优达学城
UDACITY

人工智能编程基础
第一章节
Intro to Python

目录
CONTENT

01 Python 基础破冰

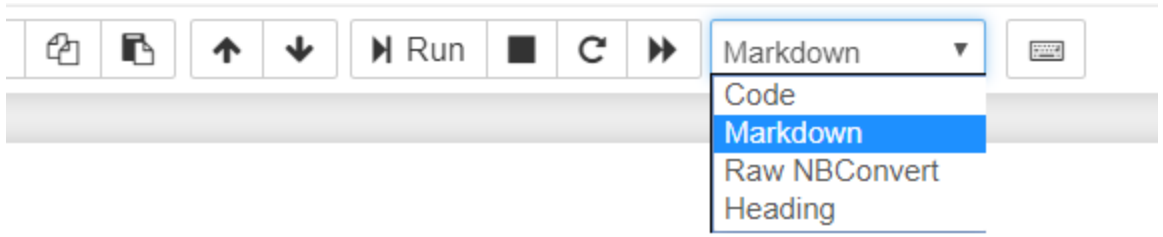
对项目中用到的最基础的 Python 语法进行讲解，让零基础学员更容易下手

01 Python 基础破冰

本章考虑到有部分没有编程基础的零基础学员上手项目困难，将课程中没有提到的基础内容在此呈现，
零基础学员可以多次查看。

文本格式

文本格式建议使用 Markdown 输出（这里有一个很简单的教程）



人工智能编程基础（试学班）

项目：我的微信好友

01 Python 基础破冰

本章考虑到有部分没有编程基础的零基础学员上手项目困难，将课程中没有提到的基础内容在此呈现，
零基础学员可以多次查看。

代码运行

```
In [1]: ### 以下内容无需改动，直接运行即可
        print("我的好友中共有", sex['male'],"位男性、", sex['female'], "位女性，有", sex['unknown'], "位好友未填写。")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-0116556c0048> in <module>()
      1 ### 以下内容无需改动，直接运行即可
----> 2 print("我的好友中共有", sex['male'],"位男性、", sex['female'], "位女性，有", sex['unknown'], "位好友未填写。")

NameError: name 'sex' is not defined
```

当重新打开编程环境时，需要**从最上方运行代码**，不然会得到明明定义过的问题出现未定义的错误

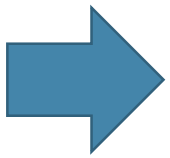
01 Python 基础破冰

本章考虑到有部分没有编程基础的零基础学员上手项目困难，将课程中没有提到的基础内容在此呈现，
零基础学员可以多次查看。

行和缩进

学习 Python 与其他语言最大的区别就是，Python 的代码块不使用大括号 {} 来控制类，函数以及其他逻辑判断。Python 最具特色的就是用**缩进**来写模块。

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



```
if True:
    print "Answer"
    print "True"
else:
    print "Answer"
    # 没有严格缩进，在执行时会报错
  print "False"
```

01 Python 基础破冰

本章考虑到有部分没有编程基础的零基础学员上手项目困难，将课程中没有提到的基础内容在此呈现，
零基础学员可以多次查看。

引号

Python 可以使用引号(‘)、双引号(“)、三引号 来表示字符串，引号的开始与结束必须的相同类型的（**都为英文标点**）。

其中三引号可以由多行组成，编写多行文本的快捷语法，常用于文档字符串，在文件的特定地点，被当做注释。

```
word = 'word'
sentence = "这是一个句子。"
paragraph = """这是一个段落。
包含了多个语句。"""
```

