

Python编程基础

王启睿

内容

- 为什么介绍Python
- Python是什么
- 如何运行Python
- Python基本语法
 - 数据类型
 - 控制结构

内容

- 为什么介绍Python
- Python是什么
- 运行Python
- Python基本语法

为什么介绍Python - TIOBE Index

Jun 2016	Jun 2015	Change	Programming Language	Ratings	Change
1	1		Java	20.794%	+2.97%
2	2		C	12.376%	-4.41%
3	3		C++	6.199%	-1.56%
4	6	⬆	Python	3.900%	-0.10%
5	4	⬇	C#	3.786%	-1.27%
6	8	⬆	PHP	3.227%	+0.36%
7	9	⬆	JavaScript	2.583%	+0.29%
8	12	⬆	Perl	2.395%	+0.64%
9	7	⬇	Visual Basic .NET	2.353%	-0.82%
10	16	⬆	Ruby	2.336%	+0.98%
11	11		Visual Basic	2.254%	+0.41%
12	23	⬆	Assembly language	2.119%	+1.36%
13	10	⬇	Delphi/Object Pascal	1.939%	+0.07%
14	14		Swift	1.831%	+0.39%

为什么介绍Python – Python优点

- 简单易学
- 代码简洁
- 拥有丰富的库

为什么介绍Python – 队式17中的Python

- 逻辑
- 平台
- 初/决赛对战脚本
- 网络对战后端

内容

- 为什么介绍Python
- Python是什么
- 运行Python
- Python基本语法

Python是什么



- 大蟒蛇
- Since 1989

Python是什么

- 面向对象
- 解释型
- 强类型
- 动态类型

内容

- 为什么介绍Python
- Python是什么
- 运行Python
- Python基本语法

运行Python – 交互界面

- 打开终端
- 输入python（或者是python3），按下回车即可
- 输入exit()，按下回车即可退出

运行Python – 运行文件

- 在终端里输入python，加上一个空格，再加上要被执行的文件的路径即可执行

内容

- 为什么介绍Python
- Python是什么
- 运行Python
- Python基本语法

Python基本语法 – 语法特点

- 缩进敏感：4空格缩进，不需要大括号
 - 大小写敏感
 - 语句结束不需要分号 # 也可以有
 - 多行语句在行末加反斜杠（\） # 不引起歧义也可不加
 - 使用变量前不需要声明，直接对它赋值即可
- # 单行注释用#号开头

Python基本语法 – 数据类型

- 基本类型：数、字符串
- 集合类型：列表（list）、字典（dict）、
元组（tuple）、集合（set）

Python基本语法 – 数

- 整数 (int)

- -12345

- 12345678909876543212345678987654321

-

Python的整型不会溢出

- 0xffe1dead

十六进制

- 0o1234567

八进制

- 0b10101100

二进制

Python基本语法 – 数

- 浮点数 (float)

- -12345.

- 1.2345e-8

科学记数法

- 精度有限，范围有限

Python基本语法 – 数

- 复数 (complex)
 - $1j$
 - $12.3 + 45.6j$
- 实部和虚部是浮点数

Python基本语法 – 数

- 布尔型 (bool)

- True # 1

- False # 0

Python基本语法 – 数

● 数的操作

- `1+2*3` # 7 运算符有优先级
- `3**2` # 9 **表示幂
- `5/3` # 1.66666666667 整数除法一定会变成小数
- `5//3` # 整数除法，和c中类似，也可以用在浮点数
- `0x12345678 & 0xf0f0f0f0` # 按位逻辑运算

Python基本语法 – 优先级表

低

Operator

[lambda](#)

[if – else](#)

[or](#)

[and](#)

[not](#) x

[in](#), [not in](#), [is](#), [is not](#), <, <=, >, >=, !=, ==

|

^

&

<<, >>

+, -

*, @, /, //, %

+x, -x, ~x

**

await x

x[index], x[index:index], x(arguments...),
x.attribute

(expressions...), [expressions...], {key: value...},
{expressions...}

Description

Lambda expression

Conditional expression

Boolean OR

Boolean AND

Boolean NOT

Comparisons, including membership tests and
identity tests

Bitwise OR

Bitwise XOR

Bitwise AND

Shifts

Addition and subtraction

Multiplication, matrix multiplication division,
remainder [\[5\]](#)

Positive, negative, bitwise NOT

Exponentiation [\[6\]](#)

Await expression

Subscription, slicing, call, attribute reference

Binding or tuple display, list display, dictionary
display, set display

高

Python基本语法 – 运算符

- 逻辑运算符 `and` `or` （都具有短路性） `not`
- `a or b` 相当于 `a or b` `a and b` 相当于 `a and b`
- `if a:` `if not a:`
- `return a` `return a`
- `else:` `else:`
- `return b` `return b`
- 三目运算符 `b if a else c` （相当于 `a?b:c`）
- `a=a+1` 可以写成 `a+=1` 不能写成 `a++` 也不能写成 `++a`

