Python3

本次课程

- 将介绍python3.x
- 将介绍python的基础语法
- 将介绍python中的变量
- 将介绍python中的基本数据类型
- 将介绍python中的基本运算

Python 3.x

- Python 是一种解释型、面向对象、动态数据类型的高级程序设计语言。
- Python 的 3.0 版本，常被称为 Python 3000，或简称 Py3k。相对于 Python 的早期版本，这是一个较大的升级。为了不带入过多的累赘，Python 3.0 在设计的时候没有考虑向下兼容。
- 官方宣布，2020 年 1 月 1 日， 停止 Python 2 的更新。
- 下载Python 3.8.1，访问https://www.python.org/downloads/。

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
```

- 以上命令的运行结果如下:

```
Python 3.8.1
```

- 你也可以进入Python的交互式编程模式，查看版本：

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

Hello World！

- 对于大多数程序语言，第一个入门编程代码便是"Hello World！"，以下代码为使用Python输出"Hello World！"：

```
print "Hello World!"
```

- 你可以直接在命令窗口输入以上代码，也可以将以上代码保存在hello.py的文件中并使用 python 命令执行该文件：

```
python3 hello.py
```

- 以上命令的输出结果为：

```
Hello World!
```

为何学习python？

- Python 是一种解释型语言： 这意味着开发过程中没有了编译这个环节。类似于PHP和Perl语言。
- Python 是交互式语言： 这意味着，您可以在一个 Python 提示符 >>> 后直接执行代码。
- Python 是面向对象语言: 这意味着Python支持面向对象的风格或代码封装在对象的编程技术。
- Python 是初学者的语言： Python 对初级程序员而言，是一种伟大的语言，它支持广泛的应用程序开发，从简单的文字处理到 WWW 浏览器再到游戏。

优点

- 易于学习：Python有相对较少的关键字，结构简单，和一个明确定义的语法，学习起来更加简单。
- 易于阅读：Python代码定义的更清晰。
- 易于维护：Python的成功在于它的源代码是相当容易维护的。
- 一个广泛的标准库：Python的最大的优势之一是丰富的库，跨平台的，在UNIX，Windows和Macintosh兼容很好。
- 互动模式：互动模式的支持，您可以从终端输入执行代码并获得结果的语言，互动的测试和调试代码片断。
- 可移植：基于其开放源代码的特性，Python已经被移植（也就是使其工作）到许多平台。
- 可扩展：如果你需要一段运行很快的关键代码，或者是想要编写一些不愿开放的算法，你可以使用C或C++完成那部分程序，然后从你的Python程序中调用。
- 数据库：Python提供所有主要的商业数据库的接口。
- GUI编程：Python支持GUI可以创建和移植到许多系统调用。
- 可嵌入: 你可以将Python嵌入到C/C++程序，让你的程序的用户获得"脚本化"的能力。

编码

- 默认情况下，Python 3 源码文件以 UTF-8 编码，所有字符串都是 unicode 字符串。

标识符

- 第一个字符必须是字母表中字母或下划线 _ 。
- 标识符的其他的部分由字母、数字和下划线组成。
- 标识符对大小写敏感。

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：

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', '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', 'y
ield']
```

注释

- Python中单行注释以 # 开头，实例如下：

```
# 注释1
print("Hello World!") # 注释2
```

- 多行注释可以用多个 # 号，还有 ''' 和 """
- 注释只供读者使用，作用是提高代码的可读性

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：

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

行与缩进

- 如果代码语句缩进数的空格数不一致，会导致运行错误：

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

- 运行结果如下：

```
  File "test.py", line 6
    print ("False")
                  ^
IndentationError: unindent does not match any outer indentation level
```

多行语句

- Python 通常是一行写完一条语句，但如果语句很长，我们可以使用反斜杠(\)来实现多行语句，例如：

```
total = item_one + \
        item_two + \
        item_three
```

- 在 [], {}, 或 () 中的多行语句，不需要使用反斜杠(\)，例如：

```
total = ['item_one', 'item_two', 'item_three',
        'item_four', 'item_five']
```

Print

- 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="":

```
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()
```

Print（续）

- 在上面的例子中，输出结果如下：

```
a
b
----------
a  b
```

变量

- 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。
- 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：

```
counter = 100          # 整型变量
miles   = 1000.0       # 浮点型变量
name    = "runoob"     # 字符串
```

变量

- Python允许同时为多个变量赋值。例如：

```
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。
- 你也可以为多个对象指定多个变量。例如：

