第一章 Python 入门导学

1-1 导学

Python 不是一门新兴的语言，上世纪 90 年代初
TIOBE 语言排行 Python 是 第四名 https://www.tiobe.com/tiobe-index/
java C C++ Python C# VB.net PHP Javascript SQL R Ruby Go

Python ┌ Python2
       └ Python3

课程使用 Python3 从三个大方面教学
基础语法：基础变量 ----> 高阶函数
面向对象：讲思维
常见误区，Pythonic，总结经验，原生爬虫

了解语法是编程的先决条件，精通语法是编好程的必要条件

Pythonic ----> 很。Python 简介就是 Python 的特点，Life is simple, i use Python.

Python 能做什么? ----> 爬虫，大数据，测试，Web，AI， 脚本处理

1-2 Python 的特性

语言：Java Python Go…..
框架：以语言为基础构建的一系列基础功能集合

Python 的特性:

1. 简洁，优雅 life is short, I use Python.
2. 跨平台 Windows Linux MacOS
3. 易于学习
4. 极为强大的而丰富的标准库与第三方库
5. Python 是面向对象的语言

1-3 我为什么喜欢 Python

Python 之禅:(之中的 2 句话)

Simple is better than complex.
Now is better than never.
Although never is often better than *right* now.

在 Python 环境中查看 Python 之禅

```
>>>import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

1-4 Python 的缺点

慢 ----> (对比与 C,C++,Java)运行效率慢

计算机语言：分两类

- 编译型语言: C, C++
- 解释型语言: Javascript, Python

运行效率 与 开发效率

- 运行效率：开发完代码后 运行代码的速度 (性能)
  - 虽然快 但是 如果代码写的不好也有可能会运行的很慢
- 开发效率：开发代码时 写代码的速度
  - Python 开发效率快

1-5 一个经典误区

编程  Web 编程

可以学习一点 Web 编程 Web 是基础

1-6 Python 能做些什么

(几乎是万能的) 哈?

1. 爬虫
2. 大数据与数据分析(Spark 框架)
3. 自动化运维与自动化测试
4. Web 开发:Flask 框架, Django 框架
5. 机器学习:Tensor Flow
6. 胶水语言:混合其他语言来编程，如: C++，Java
    a. 能够把用其他语言制作的各种模块(尤其是 C/C++)很轻松地联合在一起

1-7 课程内容与特点

- 基础语法　一定要扎实
- Pythonic　很-Python

    代码的复杂程度对比 ---- A B 两个变量交换 值

    Java 实现

    int temp = a;

    a = b;

    b = temp;

    三行代码才实现

    Python 实现:

    a,b = b,a　　一行代码就实现了

- Python 高性能与优化

    写代码不要只满足功能 要选择一种最佳的代码方案 追求高性能

- 数据结构

    也是非常基础的 不同的编程语言都是通用的

1-8 Python 的前景

嗯 就是很有前景 就对了

1-9 课程维护与提问

先 Debug　　微信公众号: 林间有风

QQ 群: 299631895

验证信息:1801281026544986

第二章 Python 环境安装

2-1 下载 Python 环境安装

Python 官网 https://www.python.org/
去 Download 下载 Python2 & Python3

2-2 安装 Python

先安装 Python 3 新建一个文件夹
运行安装文件 勾选 Add Path 3.x to PATH

在安装 Python 2 新建一个文件夹
计算机 右键 properties, advanced system settings
Environment variables, System variables
Path 加一条 Python 2 的安装路径

Rename Python3 文件夹中的 Python.exe 成 Python3.exe

然后在 Shell 中 输入 python 进入 Python 2
输入 python3 进入 Python 3

2-3 IDLE 与第一段 Python 代码

IDLE 分为 Python 2 和 Python 3
用 search bar 找到不同的 IDLE 可以在需要时使用

Python 2 print “hello world”
Python 3 print(“hello world”)

第三章 理解什么是写代码与 Python 的基本类型

3-1 什么是代码, 什么是写代码

代码：是实现世界事务在计算机世界中的映射
写代码：是将现实世界中的事物用计算机语言来描述

3-2 数字:整形与浮点型

Number 数字{int 整数型，float 浮点型，complex 复数，bool 布尔值(True 或 False)}
Python 2 中还有一个类型为 Long 任何精度的整数 只在 Python2 中

int 整数型 正 负都行
float 浮点数 就是带小数点的数
(其他语言中有分 单精度 float 和 双精度 double)
(在 Python3 中没有单双精度的区分！)
(其他语言 在整数型中 有分 short int long)
(在 Python3 中都没有 在 Python 2 中有 long)

type() 函数
type() 函数 如果只有一个参数返回对象的类型

