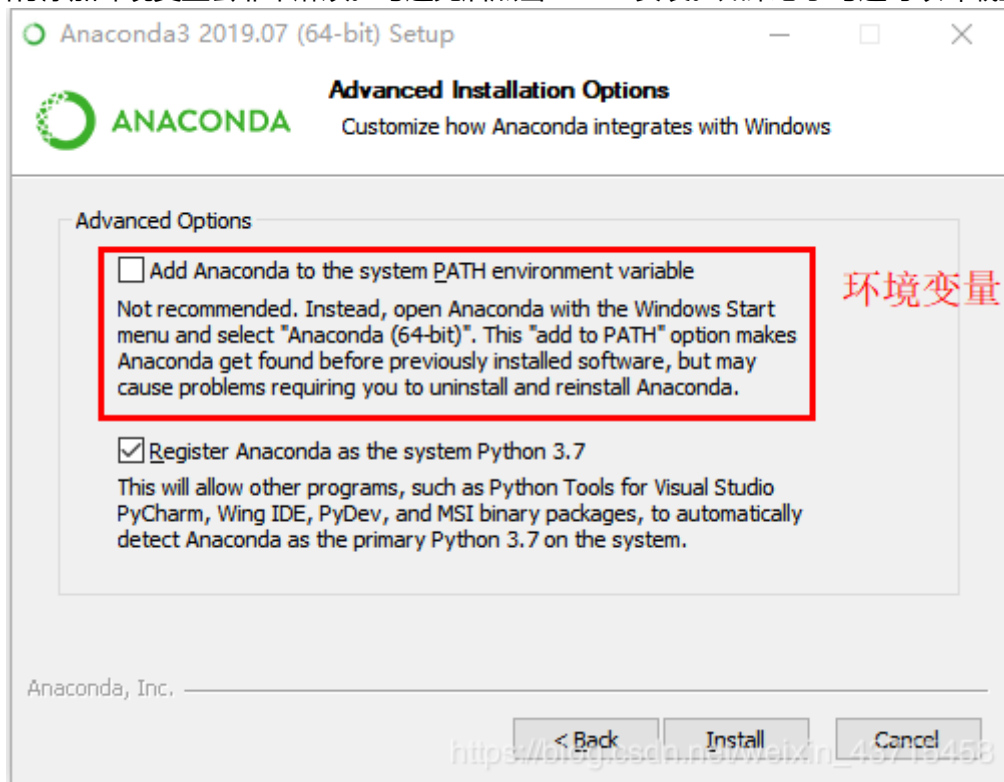
然后一直下一步即可

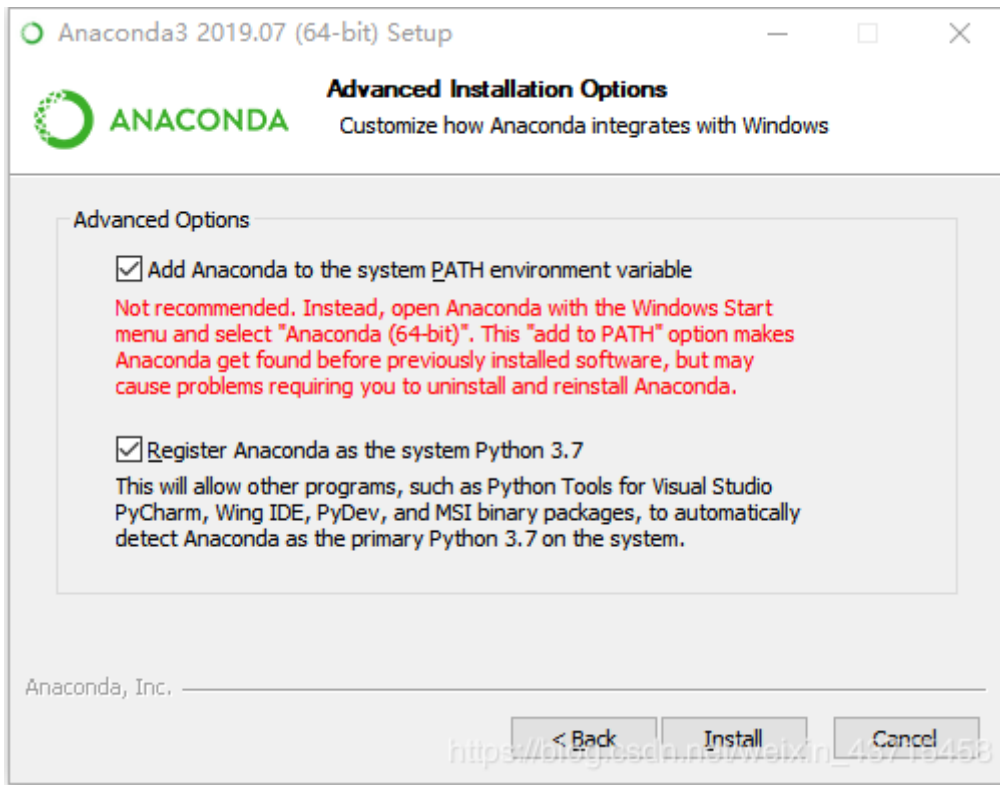


选择安装路径，这里建议装在C盘，也就是默认安装位置。（我的用户名是50454，大家选择Just me之后，路径就会自动选择在自己的windows账户下，默认是Administrator）安装完大概3个G，如果C盘空间很紧张也可以装在其他盘，但将来我们在使用时在读取速率上可能会有一定的影响。选择好了之后点击Next;

2.4.2 注意环境的安装

接下来是重中之重，第一个选项是添加环境变量，默认是没有勾选的，请务必勾选上，如果这里不勾选，后续安装完成后想要自行添加环境变量会非常麻烦。勾选完后点击 Install 安装。如果忘了勾选可以卸载重装。





2.5 IDEL文件

会在IDLE里面新建文件
知道Python文件的后缀名是.py
能区分哪些是Python文件

2.6 编程规范

2.6.1 python文件组成部分

Linux 平台上，一个 python 源码文件应该以下部分组成。Windows 平台上，可以省略第一项。

解释器声明
编码格式声明
模块注释或文档字符串
模块导入
常量和全局变量声明
顶级定义（函数或类定义）
执行代码

2.6.1.1 案例：

In []:

```
#!/usr/bin/env python    #解释器声明 (Linux平台需要声明)
# -*- coding: utf-8 -*-  #编码声明
# coding = utf-8         #这两种都可以 (我比较习惯这种)

"""通常这里是关于本文档的说明 (docstring)，须以半角的句号、 问号或惊叹号结尾!

本行之前应当空一行，继续完成关于本文档的说明
如果文档说明可以在一行内结束，结尾的三个双引号不需要换行；否则，就要像下面这样
"""

import os, time
import datetime
import math

import numpy as np
import xlrd, xlwt, xlutils

import youth_mongodb
import youth_curl

BASE_PATH = r"d:\YouthGit"
LOG_FILE = u"运行日志.txt"

class GameRoom(object):
    """对局室"""

    def __init__(self, name, limit=100, **kwds):
        """构造函数!

        name        对局室名字
        limit        人数上限
        kwds         参数字典
        """

        pass

    def craete_and_start():
        """创建并启动对局室"""

        pass

if __name__ == '__main__':
    # 开启游戏服务
    start()
```

2.7 代码注释

知道代码注释使用的是哪一个符号：#

多行注释可以用多个 # 号，还有 ''' 和 """：

In [2]:

```
# 代码注释
#!/usr/bin/python3

# 第一个注释
print ("Hello, Python!") # 第二个注释
```

Hello, Python!

In []:

```
#!/usr/bin/python3

# 第一个注释
# 第二个注释

'''
第三注释
第四注释
'''

"""
第五注释
第六注释
"""

print ("Hello, Python!") # 我的第一个程序
```

Type *Markdown* and LaTeX: α^2

2.8 行与缩进

python最具特色的就是使用缩进来表示代码块，不需要使用大括号 {}。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。实例如下：

In [2]:

```
if True:
    print ("True")
else:
    print ("False")
```

True

以下代码最后一行语句缩进数的空格数不一致，会导致运行错误：

In [3]:

```
if True:
    print ("Answer")
    print ("True")
else:
    print ("Answer")
    print ("False")    # 缩进不一致，会导致运行错误
```

```
File "<ipython-input-3-a211152115b6>", line 6
    print ("False")    # 缩进不一致，会导致运行错误
    ^
```

IndentationError: unindent does not match any outer indentation level

3 编程基础

3.1 Print 输出

3.1.1 作用

用来输出显示

3.1.2 用法

如果后面显示的是字符串，那么需要加引号

In [1]:

```
print('我今天学习了Python!')
```

我今天学习了Python!

In [2]:

```
print("我是向老师!")
```

我是向老师!

In [5]:

```
print('我是向老师!')
```

我是向老师!