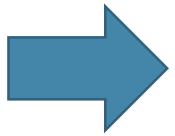
01 Python 基础破冰

本章考虑到有部分没有编程基础的零基础学员上手项目困难，将课程中没有提到的基础内容在此呈现，
零基础学员可以多次查看。

单行注释

Python 中单行注释采用 # 开头。

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
# 文件名：test.py

# 第一个注释
print "Hello, Python!";  # 第二个注释
```



```
Hello, Python!
```

01 Python 基础破冰

本章考虑到有部分没有编程基础的零基础学员上手项目困难，将课程中没有提到的基础内容在此呈现，
零基础学员可以多次查看。

多行注释

Python 中多行注释使用三个单引号(''')或三个双引号(""")。

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
# 文件名：test.py


'''
这是多行注释，使用单引号。
这是多行注释，使用单引号。
这是多行注释，使用单引号。
'''

"""
这是多行注释，使用双引号。
这是多行注释，使用双引号。
这是多行注释，使用双引号。
"""
```

01 Python 基础破冰

本章考虑到有部分没有编程基础的零基础学员上手项目困难，将课程中没有提到的基础内容在此呈现，
零基础学员可以多次查看。

大小写敏感

Python 是一个大小写敏感的语言，比如右边就是四条完全不同的代码，可能有的报错有的就能运行成功

```
sex = {'Sex':'male'}


Sex['Sex']
Sex['sex']
sex['Sex']
sex['sex']
```