```
type(1)     <class 'int'>
type(1+1.0) <class 'float'>   1+1.0 = 2.0 ----> float
type(1*1.0) <class 'float'>
type(1.1)   <class 'float'>
type(1+1)   <class 'int'>
type(1*1)   <class 'int'>

type(2/2)   <class 'float'>
   2/2 ----> 1.0 ----> float
   2//2 ----> 1 ----> int     // 整除 只留整数部分
```

int / int 如果能整除 结果就是 int 不能整除 结果就是 float
int // int 就算不能整除 也只保留整数部分

3-3 十进制 二进制 八进制 十六进制

十进制 0 1 2 3 4 5 6 7 8 9 10
以 0 开始 满 10 个数字进 1

二进制 0 1 10 11 100 101 110 111 1000
以 0 开始 满 2 个数字进 1

八进制 0 1 2 3 4 5 6 7 10 11 12 13 14
以 0 开始 满 8 个数字进 1

十六进制 0 1 2 3 4 5 6 7 8 9 A B C D E F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20
以 0 开始 满 16 个数字进 1 9 以后的数字以 A B C D E F 来表示

十进制 二进制 八进制 十六进制 中的 10 都代表不容的意思 虽然都是 10 但是意义不同 不是同一个真实的数字

还有其他的进制 时间 60 秒 ----> 1 分钟
60 分钟 ----> 1 小时

3-4 各进制的表示与转换

Python 中表示二进制:

在数字前加上 0b (零 b) 2 进制 --> 10 进制

```
>>> 0b10
2
>>> 0b11
3
```

IDLE 中 用 0b10 自动将 2 进制转到 10 进制 直接回车

Python 中表示八进制:

在数字前加上 0o (零 o) 8 进制 --> 10 进制

```
>>> 0o10
8
>>> 0o13
11
```

IDLE 中 直接会将 8 进制转到 10 进制

Python 中默认为 10 进制 直接输入就好

Python 中表示十六进制:

在数字前加上 0x (零 x)

```
>>> 0x10
16
>>> 0x1F
31
```

IDLE 中直接会将 16 进制 转为 10 进制

各进制转换

1. 其他进制的数 转成 2 进制 bin()

   bin(50) ----> '0b110010' 10 进制 ----> 2 进制

   bin(0o7) ----> '0b111' 8 进制 ----> 2 进制

   bin(0xE) ----> '0b1110' 16 进制 ----> 2 进制

2. 其他进制的数 转成 10 进制 int()

   int(0b111) ----> 7 2 进制 ----> 10 进制

   int(0o777) ----> 511 8 进制 ----> 10 进制

   int(0x1F) ----> 31 16 进制 ----> 10 进制

3. 其他进制的数 转成 8 进制 oct()
   oct(0b111) ----> '0o7'   2 进制 ----> 8 进制
   oct(0x777) ----> '0o3567' 16 进制 ----> 8 进制
   oct(20)   ----> '0o24'   10 进制 ----> 8 进制

4. 其他进制的数 转乘 16 进制 hex()
   hex(888)   ----> '0x378' 10 进制 ----> 16 进制
   hex(0o777) ----> '0x1ff' 8 进制 ----> 16 进制
   hex(0b111) ----> '0x7'   2 进制 ----> 16 进制

3-5 数字: 布尔类型 与 复数

Number 除 int 和 float 外 还有两种数据类型 bool 和 complex

bool 布尔类型 表示真假
complex 表示复数

bool: True 表示为真 T 为大写
    False 表示为假 F 为大写

bool 是 Number 的一个类型 为什么呢?

