

Python语言程序设计

张幸林 副教授

csxlzhang@scut.edu.cn

Python语言程序设计

- 参考资料

- <<Python编程-从入门到实践>>
- <<Python语言程序设计基础(第2版)>>
- 网络资源
-

Python语言程序设计

- 预期主要内容：
 - 数据类型
 - 函数
 - 面向对象基础
 - Python常见库介绍
 - Python应用举例（科学计算与数据可视化，Web应用？）

Python语言程序设计

- 实验作业：
 - 每周实验习题，课后完成，3-5道题，几乎都是通过编程完成，提交电子版实验报告和源代码。
- 期末考核：
 - 6-10道编程习题
 - 各自完成一个编程项目，可以为数据分析、小型游戏、个人网站等，并编写一份文档，对项目进行简单介绍。
- 课程成绩：
 - 平时成绩：课程考勤+平时实验完成情况
 - 期末考核：期末编程习题完成情况+项目难度及完成情况

Python语言程序设计

- 实验作业布置与提交：
 - 每周课程结束后，作业布置在QQ群里。作业由各班课代表收集，作业按照实验报告模板提交，打包后发到邮箱：542430631@qq.com。



QQ群号：861650286

群名称：2019Python课程

群 号：861650286

Python语言程序设计

- 提交文件夹的格式

 学号_姓名		2019/8/22 16:28	文件夹
此电脑 > DATA (D:) > 学号_姓名			
名称 ^		修改日期	类型
 源代码		2019/8/22 16:28	文件夹
 实验一.docx		2019/8/22 16:28	Microsoft Word ...

Python语言概述

Python语言的诞生



Guido van Rossum

Python 1: 1994

Python 2: 2000

Python 3: 2008

应用领域

- 数据分析、组件集成、网络服务、图像处理、数值计算和科学计算等众多领域。
- 大中型互联网企业都在使用Python，如：Youtube、Dropbox、Quora、豆瓣、知乎、Google、Yahoo!、Facebook、NASA、百度、腾讯等

TIOBE Index for August 2019

Aug 2019	Aug 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.028%	-0.85%
2	2		C	15.154%	+0.19%
3	4	⬆	Python	10.020%	+3.03%
4	3	⬇	C++	6.057%	-1.41%
5	6	⬆	C#	3.842%	+0.30%
6	5	⬇	Visual Basic .NET	3.695%	-1.07%
7	8	⬆	JavaScript	2.258%	-0.15%
8	7	⬇	PHP	2.075%	-0.85%
9	14	⬆	Objective-C	1.690%	+0.33%
10	9	⬇	SQL	1.625%	-0.69%

PYPL Index for August 2019

Rank	Change	Language	Share	Trend
1		Python	28.73 %	+4.5 %
2		Java	20.0 %	-2.1 %
3		Javascript	8.35 %	-0.1 %
4		C#	7.43 %	-0.5 %
5		PHP	6.83 %	-1.0 %
6		C/C++	5.87 %	-0.3 %
7		R	3.92 %	-0.2 %
8		Objective-C	2.7 %	-0.6 %
9		Swift	2.41 %	-0.3 %
10		Matlab	1.87 %	-0.3 %

“Hello World” in Java

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello world!");  
    }  
}
```

“Hello World” in C++

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello world!" << endl;
}
```

“Hello World” in Python

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

“Hello World” in Python

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

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语言的特点

- Python语言是通用语言
- Python语言是脚本语言
- Python语言是开源语言
- Python语言是跨平台语言

Python开发环境配置

安装

- 到Python主页下载并安装Python基本开发和运行环境，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.X系列版本程序

安装



The screenshot shows the Python.org website with a dark blue header. The Python logo is on the left, and navigation links (About, Downloads, Documentation, Community, Success Stories, News, Events) are in the center. On the right, there is a 'Donate' button, a search bar with a 'GO' button, and a 'Socialize' link. Below the header, the 'Downloads' section is highlighted. The main heading is 'Download the latest version for Windows'. A yellow button labeled 'Download Python 3.7.4' is circled in red. Below this button, there are links for 'Python for Windows, Linux/UNIX, Mac OS X, Other', 'Pre-releases', 'Docker images', and a note about Python 2.7.

python™

Donate Search GO Socialize

About Downloads Documentation Community Success Stories News Events

Download the latest version for Windows

Download Python 3.7.4

Looking for Python with a different OS? Python for [Windows](#),
[Linux/UNIX](#), [Mac OS X](#), [Other](#)

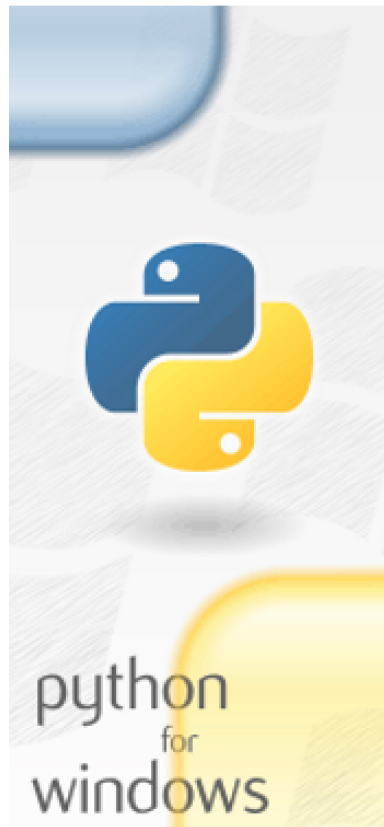
Want to help test development versions of Python? [Pre-releases](#),
[Docker images](#)

Looking for Python 2.7? See below for specific releases



安装

Python 3.7.4 (32-bit) Setup



Install Python 3.7.4 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.



