Python3.X编程初步

人工智能夏令营学习材料参考说明

编辑：Dr. 李佳

我们自己总结了一些基本知识点，也参考了一些网上资源。

译者注：本文智能单元首发，翻译自斯坦福CS231n课程笔记Python Numpy Tutorial，由课程教师Andrej Karpathy授权进行翻译。本篇教程由杜客翻译完成，Flood Sung、SunisDown、巩子嘉和一位不愿透露ID的知友对本翻译亦有贡献。

本部程序参考了廖雪峰的官方主页。

内容列表：

Python

Python是一种高级的，动态类型的多范型编程语言。很多时候，大家会说Python看起来简直和伪代码一样，这是因为你能够通过很少行数的代码表达出很有力的思想。举个例子，下面是用Python实现的经典的快速排序（快排）算法例子：

```
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) / 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quicksort(left) + middle + quicksort(right)

print(quicksort([3,6,8,10,1,2,1]))
# Prints "[1, 1, 2, 3, 6, 8, 10]"
```

Python版本

Python有两个支持的版本，分别是2.X和3.X。这有点让人迷惑，3.0向语言中引入了很多不向后兼容的变化，2.X下的代码有时候在3.X下是行不通的。在这个课程中，我们使用的是3.6版本。

如何查看版本呢？使用**python --version**命令。

基本数据类型

和大多数编程语言一样，Python拥有一系列的基本数据类型，比如整型、浮点型、布尔型和字符串等。这些类型的使用方式和在其他语言中的使用方式是类似的。

数字**Numbers**：整型和浮点型的使用与其他语言类似。

```
x = 3
print(type(x)) # 打印变量x的类型 "<class 'int'>"
print(x)       # Prints "3"
print(x + 1)   # 加法，  prints "4"
print(x - 1)   # 减法; prints "2"
print(x * 2)   # 乘法; prints "6"
print(x ** 2)  # 平方; prints "9"
x += 1
print(x)  # Prints "4"
x *= 2
print(x)  # Prints "8"
y = 2.5
print(type(y)) # Prints "<class 'float'>"
print(y, y + 1, y * 2, y ** 2) # 一次打印多个数字（对象） "2.5 3.5 5.0 6.25"
```

补充讲解：python程序的注释
1. # 注释，如上面这个例子
2. 模块或者对象（class、 funcntion)__doc__ 注释""" """，我们将放在最后介绍。

需要注意的是，Python中没有 x++（自加） 和 x— （自减）的操作符。

Python也有内置的长整型和复杂数字类型，具体细节可以查看文档。

布尔型**Booleans**：Python实现了所有的布尔逻辑，但用的是英语，而不是我们习惯的操作符（比如&&和||等）。

```
t = True
f = False
print(type(t)) # Prints "<class 'bool'>"
print(t and f) # Logical AND; prints "False"; 与运算
print(t or f)  # Logical OR; prints "True"; 或运算
print(not t)   # Logical NOT; prints "False"; 非运算
print(t != f)  # Logical XOR; prints "True"; 判断t是否不等于f
```

字符串**Strings**：Python对字符串的支持非常棒。

```
hello = 'hello'    # String literals can use single quotes; 一对单引号将字符串包围。
world = "world"    # or double quotes; it does not matter. 或者用双引号包围。
print(hello)       # Prints "hello"
```

```
print(len(hello))  # String length; prints "5"；打印字符串的长度，即输出这个字符串有几个字母。
hw = hello + ' ' + world  # String concatenation；字符串的拼接。
print(hw)  # prints "hello world"
hw12 = '%s %s %d' % (hello, world, 12)  # sprintf style string formatting；制定字符串的占位符。
print(hw12)  # prints "hello world 12"
```