In [6]:

```
print(4+5)
```

9

In [7]:

```
print('4+5')
```

4+5

In [9]:

```
print('4+5=', 4+5)
```

4+5= 9

print 默认输出是换行的，如果要实现不换行需要在变量末尾加上 end=""：

In [24]:

```
#!/usr/bin/python3
```

```
x="a"
```

```
y="b"
```

```
# 换行输出
```

```
print( x )
```

```
print( y )
```

```
print('-----')
```

```
# 不换行输出
```

```
print( x, end=" " )
```

```
print( y, end=" " )
```

```
print()
```

a

b

a b

3.2 变量的命名和使用

3.2.1 变量的创建

Python 中的变量不需要声明。每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建。在 Python 中，变量就是变量，它没有类型，我们所说的"类型"是变量所指的内存中对象的类型。

3.2.2 变量的赋值

等号 (=) 用来给变量赋值。

等号 (=) 运算符左边是一个变量名,等号 (=) 运算符右边是存储在变量中的值。例如：

In [4]:

```
#!/usr/bin/python3

counter = 100          # 整型 这里说的类型指不是变量类型，是数据类型
miles   = 1000.0       # 浮点型
name    = "runoob"     # 字符串

print (counter)
print (miles)
print (name)
```

```
100
1000.0
runoob
```

3.2.3 变量的命名

变量的名称可以使用半角的英文和“_”下划线，但是变量的首字母不能使用数字。同时也不能使用Python中的保留字符（见后面的课程）

In [1]:

```
2a=67
```

```
File "<ipython-input-1-2c805b6048b4>", line 1
  2a=67
    ^
```

```
SyntaxError: invalid syntax
```

In [2]:

```
a2=67
```

3.2.4 多个变量赋值

Python允许你同时为多个变量赋值。例如：

In [5]:

```
a = b = c = 1
```

以上实例，创建一个整型对象，值为 1，从后向前赋值，三个变量被赋予相同的数值。您也可以为多个对象指定多个变量。例如：

In [6]:

```
a, b, c = 1, 2, "runoob"
```

以上实例，两个整型对象 1 和 2 的分配给变量 a 和 b，字符串对象 "runoob" 分配给变量 c。

3.2.5 保留字

python保留字：保留字即关键字，我们不能把它们用作任何标识符名称。Python 的标准库提供了一个 keyword 模块，可以输出当前版本的所有关键字：

In [3]:

```
import keyword
keyword.kwlist
```

Out[3]:

```
['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']
```

3.2.6 练习一

3.2.6.1 题目要求

计算a、b的加、减、乘、除的值

3.2.6.2 分析

直接写print() 答案的方式不合适，需要用到变量。

3.2.6.3 参考程序

In [5]:

```
a=50
b=5
print(a+b, a-b, a*b, a/b)
```

55 45 250 10.0

3.2.7 练习二

3.2.7.1 题目要求

有下面几句话：

我是帅哥！

我是美女！

我是小鲜肉！

然后分别组合出来不同的语句。

3.2.7.2 分析

同样，直接写print()写不合适，需要用到变量。

3.2.7.3 参考程序

In [7]:

```
a="我是帅哥！"
b="我是美女！"
c="我是小鲜肉！"
print(a+b+c)
print(b+a+c)
print(c+b+a)
```

我是帅哥！我是美女！我是小鲜肉！

我是美女！我是帅哥！我是小鲜肉！

我是小鲜肉！我是美女！我是帅哥！

3.2.8 练习三

3.2.8.1 题目要求

比较一下下面的代码的不同。

abc="放假了，太开心了！"

print("abc")

print(abc)

3.2.8.2 参考程序

In [9]:

```
开心="放假了，太开心了！"  
print("abc")  
print(开心)
```

abc
放假了，太开心了！

3.3 数据类型

Python3 中有六个标准的数据类型：

不可变数据（3 个）：Number（数字）、String（字符串）、Tuple（元组）
可变数据（3 个）：List（列表）、Dictionary（字典）、Set（集合）

3.3.1 数据类型的查询

像大多数语言一样，数值类型的赋值和计算都是很直观的。内置的 `type()` 函数可以用来查询变量所指的对象类型。

In [8]:

```
a, b, c, d = 20, 5.5, True, 4+3j  
print(type(a), type(b), type(c), type(d))
```

<class 'int'> <class 'float'> <class 'bool'> <class 'complex'>

3.3.2 数字(Number)类型

python中数字有四种类型：整数、布尔型、浮点数和复数。

int (整数), 如 1, 只有一种整数类型 int, 表示为长整型, 没有 python2 中的 Long

bool (布尔), 如 True

float (浮点数), 如 1.23、3E-2

complex (复数), 如 1 + 2j、1.1 + 2.2j

3.3.3 字符串(String)

Python中的字符串是用引号引起来的；Python中的字符串不能改变。Python 没有单独的字符类型，一个字符就是长度为 1 的字符串。

In [11]:

```
print("大家好！")
```

大家好！

3.3.4 引号

python中单引号和双引号使用完全相同；

In [19]:

```
word = '字符串'  
sentence = "这是一个句子。"
```

但是引号前后要一致

In []:

```
print("Hello!")
```

3.3.5 三引号

使用三引号("或''')可以指定一个多行字符串。

In [12]:

```
print("""这是一个段落，  
可以由多行组成""")
```

这是一个段落，
可以由多行组成

3.3.6 转义符

“\”反斜杠可以用来转义，使用r可以让反斜杠不发生转义。。如 r"this is a line with \n" 则\n会显示，并不是换行。

按字面意义级联字符串，如"this " "is " "string"会被自动转换为this is string。

In [16]:

```
print("this " "is " "string")           #按字面意义级联字符串  
print("this is a line with \n" "Hello!") # 使用反斜杠(\)+n转义特殊字符  
print(r"this is a line with \n")        # 在字符串前面添加一个 r，表示原始字符串，不会发生转义
```

this is string
this is a line with
Hello!
this is a line with \n

3.3.7 字符串加法和乘法

可以用 + 运算符连接在一起，用 * 运算符重复。

In [20]:

```
print("H"+"ello"+" "+"World!")
```

Hello World!

In [23]:

```
print("hello "*3)    #注意：被引用的空格也是字符串的一部分。
```

hello hello hello

3.4 Python数据类型转换

字符串及数值类型转换 尤其是看起来像数值的类型，不一定就是数值的，可以互相转换，转换命令如下：

3.4.1 int() 函数

3.4.1.1 描述

int() 函数用于将一个字符串或数字转换为整型。

3.4.1.2 语法

以下是 int() 方法的语法：

```
int(x, base=10)
```

3.4.1.3 参数

x -- 字符串或数字。 base -- 进制数，默认十进制。

3.4.1.4 返回值

返回整型数据。

3.4.1.5 实例

以下展示了使用 int() 方法的实例：

In [1]:

```
int()    # 不传入参数时，得到结果0
```

Out[1]:

0

In [38]:

```
int(3)
```

Out[38]:

3

In [2]:

```
int(3.6)
```

Out[2]:

3

In [40]:

```
int('12', 16)      # 如果是带参数base的话, 12要以字符串的形式进行输入, 12 为 16进制
```

Out[40]:

18

In [41]:

```
int('0xa', 16)
```

Out[41]:

10

In [36]:

```
int('10', 8)
```

Out[36]:

8

3.4.2 str() 函数

3.4.2.1 描述

str() 函数将对象 x 转换为字符串。

3.4.2.2 语法

以下是 str() 方法的语法:

```
str(object="")
```

3.4.2.3 参数

object -- 对象。

3.4.2.4 返回值

返回一个对象的字符串格式。

3.4.2.5 实例

以下展示了使用 str() 方法的实例:

In [2]:

```
s = '4+6'  
str(s)
```

Out[2]:

```
'4+6'
```

3.4.3 float() 函数

3.4.3.1 描述

float() 函数用于将整数和字符串转换成浮点数。

3.4.3.2 语法

float()方法语法:

```
float(x)
```

3.4.3.3 参数

x -- 整数或字符串

3.4.3.4 返回值

返回浮点数。

3.4.3.5 实例

以下实例展示了 float() 的使用方法:

In []:

```
float(1)
```

In []:

```
float(112)
```

In []:

```
float(-123.6)
```

In [1]:

```
float('123')    # 字符串
```

Out[1]:

```
123.0
```

3.5 input()语句

3.5.1 描述

其接收任意任性输入，将所有输入默认为字符串处理，并返回字符串类型。

3.5.2 语法

```
input([prompt])
```

3.5.3 参数说明:

prompt: 提示信息

3.5.4 返回值

返回字符串类型。

3.5.5 实例

请在下面的程序运行后，输入数字、字符、公式，看看最后输出的是哪一种类型？

In [3]:

```
a = input("请输入:")  
type(a)           # 显示返回值的类型
```

请输入:向金

Out[3]:

str

3.6 编程基础例题练习

3.6.1 练习一

3.6.1.1 题目要求

在屏幕上显示如下图案：

```
*  
**  
***  
****  
*****  
*****  
*****
```

3.6.1.2 考核点

print() 语句、引号

3.6.1.3 参考程序

In [4]:

```
print('''*
**
***
****
*****
*****
*****''')
```

```
*
**
***
****
*****
*****
*****
```