Install Now

C:\Users\Lenovo\AppData\Local\Programs\Python\Python37-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations



Customize installation

Choose location and features

☒ Install launcher for all users (recommended)

☒ Add Python 3.7 to PATH



Cancel

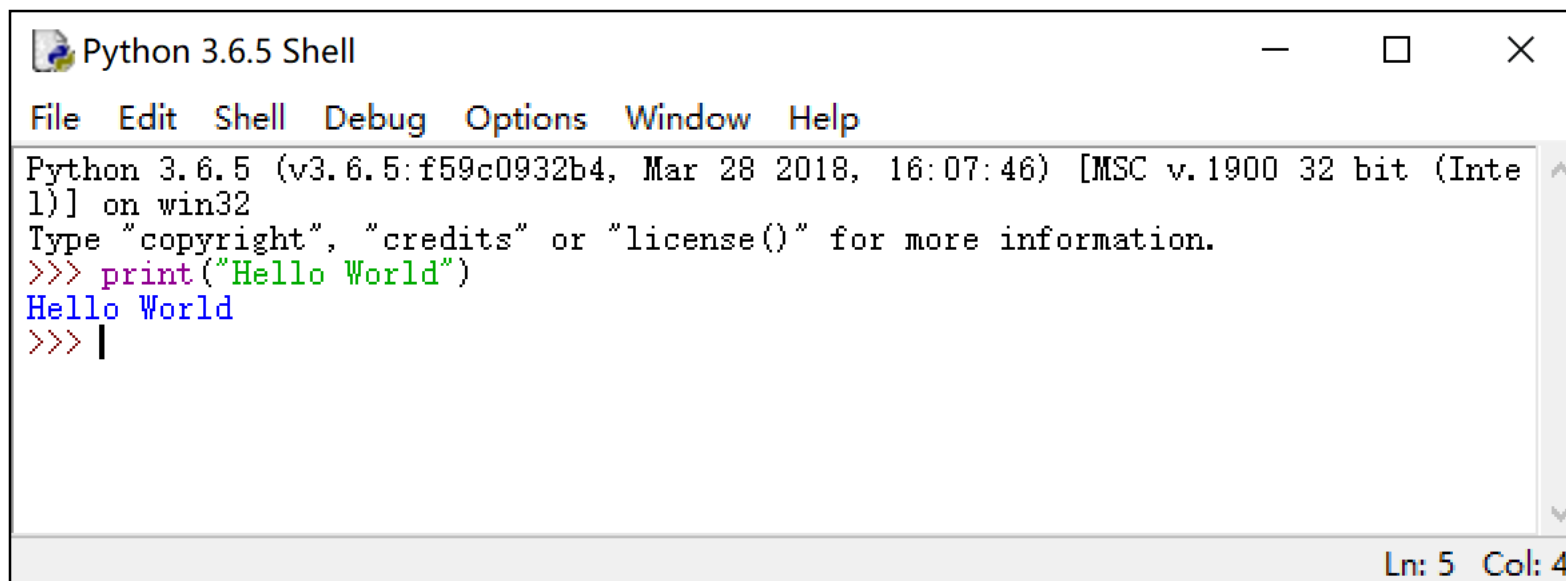
启动

- **方法1:** 启动Windows命令行工具，输入python

```
C:\Users\Lenovo>python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>>
```

启动

- **方法2:** 调用IDLE来启动Python图形化运行环境



The screenshot shows a window titled "Python 3.6.5 Shell" with a standard menu bar (File, Edit, Shell, Debug, Options, Window, Help). The main text area displays the following text:

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>> |
```

The status bar at the bottom right indicates "Ln: 5 Col: 4".

启动

- **方法3:** 按照语法格式编写代码，编写可以用任何文本编辑器，保存为文件。

```
D:\>python hello_world.py  
Hello World
```

启动

- **方法4:** 打开IDLE, 点击Ctrl+N打开一个新窗口, 输入语句并保存, 使用快捷键F5即可运行该程序
- **方法5:** 将Python集成到Eclipse、PyCharm等面向较大规模项目开发的集成开发环境中

温度转换程序实例

温度体系

- 温度刻画存在不同体系，摄氏度以1标准大气压下水的结冰点为0度，沸点为100度，将温度进行等分刻画。华氏度以1标准大气压下水的结冰点为32度，沸点为212度，将温度进行等分刻画。

温度转换实例

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

温度转换实例

■ 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

温度转换实例

■ 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

温度转换实例

■ 步骤4：编写程序

```
#TempConvert.py
TempStr = input("请输入带有符号的温度值：")
if TempStr[-1] in ['F', 'f']:
    C = (eval(TempStr[0:-1]) - 32)/1.8
    print("转换后的温度是{:.2f}C".format(C))
elif TempStr[-1] in ['C', 'c']:
    F = 1.8*eval(TempStr[0:-1]) + 32
    print("转换后的温度是{:.2f}F".format(F))
else:
    print("输入格式错误")
```

Python语法元素分析

程序的格式框架

- Python语言采用严格的“缩进”来表明程序的格式框架。
 - 缩进指每一行代码开始前的空白区域，用来表示代码之间的包含和层次关系。
 - 1个缩进 = 4个空格
 - 缩进是Python语言中表明程序框架的唯一手段

程序的格式框架

单层缩进