```
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。

标准数据类型

- Python3 中有六个标准的数据类型：
  - Number（数字）
  - String（字符串）
  - List（列表）
  - Tuple（元组）
  - Set（集合）
  - Dictionary（字典）
- Python3 的六个标准数据类型中：
  - 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）。

数字类型

- Python3 支持 int、float、bool、complex（复数）。
- 在Python 3里，只有一种整数类型 int，表示为长整型，没有 python2 中的 Long。
- 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，例如：

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

- 输出结果如下：

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

数值运算

```
>>>5 + 4
9
>>>4.3 - 2
2.3
>>>3 * 7
21
>>>2 / 4
0.5
>>>17 % 3
2
>>>2 **5
32
```

数值运算

- Python 解释器可以作为一个简单的计算器，你可以在解释器里输入一个表达式，它将输出表达式的值。
- 表达式的语法很直白： +, -, * 和 /, 和其它语言（如Pascal或C）里一样。例如：

```
>>>2 + 2
4
>>>50 - 5*6
20
>>>(50 - 5*6) / 4
5.0
>>>8 / 5 #总是返回一个浮点数
1.6
```

数值运算

- 注意：在不同的机器上浮点运算的结果可能会不一样。
- 在整数除法中，除法 / 总是返回一个浮点数，如果只想得到整数的结果，丢弃可能的分数部分，可以使用运算符 // ：

```
>>>17 / 3
5.66666666667
>>>17 // 3
5
```

数值运算

- 注意：// 得到的并不一定是整数类型的数，它与分母分子的数据类型有关系。

```
>>>7 // 2
3
>>>7.0 // 2
3.0
>>>
```

数值运算

- 等号 = 用于给变量赋值。赋值之后，除了下一个提示符，解释器不会显示任何结果。

```
>>> width = 20
>>> height = 5*9
>>> width * height
900
```

数值运算

- 变量在使用前必须先"定义"（即赋予变量一个值），否则会出现错误：

```
>>> n  # 尝试访问一个未定义的变量
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'n' is not defined
```

数值运算

- 不同类型的数混合运算时会将整数转换为浮点数：

```
>>> 3 * 3.75 / 1.5
7.5
>>> 7.0 / 2
3.5
```

数值运算

- 在交互模式中，最后被输出的表达式结果被赋值给变量 _ 。例如:

```
>>> tax = 12.5 / 100
>>> price = 100.50
>>> price * tax
12.5625
>>> price + _
113.0625
>>> round(_, 2)
113.06
```

- 此处， _ 变量应被用户视为只读变量。

数值运算符

- + 加法
- - 减法
- * 乘法
- / 除法
- % 取余
- ** 乘方
- // 取整除

比较运算符

运算符	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。
>	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。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。	(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。

赋值运算符

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

:=	海象运算符，可在表达式内部为变量赋值。Python3.8 版本新增运算符。	在这个示例中，赋值表达式可以避免调用 len() 两次:
		`if (n := len(a)) > 10:`
		`    print(f"List is too long ({n} elements, expected <= 10)")`

其他运算符

- Python位运算符
- Python逻辑运算符
- Python成员运算符
- Python身份运算符
- ...

字符串

- 字符串是 Python 中最常用的数据类型。我们可以使用引号(' 或 ")来创建字符串。
- 创建字符串很简单，只要为变量分配一个值即可。例如：

```
var1 = 'Hello World!'
var2 = "Runoob"
```

字符串

- Python 不支持单字符类型，单字符在 Python 中也是作为一个字符串使用。
- Python 访问子字符串，可以使用方括号来截取字符串，如下实例：

```
var1 = 'Hello World!'
var2 = "Runoob"

print ("var1[0]: ", var1[0])
print ("var2[1:5]: ", var2[1:5])
```

- 以上实例执行结果：

```
var1[0]:  H
var2[1:5]:  unoo
```

修改字符串

- 你可以截取字符串的一部分并与其他字段拼接，如下实例:

```
var1 = 'Hello World!'

print ("已修改字符串 : ", var1[:6] + 'Runoob!')
```

- 输出结果如下:

```
已修改字符串 :  Hello Runoob!
```

小结

- Python 是一种解释型、面向对象、动态数据类型的高级程序设计语言。
- Python允许同时为多个变量赋值，也可以对多个对象指定多个变量。
- Python数值运算的基本运算符：+、-、*、/、%、//...
- 字符串是 Python 中最常用的数据类型。