【课余扩展】字符串对象有一系列有用的方法，比如：

```
s = "hello"
print(s.capitalize())  # Capitalize a string; prints "Hello"
print(s.upper())       # Convert a string to uppercase; prints "HELLO"
print(s.rjust(7))      # Right-justify a string, padding with spaces; prints "  hello"
print(s.center(7))     # Center a string, padding with spaces; prints " hello "
print(s.replace('l', '(ell)'))  # Replace all instances of one substring with another;
                                # prints "he(ell)(ell)o"
print('  world '.strip())  # Strip leading and trailing whitespace; prints "world"
```

变量

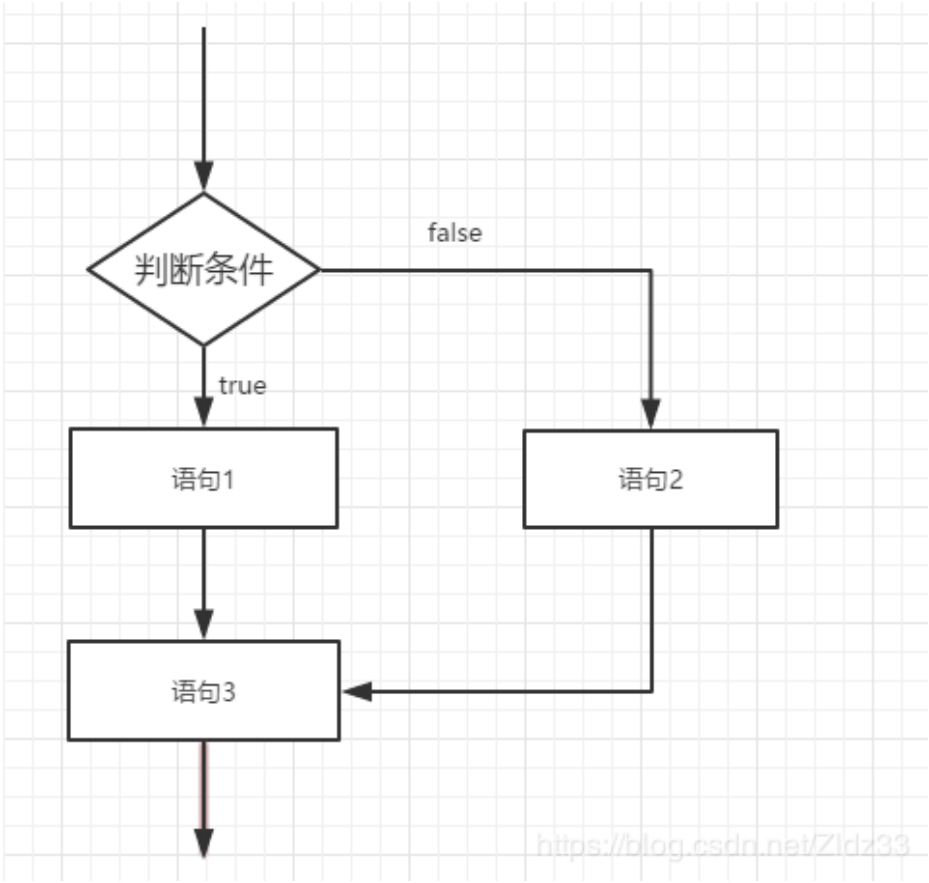
在说到变量之前，先来说一下常量。所谓常量就是不能变的变量，比如常用的数学常数π=3.1415926……就是一个常量。
变量的概念基本上和代数中方程变量x是一致的，只是在计算机程序中，变量不仅可以是数字，还可以是任意数据类型。
变量在程序中就是用一个变量名表示了，变量名必须是大小写英文、数字和_的组合，且不能用数字开头，比如：

```
a = 1
print(a)
a = 123.5  # 现在变量a不再是1了，而变成了123.5
print(a)
a = True
a = '哈哈，你好呀！'  #  这里的变量a变成了字符串
```