```
#e1.1TempConvert.py
TempStr = input("请输入带有符号
if TempStr[-1] in ['F','f']:
    ↪ C = (eval(TempStr[0:-1]) -
    print("转换后的温度是{:.2f}C
elif TempStr[-1] in ['C','c']:
    ↪ F = 1.8*eval(TempStr[0:-1])
    print("转换后的温度是{:.2f}F
else:
    ↪ print("输入格式错误")
```

多层缩进

```
DARTS = 1000
hits = 0.0
clock()
for i in range(1, DARTS):
    ↪ x, y = random(), random()
    ↪ dist = sqrt(x ** 2 + y ** 2)
    ↪ if dist <= 1.0:
    ↪     ↪ hits = hits + 1
pi = 4 * (hits/DARTS)
print("Pi的值是{:.2f}".format
```

注释

- 单行注释

Single line comments start with a '#'

- 多行注释

"""

Multiline strings can be written using three "s, and are often used as function and module comments

"""

命名与保留字

- **常量**：程序中值不发生改变的元素
- **变量**：程序中值发生改变或者可以发生改变的元素

Python语言允许采用大写字母、小写字母、数字、下划线(_)和汉字等字符及其组合给变量命名，但名字的首字符不能是数字，中间不能出现空格，长度没有限制

注意：标识符对**大小写敏感**，python和Python是两个不同的名字

命名与保留字

- 保留字，也称为关键字，指被编程语言内部定义并保留使用的标识符。
- 程序员编写程序不能定义与保留字相同的标识符。
- 每种程序设计语言都有一套保留字，保留字一般用来构成程序整体框架、表达关键值和具有结构性的复杂语义等。
- 掌握一门编程语言首先要熟记其所对应的保留字。

命名与保留字

- Python 3.x保留字列表 (33个)

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

变量

```
x = 2
```

无需分号

```
x * 7
```

```
# => 14
```

```
x = "Hello, I'm "
```

```
x + "Python!"
```

```
# => "Hello, I'm Python!"
```

变量类型？

In Java or C++

```
int x = 0;
```

变量类型？

In Python

x = 0

变量类型？

- **动态类型(dynamically-typed)**: 无需声明，在赋值时变量可以重新赋值为任意值。
- Python中“万物皆对象”，变量名与对象是划分开的，对象有相应的类型

变量类型？

- **动态类型(dynamically-typed)**: 无需声明，在赋值时变量可以重新赋值为任意值。
- Python中“万物皆对象”，变量名与对象是划分开的，对象有相应的类型

```
type(1)           # => <class 'int'>
type("Hello")    # => <class 'str'>
type(None)        # => <class 'NoneType'>
```

变量类型？

- **动态类型(dynamically-typed)**: 无需声明，在赋值时变量可以重新赋值为任意值。
- Python中“万物皆对象”，变量名与对象是划分开的，对象有相应的类型

```
type(1)           # => <class 'int'>
type("Hello")    # => <class 'str'>
type(None)        # => <class 'NoneType'>
type(int)         # => <class 'type'>
type(type(int))   # => <class 'type'>
```

字符串

- Python语言中，字符串是用两个双引号 “ ” 或者单引号 ‘ ’ 括起来的一个或多个字符。

字符串

- Python语言中，字符串是用两个双引号 “ ” 或者单引号 ‘ ’ 括起来的一个或多个字符。

