# Notes for Computer Science

**by 王一粟 2200015507**

# Course 1：数算课程导论

## 一、计算机科学

**1.计算机科学相关概念的简单介绍**

计算机科学：研究问题及其解决方案、以及研究目前无解的问题的学科，它的研究对象是问题、解决问题的过程，以及通过该过程得到的解决方案。

算法：具有有限步骤、能够逐步解决给定问题的解决方案

可计算：存在能够解决该问题的算法，则该问题为可计算的

抽象：从逻辑视角和物理视角来看待问题和解决方案

接口：即指设计者利用计算机所提供的功能，使用者无需了解功能的实现细节，从逻辑视角看待计算机，但计算机科学家必须知道底层细节。一个简单的例子即是python的模块类，如math，这种计算有时候也被称为过程的“黑盒”视角。

编程：为解决方案创造的表达方式，编程语言提供了表达过程和数据的标记方式

编程语言用于描述算法的条件：能够进行顺序执行、决策分支、循环迭代

数据类型：赋予计算机中二进制字符串实际的意义。内建的底层数据类型（原生数据类型）提供了算法开发的基本单元

**2.数据结构与算法**

为了控制问题及其求解过程的复杂度，计算机科学利用抽象（对问题进行建模）来高效解决问题

过程抽象将功能的实现细节隐藏起来；抽象数据类型（ADT）从逻辑上描述了如何看待数据及其对应运算而无需考虑具体实现，我们仅需关心数据代表了什么，而可以忽略它们的构建方式（封装具体的实现细节）。抽象数据类型的实现常被称为数据结构

各种算法之间存在效率、速度、内存等方面的巨大差异，我们需要比较不同算法的分析技巧、区分有解和无解的问题

## 二、Python基础

python是面向对象的解释性语言（只需要查看和描述交互式会话）

**1 数据**

python支持面向对象（客观世界的实体在计算机逻辑中的映射）编程范式，即数据是解决问题过程中的关键点

类（class）是对数据的构成（状态）以及数据能做什么（行为）的描述。类与ADT是类似的。在python中，数据项被称作对象，一个对象就是类的一个实例

**内建原子数据类型：**

python有两大内建数据类实现了整数类型和浮点数类型，即所谓int与float，配合数学运算符+、-、*、/、**、%、//一起使用

python通过bool类实现逻辑运算，可能状态值为True与False，布尔运算符有and、or和not。同时，布尔对象也被用作相等==、大于>、小于等于<=、不等于!=等比较运算符的计算结果

标识符在编程语言中被用作名字，python中以字母或者下划线_开头，且区分大小写。当一个名字第一次出现在赋值语句的左边部分时，会创建对应的python变量。赋值语句将名字与值关联起来，变量存的是指向数据的引用，而非数据本身。

**内建集合数据类型**

python还拥有众多强大的内建集合类：列表、字符串以及元组是概念上非常相似的有序集合，字典和集是无序集合。

**列表**是零个或多个指向python数据对象的引用的有序集合，通过在方括号内以逗号分隔的一系列值来表达。空列表就是[]。列表是异构的：其指向的数据对象不需要都是同一个类，并且这一集合可以被赋值给一个变量。

需注意：列表和序列的下标从0开始，切片初始值包括但终端不包括；快速初始化列表可通过重复运算来实现，如 mylist = [0] * 6（but，重复运算返回的结果是序列中指向数据对象的引用的重复）

| Operation Name | Operator | Explanation |
|---|---|---|
| indexing | [ ] | Access an element of a sequence |
| concatenation | + | Combine sequences together |
| repetition | * | Concatenate a repeated number of times |
| membership | in | Ask whether an item is in a sequence |
| length | len | Ask the number of items in the sequence |
| slicing | [ : ] | Extract a part of a sequence |

| ethod Name | Use | Explanation |
|---|---|---|
| `append` | `alist.append(item)` | Adds a new item to the end of a list |
| `insert` | `alist.insert(i,item)` | Inserts an item at the ith position in a list |
| `pop` | `alist.pop()` | Removes and returns the last item in a list |
| `pop` | `alist.pop(i)` | Removes and returns the ith item in a list |
| `sort` | `alist.sort()` | Modifies a list to be sorted |
| `reverse` | `alist.reverse()` | Modifies a list to be in reverse order |
| `del` | `del alist[i]` | Deletes the item in the ith position |
| `index` | `alist.index(item)` | Returns the index of the first occurrence of `item` |
| `count` | `alist.count(item)` | Returns the number of occurrences of `item` |
| `remove` | `alist.remove(item)` | Removes the first occurrence of `item` |

**range**会生成一个代表值序列的范围对象。默认情况下其从0开始；同样地，range包括初始不包括终值，也可在终值后加入步长。

**字符串**是零个或多个字母、数字和其他符号组成的有序集合。常量字符串通过单引号或双引号与标识符进行区分。由于字符串是序列，因此所有的序列运算符都能用于字符串（即表1）。此外，字符串还有一些其他的特有方法。

| Method Name | Use | Explanation |
| --- | --- | --- |
| `center` | `astring.center(w)` | Returns a string centered in a field of size `w` |
| `count` | `astring.count(item)` | Returns the number of occurrences of `item` in the string |
| `ljust` | `astring.ljust(w)` | Returns a string left-justified in a field of size `w` |
| `lower` | `astring.lower()` | Returns a string in all lowercase |
| `rjust` | `astring.rjust(w)` | Returns a string right-justified in a field of size `w` |
| `find` | `astring.find(item)` | Returns the index of the first occurrence of `item` |
| `split` | `astring.split(schar)` | Splits a string into substrings at `schar` |