条件判断和循环

条件判断

Python中的条件判断以及分支结构是使用if-else语句实现的。计算机之所以能做很多自动化的任务，因为它可以自己做条件判断。



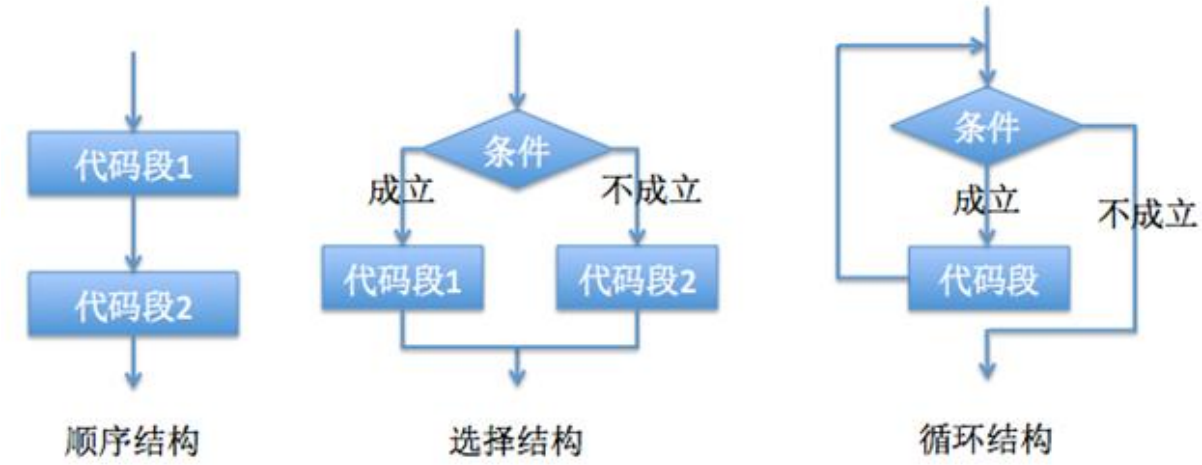

```
score = 75
if score >= 60:
  print('考试成绩及格')
else:
  print('考试成绩不及格')
```

分支结构由if-elif-else实现，比如：

```
age = 3
if age >= 18:
    print('成年人')
elif age >= 6:
    print('小朋友')
else:
    print('幼儿')
```

循环结构

我们先考虑一个问题，用Python计算1+2+3=？
接下来我们计算1+2+3+…+100=？这个时候我们不可能一个个手动把数字输入进去了，需要使用循环结构。



Python的循环有两种，一种是for...in循环

```
names = ['Michael', 'Bob', 'Tracy']
for name in names:
    print(name)

sum = 0
for i in range(1, 101):
  sum = sum + i
print(sum)
```

第二种循环是while循环，只要条件满足，就不断循环，条件不满足时退出循环。

```
sum = 0
n = 1
while n < 101:
    sum = sum + n
    n = n + 1
print(sum)
```

容器Containers

列表Lists

列表就是Python中的数组，但是列表长度可变，且能包含不同类型元素。

概念补充：索引（序号）


```
xs = [3, 1, 2]    # Create a list; 创建一个列表
print(xs, xs[2])  # Prints "[3, 1, 2] 2" ; ### 看看列表的索引是从0开始还是从1开始的?
print(xs[-1])     # prints "2" ; -1是索引，打印最后一个列表成员，负数表示从最后一个开始倒数。
xs[2] = 'foo'     # Lists can contain elements of different types; 更改列表索引为2的成员的值。
print(xs)         # Prints "[3, 1, 'foo']"
xs.append('bar')  # Add a new element to the end of the list; 在列表的最后追加元素。
print(xs)         # Prints "[3, 1, 'foo', 'bar']"
x = xs.pop()      # Remove and return the last element of the list; 弹出列表最后一个元素。
print(x, xs)      # Prints "bar [3, 1, 'foo']"
```