```
>>> int(True)
1
>>> int(False)
0
```

```
>>> bool(1)
True
>>> bool(0)
False
```

```
>>> bool(2)
True
>>> bool(2.2)
True
```

```
>>> bool(-1.1)
True
>>> bool(0b10)
True
```

bool 只有是非 0 就是 True 如果是 0 或者 None 就是 False
Python 中任何空值 都是 False 非空值 都为 True

```
>>> bool(0)
False
>>> bool(None)
False
```

None 类型
None 类型 表示一个 Null 对象(没有值得对象)。 Python 提供了一个 Null 对象，在程序中表示为 None。
如果一个函数没有显式的返回值，则会返回该对象。None 经常用作可选参数的默认值，以便让函数检测调用者是否为该参数实际传递了值。
None 没有任何属性，在布尔表达式中求值时为 False。

不只是整数类型可以与 bool 互换，字符串也可以。

```
>>> bool('abc')
True
>>> bool(' ')
True
>>> bool('')
False
>>> bool()
False
```

```
>>> bool([1,2,3])
True
>>> bool([])
False
>>> bool({})
False
>>> bool({1,2,3})
True
```

复数 ---- complex
用 j 表示复数 一般不会经常用 以后可以研究一下

3-6 安装 Python

字符串 ---> 非常常用的基本类型
字符串 str 表示文字类型的数据

str{单引号 ''   双引号" " 三引号 }

```
>>> 'Helloworld'
'Helloworld'
>>> "helloworld"
'helloworld'
```

这两个都是可以的 但是区别就是在于
单引号无法表示 字符串中的引号

```
>>> 'let''s'
'lets'
>>> 'let''s go'
'lets go'
```

如果想使用单引号表示 字符串中的引号 则需要用到转义字符 \

```
>>> 'let\'s go'
"let's go"
```

但是双引号可以表示

```
>>> "let's go"
"let's go"
```

无论任何符号 在 Python 中都必须是英文符号

1 与 '1' 不同 因为 1 是 int , 而 '1' 是 str

3-7 多行字符串

三引号 超长的字符串 不适宜阅读
所以 Python 中规定(建议) 每行的宽度为 79 个字符

三引号 可以使三个单引号 或者 三个双引号 要成对出现 可以回车
但是 IDLE 中会在你每一次回车的时候 加上 \n

还是使用三个双引号吧 虽然在 IDLE 中可以使用三个单引号 但是三个单引号代表多行注释

```
>>> """ helloworld
        helloworld
        helloworld"""
' helloworld\nhelloworld\nhelloworld'
```

为什么在 IDLE 中会有\n 出现?
因为在 Python 中 >>> 表示要接收一个输入 无论是可见的 abcd 或者 是不可见的 tab enter 等
IDLE 要把所有的输入都要显示出来，所以会用转义字符 来表示无法显示的字符

```
但是 >>> """helloworld\nhelloworld\nhelloworld"""
        'helloworld\nhelloworld\nhelloworld'
      如果这样收入 同时也会输出一样的字符

      print() 可以吧\n 等转义字符解析并实现
      >>> print("""hello world\nhelloworld\nhelloworld\n""")
      hello world
      hello world
      hello world
```

IDLE 单引号换行