```
greeting = 'Hello'
```

```
group = "world"
```

字符串

- Python语言中，字符串是用两个双引号 “ ” 或者单引号 ‘ ’ 括起来的一个或多个字符。

```
greeting = 'Hello'
```

```
group = "world"
```

```
greeting + ' ' + group + '!' # => 'Hello  
world!'
```

索引(indexing)

0 1 2 3 4 5 6
S = 'Arthur'

索引(indexing)

0 1 2 3 4 5 6
s = 'Arthur'

s[0]	==	'A'
s[1]	==	'r'
s[4]	==	'u'
s[6]	#	Bad!

逆向索引(negative indexing)

$S =$ 'Arthur'

0	1	2	3	4	5	6
-6	-5	-4	-3	-2	-1	0

逆向索引(negative indexing)

`S = 'Arthur'`

	0	1	2	3	4	5	6
	A	r	t	h	u	r	
-6	-5	-4	-3	-2	-1	0	

<code>s[-1]</code>	<code>==</code>	<code>'r'</code>
<code>s[-2]</code>	<code>==</code>	<code>'u'</code>
<code>s[-4]</code>	<code>==</code>	<code>'t'</code>
<code>s[-6]</code>	<code>==</code>	<code>'A'</code>

切片(slicing)

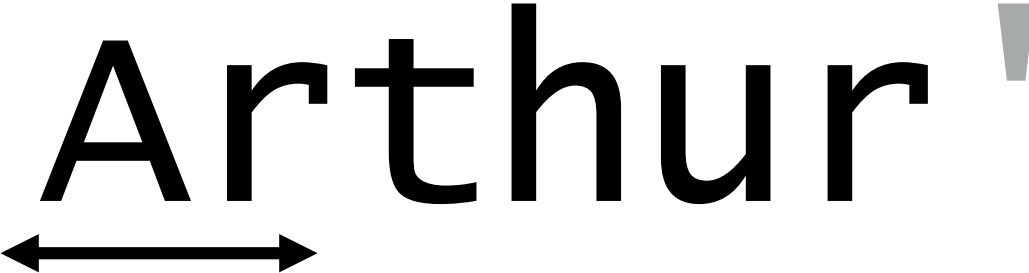
S = 'Arthur'

0 1 2 3 4 5 6

切片(slicing)

S = 'Arthur'

0 1 2 3 4 5 6




The diagram illustrates string slicing on the word 'Arthur'. Above the letters, indices 0 through 6 are shown. A double-headed arrow is positioned below the first two letters, 'A' and 'r', indicating a slice from index 0 to index 2 (exclusive of index 2).

切片(slicing)

S = 'Arthur'

0 1 2 3 4 5 6




`s[0:2] == 'Ar'`

切片(slicing)

S = 'Arthur'

0 1 2 3 4 5 6



`s[0:2] == 'Ar'`

切片(slicing)

S = 'Arthur'

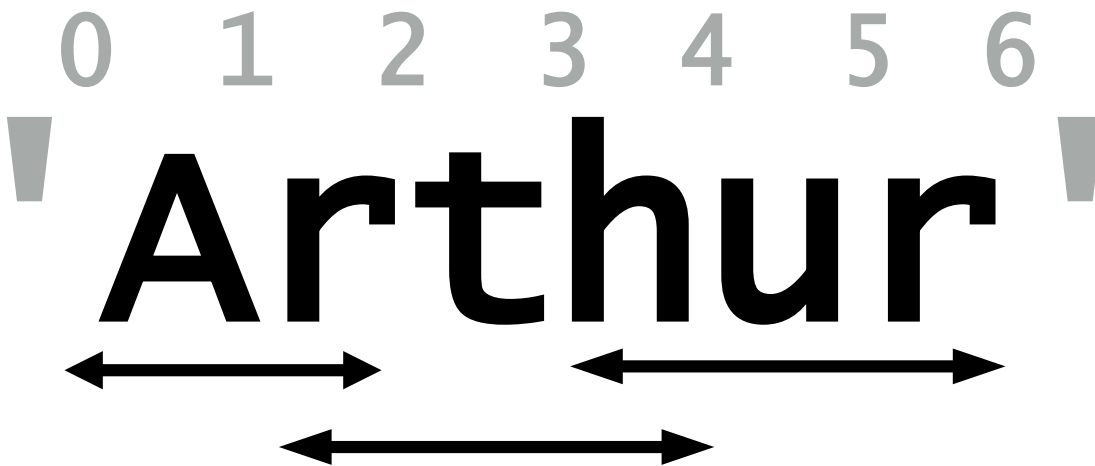
0 1 2 3 4 5 6

The diagram shows the string 'Arthur' with indices 0 through 6 above each character. Below the string, two double-headed arrows indicate slicing ranges: the first arrow spans from index 0 to index 2 (covering 'Ar'), and the second arrow spans from index 3 to index 6 (covering 'hur').

<code>s[0:2]</code>	<code>==</code>	<code>'Ar'</code>
<code>s[3:6]</code>	<code>==</code>	<code>'hur'</code>

切片(slicing)

`s = 'Arthur'`



The diagram illustrates string slicing on the string 'Arthur'. Above the string, indices 0 through 6 are shown. Below the string, three double-headed arrows indicate the ranges: the first arrow spans from index 0 to 2 (covering 'Ar'), the second arrow spans from index 3 to 6 (covering 'hur'), and a third arrow spans from index 1 to 4 (covering 'rth').

<code>s[0:2]</code>	<code>==</code>	<code>'Ar'</code>
<code>s[3:6]</code>	<code>==</code>	<code>'hur'</code>
<code>s[1:4]</code>	<code>==</code>	<code>'rth'</code>

切片(slicing)

0 1 2 3 4 5 6
S = 'Arthur'

Implicitly starts at 0

s[:2] == 'Ar'
s[3:] == 'hur'

Implicitly ends at
the end

切片(slicing)

S = 'Arthur'

0 1 2 3 4 5 6

切片(slicing)

0 1 2 3 4 5 6
s = 'Arthur'

Can also pass a step size

s[1:5:2] == 'rh'
s[4::-2] == 'utA'

切片(slicing)

0 1 2 3 4 5 6
s = 'Arthur'

Can also pass a step size

Reversing a string

```
s[1:5:2] == 'rh'  
s[4::-2] == 'utA'  
s[::-1] == 'ruhtrA'
```

类型转换

`str(42)` `#` `=>` `"42"`

`int("42")` `#` `=>` `42`

`float("2.5")` `#` `=>` `2.5`

`float("1")` `#` `=>` `1.0`

赋值语句

- Python语言中，= 表示“赋值”，即将等号右侧的值计算后将结果值赋给左侧变量，包含等号(=)的语句称为“赋值语句”
- 同步赋值语句：同时给多个变量赋值
<变量1>, ..., <变量N> = <表达式1>, ..., <表达式N>

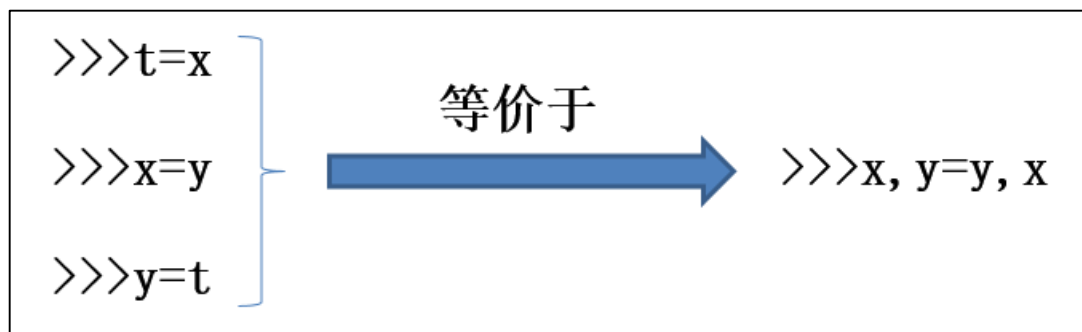
赋值语句

例：将变量x和y交换

- 采用单个赋值，需要3行语句：

即通过一个临时变量t缓存x的原始值，然后将y值赋给x，再将x的原始值通过t赋值给y。

- 采用同步赋值语句，仅需要一行代码：



Console I/O – input()函数

- 获得用户输入之前，input()函数可以包含一些提示性文字

<变量> = input(<提示性文字>)

Read a string from the user