3.6.2 练习二

3.6.2.1 题目要求

在屏幕上用a显示出菱形图案： 如下图：

In [5]:

```
print('''
  a
 aaa
aaaaa
aaaaaaa
aaaaaaaa
aaaaaaa
aaaaa
  aaa
    a
''')
```

```

  a
 aaa
aaaaa
aaaaaaa
aaaaaaaa
aaaaaaa
aaaaa
  aaa
    a
```

3.6.2.2 考核点

print()语句、三引号、格式输出

3.6.2.3 参考程序

In [2]:

```
print("""
      ^                / |
    / \ 7            /  _/
   /   |            /   /
  |   Z _ , <      /   / ^ \
  |           \     /   /   }
  Y           ,    /   /
1● 、 ●      CD <   /
() ^         |   \ <
>- 、 -   1   |   / /
/ ^      /   / < | \ \
\ _/    ( _/   |   / /
7        |   / /
>-r_____-- """)
```

[illegible]

把下面的话分成两行打印，但必须在一行程序中写出来：
原文：我是你们的老师，我很喜欢你们，但是，请你们不要嫌弃我很老哦！
打印成为：
我是你们的老师，
我很喜欢你们，
但是，
请你们不要嫌弃我很老哦！

print() 函数

print() 函数为打印函数，如括号内数据带引号，print() 函数会原样打印引号内的数据。

转义字符

\n 代表换行；\' 代表单引号。

三引号

三引号可以把内容里的单引号打印出来且可以换行

3.6.4.3 参考程序

In [11]:

```
print("我是你们的老师,\n我很喜欢你们,\n但是, 请你们不要嫌弃我很老哦!")
```

我是你们的老师,
我很喜欢你们,
但是, 请你们不要嫌弃我很老哦!

3.6.5 练习五

3.6.5.1 题目要求

程序运行后要求输入一个（或者许多个）数字或者字符，输完后按回车键，屏幕显示出三个该数字或者字符；

3.6.5.2 考核点

input() 函数
print() 函数
变量及命名
数据类型

3.6.5.3 参考程序

In [12]:

```
a=input("请输入内容：")  
print(a)  
print(a)  
print(a)
```

请输入内容：我是向老师
我是向老师
我是向老师
我是向老师

3.6.6 练习六

3.6.6.1 题目要求

程序运行后要求输入一个数字（大小不限），输完后按回车，屏幕按照输入的数字显示多少个A；

3.6.6.2 考核点

input() 函数
print() 函数
变量及命名
数据类型
字符串的乘法

3.6.6.3 参考程序

In [16]:

```
cishu=input("请输入显示A的数量：")  
print("A"*int(cichu))
```

请输入显示A的数量：90

AA
AAAAAA

3.6.6.4 程序分析

重点要理解：

- 1、为什么A要加引号？因为A是字符串，不是变量，所以必须要加引号；
- 2、问什么cishu不加引号？因为cishu是一个变量，不是字符串；
- 3、问什么不能直接乘以cishu？因为变量cishu是一个字符串变量，不是数值型变量；

3.6.7 练习七

3.6.7.1 题目要求

程序运行后要求输入一个数字（大小不限），输完后按回车，程序要求再次输入一个一句话，回车后，屏幕将安装刚才给的数量重复第二次输入的话；

3.6.7.2 考核点

input() 函数
print() 函数
变量及命名
数据类型
字符串的乘法

3.6.7.3 参考程序

In [24]:

```
a=input("请输入数量：")
b=input("请输入一句话：")
print(b*int(a))
```

请输入数量：3
请输入一句话：Python
Python Python Python

3.6.7.4 思考题:

- 1、为什么b不用加引号了？
- 2、如何让输入的句子中间间隔开来？

4 运算符

4.1 赋值运算符

以下假设变量a为10，变量b为20：

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c = a 等效于 c = c a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c = a 等效于 c = c a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

4.2 算术运算符

下面的案例：a为10， b为21

运算符	名称	描述	实例
+	加	两个对象相加	a+b输出结果 31
-	减	得到负数或是一个数减去另一个数	a-b输出结果-11
*	乘	两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果210
/	除	x除以y	b/a输出结果 2.1
%	取模	返回除法的余数	b%a输出结果1
**	幂	返回x的y次幂	a**b为10的21次方
//	取整除	向下取接近除数的整数	9//2为4

在Python中，可以直接进行算术运算符的运算

4.2.1 练习一

4.2.1.1 题目要求

直接计算5的7次方的值；

4.2.1.2 考核点

算术运算符中的**（幂运算）

4.2.1.3 参考程序

In [25]:

```
5**7
```

Out[25]:

78125

4.2.2 练习二

4.2.2.1 题目要求

输入两个数，然后计算这两个数的和、差、积、相除的得数；

4.2.2.2 考核点

算术运算符中的加减乘除运算
数据类型的转换

4.2.2.3 参考程序

In [4]:

```
a=int(input("请输入第一个数："))
b=int(input("请输入第二个数："))
print(a+b)
print(a-b)
print(a*b)
print(a/b)
```

请输入第一个数：36

请输入第二个数：5

41

31

180

7.2

4.2.2.4 思考：

- 1、为什么要在input语句前加int（）函数？
- 2、为什么print里面没有引号？

4.2.3 练习三

4.2.3.1 题目要求

输入两个数，然后计算这两个数的余数、幂、除数取整；

4.2.3.2 考核点

算术运算符

数据类型的转换

4.2.3.3 参考程序

In [5]:

```
a=int(input("请输入第一个数："))
b=int(input("请输入第二个数："))
print(a%b)
print(a**b)
print(a//b)
```

请输入第一个数：12

请输入第二个数：3

0

1728

4

4.2.3.4 注意事项

除数取整是向下取整，所以如果是负数，那么是比商小，如下：

In [6]:

```
-23//6
```

Out[6]:

-4

In [7]:

```
23//6
```

Out[7]:

3

4.2.4 练习四

4.2.4.1 题目要求

输入两个数，然后按照下面的格式显示出来：
您输入的是**和**；
+=（显示正确的答案）
除以等于（显示正确答案）
-等于（显示正确答案）
如下面的案例：
请输入第一个数：34
请输入第二个数：6
您输入的是 34 和 6
34 + 6 = 40
34 除以 6 等于 5.666666666666667
34 - 6 = 28

4.2.4.2 考核点

算术运算符
数据类型的转换
print() 输出语句的综合运用

4.2.4.3 参考程序

In [10]:

```
a=int(input("请输入第一个数："))
b=int(input("请输入第二个数："))
print("您输入的是",a,"和",b)
print(a,"+",b,"=",a+b)
print(a,"除以",b,"等于",a/b)
print(a,"-",b,"=",a-b)
```

请输入第一个数：34
请输入第二个数：6
您输入的是 34 和 6
34 + 6 = 40
34 除以 6 等于 5.666666666666667
34 - 6 = 28

4.2.4.4 思考：

- 1、问什么在print中，+号-号=号等需要加引号？
- 2、显示的34+6能否直接写成“a+b”（加上引号）输出？
- 3、如果不可以，为什么？

4.3 比较运算符

以下假设变量a为10，变量b为20：所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。注意，这些变量名的大写。

运算符	名称	描述	实例
==	等于	比较对象是否相等	(a == b) 返回 False。
!=	不等于	比较两个对象是否不相等	(a != b) 返回 True。

运算符	名称	描述	实例
>	大于	返回x是否大于y	(a > b) 返回 False。
<	小于	返回x是否小于y	(a < b) 返回 True。
>=	大于等于	返回x是否大于等于y。	(a >= b) 返回 False。
<=	小于等于	返回x是否小于等于y。	(a <= b) 返回 True。

4.3.1 练习一

4.3.1.1 题目要求

a、b两个数分别为21和10，自动判断两个数是否等于、是否不等于.....（把比较运算符整个过一遍）

4.3.1.2 考核点

- 1、比较运算符；
- 2、条件语句；

4.3.1.3 参考程序

In [12]:

```
#!/usr/bin/python3

a = 21
b = 10

if ( a == b ):
    print ("1: a 等于 b")
else:
    print ("1: a 不等于 b")

if ( a != b ):
    print ("2: a 不等于 b")
else:
    print ("2: a 等于 b")

if ( a < b ):
    print ("3: a 小于 b")
else:
    print ("3: a 大于等于 b")

if ( a > b ):
    print ("4: a 大于 b")
else:
    print ("4: a 小于等于 b")

# 修改变量 a 和 b 的值
a = 5;
b = 20;
if ( a <= b ):
    print ("5: a 小于等于 b")
else:
    print ("5: a 大于 b")

if ( b >= a ):
    print ("6: b 大于等于 a")
else:
    print ("6: b 小于 a")
```

```
1: a 不等于 b
2: a 不等于 b
3: a 大于等于 b
4: a 大于 b
5: a 小于等于 b
6: b 大于等于 a
```