```
>>> 'hello\
    world'
'helloworld'
>>> "hello\
    world"
'helloworld'
```

3-8 转义字符

转义字符 ----> 特殊的字符{表示无法"看到"的字符 换行 等，与语言本身语法有冲突的字符}

转义字符
\n 换行
\` 单引号
\t 横向制表符
etc 还有很多 可以自行百度

\n \r 容易搞混 不是同一个概念 可以找一下 百度 google
\n 换行 \r 回车

print 输出 hello \n world

print('hello \\n world') 这与 let\'s go 同理
这个可以再输出文件夹路径的时候用到
>>> print('C:\\fatherfoldername\\childfoldername')
C:\fatherfoldername\childfoldername

但是要是有太多的 \\ 这样做也是真的很麻烦

所以在路径前面加一个 r 表示 r 后面的字符串不是一个普通字符串 而是一个原始字符串
r 的大小写不影响
这个原始字符串 所见即所得
print(r'C:\fatherfoldername\childfoldername')

注意 这种情况不是普遍适用的
如果字符串里面有单引号就不能用
print(r'let's go') 这种情况就会报错 因为出现了三个单引号
所以换成双引号就没问题了啊
print(r"let's go")

3-9 included_in_3-8 原始字符串

3-10_3-12 字符串运算 一、二、三

合并两个字符串 & 截取字符串

合并 拼接 用 + (加号) * (乘号)

"Hello" + "world" ========> "Helloworld"
"hello" * 3 ========> "hellohellohello"
"hello" * "world" ========> 报错 字符不可以与字符相乘
字符串只能与数字相乘

截取 获取字符串 中的单个字符
"Helloworld"[0] ==> H
"Helloworld"[3] ==> l

只要输入字符的字号就能得到该字符 字号从 0 开始

"H e l l o w o r l d"
"0 1 2 3 4 5 6 7 8 9"

如果中间带空格 空格也算一位
"H e l l o w o r l d"
0 1 2 3 4 5 6 7 8 9 10 这是字号
-11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 这是 n 次的字符

"hello world"[-3] ====> r
字符串后面的方括号[]中 如果是正数 代表字号 从 0 开始
字符串后面的方括号[]中 如果是负数 从该字符串最后向前数 多少个位的字符
-n 就是从后往前数 第 n 个字符

如果要得到 w 在 "hello world" 中
"Hello world"[6] ====> w
"Hello world"[-5] ====> w

截取 获取字符串 中的一组字符 要从截取字符串的起点 并且往后截取几个字符
"hello world"[0:5] ====> hello
"hello world"[0:-1] ====> hello worl 我们发现 d 消失了
这里面的-1 代表了 "步长" 往回数 1 个字符
"hello world"[0:-3] ====> hello wo

Question: 截取 world 的两种不同的方法
"Hello world"[6:11] ====> world 但是奇怪的是 这个字符串没有第 11 个字符 所以无论你写什么数字 只要大于 10 就没问题
"Hello world"[6:] ====> world 可以使用这个样子 系统就明白从第六个字符开始截取 一直截取到字符串结束

"hello python java c# javascript php ruby"
截取 编程语言 去掉 hello
"hello python java c# javascript php ruby"[6:] ====> python java c# javascript php ruby
这样就去掉了 hello

[6:] 从第 6 位向右截取到字符串结束
[:4] 从字符串 0 位向后截取 4 个字符
[:-4] 从字符串 0 位向前截取 4 个字符 字符串第一个字符永远是 0 号字符
[-4:] 从字符串 -4 位向后截取到字符串结尾。 ====> ruby

第四章 Python 中表示 '组' 的概念与定义

4-1 列表的定义

序列 组的概念 就是一个或多个元素组成在一起

列表 也是组的一种 列表 [] [list]

python 中定义列表 [1,2,3,4,5] 这个列表有 5 个元素