```
>>> name = input("what is your name? ")
```


Console I/O – input()函数

- 获得用户输入之前，input()函数可以包含一些提示性文字

<变量> = input(<提示性文字>)

```
# Read a string from the user
```

```
>>> name = input("what is your name? ")
```

```
what is your name?
```

Console I/O – input()函数

- 获得用户输入之前，input()函数可以包含一些提示性文字

<变量> = input(<提示性文字>)

Read a string from the user

```
>>> name = input("what is your name? ")
```

What is your name? Sam

Console I/O – print()函数

- `print()`函数用来输出字符信息，或以字符形式输出变量。
- `print()`函数可以输出各种类型变量的值。
- `print()`函数通过%来选择要输出的变量。

Console I/O – print()函数

- print()函数用来输出字符信息，或以字符形式输出变量。
- print()函数可以输出各种类型变量的值。
- print()函数通过%或者str.format函数来输出变量。

```
# Read a string from the user
```

```
>>> name = input("What is your name? ")
```

```
What is your name? Sam
```

```
>>> print("I'm Python. Nice to meet you,", name)
```

```
I'm Python. Nice to meet you, Sam
```

eval()函数

- `eval(<字符串>)`函数是Python语言中一个十分重要的函数，它能够以Python表达式的方式解析并执行字符串，将返回结果输出

eval()函数

- `eval(<字符串>)`函数是Python语言中一个十分重要的函数，它能够以Python表达式的方式解析并执行字符串，将返回结果输出

```
>>> x = 1
>>> eval("x + 1")
2
```

eval()函数

- `eval(<字符串>)`函数是Python语言中一个十分重要的函数，它能够以Python表达式的方式解析并执行字符串，将返回结果输出

```
>>> x = 1
```

```
>>> eval("x + 1")
```

```
2
```

```
>>> eval("1.1 + 2.2")
```

```
3.3
```

分支语句

- 分支语句是控制程序运行的一类重要语句，它的作用是根据判断条件选择程序执行路径，使用方式如下：

分支语句

- 分支语句是控制程序运行的一类重要语句，它的作用是根据判断条件选择程序执行路径，使用方式如下：

No parentheses

Colon

```
if the_world_is_flat:
```

No curly braces!

```
    print("Don't fall off!")
```

Use 4 spaces to indent

分支语句

```
if some_condition:
    print('Some condition holds')
elif other_condition:
    print('Other condition holds')
else:
    print('Neither condition holds')
```

Zero or more elifs

else is optional

An Aside

Python has no switch statement,
opting for if/elif/else chains

分支语句

- 回文(Palindrome)?

```
# Palindrome Check
```

```
word = input("Please enter a word: ")
```

```
reversed_word = word[::-1]
```

```
if word == reversed_word:
```

```
    print("Hooray! You entered a palindrome")
```

```
else:
```

```
    print("You did not enter a palindrome")
```

循环语句

- 循环语句：控制程序运行，根据判断条件或计数条件确定一段程序的运行次数

for循环语句

- 循环语句：控制程序运行，根据判断条件或计数条件确定一段程序的运行次数

loop explicitly over data

strings, lists, etc.

for <循环变量> **in** <遍历结构>:
 <语句块>

no loop counter

for循环语句

- else语句扩展

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

- 当for循环正常执行之后，程序会继续执行else语句中内容。else语句只在循环正常执行之后才执行并结束，
- 因此，可以在<语句块2>中放置判断循环执行情况的语句。

for循环语句

```
for s in "BIT":  
    print("循环进行中: " + s)  
else:  
    s = "循环正常结束"  
print(s)
```

```
>>>
```

```
循环进行中: B
```

```
循环进行中: I
```

```
循环进行中: T
```

```
循环正常结束
```

range

```
range(3)  
# generates 0, 1, 2
```

used to iterate over a
sequence of numbers

```
range(5, 10)  
# generates 5, 6, 7, 8, 9
```

```
range(2, 12, 3)  
# generates 2, 5, 8, 11
```

```
range(-7, -30, -5)  
# generates -7, -12, -17, -22, -27
```

`range(stop)` or `range(start, stop[, step])`

while循环语句

- Python通过while关键字实现无限循环
- while循环也有保留字else的扩展模式

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

while循环语句

```
s, idx = "BIT", 0
while idx < len(s):
    print("循环进行中: " + s[idx])
    idx += 1
else:
    s = "循环正常结束"
print(s)
```

```
>>>
```

```
循环进行中: B
```

```
循环进行中: I
```

```
循环进行中: T
```

```
循环正常结束
```

循环保留字: break和continue

- 循环结构有两个辅助保留字: break和continue, 它们用来辅助控制循环执行
- break用来跳出最内层for或while循环, 脱离该循环后程序从循环后代码继续执行