发射，子弹弹出！

切片**Slicing**：为了一次性地获取列表中的元素，Python提供了一种简洁的语法，这就是切片。

```
nums = list(range(5))     # range is a built-in function that creates a list of integers
print(nums)               # Prints "[0, 1, 2, 3, 4]"
print(nums[2:4])          # prints "[2, 3]"; 切出索引从2～4号（不包括4号）的切片
print(nums[2:])           # Get a slice from index 2 to the end; prints "[2, 3, 4]"
print(nums[:2])           # Get a slice from the start to index 2 (exclusive); prints "[0, 1]"
print(nums[:])            # Get a slice of the whole list; prints "[0, 1, 2, 3, 4]"
print(nums[:-1])          # Slice indices can be negative; prints "[0, 1, 2, 3]"
```

```
nums[2:4] = [8, 9]        # Assign a new sublist to a slice
print(nums)               # Prints "[0, 1, 8, 9, 4]"
```

循环Loops：我们可以这样遍历列表中的每一个元素：

```
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
# Prints "cat", "dog", "monkey", each on its own line.
```

如果想要在循环体内访问每个元素的指针（也就是索引，或者说是序号），可以使用内置的**enumerate**函数

```
animals = ['cat', 'dog', 'monkey']
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
# Prints "#1: cat", "#2: dog", "#3: monkey", each on its own line
```

【课余扩展】列表推导List comprehensions：在编程的时候，我们常常想要将一种数据类型转换为另一种。下面是一个简单例子，将列表中的每个元素变成它的平方。

```
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print(squares)   # Prints [0, 1, 4, 9, 16]
```

【课余扩展】使用列表推导，你就可以让代码简化很多：

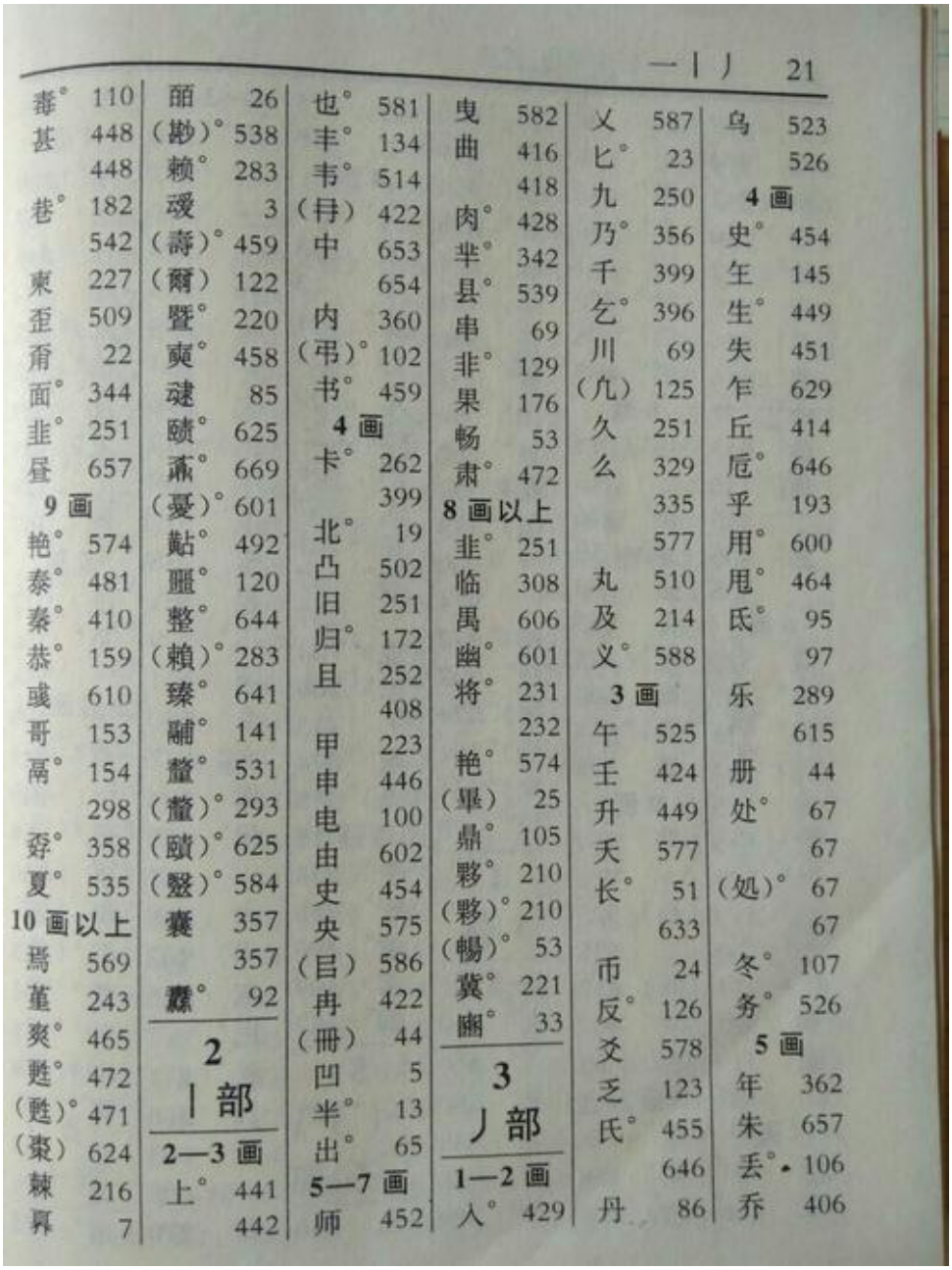
```
nums = [0, 1, 2, 3, 4]
squares = [x ** 2 for x in nums]
print(squares)   # Prints [0, 1, 4, 9, 16]
```