完成定义后可以使用 type() 来检测一下
type([1,2,3,4,5]) ======> <class 'list'>

列表中的元素 可以为任何类型的数据 比如 string boolean 或者 不同的数据类型混合在一起也可以。

["Hello","world",1,9,True,False] 多种数据类型混合在一个列表中 作为该列表的不同元素

在列表中 也可以用列表作为元素 这就是 二维列表 或 多维列表
[[1,2],[3,4],[5,6],[7,8],[True,False]]

type([[1,2],[3,4],[5,6],[7,8],[True,False]]) =========> <class 'list'>

二维数组 和 多维数组 在 Python 中叫做嵌套列表。

4-2 列表的基本操作

访问列表中的某一个或多个元素 以 0 位开始

[1,2,3,4,5,6][0] ========> 1 <class 'int'>
[1,2,3,4,5,6][0:2] ========> [1,2] [0:2] 从 0 位开始打印 2 个元素
[1,2,3,4,5,6][-1:] ========> [6]

这跟字符串中的字号相似

[1, 2, 3, 4, 5, 6] 列表
0 1 2 3 4 5 列表的元素序列号
-6 -5 -4 -3 -2 -1 同样也是元素的序列号

列表用 : 英文感叹号访问 多个元素会返回一个列表

更改列表内的元素的值
==========

1. 列表相加 会合成为一个列表
[1,2,3,4,5] + [6,7] =======> [1,2,3,4,5,6,7]

2. 两个列表相乘 没有这种操作 会报错 列表不可以乘列表
列表可以乘数字
[1,2,3,4,5] * 3 =======> [1,2,3,4,5,1,2,3,4,5,1,2,3,4,5]
[列表]*n 会合成为一个列表 并且把该列表重复 n 次 也就是列表中的所有元素乘 n

3. 列表没有减法 没有这种操作 会报错

用嵌套列表的方式来表示世界杯小组赛
A1 B1 C1 D1 E1 F1 G1 H1
A2 B2 C2 D2 E2 F2 G2 H2
A3 B3 C3 D3 E3 F3 G3 H3
A4 B4 C4 D4 E4 F4 G4 H4

[[A1,A2,A3,A4],[B1,B2,B3,B4],[C1,C2,C3,C4],[D1,D2,D3,D4],[E1,E2,E3,E4],[F1,F2,F3,F4],[G1,G2,G3,G4],[H1,H2,H3,H4]]

4-3 元组 tuple

元组用小括号表示 用逗号将元素隔开 元组中的元素可以为不同类型 (元素,元素,元素,元素,元素)

访问元组 元组相加 元祖乘数字 与列表相同

(0,1,2,3,4)[0] ====> 0
(0,1,2,3,4)[0:2] ====> (0, 1)
(0,1,2) + (3,4,5) ====> (0,1,2,3,4,5)
(0,1,2) * 3 ====> (0, 1, 2, 0, 1, 2, 0, 1, 2)

如果元组中只有一个元素 则不会认为是元组类型 而是会认定为元素的类型
type((1)) ====> <class 'int'>

==========
因为在 python 中 () 也可以表示一种数学的运算 如 (1+1)*2
()就叫做运算优先级括号 这就会和元组产生冲突
当有歧义的时候 Python 中规定 当有一个括号并且其中只有一个元素 就当做运算符号来进行计算
==========
但是这在 list 中就不会有这样的顾虑
==========
但如何定义只有一个元素的元组呢？ (1,) 加一个都好加装后面有一个元素
如何定义一个元素都没有的元组呢？ ( ) type(( )) ====> <class 'tuple'>

4-4 序列总结　　序列是有序的

基本类型总结

数字基本类型 (int　float　bool)

str　list　tuple　这三种基本数据类型非常相似 都是序列 str 是由单个字母组成的一串字母 所以叫做字符串

==========

序列有哪些共同特点?

1. 可以在序列后面加 [ ] 进行访问

序列内的每一个元素都会被分配一个序号 通过这个需要对学历进行访问

"String"[2] ====> r

2. 切片　就是序列都能切片

[1,2,3,4,5][0:3]　====>　[1,2,3]

[1,2,3,4,5][-1:]　====>　[5]

'helloworld'[0:8:2]　====>　hloo

3. 序列可以进行 加法 和 乘法

4. 序列中是否存在某个元素　在这里引用一个新的符号　in　not in　会返回一个 bool 类型

某元素是否在序列中　　3 是否在序列[1,2,3] 中

3 in [1,2,3]　====>　True

