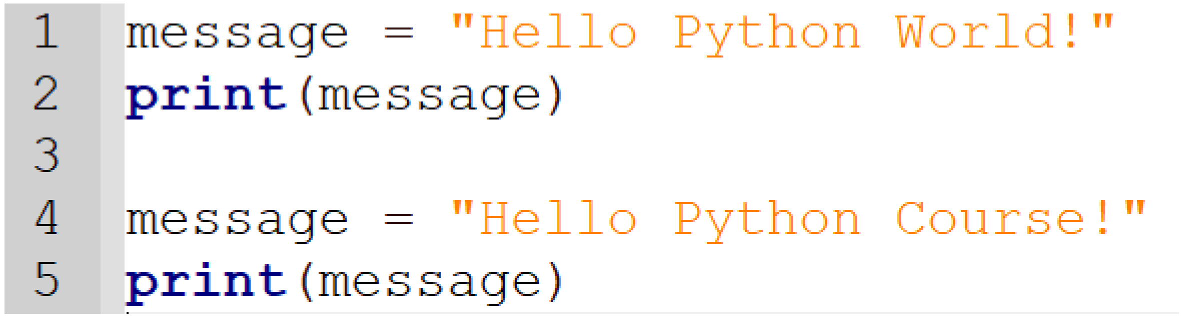
第二讲：变量和简单数据类型

Python的变量



```python
message = "Hello Python World!"
print(message)

message = "Hello Python Course!"
print(message)
```

- 这里message就是变量，它关联的值在程序运行的过程中发生了改变
- Python的变量不像C++需要先声明类型

变量命名注意事项

- 变量名只能包含字母、数字和下划线，变量名可以字母或下划线打头，但不能以数字打头
- 变量名不能包含空格，但可使用下划线来分隔其中的单词
- 不要将Python关键字和函数名用作变量名，如print （参见附录A.4）
- 变量名应既简短又具有描述性

字符串

- 字符串 就是一系列字符
- 在Python中，用引号括起的都是字符串，其中的引号可以是单引号，也可以是双引号，例如：

  "This is a string."

  'This is also a string.'
- 这种灵活性让你能够在字符串中包含双引号和单引号

  'I told my friend, "Python is my favorite language!"'

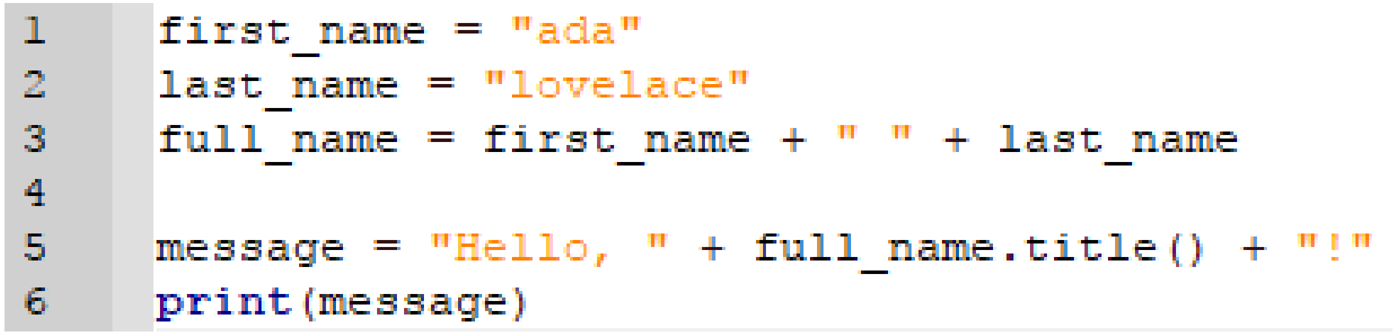
  "The language 'Python' is named after Monty Python, not the snake."

  "One of Python's strengths is its diverse and supportive community."