Python基本语法 – 字符串

- 字符串 (string)

- 'Hello, world' # 字符串可以用单引号括起

- "Hello, world" # 也可以用双引号

- '''Hello,

- World

- '''

- # 用三个引号括起来的字符串可以换行

- # 也可以写成'Hello,\nWorld\n'

Python基本语法 – 字符串操作

- 连接两个字符串

- `'Hello, ' + 'world' # 'Hello, world'`

- 字符串乘以整数

- `'Hello' * 3 # 'HelloHelloHello'`

Python基本语法 – 字符串操作

- 字符串的长度

- `len('Hello, world')` # 12

- 查找子字符串

- `'or' in 'Hello, world'` # True

- `'Hello, world'.count('o')` # 2

- `'Hello, world'.find('or')` # 8

Python基本语法 – 字符串操作

- 判断前后缀

- `'Hello, world'.startswith('Hell')` `# True`

- `'Hello, world'.endswith('old')` `# False`

- 替换

- `'Hello, world'.replace('o','^_^')`

`# 'Hell^_^, w^_^rld'`

Python基本语法 – 字符串的切分

```
a = ' H e l l o , _ w o r l d '
```

0 1 2 3 4 5 6 7 8 9 10 11

- `a[1]` # `'e'`
- `a[2:5]` # `'llo'`
- `a[:6]` # `'Hello,'`

Python基本语法 – 字符串的切分

```
a = ' H e l l o , _ w o r l d '
```

	0	1	2	3	4	5	6	7	8	9	10	11
	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- `a[-1]` # `'d'`
- `a[-3:]` # `'rld'`
- `a[3:-3]` # `'lo, wo'`

Python基本语法 – 格式化字符串

- `'My name is %s.' % 'wangqr'`
- `'Exapmle # %d: ' % 2`
- `'%d %d %d %s' % (1,2,3,'a')` # 涉及到将要讲的tuple
- `'%(height)f %(age)d' % {'age': 15, 'height': 143.0}`
涉及到将要讲的dict

注1：自动补全在cmd中可能无法正常工作，IDLE带有自动补全功能但稍有不同

注2：自动补全的机制将在接下来的Linux课程中详细介绍

* Python技巧 – 自动补全

- `a='Hello, world'`
- 输入 `a.end` 按下Tab键，会自动补全到 `a.endswith(`
- 输入 `a.isa` 按下Tab键，会自动补全到 `a.isal`
- 再次按下Tab键，会列出所有的备选项
 - 第一次按Tab键会补全到尽可能长的位置
 - 第二次按Tab键会列出所有可能的选项
 - 输入`a.`后按下两次Tab键可以看到a的所有成员

注：在*nix系统（Mac OS X、Linux）上，help页的操作方法与less、man等程序一致，将在接下来的Linux课程中详细介绍

* Python技巧 – 一些内建函数

- `a='Hello, world'`

- `type(a)` `# <class 'str'>`

- `type(1)` `# <class 'int'>`

- `help(str)` `# 如果卡在帮助里了，按q退出`

Python基本语法 – 列表

- `[]`
- `[1]`
- `[1, '2']`
- `[x for x in [1,2,3,4] if x % 2 == 0]`
- `[2*x for x in [1,2,3,4]]`

注：可以与集合的写法对照看： $\{x|x \in S, x \text{为偶数}\}$ ，
但需要注意列表中可以有重复元素

Python基本语法 – 列表生成

- `[<expr> for <var> in <iterable>]`
- `[<expr> for <var> in <iterable> if <expr>]`
- `[x for x in [1,2,3,4] if x % 2 == 0]`
- `[2*x for x in [1,2,3,4]]`
- `[x+y for x in [0,1] for y in [10,100]]`

Python基本语法 – 列表操作

- 与字符串类似，可以：
- `[1,2][0]` # 1 （按下标访问）
- `[1,2] + ['a','b']` # `[1, 2, 'a', 'b']`
- `len(['apple','banana'])` # 2
- `['a','b','c','d'][1:3]` # `['b', 'c']`

注1：字符串不能进行该页中列出的所有操作，但字符串可以用+=号（为什么？）

注2：当一个值在列表中出现多次时，remove只删除第一个

Python基本语法 – 列表操作

● 元素操作	# 假设我们有 a = [1,2,3,4,5,6,7,8,9]
a[3] = 0	# 修改元素 a=[1,2,3,0,5,6,7,8,9]
a.append(10)	# 追加 a=[1,2,3,0,5,6,7,8,9,10]
a.insert(7, 7.5)	# 按下标插入 插入的元素将位于a[7]
del a[:5]	# 按下标删除 a=[6,7,7.5,8,9,10]
a.extend([13,14,15])	# 相当于 a+= [13,14,15]
a.remove(9)	# 删除元素 a=[6,7,7.5,8,10,13,14,15]
a.clear()	# 删除全部元素 a=[]