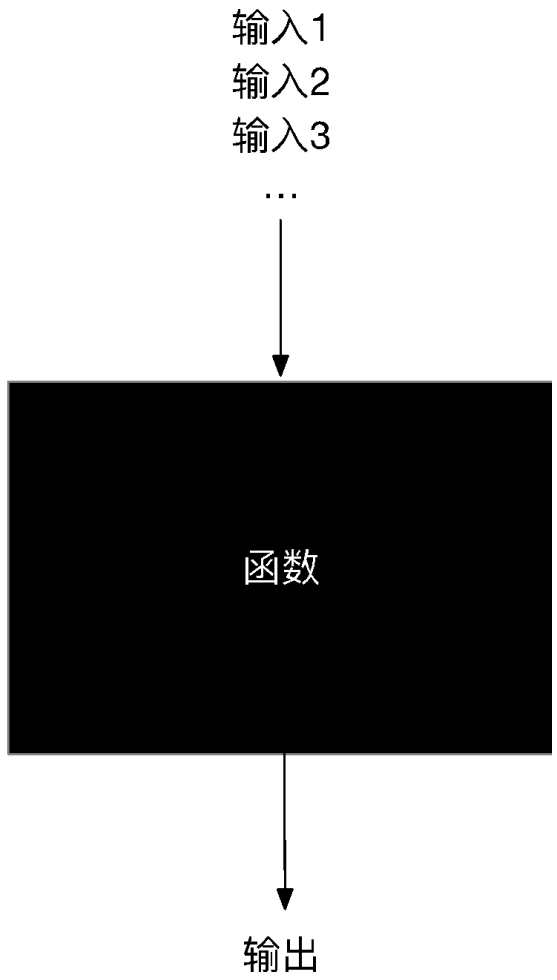
02 Python 函数

本章中将对Python中的函数做详细解释

02 Python 函数

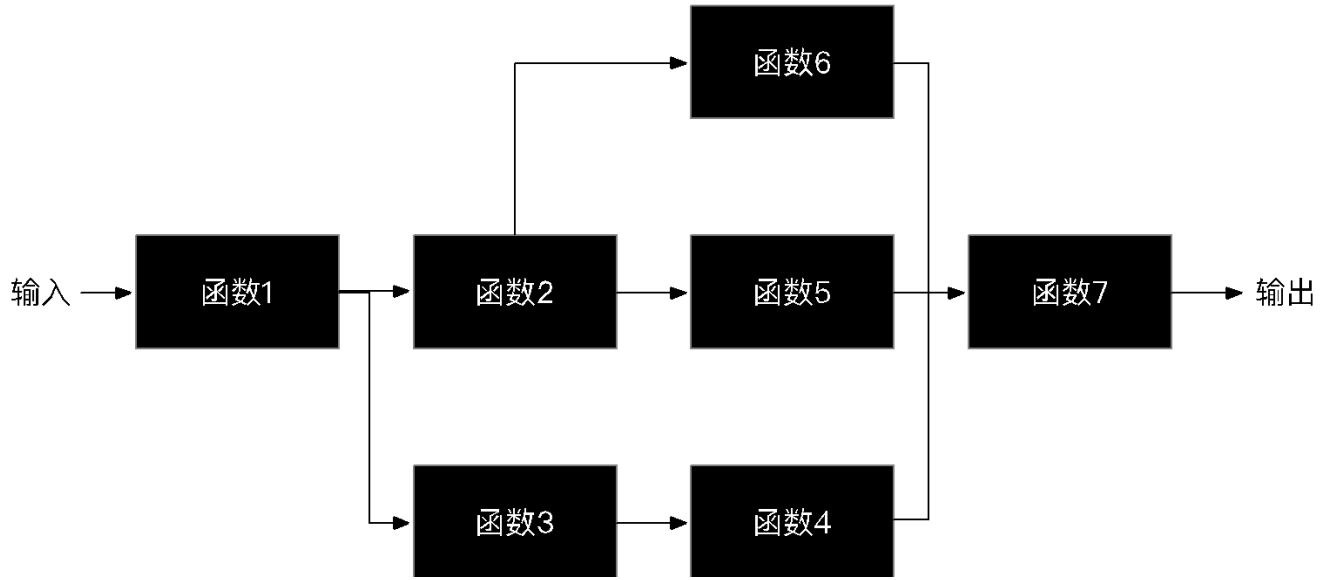
函数是什么?

- 函数是一个黑盒，投之以桃李，报之以琼瑶
- 函数也可以是一个容器，将多个稀碎的操作封装为一个内容可能很复杂但接口依然清晰的个体
- 函数是一段代码，可自定义其中的执行处理逻辑
- 函数可以被重复使用，编写一次即可受用无穷



为什么使用函数?

- 程序由函数组合而成，思路清晰，逻辑分明
- 功能模块化，步步为营，易于质量保证
- 将通用逻辑封装为函数，减少重复代码的编写
- 子功能职责边界清晰，程序逻辑修改方便
- 便于依赖管理，避免双向依赖
- 强鲁棒性，一个函数的异常不会影响其它函数的正确性



02 Python 函数

函数的定义：def <函数名>(<输入参数>):

- 函数定义关键字： def
- 函数名称：可自行定义，注意避免重名
- 输入参数：数量不限，也可以没有输入参数
- 返回关键字：将变量作为函数的输出返回

- 函数内部代码需缩进一层
- 函数的一次运行至多只会执行一次 return
- 函数不一定需要返回数值
- 在执行完 return 语句后，函数终止执行

```python
def factorial( x ):
    if x <= 0:
        return None
    elif x == 1:
        return x
    else:
        return x * factorial(x - 1)
```

02 Python 函数

函数的执行：<函数名>(<输入参数>)

- 函数调用：括号内为实际输入参数
- 函数输出值：将输出值赋值给新建的变量：result
- 在函数的定义中可以调用函数自身，形成循环调用，注意无限循环！

- 动手算一算：factorial(10)的值为：10*9*8*7*6*5*4*3*2*1
- 如果要计算20的阶乘，通过额外一行代码即可：

```
1   def factorial( x ):
2       if x <= 0:
3           return None
4       elif x == 1:
5           return x
6       else:
7           return x * factorial(x - 1)
8
9   result = factorial ( 10 )
10  print('Factorial of 10 equals: ', result)
```

```
问题  输出  调试控制台  终端                    1: zs

~/Desktop/python_demo  python3 demo.py
Factorial of 10 equals:  3628800
```

02 Python 函数

函数 vs. 生成器

```python
def fib_function(number):
    n, a, b, result = 0, 0, 1, []
    while n < number:
        result.append(b)
        a, b = b, a + b
        n = n + 1
    return result

print(fib_function(10))
```

```
问题  输出  调试控制台  终端

~/Desktop/python_demo  python3 demo.py
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

vs

```python
def fib_generator(number):
    n, a, b = 0, 0, 1
    while n < number:
        yield b
        a, b = b, a + b
        n = n + 1
    return 'OK'

print(fib_generator(10))
```

```
问题  输出  调试控制台  终端  1:

~/Desktop/python_demo  python3 demo.py
<generator object fib_generator at 0x103825bf8>
```

02 Python 函数

Why 生成器

为迭代而生!