大小写转换和字符串的连接

- title(): 将字符串中每个单词的首字母改成大写
- upper(): 将字符串中的所有字母都改为大写
- lower():将字符串中的所有字母都改为大写
- +: 连接两个字符串

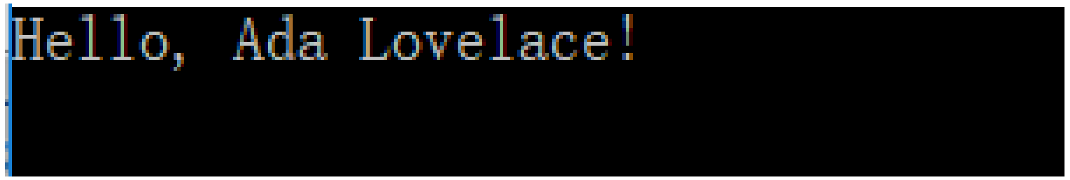
例子



```python
first_name = "ada"
last_name = "lovelace"
full_name = first_name + " " + last_name

message = "Hello, " + full_name.title() + "!"
print(message)
```

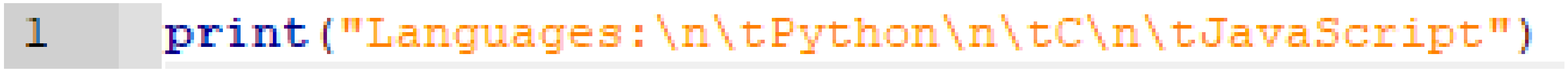
运行结果：



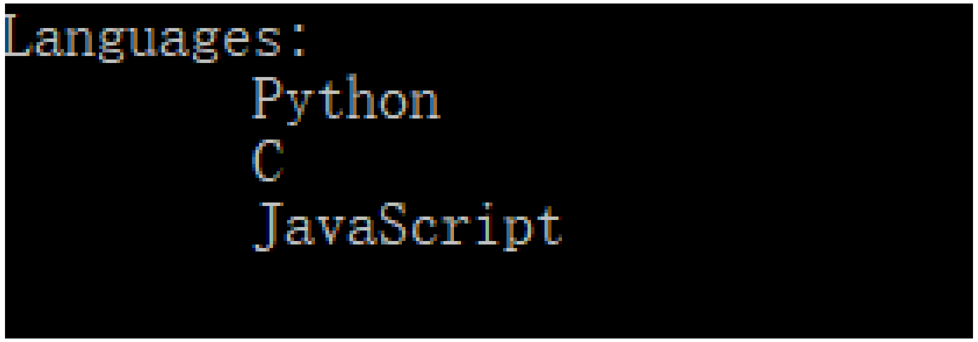
```
Hello, Ada Lovelace!
```

字符串中包含制表符和换行符

- Python也可以像C++一样使用转义字符，\n表示换行符，\t表示制表符，例如



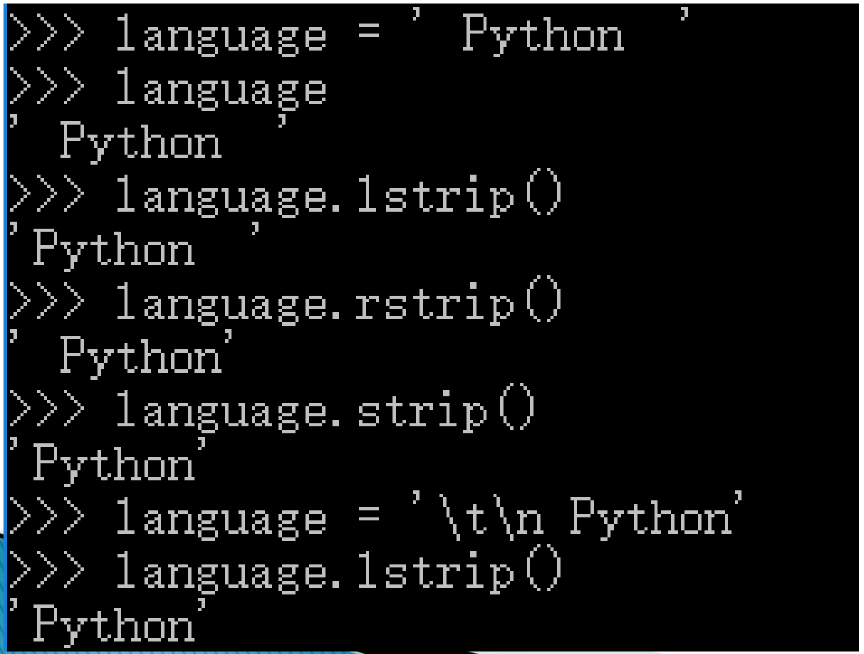
```python
print("Languages:\n\tPython\n\tC\n\tJavaScript")
```



```
Languages:
	Python
	C
	JavaScript
```