```
for s in "BIT":  
    for i in range(10):  
        print(s, end="")  
        if s=="I":  
            break
```

```
>>>  
BBBBBBBBBBBITTTTTTTTTT
```

循环保留字: break和continue

- continue用来结束当前当次循环，即跳出循环体中下面尚未执行的语句，但不跳出当前循环。
- 对于while循环，继续求解循环条件。而对于for循环，程序流程接着遍历循环列表
- 对比continue和break语句，如下

```
for s in "PYTHON":  
    if s=="T":  
        continue  
    print(s, end="")
```

```
>>>
```

```
PYHON
```

```
for s in "PYTHON":  
    if s=="T":  
        break  
    print(s, end="")
```

```
>>>
```

```
PY
```

循环保留字: break和continue

- for循环和while循环中都存在一个else扩展用法。
- else中的语句块只在一种条件下执行，即for循环正常遍历了 所有内容没有因为break或return而退出。
- continue保留字对else没有影响。看下面两个例子

```
for s in "PYTHON":  
    if s=="T":  
        continue  
    print(s, end="")  
else:  
    print("正常退出")
```

```
>>>
```

```
PYHON正常退出
```

```
for s in "PYTHON":  
    if s=="T":  
        break  
    print(s, end="")  
else:  
    print("正常退出")
```

```
>>>
```

```
PY
```

函数

- 函数可以理解为对一组表达特定功能表达式的封装

函数

- 函数可以理解为对一组表达特定功能表达式的封装

the def keyword
defines a function

parameters have
no explicit types

```
def fn_name(param1, param2):  
    value = do_something()  
    return value
```

return is optional
if either return or its value are omitted,
implicitly returns None

Prime Number Generator

```
def is_prime(n)
    for i in range(2, n):
        if n % i == 0
            return False
    return True

n = input("Enter a number: ")
for x in range(2, int(n)):
    if is_prime(x):
        print(x, " is prime")
    else
        print(x, " is not prime")
```

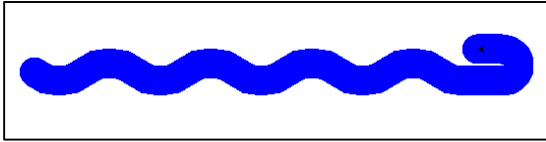

turtle库和蟒蛇绘制程序

Python小蛇

- Python英文是蟒蛇的意思
- 通过下面的例子，来实践用Python语言输出图形效果。



Python蟒蛇绘制实例