注：当一个值在列表中出现多次时，index只返回第一个的下标

Python基本语法 – 列表操作

● 元素查找 # a=[6,7,7.5,8,10,13,14,15]

3 in a # False

13 in a # True 判断元素是否在列表中

a.index(13) # 5 取得元素下标

a.count(14) # 1 统计元素出现次数

Python基本语法 – 列表操作

- 元素排序

```
a+=[9.5, 14.3, 12.8]; a.sort();
```

```
# a=[6, 7, 7.5, 8, 9.5, 10, 12.8, 13, 14, 14.3, 15]
```

- 元素逆序

```
a.reverse()
```

```
# a=[15, 14.3, 14, 13, 12.8, 10, 9.5, 8, 7.5, 7, 6]
```

列表与字典

list

0	'Alice'
1	'Bob'
2	'Carol'
3	'Dan'
4	'Eve'
5	'Frank'
6	'Grace'

下标与数据对应

dict

'hair'	'blue'
'eye'	'green'
'height'	143.0
'weight'	32.0
'age'	15
'birthday'	'5/28'
'blood type'	'A'

键与值对应

Python基本语法 – 字典

- `{}`
- `{'a': 'apple', 'b': 'banana'}`

Python基本语法 – 字典操作

● 元素操作 # 假设我们有a={'a':'apple','b':'banana'}

a['a'] # 'apple' 访问元素

a['c'] = 'cake' # 添加元素 a={ ... , 'c': 'cake' }

a['b'] = 'baby' # 修改元素

del a['b'] # 删除元素

a.clear() # 删除所有元素

注：准确的说，元素在字典中是按照键的哈希值排列的，但是我们不了解Python内部哈希函数的具体实现，从而无法确定顺序

Python基本语法 – 字典操作

- 元素查找 # 假设我们有a={'a':'apple','b':'banana'}

'b' in a # True 实际上查找的是键

- 获得所有键 a.keys()
- 获得所有值 a.values()
- 获得所有元素 a.items()
- 元素在字典中是不保证顺序的

Python基本语法 – 元组

- `()`

- `(1,)`

单元素元组的表示方法

- `(1, 2, 3)`

- 与列表类似，但是不可以修改元素

Python基本语法 – 集合

- `set()` # 空集合的表示方法
- `{1, 2, 3}`
- `{x for x in range(50)}` # 参考 列表生成

Python基本语法 – 集合操作

● 元素操作 # 假设我们有a={'a','b','c'}

a.add(123) # 添加元素 a={'a','b','c',123}

a.remove('c') # 删除元素 a={'a','b',123}

a.union({123, 456}) # 集合并a={'a','b',123,456}

a.clear() # 删除所有元素

Python基本语法 – 集合操作

● 元素查找 # 假设我们有 `a={'a','b',123,456}`

`'b' in a` # True

`'rabbit' in a` # False

Python基本语法 - 类型转换

- `<type>(<expr>)`

- `str(1)` `# '1'`

- `int('2')` `# 2`

- `bool(0)` `# False`

- `list({1,2,3})` `# [1,2,3]` 顺序可能不同

Python基本语法 – 控制结构

- `if <expr>:`
- `lines_of_code`
- `elif <expr>:`
- `lines_of_code`
- `else:`
- `lines_of_code`

Python基本语法 – 布尔转换规则

- 在前述类型中，只有以下值被转换为False：
- 值为零 (0, 0.0, 0j, False)
- 内容为空 ('', [], {}, (), set())
- 其他值转换到布尔型均为True，例如
- `bool('False')` # True

Python基本语法 – 控制结构

- `while <expr>:`
- `lines_of_code`

Python基本语法 – 控制结构

- `for <var> in <iterable>:`
- `lines_of_code`

Python基本语法 – iterable

- 按照一定的次序遍历一些值
- str, list, dict (实际上是它的键), tuple, set
- 常用的iterable： range
- range(3) # [0, 1, 2]
- range(4, 8) # [4, 5, 6, 7]
- range(10, 30, 5) # [10, 15, 20, 25]

输入输出

- `print(...)` # 输出括号内的值，并换行
- `input()` # 读入一行，并丢弃换行符
- `print('No newline!', end='')` # 输出后不换行
- `print(12345, end=', ')` # 输出12345,␣

作业1

- 编写一个程序，实现如下功能：
- 输入一个句子（用空格分割的小写字母串），输出每个单词出现的次数，输出的顺序不做要求
- 提交方式：将py文件发送至

wangqr@wangqr.tk

作业1

- 样例输入

a bbb c ddg dd bb bbb

- 样例输出

a 1

bb 1

bbb 2

c 1

dd 1

ddg 1

作业1 – 提示

- 字符串切分：

```
words = 'Beautiful is better than ugly.'.split()
```

```
# 得到一个list: words=['Beautiful','is','better','than','ugly.']
```

- 打印一个dict，用空格分隔键和值： # 假设这个dict叫 a

```
for x, y in a.items():
```

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

```
    print(y)
```

下节内容

- 函数
- 类
- 异常处理