4.3.1.4 程序说明

- 1、if else 是Python中的条件语句，在2级中会学习到；这几道练习题均为2级考题范围。
- 2、变量只保留最后一次的赋值；

4.3.2 练习二

4.3.2.1 题目要求

分别输入两个数，判断两个数是否相等，并且显示出来；

4.3.2.2 考核点

- 1、比较运算符；
- 2、条件语句；

4.3.2.3 参考程序

In [22]:

```
a=int(input("请输入第一个数："))
b=int(input("请输入第二个数："))
if a==b:
    print("您输入的两个数字相等！")
else:
    print("你输入的两个数字不相等！")
```

请输入第一个数：34
请输入第二个数：34
您输入的两个数字相等！

4.3.3 练习三

4.3.3.1 题目要求

自动判断输入的数字是否是偶数，并显示出来。

4.3.3.2 考核点

- 1、比较运算符；
- 2、条件语句；
- 3、算术运算符；

4.3.3.3 思路分析

偶数一定能被2整除，根据这个特点，我们只需要判断输入的数除以2的余数是否为0就知道是不是偶数。

4.3.3.4 参考程序

In [3]:

```
a=int(input("请输入第一个数："))
if (a%2==0):
    print("您输入的是：",a,"；是一个偶数。")
else:
    print("您输入的是：",a,"；是一个奇数。")
```

请输入第一个数：34
您输入的是： 34 ；是一个偶数。

4.3.4 练习四

4.3.4.1 题目要求

分别输入两个数，自动判断两个数哪个数大，并且按照如下格式显示：
您输入的数字是2和5
5大于2（如果相等则输出：6等于6，后面第三行不再显示）
所以大的数字是5

4.3.4.2 考核点

- 1、比较运算符；
- 2、条件语句；

4.3.4.3 参考程序

In [5]:

```
a=int(input("请输入第一个数："))
b=int(input("请输入第二个数："))
print("您输入的数字是",a,"和",b)
if (a==b):
    print(a,"等于",b)
else:
    if(a>b):
        print(a,"大于",b)
        print("所以大的数字是:",a)
    else:
        print(b,"大于",a)
        print("所以大的数字是",b)
```

请输入第一个数：23
请输入第二个数：32
您输入的数字是 23 和 32
32 大于 23
所以大的数字是 32

4.4 逻辑运算符

Python语言支持逻辑运算符，以下假设变量 a 为 10, b为 20:

运算符	逻辑表达式	名称	描述	实例
and	x and y	布尔"与"	如果 x 为 False, x and y 返回 False, 否则它返回 y 的计算值。	(a and b) 返回 20。
or	x or y	布尔"或"	如果 x 是 True, 它返回 x 的值, 否则它返回 y 的计算值。	(a or b) 返回 10。
not	not x	布尔"非"	如果 x 为 True, 返回 False 。如果 x 为 False, 它返回 True。	not(a and b) 返回 False。

将上面的内容转换为程序，如下：

In [7]:

```
#!/usr/bin/python3

a = 10
b = 20

if ( a and b ):
    print ("1: 变量 a 和 b 都为 true")
else:
    print ("1: 变量 a 和 b 有一个不为 true")

if ( a or b ):
    print ("2: 变量 a 和 b 都为 true, 或其中一个变量为 true")
else:
    print ("2: 变量 a 和 b 都不为 true")

# 修改变量 a 的值
a = 0
if ( a and b ):
    print ("3: 变量 a 和 b 都为 true")
else:
    print ("3: 变量 a 和 b 有一个不为 true")

if ( a or b ):
    print ("4: 变量 a 和 b 都为 true, 或其中一个变量为 true")
else:
    print ("4: 变量 a 和 b 都不为 true")

if not( a and b ):
    print ("5: 变量 a 和 b 都为 false, 或其中一个变量为 false")
else:
    print ("5: 变量 a 和 b 都为 true")
```

- 1: 变量 a 和 b 都为 true
- 2: 变量 a 和 b 都为 true, 或其中一个变量为 true
- 3: 变量 a 和 b 有一个不为 true
- 4: 变量 a 和 b 都为 true, 或其中一个变量为 true
- 5: 变量 a 和 b 都为 false, 或其中一个变量为 false

4.5 运算符优先级

以下表格列出了从最高到最低优先级的所有运算符：

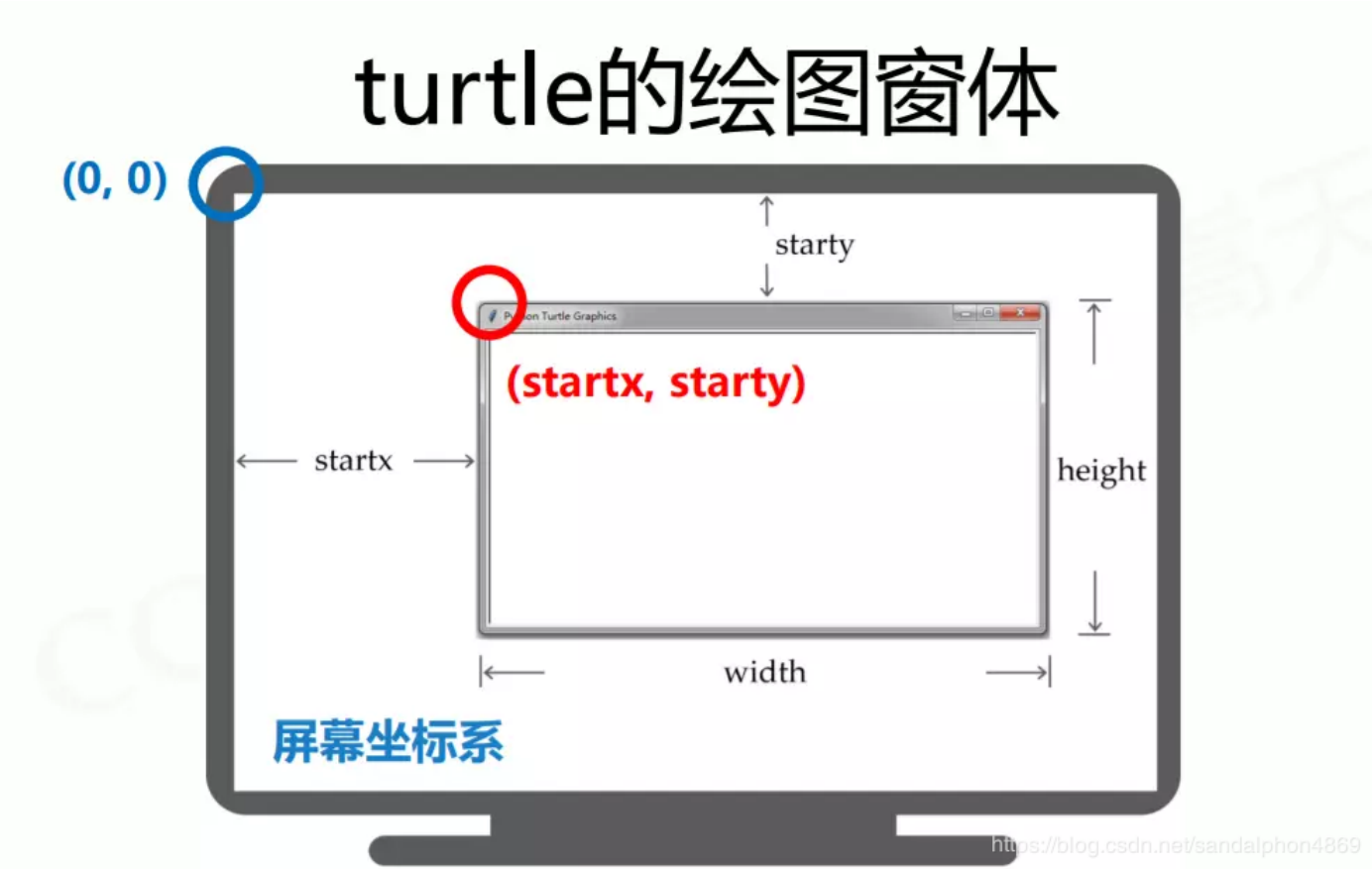
运算符	描述
**	指数 (最高优先级)
~ + -	按位翻转, 一元加号和减号 (最后两个的方法名为 +@ 和 -@)
* / % //	乘, 除, 求余数和取整除
+ -	加法减法
>> <<	右移, 左移运算符
&	位 'AND'
^	位运算符
<= < > >=	比较运算符

运算符	描述
== !=	等于运算符
= %= /= //= -= += = * =	赋值运算符
is is not	身份运算符
in not in	成员运算符
not and or	逻辑运算符