删除字符串前后的空白

- lstrip(): 删除字符串前面的空白字符
- rstrip(): 删除字符串后面的空白字符
- strip(): 删除字符串前后的空白字符
- 空白字符除了空格外，也包含换行符制表符等，例如：

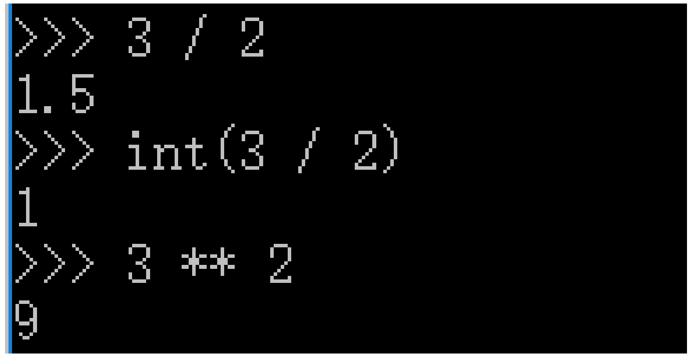


```
>>> language = '  Python  '
>>> language
'  Python  '
>>> language.lstrip()
'Python  '
>>> language.rstrip()
'  Python'
>>> language.strip()
'Python'
>>> language = '\t\n Python'
>>> language.lstrip()
'Python'
```

整数及其运算

- Python的整数运算+, -, *, %与C++相同
- 在Python3中，整数除法不会自动取整
- 如果要取整，可以用强制类型转换
- 用**来求幂

例如：

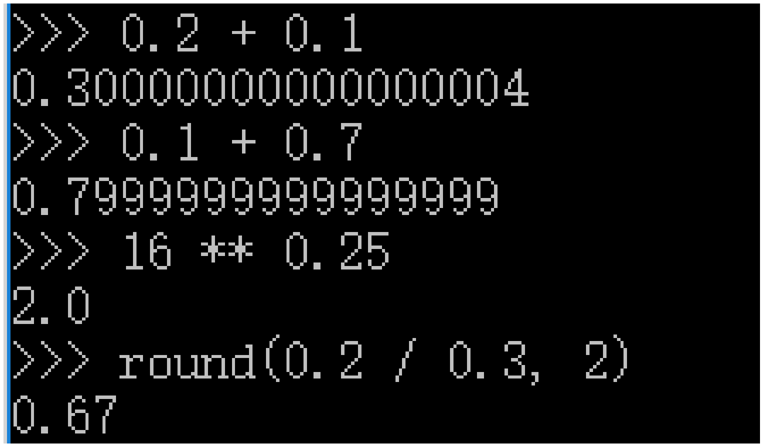


```
>>> 3 / 2
1.5
>>> int(3 / 2)
1
>>> 3 ** 2
9
```

浮点数及其运算

- 运算+, -, *, /与C++相同
- 注意会有浮点误差
- round(x, k)表示四舍五入保留小数点后k位
- 用**来求幂

例如：

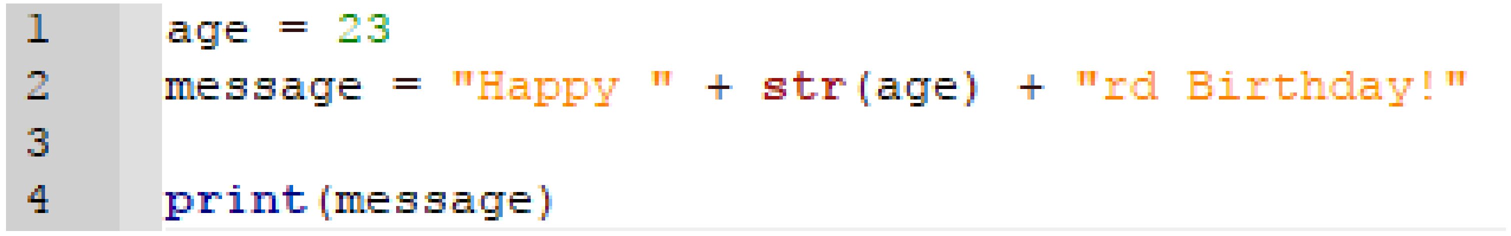


```
>>> 0.2 + 0.1
0.30000000000000004
>>> 0.1 + 0.7
0.7999999999999999
>>> 16 ** 0.25
2.0
>>> round(0.2 / 0.3, 2)
0.67
```