```python
def fib_generator(number):
    n, a, b = 0, 0, 1
    while n < number:
        yield b
        a, b = b, a + b
        n = n + 1
    return 'OK'

f_generator = fib_generator(10)
print([i for i in f_generator])
```

```
问题  输出  调试控制台  终端

~/Desktop/python_demo  python3 demo.py
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

无限循环？！

```python
def fib_generator():
    a, b = 0, 1
    while True:
        yield b
        a, b = b, a + b
    return 'OK'

f_generator = fib_generator()
```

Easy… 要多少拿多少

```python
def fib_generator():
    a, b = 0, 1
    while True:
        yield b
        a, b = b, a + b
    return 'OK'

f_generator = fib_generator()
result = []
for i in range(10):
    result.append(f_generator.__next__())
print(result)
```

```
问题  输出  调试控制台  终端  1: zs

~/Desktop/python_demo  python3 demo.py
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

作用域

- Python 中，程序的变量并不是在哪个位置都可以访问的，访问权限决定于这个变量是在哪里赋值的。
- 变量的作用域决定了在哪一部分程序可以访问哪个特定的变量名称。
- Python的作用域一共有4种，分别是：

1. L （Local） 局部作用域
2. E （Enclosing） 闭包函数外的函数中
3. G （Global） 全局作用域
4. B （Built-in） 内建作用域

- 以 L –> E –> G –>B 的规则查找，即：在局部找不到，便会去局部外的局部找（例如闭包），再找不到就会去全局找，再者去内建中找。
- Python 中只有模块（module），类（class）以及函数（def、lambda）才会引入新的作用域，其它的代码块（如 if/elif/else/、try/except、for/while等）是不会引入新的作用域的，也就是说这些语句内定义的变量，外部也可以访问

```
x = int(2.9)          # 内建作用域
g_count = 0           # 全局作用域
def outer():
    o_count = 1       # 闭包函数外的函数中
    def inner():
        i_count = 2 # 局部作用域
```

```
1    a = 5
2    def print_a():
3        a = 10
4        print(a)
5    print_a()
6    print(a)
```

```
问题    输出    调试控制台    终端

~/Desktop/python_demo  python3 demo.py
10
5
```

03 拓展内容

经过拓展以后的学员就是不一样了呢，忍不住了让我皮一下 ☺

03 内容拓展

Immutable（不可变） vs. Mutable（可变）

- Immutable: int, float, tuple, string – 值传递
- Mutable: 其它 – 引用传递

```python
a_list = [1,2]
a_tuple = (1,2)