5 Turtle库

5.1 turtle坐标系

5.1.1 turtle绘图窗体布局



这个是绘画时候的窗口的坐标，不是小海龟的坐标。

(1) 最小单位是像素，左上角是（0，0）

(2) startx与starty：绘图窗体出现在屏幕的哪里，后两个参数可选，默认正中心

5.1.1.1 对应的命令行

setup() 设置窗体大小及位置 格式：turtle.setup(width,height,startx,starty)。

4个参数中后两个可选。

setup函数不是必须的，只有当需要控制绘图窗体大小的时候才调用。

5.1.1.2 案例

比较下面两个命令的不同

In [8]:

```
import turtle
turtle.setup(800, 600)
```

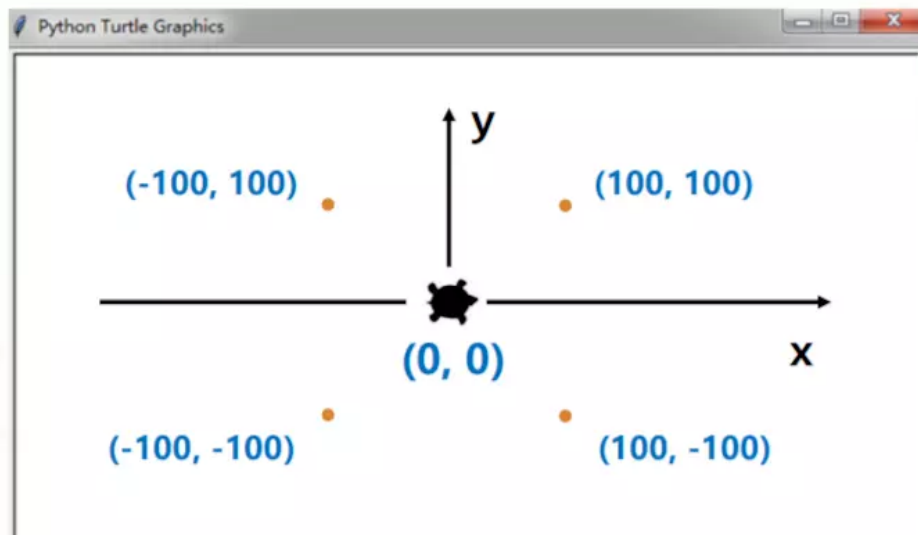
In [2]:

```
import turtle
turtle.setup(800, 600, 0, 0)
```

5.1.2 小海龟绝对坐标

turtle空间坐标体系

绝对坐标



<https://blog.csdn.net/sandalphon4869>

这里指的是海龟在绘图窗口里面的坐标。

绝对坐标：就是标准的x0y坐标系，上y右x，中央点是（0，0）。

5.1.2.1 对应的命令行

```
turtle.goto(x, y)
```

小海龟前进到指定的坐标位置。

5.1.2.2 案例

In [1]:

```
import turtle
turtle.goto(100, 100)
turtle.done()
```

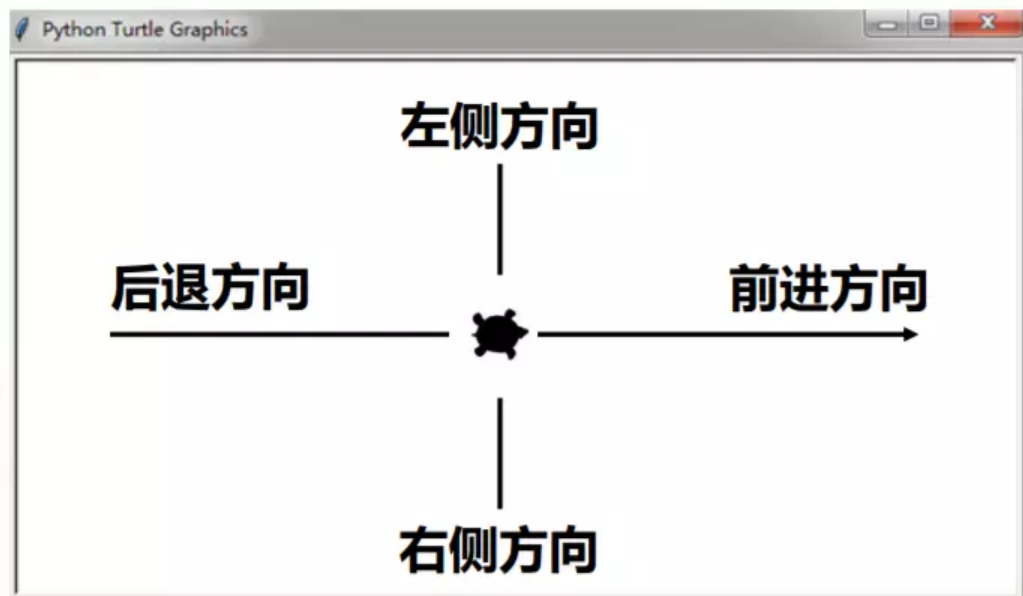
5.1.2.3 程序说明

- 1、在绘图前必须要引入turtle库文件；
- 2、可以不用设置窗口，那么窗口将是采用默认的大小和位置；
- 3、建议结尾加上turtle.done()命令：用来停止画笔绘制，但绘图窗体不关闭。这样就不会出现死机现象。

5.1.3 海龟方向坐标

turtle空间坐标体系

海龟坐标



<https://blog.csdn.net/sandaiphon4869>

standard模式下（默认）：头朝的方向就是前方：如当龟头朝右时，右侧就为前方。开始默认头朝右。

5.1.3.1 对应的命令行

- 1、turtle.forward(a) 向当前画笔方向移动a像素长度。
- 2、turtle.backward(a) 向当前画笔相反方向移动a像素长度。
- 3、turtle.circle(radius, extent=None, steps=None) radius(半径) - 半径为正(负)，表示圆心在画笔的左边(右边)画圆；extent(弧度) (optional)；steps (optional) - 做半径为radius的圆的内切正多边形，多边形边数为steps

5.1.3.2 案例

In [1]:

```
# 在正前方画一条线，
import turtle
turtle.forward(100)
turtle.done()
```

In [5]:

```
# 在正后方画一条线,  
import turtle  
turtle.backward(100)  
turtle.done()
```

In [3]:

```
# 画一个圆  
import turtle  
turtle.circle(-100)  
turtle.done()
```

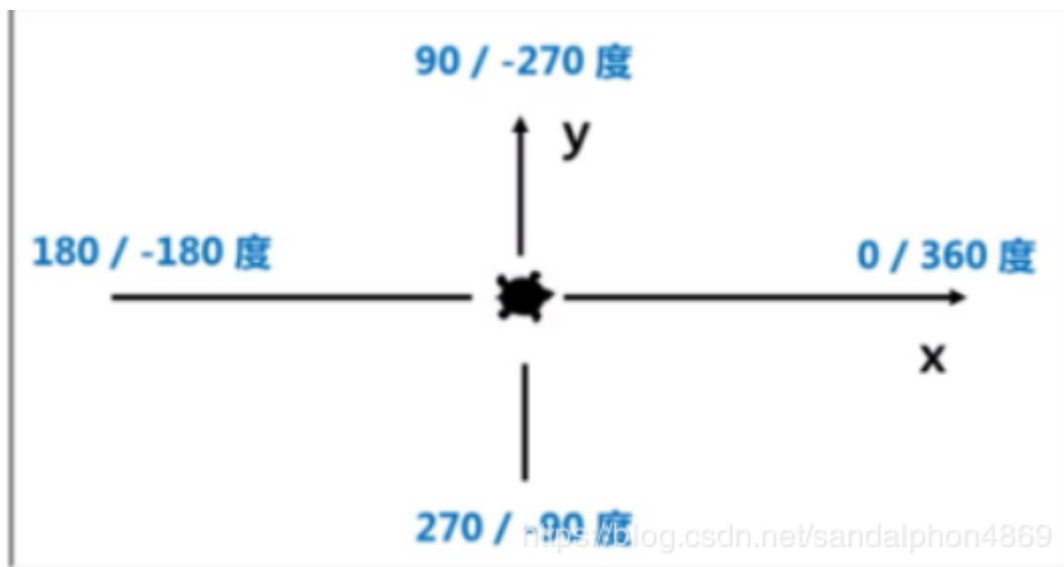
In [7]:

```
# 画正方形  
import turtle  
turtle.circle(100, 360, 4)  
turtle.done()
```

In [9]:

```
# 画一段曲线  
import turtle  
turtle.circle(-40, 80)  
turtle.done()
```

5.1.4 海龟角度坐标体系



角度坐标体系：就是数学上的坐标轴角度，绕x轴逆时针角度从0° 到360°

5.1.4.1 对应的命令行

(1) 绝对方向:

`turtle.setheading(angle)` 别名 `turtle.seth(angle)`: 只改变方向不行进。如 `turtle.seth(90)`: 海龟的朝向为90度

注意: 与当前海龟头的朝向没有关系, 90度就一定向上

(2) 相对方向:

`turtle.left(angle)` 与 `turtle.right(angle)`: 在海龟当前头的方向上再向左/右转多少度

5.1.4.2 案例

In [5]:

```
import turtle
turtle.forward(100)
turtle.left(60)
turtle.forward(100)
turtle.seth(90)
turtle.forward(100)
turtle.done()
```