【课余扩展】列表推导还可以包含条件：

```
nums = [0, 1, 2, 3, 4]
even_squares = [x ** 2 for x in nums if x % 2 == 0]
print(even_squares)  # Prints "[0, 4, 16]"
```

字典Dictionaries

字典用来储存（键, 值）对，这和Java中的Map差不多。
打个比喻，键—偏旁比划索引，值—待查陌生字在字典中的页数。




```
d = {'cat': '猫', 'dog': '狗'}  # Create a new dictionary with some data
print(d['cat'])       # Get an entry from a dictionary; prints “猫”
print('cat' in d)     # Check if a dictionary has a given key; prints "True"
d['fish'] = '鱼'     # Set an entry in a dictionary
print(d['fish'])      # Prints “鱼”
# print(d['monkey'])  # KeyError: 'monkey' not a key of d
# print(d.get('monkey', 'N/A'))  # Get an element with a default; prints "N/A"
# print(d.get('fish', 'N/A'))    # Get an element with a default; prints "wet"
# del d['fish']         # Remove an element from a dictionary
# print(d.get('fish', 'N/A')) # "fish" is no longer a key; prints "N/A"
```

如果你想要访问键和对应的值，那就使用**iteritems**迭代方法：

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal, legs in d.items():
    print('A %s has %d legs' % (animal, legs))
# Prints "A person has 2 legs", "A cat has 4 legs", "A spider has 8 legs"
```

元组**Tuples**

元组是一个值的有序列表（不可改变）。从很多方面来说，元组和列表都很相似。和列表最重要的不同在于，元组可以在字典中用作键。

```
d = {(x, x + 1): x for x in range(10)}  # Create a dictionary with tuple keys; 创建元组
t = (5, 6)        # Create a tuple
print(type(t))    # Prints "<class 'tuple'>", 打印元组的类型
print(d[t])       # Prints "5"
print(d[(1, 2)])  # Prints "1" ; 元组和列表一样，也可以进行索引求值。
```

函数Functions

函数是组织好的，可以被重复使用的，用来实现单一或相关功能的代码块。
Python函数使用def来定义函数.