def manipulate(a_list, a_tuple):
    a_list.append(3)
    a_tuple = (1,2,3)
    print('list value inside function: ', a_list)
    print('tuple value inside function: ', a_tuple)
manipulate(a_list, a_tuple)
print('list value outside function: ', a_list)
print('tuple value outside function: ', a_tuple)
```

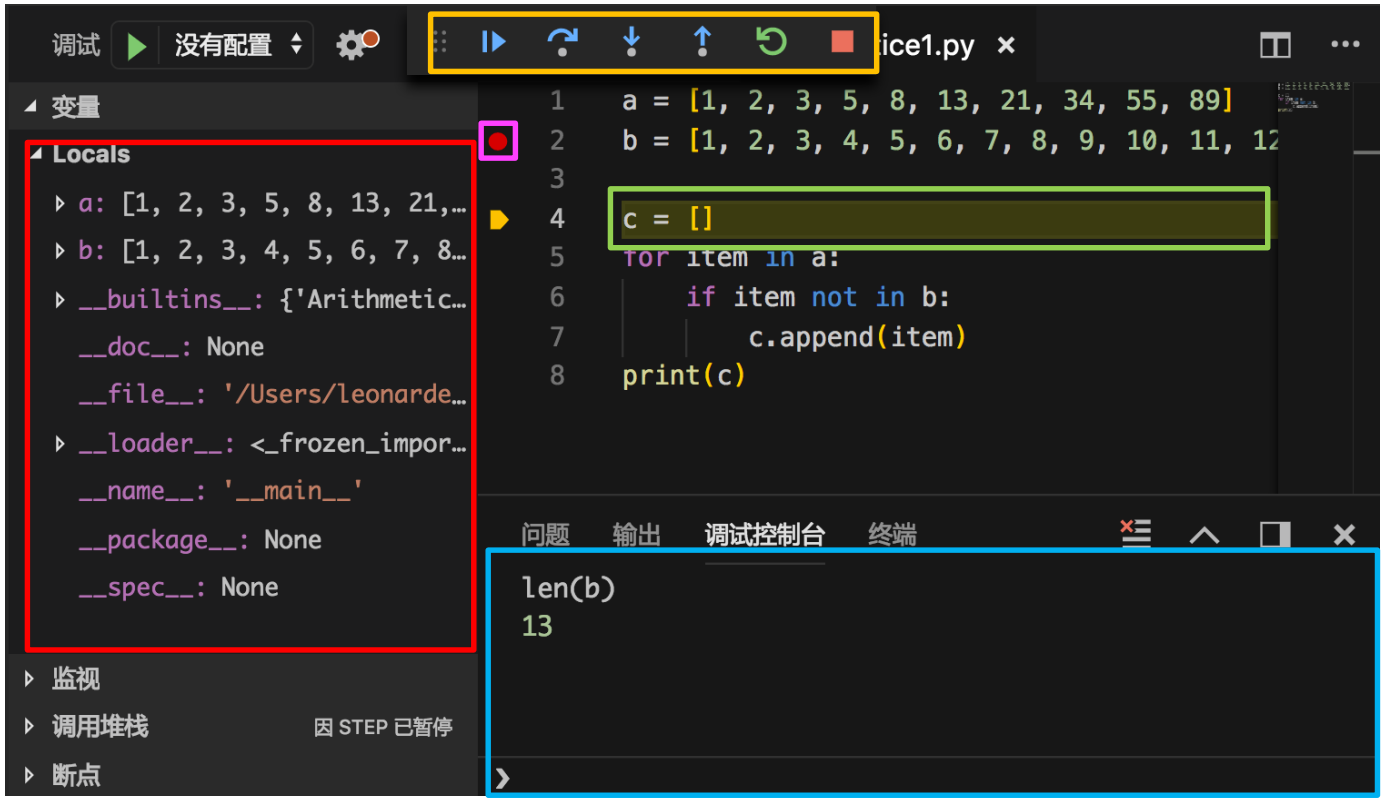
```
问题  输出  调试控制台  终端                    2: Python Debug C

~/Desktop/python_demo  python3 demo.py
list value inside function:  [1, 2, 3]
tuple value inside function:  (1, 2, 3)
list value outside function:  [1, 2, 3]
tuple value outside function:  (1, 2)
```

03 内容拓展

代码调试（以VS Code为例）

- 断点，可自定义任意数量断点
- 内存中的变量值查看
- 继续、单步跳过、单步调试、单步跳出
- 即将执行的步骤
- 调试控制台：可对当前内存变量进行代码操作

- 可以通过单步调试查看完整的代码运行过程
- 对于新手理解if、for、while等操作流很有帮助
- 让逻辑错误无处遁形 ☺



05 学员问题讲解（可选）

本章中将会对课程中学员提到较多的问题或易错点进行选择讲解

04 参考资料

本章会罗列出一些Python的参考资料与入门机器学习的一些参考资料

04 参考资料

资料推荐

[廖雪峰博客](#)　[编程小白学 Python](#)　[菜鸟教程](#)　[优达的免费机器学习介绍课](#)

《Python基础教程(第3版)》

《统计学习方法》

《机器学习》


优达学城
UDACITY
Thank you for listening!
Q & A