5.2 函数纵览

5.2.1 海龟动作

移动和绘制

`forward()` | `fd()` 前进
`backward()` | `bk()` | `back()` 后退
`right()` | `rt()` 右转
`left()` | `lt()` 左转
`goto()` | `setpos()` | `setposition()` 前往/定位
`setx()` 设置x坐标
`sety()` 设置y坐标
`setheading()` | `seth()` 设置朝向
`home()` 返回原点
`circle()` 画圆
`dot()` 画点
`stamp()` 印章
`clearstamp()` 清除印章
`clearstamps()` 清除多个印章
`undo()` 撤消
`speed()` 速度

获取海龟的状态

`position()` | `pos()` 位置
`towards()` 目标方向
`xcor()` x坐标
`ycor()` y坐标
`heading()` 朝向
`distance()` 距离

设置与度量单位

`degrees()` 角度
`radians()` 弧度

5.2.2 画笔控制

绘图状态

`pendown()` | `pd()` | `down()` 画笔落下
`penup()` | `pu()` | `up()` 画笔抬起
`pensize()` | `width()` 画笔粗细
`pen()` 画笔
`isdown()` 画笔是否落下

颜色控制

`color()` 颜色
`pencolor()` 画笔颜色
`fillcolor()` 填充颜色

填充

`filling()` 是否填充
`begin_fill()` 开始填充
`end_fill()` 结束填充

更多绘图控制

`reset()` 重置
`clear()` 清空
`write()` 书写

5.2.3 海龟状态

可见性

`showturtle()` | `st()` 显示海龟
`hideturtle()` | `ht()` 隐藏海龟
`isvisible()` 是否可见

外观

`shape()` 形状
`resizemode()` 大小调整模式
`shapesize()` | `turtlesize()` 形状大小
`shearfactor()` 剪切因子
`settiltangle()` 设置倾角
`tiltangle()` 倾角
`tilt()` 倾斜
`shapetransform()` 变形
`get_shapepoly()` 获取形状多边形

5.2.4 使用事件

`onclick()` 当鼠标点击
`onrelease()` 当鼠标释放
`ondrag()` 当鼠标拖动

5.2.5 特殊的海龟方法

`begin_poly()` 开始记录多边形
`end_poly()` 结束记录多边形
`get_poly()` 获取多边形
`clone()` 克隆
`getturtle()` | `getpen()` 获取海龟画笔
`getscreen()` 获取屏幕
`setundobuffer()` 设置撤消缓冲区
`undobufferentries()` 撤消缓冲区条目数

5.2.6 其他命令

窗口控制

`bgcolor()` 背景颜色
`bgpic()` 背景图片
`clear()` | `clearscreen()` 清屏
`reset()` | `resetscreen()` 重置
`screensize()` 屏幕大小
`setworldcoordinates()` 设置世界坐标系

动画控制

`delay()` 延迟
`tracer()` 追踪
`update()` 更新

使用屏幕事件

`listen()` 监听
`onkey()` | `onkeyrelease()` 当键盘按下并释放
`onkeypress()` 当键盘按下
`onscreenclick()` 当点击屏幕
`ontimer()` 当达到定时
`mainloop()` | `done()` 主循环

设置与特殊方法

`mode()` 模式
`colormode()` 颜色模式
`getcanvas()` 获取画布
`getshapes()` 获取形状
`register_shape()` | `addshape()` 添加形状
`turtles()` 所有海龟
`window_height()` 窗口高度
`window_width()` 窗口宽度

输入方法

`textinput()` 文本输入
`numinput()` 数字输入

Screen 专有方法

`bye()` 退出
`exitonclick()` 当点击时退出
`setup()` 设置
`title()` 标题

<https://blog.csdn.net/sandalphon4869>

5.3 重点考核点

5.3.1 画布设置

画布就是turtle为我们展开用于绘图区域，我们可以设置它的大小和初始位置。
设置画布大小

5.3.1.1 命令一

`turtle.screensize(canvwidth=None, canvheight=None, bg=None)`，
参数分别为画布的宽(单位像素)，高，背景颜色。
如：`turtle.screensize(800,600, "green")`

`turtle.screensize()` #返回默认大小(400, 300)

5.3.1.2 命令二

`turtle.setup(width=0.5, height=0.75, startx=None, starty=None)`，
参数：`width`, `height`：输入宽和高为整数时，表示像素；为小数时，表示占据电脑屏幕的比例，(`startx`, `starty`)：这一坐标表示矩形窗口左上角顶点的位置，如果为空，则窗口位于屏幕中心。
如：`turtle.setup(width=0.6,height=0.6)`
`turtle.setup(width=800,height=800, startx=100, starty=100)`

5.3.1.3 案例一

In []:

插入案例

5.3.2 画笔

5.3.2.1 画笔的状态

在画布上，默认有一个坐标原点为画布中心的坐标轴，坐标原点上有一只面朝x轴正方向小乌龟。这里我们描述小乌龟时使用了两个词语：坐标原点(位置)，面朝x轴正方向(方向)，turtle绘图中，就是使用位置方向描述小乌龟(画笔)的状态。

5.3.2.2 画笔的属性

画笔(画笔的属性，颜色、画线的宽度等)

- 1) turtle.pensize(): 设置画笔的宽度;
- 2) turtle.pencolor(): 没有参数传入，返回当前画笔颜色，传入参数设置画笔颜色，可以是字符串如"green", "red", 也可以是RGB 3元组。
- 3) turtle.speed(speed): 设置画笔移动速度，画笔绘制的速度范围[0,10]整数，数字越大越快。

5.3.2.3 案例二

In []:

```
import turtle
turtle.forward(100)
turtle.pensize(20)
turtle.pencolor("yellow")
turtle.forward(100)
turtle.done()
```

5.3.3 绘图命令

操纵海龟绘图有着许多的命令，这些命令可以划分为3种：一种为运动命令，一种为画笔控制命令，还有一种是全局控制命令。

5.3.3.1 画笔运动命令

命令	说明
turtle.forward(distance)	向当前画笔方向移动distance像素长度
turtle.backward(distance)	向当前画笔相反方向移动distance像素长度
turtle.right(degree)	顺时针移动degree°
turtle.left(degree)	逆时针移动degree°
turtle.pendown()	移动时绘制图形，缺省时也为绘制
turtle.goto(x,y)	将画笔移动到坐标为x,y的位置
turtle.penup()	提起笔移动，不绘制图形，用于另起一个地方绘制
turtle.circle()	画圆，半径为正(负)，表示圆心在画笔的左边(右边)画圆
turtle.setx()	将当前x轴移动到指定位置
turtle.sety()	将当前y轴移动到指定位置
setheading(angle)	设置当前朝向为angle角度
turtle.home()	设置当前画笔位置为原点，朝向东。
turtle.dot(r)	绘制一个指定直径和颜色的圆点

5.3.3.2 案例三

In [17]:

```
import turtle
turtle.pendown()
turtle.forward(100)
turtle.sety(250)
turtle.penup()
turtle.forward(100)
turtle.home()
turtle.done()
```

5.3.3.3 画笔控制命令

命令	说明
turtle.fillcolor(colorstring)	绘制图形的填充颜色
turtle.color(color1, color2)	同时设置pencolor=color1, fillcolor=color2
turtle.filling()	返回当前是否在填充状态
turtle.begin_fill()	准备开始填充图形
turtle.end_fill()	填充完成
turtle.hideturtle()	隐藏画笔的turtle形状
turtle.showturtle()	显示画笔的turtle形状

5.3.3.4 案例四

In [21]:

```
# 画红色的菱形
import turtle
turtle.fillcolor("red")
turtle.begin_fill()
turtle.circle(100,360,4)
turtle.end_fill()
turtle.done()
```

5.3.3.5 全局控制命令

命令	说明
turtle.clear()	清空turtle窗口，但是turtle的位置和状态不会改变
turtle.reset()	清空窗口，重置turtle状态为起始状态
turtle.write(s [,font=("font-name",font_size,"font_type")])	写文本，s为文本内容，font是字体的参数，分别为字体名称，大小和类型；font为可选项，font参数也是可选项

5.3.3.6 案例五

In [25]:

```
# 画红色的菱形
import turtle
turtle.fillcolor("red")
turtle.begin_fill()
turtle.circle(100,360,4)
turtle.end_fill()
turtle.clear()
turtle.done()
```

5.3.3.7 其他命令

`turtle.done()` 必须是乌龟图形程序中的最后一个语句。

5.4 练习一

5.4.1 题目要求

绘制一个半径为100的红色的圆

5.4.2 参考程序

In [27]:

```
import turtle
turtle.color("yellow", "red")
turtle.begin_fill()
turtle.circle(100)
turtle.end_fill()
turtle.done()
```

5.5 练习二

5.5.1 题目要求

绘制一个8

5.5.2 参考程序

In [29]:

```
import turtle
turtle.pensize(20)
turtle.circle(100)
turtle.circle(-100)
turtle.done()
```

5.6 练习三

5.6.1 题目要求

绘制一个正方形，边框为红色，填充为绿色

5.6.2 参考程序

In [37]:

```
import turtle
turtle.pensize(10)
turtle.color("red", "green")
turtle.begin_fill()
turtle.forward(200)
turtle.left(90)
turtle.forward(200)
turtle.left(90)
turtle.forward(200)
turtle.left(90)
turtle.forward(200)
turtle.left(90)
turtle.end_fill()
turtle.done()
```