数值转化为字符串

- 类似于用int()来把浮点数转化为整数，我们也可以用str()来把数值转化为字符串，

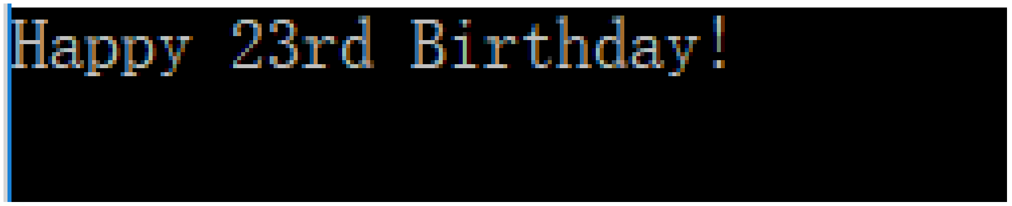
例如：



```python
age = 23
message = "Happy " + str(age) + "rd Birthday!"

print(message)
```

运行结果：



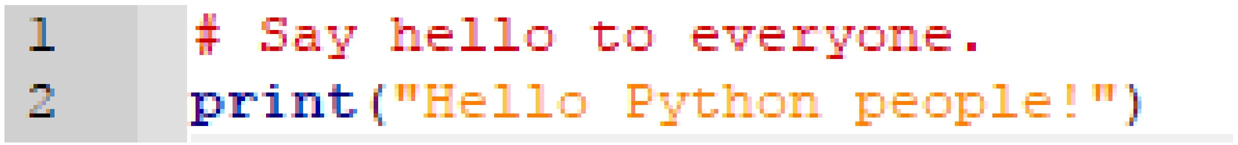
```
Happy 23rd Birthday!
```

注意：如果直接将数值和字符串进行连接会导致出错！

注释

- 以#号开始的一行是注释，不被执行

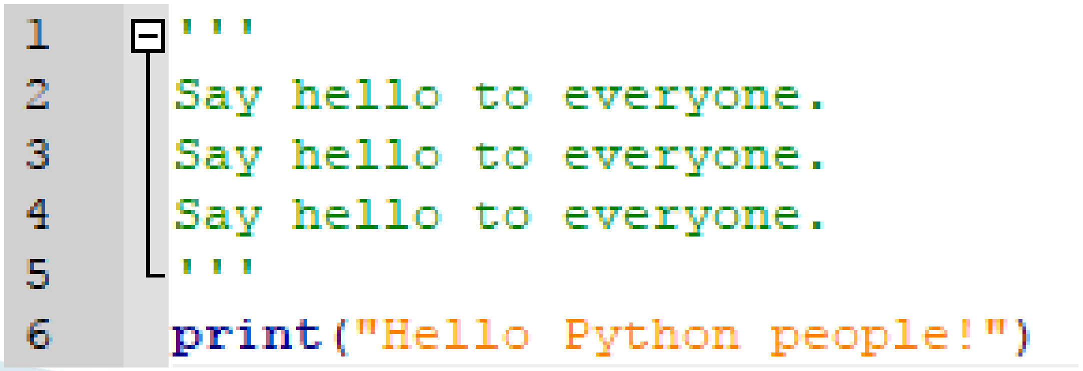
例如：



```python
# Say hello to everyone.
print("Hello Python people!")
```

- 多行注释可以前后各用三个单引号（或双引号），分别表示注释的开始和结束

例如：



```python
'''
Say hello to everyone.
Say hello to everyone.
Say hello to everyone.
'''
print("Hello Python people!")
```

Python之禅

在命令行界面输入import this可以看到”Python之禅”，其中列举了写Python程序的原则：

- 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.

Python之禅

- 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.

Python之禅

- 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!

总结

- 字符串类型及其常用操作
- 整数和浮点数的常用运算
- 注释
- Python之禅

- 下节课我们将学习重要的数据类型--列表

作业

- 教材中课后的练习，2-1到2-11，选一些写到你的博客上

谢谢！