```
import turtle
def drawSnake(radius, angle, length, neckrad):
    for i in range(length):
        turtle.circle(radius, angle)
        turtle.circle(-radius, angle)
    turtle.circle(radius, angle/2)
    turtle.fd(radius)
    turtle.circle(neckrad+1, 180)
    turtle.fd(radius*2/3)
def main():
    turtle.setup(1300, 800, 0, 0)
    turtle.penup()
    turtle.fd(-250)
    turtle.pendown()
    pythonsize = 30
    turtle.pensize(pythonsize)
    turtle.pencolor("blue")
    turtle.seth(-40)
    drawSnake(40, 80, 4, pythonsize/2)
    turtle.done()
main()
```

Python语法元素

■ **import** turtle

■ **import**是一个关键字，用来引入一些外部库，这里的含义是引入一个名字叫**turtle**的函数库

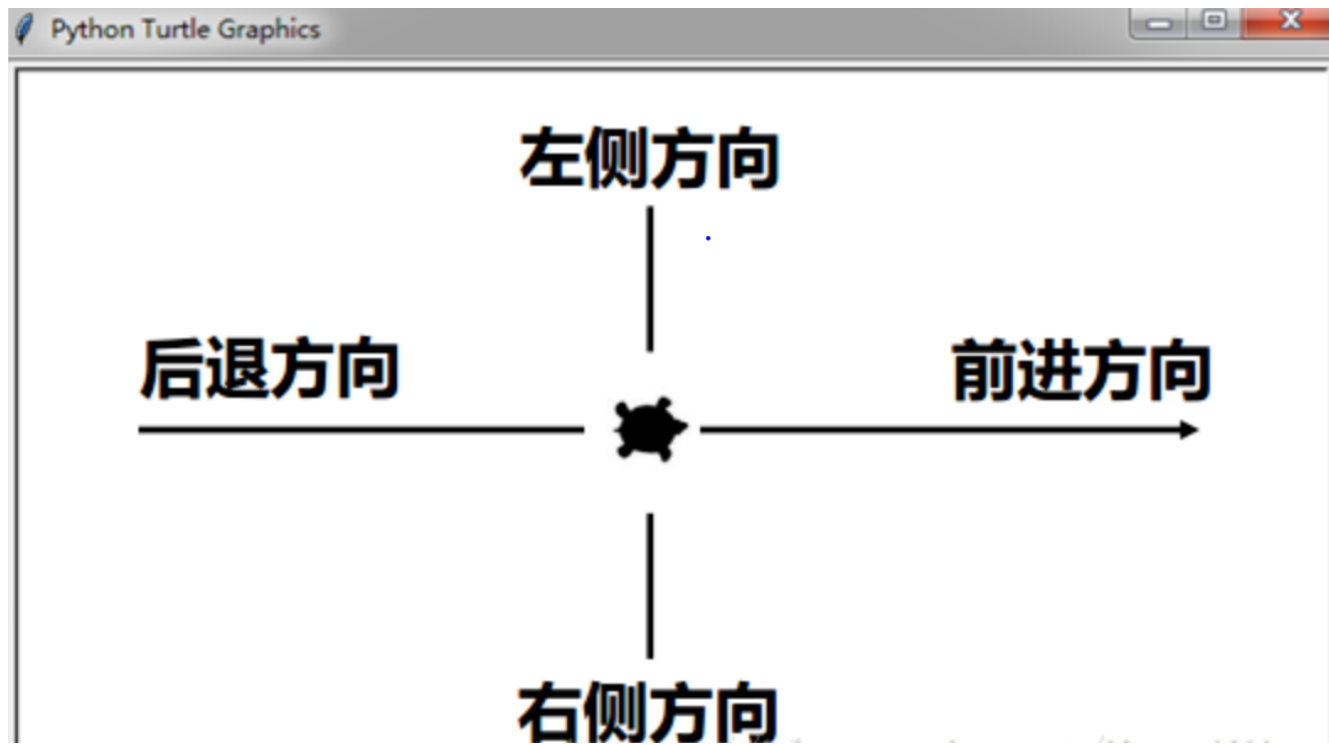
Turtle库

- Turtle库是Python语言中一个很流行的绘制图像的函数库

- 使用turtle库，头脑里需要有这样一个概念：

- 想象一个小乌龟，在一个横轴为x、纵轴为y的坐标系原点，(0,0)位置开始

- 它根据一组函数指令的控制，在这个平面坐标系中移动，从而在它爬行的路径上绘制了图形



程序运行

- **def** 用于定义函数，这段程序中，共出现两次 **def** 关键词，包含两个函数 **drawSnake** 和 **main**。
- **def** 所定义的函数在程序中未经调用不能直接执行，需要通过函数名调用才能够执行。
- 由于 **def** 定义的函数在程序中未经调用不会被执行，整个程序第一条执行的语句是 **main()**，它表示执行名字为 **main()** 的函数。

程序运行

- 从而，该程序跳转到 **main()** 函数定义的一组语句中执行，即开始执行 **turtle.setup()** 语句
- 同样的，**main()** 函数的最后一条语句调用了 **drawSnake()** 函数，当执行到这条语句时，程序跳转到 **drawSnake()** 函数中运行。

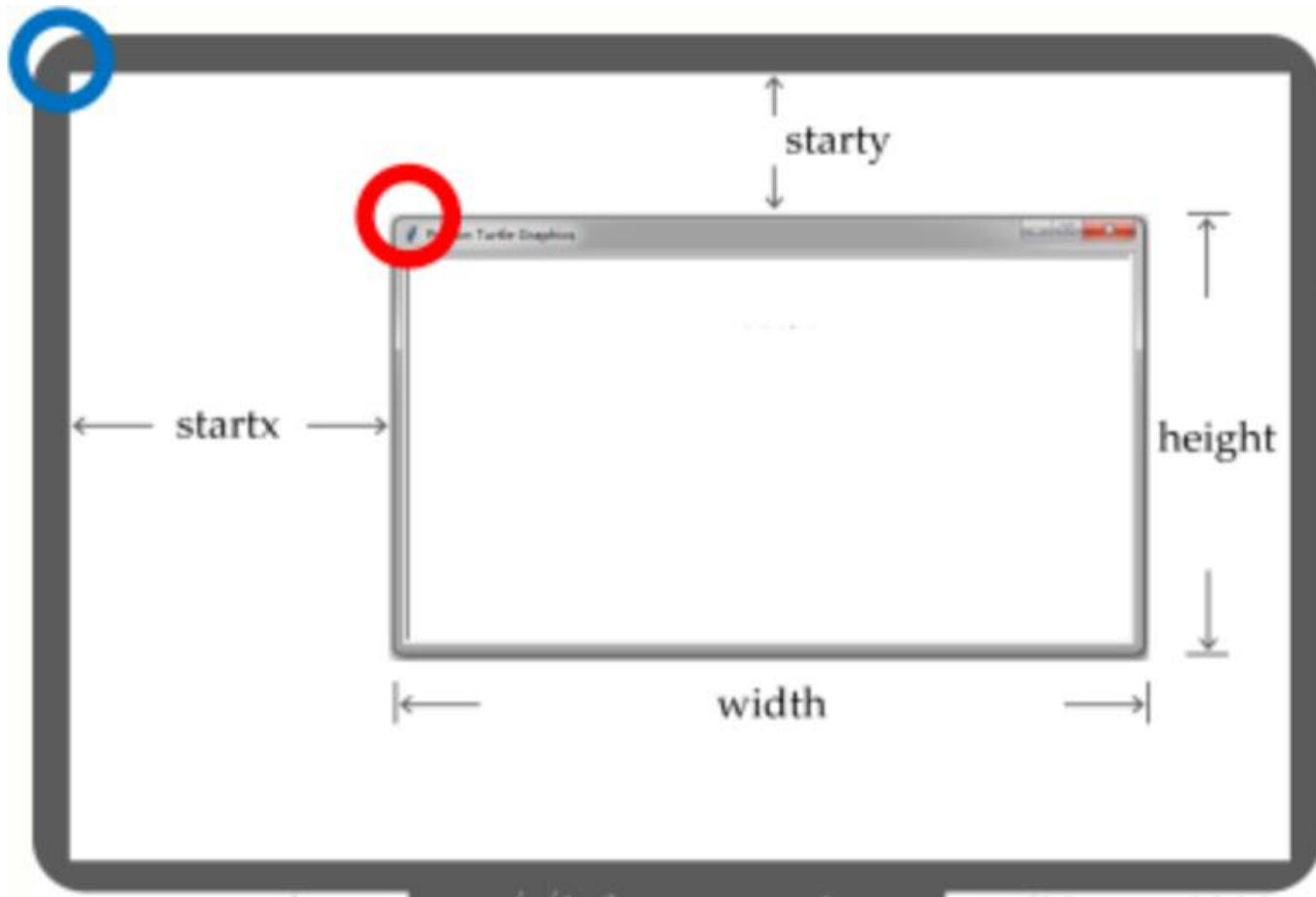
Python蟒蛇程序分析

Python小蛇实例