5.7 练习四

5.7.1 题目要求

绘制一段曲线

5.7.2 参考程序

In [35]:

```
import turtle
turtle.penup()
turtle.goto(-200, -200)
turtle.pendown()
turtle.pensize(10)
turtle.circle(100, 30)
turtle.circle(-100, 30)
turtle.circle(100, 30)
turtle.circle(-100, 30)
turtle.circle(100, 30)
turtle.circle(-100, 30)
turtle.circle(100, 30)
turtle.circle(-100, 30)
turtle.circle(100, 30)
turtle.circle(-100, 30)
turtle.done()
```

5.8 练习五

5.8.1 题目要求

5.8.2 参考程序

In [47]:

5.8.2.2 方法二

In [45]:

5.9 练习六

5.9.1 题目要求

5.9.2 参考程序

5.9.2.1 方法一

In [7]:

```
import turtle
turtle.pensize(10)
turtle.color("yellow", "red")
turtle.begin_fill()
turtle.forward(200)
turtle.left(120)
turtle.forward(200)
turtle.left(120)
turtle.forward(200)
turtle.left(120)
turtle.end_fill()
turtle.hideturtle()
turtle.done()
```

5.9.3 方法二

In [9]:

```
import turtle
turtle.pensize(5)
turtle.color("yellow", "red")
turtle.begin_fill()
turtle.circle(100, 360, 3)
turtle.end_fill()
turtle.done()
```

5.10 练习七

5.10.1 题目要求

绘制一个五角星

5.10.2 参考程序

5.10.2.1 方法一

In [1]:

```
import turtle
turtle.pensize(5)
turtle.color("yellow", "red")
turtle.begin_fill()
for i in range(5):
    turtle.forward(200)
    turtle.right(144)
turtle.end_fill()
turtle.hideturtle()
turtle.done()
```

5.10.2.2 方法二

In [3]:

```
import turtle
turtle.pensize(10)
turtle.color("yellow", "red")
turtle.begin_fill()
turtle.forward(200)
turtle.right(144)
turtle.forward(200)
turtle.right(144)
turtle.forward(200)
turtle.right(144)
turtle.forward(200)
turtle.right(144)
turtle.forward(200)
turtle.right(144)
turtle.end_fill()
turtle.hideturtle()
turtle.done()
```

5.11 练习八

5.11.1 题目要求

绘制一个半圆

5.11.2 参考程序

In [13]:

```
import turtle
turtle.circle(100, 180)
turtle.done()
```

5.12 练习九

5.12.1 题目要求

绘制一个直径为50的蓝色的圆点

5.12.2 参考程序

In [15]:

```
import turtle
turtle.dot(50, "blue")
turtle.done()
```

5.13 练习十

5.13.1 题目要求

输入一个数字，然后绘制该数字为边数的正多边形，并且自动调整图形的大小

5.13.2 参考程序

5.13.2.1 方法一

In [33]:

```
import turtle
a=int(input("请输入边数："))
turtle.pensize(5)
turtle.color("yellow","red")
turtle.penup()
turtle.sety(-300)
turtle.pendown()
turtle.begin_fill()
turtle.circle(900/a,360,a)
turtle.end_fill()
turtle.done()
```

请输入边数：3

5.13.2.2 方法二

In [43]:

```
import turtle
a=int(input("请输入边数："))
turtle.pensize(5)
turtle.color("yellow","red")
turtle.penup()
turtle.goto(-100,-200)
turtle.pendown()
turtle.begin_fill()
for i in range(a):
    turtle.forward(900/a)
    turtle.left(360/a)
turtle.end_fill()
turtle.done()
```

请输入边数：4

5.14 练习十一

5.14.1 题目要求

画一个正方形，里面嵌套一个红色填充的圆形

5.14.2 参考程序

5.14.2.1 方法一

In [63]:

```
import turtle
turtle.penup() #画笔抬起
turtle.goto(-100,100) #回到画正方形初始位置
turtle.pendown() #落下画笔
turtle.forward(200) #从当前画笔方向移动100
turtle.right(90) #顺时针移动90°
turtle.forward(200) #从当前画笔方向移动100
turtle.right(90) #顺时针移动90°
turtle.forward(200) #从当前画笔方向移动100
turtle.right(90) #顺时针移动90°
turtle.forward(200) #从当前画笔方向移动100
turtle.right(90) #顺时针移动90°
turtle.penup() #画笔抬起
turtle.goto(0,-100) #移动到 (0, -100) 的位置，也就是画圆开始的位置
turtle.pendown() #落下画笔
turtle.fillcolor('red') #设置填充颜色为红色
turtle.begin_fill() #开始填充
turtle.circle(100) #画一个半径为100的圆，圆心在画笔左边
turtle.speed(60) #速度为60
turtle.end_fill() #填充结束
turtle.done() #停止画笔等待关闭画布
```

6 附加练习

6.1 太阳花

In [59]:

```
# coding=utf-8
import turtle
import time

# 同时设置pencolor=color1, fillcolor=color2
turtle.color("red", "yellow")

turtle.begin_fill()
for _ in range(50):
    turtle.forward(200)
    turtle.left(170)
turtle.end_fill()

turtle.mainloop()
```

6.2 螺旋彩色六边形

In [70]:

```
import turtle
turtle.pensize(2)
turtle.bgcolor("black")
turtle.speed(0)
colors = ["red", "yellow", "blue", "orange", "green", "purple"]
for i in range(100) :
    turtle.pencolor(colors[i % 6])
    turtle.forward(i)
    turtle.left(60 + 1)

turtle.hideturtle()

turtle.done()
```

6.3 时钟

In [25]:

```
# coding=utf-8

import turtle
from datetime import *

# 抬起画笔，向前运动一段距离放下
def Skip(step): #自定义一个Skip函数，该函数的功能：抬起笔前进step个像素点，然后落笔
    turtle.penup()
    turtle.forward(step)
    turtle.pendown()

def mkHand(name, length):
    # 注册Turtle形状，建立表针Turtle
    turtle.reset()
    Skip(-length * 0.1)
    # 开始记录多边形的顶点。当前的乌龟位置是多边形的第一个顶点。
    turtle.begin_poly()
    turtle.forward(length * 1.1)
    # 停止记录多边形的顶点。当前的乌龟位置是多边形的最后一个顶点。将与第一个顶点相连。
    turtle.end_poly()
    # 返回最后记录的多边形。
    handForm = turtle.get_poly()
    turtle.register_shape(name, handForm)

def Init():
    global secHand, minHand, hurHand, printer
    # 重置Turtle指向北
    turtle.mode("logo")
    # 建立三个表针Turtle并初始化
    mkHand("secHand", 135)
    mkHand("minHand", 125)
    mkHand("hurHand", 90)
    secHand = turtle.Turtle()
    secHand.shape("secHand")
    minHand = turtle.Turtle()
    minHand.shape("minHand")
    hurHand = turtle.Turtle()
    hurHand.shape("hurHand")

    for hand in secHand, minHand, hurHand:
        hand.shapesize(1, 1, 3)
        hand.speed(0)

    # 建立输出文字Turtle
    printer = turtle.Turtle()
    # 隐藏画笔的turtle形状
    printer.hideturtle()
    printer.penup()

def SetupClock(radius):
    # 建立表的外框
    turtle.reset()
    turtle.pensize(7)
    for i in range(60):
        Skip(radius)
        if i % 5 == 0:
            turtle.forward(20)
            Skip(-radius - 20)
```



```

Skip(radius + 20)
if i == 0:
    turtle.write(int(12), align="center", font=("Courier", 14, "bold"))
elif i == 30:
    Skip(25)
    turtle.write(int(i/5), align="center", font=("Courier", 14, "bold"))
    Skip(-25)
elif (i == 25 or i == 35):
    Skip(20)
    turtle.write(int(i/5), align="center", font=("Courier", 14, "bold"))
    Skip(-20)
else:
    turtle.write(int(i/5), align="center", font=("Courier", 14, "bold"))
Skip(-radius - 20)
else:
    turtle.dot(5)
    Skip(-radius)
turtle.right(6)

def Week(t):
    week = ["星期一", "星期二", "星期三",
            "星期四", "星期五", "星期六", "星期日"]
    return week[t.weekday()]

def Date(t):
    y = t.year
    m = t.month
    d = t.day
    return "%s年%d月%d日" % (y, m, d)

def Tick():
    # 绘制表针的动态显示
    t = datetime.today()
    second = t.second + t.microsecond * 0.000001
    minute = t.minute + second / 60.0
    hour = t.hour + minute / 60.0
    secHand.setheading(6 * second)
    minHand.setheading(6 * minute)
    hurHand.setheading(30 * hour)

    turtle.tracer(False)
    printer.forward(65)
    printer.write(Week(t), align="center",
                  font=("Courier", 14, "bold"))
    printer.back(130)
    printer.write(Date(t), align="center",
                  font=("Courier", 14, "bold"))
    printer.home()
    turtle.tracer(True)

    # 100ms后继续调用tick
    turtle.ontimer(Tick, 100)

def main():
    # 打开/关闭龟动画，并为更新图纸设置延迟。
    turtle.tracer(False)
    Init()
    SetupClock(160)
    turtle.tracer(True)
    Tick()
    turtle.mainloop()