某元素是否不在这个序列中

3 not in [1,2,3]　====>　False

in / not in　是逻辑运算符

5. 一个序列中一共有多少个元素　引入一个新的 function　len()

len([1,2,3,4,5,6])　====>　6

len("Hello World")　====>　11

6. 序列中找出最大的元素

max([1,2,3,4,5,6])　====>　6

7. 序列中找出最小的元素

min([1,2,3,4,5,6]) =====> 1

max 和 min 在 str 中会按照在字母表中的顺序输出 空格为最小

ASCII 码 将数字与字母相连接 可以以数字的形式代表字母 可以得出字符串中那个字母的大小。

字母的大小写在 ASCII 中都会有不同的数字表示。

max("hello world") =====> "w"

min("hello world") =====> " "

当你要查看一个数据在 ASCII 中如何表示的时候 使用 ord() function

ord()只接收一个参数 把这个参数转换成 ASCII 码

ord("w") =====> 119

ord("d") =====> 100

ord(" ") =====> 32

如果你想要吧 ASCII 转换成字母的时候使用 chr() 功能就可以了

chr(97) =====> 'a'

4-5 set 集合

除了 元组 tuple () 列表 List[] 以外 另一种基本类型为 集合 set

集合 set 的特性就是 无序

type({1,2,3,4,5,6}) =====> <class 'set'>

集合 set 没有下标索引 所以不支持切片 也不支持用下表索引访问

集合 set 内的元素不允许重复
  {1,1,2,2,3,3,4,4,5,5} =====> {1,2,3,4,5}

集合支持的操作:

  1. 长度判断 len({1,2,3,4}) =====> 4

  2. 是否有其元素 1 in {1,2,3} =====> True
    是否没有其元素 1 not in {1,2,3} =====> False

集合的特殊操作:
  除掉元素
    集合 1 {1,2,3,4,5,6} 集合 2 {3,4,7}

    {1,2,3,4,5,6} - {3,4,7} =====> {1,2,5,6} {1,2,5,6} 叫做差集 这就是两个集合的差集合

  1. 除掉两个集合共有的元素
    集合 1 - 集合 2 大集合 - 小集合

  2. 保留两个集合共有的元素
    集合 1 & 集合 2 {1,2,3,4,5,6} & {3,4,7} =====> {3,4} 这是交集

  3. 合并集合
    {1,2,3,4,5,6} | {3,4,7} =====> {1,2,3,4,5,6,7}
    最后结尾的这个集合 叫做 合集 或 并集

如何定义一个空的集合?
  set()

  验证是否为空 len(set()) = 0

4-6 安装 Python

dict 字典是 Python 的基本数据类型
dict 一般会有两个值 key value
key 就是 关键字 value 就是 相应的数据得值

dict 可以有多个 key 和 多个 value

定义一个 dict:
{key 1: value 1,key 2: value 2,key 3: value 3, ……}

{1:1, 2:2, 3:3}
{'Q':'kill','R':'kiss','w':'kim'}

字典是无序的 所以字典无法使用下标进行访问 所以 dict 通过 key 来访问
{'Q':'kill','R':'kiss','w':'kim'}{'Q'} ====> kill

dict 字典不可以有两个相同的 key
{1:'kill','1':'kiss','E':'kim','R':'business'}
这里面的 1 与 '1' 是不同的 虽然都是 1 但是第一个是数字 int 第二个是字符 string

字典 dict 中 valur 所能选取的 Python 数据类型总结:
Value: 任意 Python 基本数据类型
str int float list set dict

字典中的 value 可以使另外一个字典
type({1:'kill','1':'kiss','E':{1:1},'R':'business'}) ====> dict

key: 必须是不可变的类型 eg: int str
列表不可以 列表是可变的 list 元组也是不可以的 元组也是可变的 tuple

如何定义一个空的字典呢?
一个空的字典 表示就是一个空的大括号 {}
type({}) ====> dict

4-7 思维套图总结 基本数据类型