```
def sign(x):  # 符号函数，判断数是大于0还是小于0，即是正数还是负数。
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'

for x in [-1, 0, 1]:
    print(sign(x))
# Prints "negative", "zero", "positive"
```

【课余扩展】我们常常使用可选参数来定义函数：

```
def hello(name, loud=False):
    if loud:
        print('HELLO, %s!' % name.upper())
    else:
        print('Hello, %s' % name)

hello('Bob') # Prints "Hello, Bob"
hello('Fred', loud=True)  # Prints "HELLO, FRED!"
```

类Classes

类和对象的定义：

- 类是一个抽象的概念，是对某一类事物的抽象。举一个例子，可以把人类看作一个类，这个类的共性有：第一、站立行走，第二、有一个很发达的大脑，上面这两点都是静态的，描述的是客观的属性(attributes)。人类还需要吃饭、需要睡觉，上面这两点都是动态的行为，即方法(methods)。类可以包含函数，函数在类中就是动态的行为，即方法。
- 对象就是类的实例化，人类是一个类，而每一个人就是人类的实例化，即每一个人就是一个对象，对象具有类的属性及方法（每个人都站立行走、有一个发达的大脑，并且需要吃饭睡觉。

Python对于类的定义是简单直接的：

```
class Greeter(object):
    """ 我是一个Python类的定义，这里是注释，这就是前面提到的另一种注释方法 """
    # Constructor
    def __init__(self, name):  # 初始化函数
        self.name = name  # Create an instance variable

    # Instance method
    def greet(self, loud=False):  # 类中定义的方法
        if loud:
            print('HELLO, %s!' % self.name.upper())
        else:
            print('Hello, %s' % self.name)

g = Greeter('Fred')  # Construct an instance of the Greeter class
g.greet()            # Call an instance method; prints "Hello, Fred"
g.greet(loud=True)   # Call an instance method; prints "HELLO, FRED!"
```

Python模块的导入与使用——模块化设计的魔力

回到我们最开始展示的排序算法，可能具体算法原理我们不太明白，但这不影响我们使用。在许多工作的场合，我们的目的可能主要就是为了解决具体问题，而不是一切从零开始造一个“轮子”（轮子，英文里称作“wheel”，是计算机文件的一种）。我们要利用好别人已经开发好并且经过实践检验的工具的力量，通过函数或者对象封装，将程序划分成模块以及模块间的表达。从而把编程的过程简化，最后如图我们在玩搭积木游戏一般，解决实际问题。

实例1：导入Opencv模块，调用计算机连接的摄像头，并且实时显示。

```
import cv2  # 导入第三方模块
import numpy as np # 导入第三方模块

cap = cv2.VideoCapture(0)
while(1):
    # get a frame
    ret, frame = cap.read()
    # show a frame
    cv2.imshow("capture", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

实例2. 利用Opencv做图像的边缘检测（调用图像处理库中的Canny边缘检测）

```
# 摄像头并显示轮廓
import cv2
cap = cv2.VideoCapture(0)
i=0
while(1):
    ret, frame = cap.read()
    img_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    img_gb = cv2.GaussianBlur(img_gray, (5, 5), 0)
    edges = cv2.Canny(img_gb, 100 , 200)
    cv2.imshow("capture", edges)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
```

```
cv2.destroyAllWindows()
```

作业1. 在Pycharm上新建程序文件，运行课堂上的实例程序
作业2. 探究Python中的and, or, not的作用， False and True = ？ False or True =? Not True=?
作业3. 编写程序计算1+2+3+…+1000=?
作业4. 任意导入一个Python标准库并且使用库里的一个函数
作业5. 使用快速排序算法对自己输入的一个数字列表如 [3,6,8,10,1,2,1] 进行排序并打印输出
作业6. 用Python调用摄像头