```

```
if __name__ == "__main__":  
    main()
```

Exception in Tkinter callback

Traceback (most recent call last):

```
File "C:\ProgramData\Anaconda3\lib\tkinter\__init__.py", line 1699, in __call__  
    return self.func(*args)  
File "C:\ProgramData\Anaconda3\lib\tkinter\__init__.py", line 745, in callit  
    func(*args)  
File "<ipython-input-25-e4fd32932e97>", line 98, in Tick  
    hurHand.setheading(30 * hour)  
File "C:\ProgramData\Anaconda3\lib\turtle.py", line 1936, in setheading  
    self._rotate(angle)  
File "C:\ProgramData\Anaconda3\lib\turtle.py", line 3278, in _rotate  
    self._update()  
File "C:\ProgramData\Anaconda3\lib\turtle.py", line 2660, in _update  
    self._update_data()  
File "C:\ProgramData\Anaconda3\lib\turtle.py", line 2646, in _update_data  
    self.screen._incrementudc()  
File "C:\ProgramData\Anaconda3\lib\turtle.py", line 1292, in _incrementudc  
    raise Terminator  
turtle.Terminator
```

6.4 纪念日

In []:

```

# -*- coding: utf-8 -*-
"""
Created on Mon Jul 23 09:12:41 2018
@author: Administrator
"""
#代码写的比较乱，直接在前一个案例基础上改进的
#SevenDigitsDrawV2.py
import turtle, time# 导入海龟和时间
def drawGap(): #绘制数码管间隔
    turtle.penup() #抬笔
    turtle.fd(5) #前进5个像素
def drawLine(draw): #绘制单段数码管
    drawGap()
    turtle.pendown() if draw else turtle.penup()
    turtle.fd(40)
    drawGap()
    turtle.right(90)
def drawDigit(d): #根据数字绘制七段数码管
    drawLine(True) if d in [2,3,4,5,6,8,9] else drawLine(False)
    drawLine(True) if d in [0,1,3,4,5,6,7,8,9] else drawLine(False)
    drawLine(True) if d in [0,2,3,5,6,8,9] else drawLine(False)
    drawLine(True) if d in [0,2,6,8] else drawLine(False)
    turtle.left(90)
    drawLine(True) if d in [0,4,5,6,8,9] else drawLine(False)
    drawLine(True) if d in [0,2,3,5,6,7,8,9] else drawLine(False)
    drawLine(True) if d in [0,1,2,3,4,7,8,9] else drawLine(False)
    turtle.left(180)
    turtle.penup()
    turtle.fd(20) #前进20
def drawDate(date): #显示时间
    turtle.pencolor("red")
    for i in date:
        if i == '-':
            turtle.write('年', font=("Arial", 18, "normal"))
            turtle.pencolor("green")
            turtle.fd(40)
        elif i == '=':
            turtle.write('月', font=("Arial", 18, "normal"))
            turtle.pencolor("blue")
            turtle.fd(40)
        elif i == '+':
            turtle.write('日', font=("Arial", 18, "normal"))
            turtle.fd(40)
        else:
            drawDigit(eval(i))
def all(day):
    turtle.goto(-350,-300)
    turtle.pencolor("orange")
    turtle.write('总共', font=("Arial", 40, "normal"))
    turtle.fd(110)
    for j in day:
        drawDigit(eval(j))
    turtle.write('天', font=("Arial", 18, "normal"))
def count(t1,t2,t3):
    t=t1*365
    if t2 in [1,2]:
        t+=t2*30
    if t2 in [3]:
        t=t+91

```

```
    if t2==4:
        t+=122
    if t2==5:
        t+=152
    if t2==6:
        t+=183
    if t2==7:
        t+=213
    if t2==8:
        t+=244
    if t2==9:
        t+=275
    if t2==10:
        t+=303
    if t2==11:
        t+=334
    t+=t3
    return(str(t))
def text():
    turtle.penup()
    turtle.goto(-350,400)
    turtle.pendown()
    turtle.write('今天是:',font=("Arial", 18, "normal"))
    turtle.pensize(5)
    turtle.penup()
    turtle.goto(-350,300)
    turtle.pendown()
    drawDate(time.strftime('%Y-%m=%d+',time.gmtime()))
    turtle.penup()
    turtle.goto(-350,200)
    turtle.pensize(1)
    turtle.pendown()
    turtle.pencolor("black")
    turtle.write('小象出生在:',font=("Arial", 18, "normal"))
    turtle.penup()
    turtle.goto(-350,100)
    turtle.pendown()
    turtle.pensize(5)
    drawDate('2009-04=29+')
    turtle.penup()
    turtle.goto(-350,0)
    turtle.pensize(1)
    turtle.pendown()
    turtle.pencolor("black")
    turtle.write('他已经成长了:',font=("Arial", 18, "normal"))
    turtle.penup()
    turtle.goto(0,-100)
    turtle.pensize(1)
    turtle.pendown()
def main():
    turtle.setup(900, 900, 200, 0)
    text()
    turtle.penup()
    turtle.fd(-350)
    turtle.pensize(5)
    # drawDate('2009-04=29+')
    t1=time.gmtime()
    t2=t1.tm_year-2009
    t3=t1.tm_mon-4
    if t3<0:
        t2-=1
```

```

        t3+=12
    t4=t1.tm_mday-29
    if t4<0:
        t3-=1
        if t1.tm_mon-1 in [1,3,5,7,8,10,12]:
            t4+=31
        else:
            t4+=30
    tatol=count(t2,t3,t4)
    drawDate(str(t2)+'-'+str(t3)+'-'+str(t4)+'+')
    all(tatol)
    turtle.hideturtle()
    turtle.done()
main()

```

6.5 画树叶

In []:

```

from numpy import *
from random import random
import turtle
turtle.reset()
x = array([[.5],[.5]])
p = [0.85,0.92,0.99,1.00]
A1 = array([[.85, 0.04],
            [-0.04,.85]])
b1 = array([[0],[1.6]])
A2 = array([[0.20,-0.26],
            [0.23,0.22]])
b2 = array([[0],[1.6]])
A3 = array([[-0.15,0.28],
            [0.26,0.24]])
b3 = array([[0],[0.44]])

A4 = array([[0,0],
            [0,0.16]])

turtle.color("blue")
cnt = 1
while True:
    cnt += 1
    if cnt == 2000:
        break
    r = random()
    if r < p[0]:
        x = dot(A1 , x) + b1
    elif r < p[1]:
        x = dot(A2 , x) + b2
    elif r < p[2]:
        x = dot(A3 , x) + b3
    else:
        x = dot(A4 , x)
    #print x[1]
    turtle.up()
    turtle.goto(x[0][0] * 50,x[1][0] * 40 - 240)
    turtle.down()
    turtle.dot()

```

In [61]:

```
import turtle
turtle.penup() #画笔抬起
turtle.goto(-100,100) #回到画正方形初始位置
turtle.pendown() #落下画笔
turtle.forward(200) #从当前画笔方向移动100
turtle.right(90) #顺时针移动90°
turtle.forward(200) #从当前画笔方向移动100
turtle.right(90) #顺时针移动90°
turtle.forward(200) #从当前画笔方向移动100
turtle.right(90) #顺时针移动90°
turtle.forward(200) #从当前画笔方向移动100
turtle.right(90) #顺时针移动90°
turtle.penup() #画笔抬起
turtle.goto(0,-100) #移动到(0,-100)的位置,也就是画圆开始的位置
turtle.pendown() #落下画笔
turtle.fillcolor('red') #设置填充颜色为红色
turtle.begin_fill() #开始填充
turtle.circle(100) #画一个半径为100的圆,圆心在画笔左边
turtle.speed(60) #速度为60
turtle.end_fill() #填充结束
turtle.done() #停止画笔等待关闭画布
```

6.6 彩色螺旋

In [72]:

```
import turtle
turtle.speed(0)
turtle.pensize(2)
turtle.bgcolor("black")
colors=["red","blue","yellow","purple"]
for x in range(300):
    turtle.color(colors[x%4])
    turtle.forward(2*x)
    turtle.left(91)
turtle.done()
```

In [74]:

```
import turtle
turtle.speed(0)
turtle.delay(0)
turtle.pensize(2)
turtle.bgcolor("black")
colors=["red","blue","yellow","purple"]
for x in range(300):
    turtle.color(colors[x%4])
    turtle.forward(2*x)
    turtle.left(91)
turtle.done()
```

In [78]:

```
import turtle
turtle.tracer(0)
turtle.pensize(2)
turtle.bgcolor("black")
colors=["red","blue","yellow","purple"]
for x in range(300):
    turtle.color(colors[x%4])
    turtle.forward(2*x)
    turtle.left(91)
turtle.done()
```