```
import turtle
def drawSnake(radius, angle, length, neckrad):
    for i in range(length):
        turtle.circle(radius, angle)
        turtle.circle(-radius, angle)
    turtle.circle(radius, angle/2)
    turtle.fd(radius)
    turtle.circle(neckrad+1, 180)
    turtle.fd(radius*2/3)
def main():
    turtle.setup(1300, 800, 0, 0)
    turtle.penup()
    turtle.fd(-250)
    turtle.pendown()
    pythonsize = 30
    turtle.pensize(pythonsize)
    turtle.pencolor("blue")
    turtle.seth(-40)
    drawSnake(40, 80, 4, pythonsize/2)
    turtle.done()
main()
```

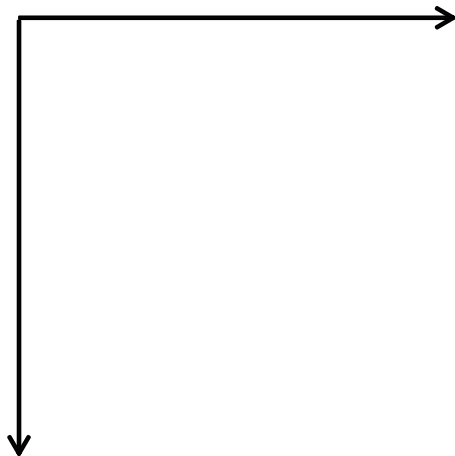
程序

- 程序运行`main()`函数中语句，遇到`setup`函数
- `Turtle`中的`turtle.setup()`函数用于启动一个图形窗口，它有四个参数
 - `turtle.setup(width, height, startx, starty)`
- 分别是：启动窗口的宽度和高度
- 表示窗口启动时，窗口左上角在屏幕中的坐标位置。



程序

- 我们所使用的显示屏幕也是一个坐标系，该坐标系以左上角为原点，向右和向下分别是x轴和y轴。
- 蟒蛇程序代码启动一个1300像素宽、800像素高的窗口，该窗口的左上角是屏幕的左上角。



程序

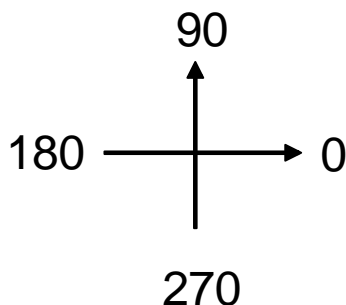
- Turtle中的`turtle.pensize()`函数表示小乌龟运动轨迹的宽度。
- 它包含一个输入参数，这里我们把它设为30像素，用`pythonsize`变量表示。

程序

- **Turtle**中的**`turtle.pencolor()`**函数表示小乌龟运动轨迹的颜色。
- 它包含一个输入参数，这里我们把它设为蓝色，**blue**，其他颜色单词也可以使用。**Turtle**采用**RGB**方式来定义颜色，如果希望获得和图片中颜色一致的小蛇，请输入
`turtle.pencolor("#3B9909")`

程序

- Turtle中的`turtle.seth(angle)`函数表示小乌龟启动时运动的方向。它包含一个输入参数，是角度值。
- 其中，0表示向东，90度向北，180度向西，270度向南；负值表示相反方向。
- 程序中，我们让小乌龟向-40度启动爬行，即：向东南方向40度。



standard mode	logo mode
0 - east	0 - north
90 - north	90 - east
180 - west	180 - south
270 - south	270 - west

程序

- **main()**函数给出了小乌龟爬行的窗体大小，爬行轨迹颜色和宽度以及初始爬行的方位。
- 最后，调用**drawSnake**函数启动绘制蟒蛇功能。
- **drawSnake**函数有四个参数，根据调用时给出的参数， 分别将40传递给rad、80给angle，5给len，15给neckrad

`turtle.circle()` 函数功能

- `turtle.circle()` 函数让小乌龟沿着一个圆形爬行
- 参数 `rad` 描述圆形轨迹半径的位置
 - 这个半径在小乌龟运行的左侧 `rad` 远位置处，如果
 - `rad` 为负值，则半径在小乌龟运行的右侧
- 参数 `angle` 表示小乌龟沿着圆形爬行的弧度值

`turtle.fd()`函数功能

- `turtle.fd()`函数也可以用`turtle.forward()`表示乌龟向前直线爬行移动
- 表示小乌龟向前直线爬行移动，它有一个参数表示爬行的距离

Python的函数封装

蟒蛇程序功能可以分成两类：

- 绘制图形前对画笔的设置，包括颜色、尺寸、初始位置等
- 以及绘制Python蟒蛇的功能。

由于蟒蛇绘制的功能相对独立，可以用函数来封装

用**turtle**画图