列表具有可修改性，但字符串是不能被修改的。

**元组**与列表一样，都是异构数据序列，但元组是不可修改的。空元组可表示为（）或tuple（）。元组适用所有的序列运算（表1）

**集set**是由0个或多个不可修改的python数据对象组成的无序集合，不允许重复元素。其写成由花括号包含、以逗号分隔的一系列值。空集由set（）表示。集是异构的。

集支持以下运算：

| Operation Name | Operator | Explanation |
| --- | --- | --- |
| membership | in | Set membership |
| length | len | Returns the cardinality of the set |
| ` | ` | `aset |
| `&` | `aset & otherset` | Returns a new set with only those elements common to both sets |
| `-` | `aset - otherset` | Returns a new set with all items from the first set not in second |
| `<=` | `aset <= otherset` | Asks whether all elements of the first set are in the second |

| Method Name | Use | Explanation |
| --- | --- | --- |
| `union` | `aset.union(otherset)` | Returns a new set with all elements from both sets |
| `intersection` | `aset.intersection(otherset)` | Returns a new set with only those elements common to both sets |
| `difference` | `aset.difference(otherset)` | Returns a new set with all items from first set not in second |
| `issubset` | `aset.issubset(otherset)` | Asks whether all elements of one set are in the other |
| `add` | `aset.add(item)` | Adds item to the set |
| `remove` | `aset.remove(item)` | Removes item from the set |
| `pop` | `aset.pop()` | Removes an arbitrary element from the set |
| `clear` | `aset.clear()` | Removes all elements from the set |

**字典**是无序结构，由相关的元素对构成（key：value）。字典由花括号包含的一系列以逗号分隔的键：值对表达。访问字典是通过键来访问，而不是使用下标。len函数对字典的功能与对其他集合的功能性相同。可以使用list函数将字典转换成列表。

| Operator | Use | Explanation |
| --- | --- | --- |
| `[]` | `myDict[k]` | Returns the value associated with `k`, otherwise its an error |
| `in` | `key in adict` | Returns `True` if key is in the dictionary, `False` otherwise |
| `del` | del `adict[key]` | Removes the entry from the dictionary |

| Method Name | Use | Explanation |
| --- | --- | --- |
| `keys` | `adict.keys()` | Returns the keys of the dictionary in a dict_keys object |
| `values` | `adict.values()` | Returns the values of the dictionary in a dict_values object |
| `items` | `adict.items()` | Returns the key-value pairs in a dict_items object |
| `get` | `adict.get(k)` | Returns the value associated with `k`, `None` otherwise |
| `get` | `adict.get(k,alt)` | Returns the value associated with `k`, `alt` otherwise |

**2 输入与输出**

**input**函数接受一个**字符串**作为参数，让用户输入数据，然后再对这些数据进行进一步处理

**print**函数为输出python程序的值提供了一种非常简便的方法。它接受0个或多个参数，并且将单个空格作为默认分隔符来显示结果，且每一次打印都默认以换行符结尾。通过设置sep这一实际参数可以改变分隔符，end则可以设置是否以换行符作为结尾

**3 控制结构**

算法需要迭代和分支两个重要的控制结构。

迭代：while语句和for语句。while语句会在给定条件为真时重复执行一段代码

分支：根据结果（条件）采取不同的行动。在python中，存在if-elif-else语句。需注意，使用elif时，else语句是必需的。

列表可以不通过迭代结构和分支结构来创建，这种方式被称为列表解析式。有表达式 [expression_if if condition else expression_else for item in iterable]

**4 异常处理**

编写程序时的两种错误：语法错误（编写语句或者表达式的错误）与逻辑错误（程序能执行完成但返回错误的结果）。我们可以用try语句来处理被抛出的异常。

**5 定义函数**

我们可以通过定义函数来隐藏任何计算的细节。函数的定义需要一个函数名，一系列参数以及一个函数体：def Function（x1,x2...）

## 三、python面向对象编程：定义类

前述均为python内建的类去展示数据和控制结构，而所谓面向对象编程语言最强大的一项特性是允许创建全新的类来对求解问题所需的数据进行建模。通过构建能实现抽象数据类型的类，可以利用抽象过程，同时为真正在程序中运用抽象提供必要的细节。每当需要实现抽象数据类型时，就可以创建新类

**1 定义新类**

在python中定义新类的做法是，提供一个类名以及一整套与函数定义语法类似的方法定义。在此，对于分数fraction类的新定义将会是比较好的认知实例。

```
class fraction:

    #所有类都需要首先提供构造方法。构造方法定义了数据对象的创建方式
    #在python中，构造方法总是命名为__init__(init前后分别有两个下划线)
    def __init__(self,top,bottom):
        self.num = top
        self.den = bottom
        #形式参数列表中，第一个总是self，是指向对象本身的特殊参数
        #在调用方法是，从来不需要提供相应的实际参数
        #self.num与self.den定义了fraction对象中有num和den两个数据对象共同组成其状态
        #在此情况下，我们就可以使用类名完成调用：
        #myfraction = Fraction(3,5)即创建了一个实际参数为3、5的数据，数据类型为fraction
        #而在后续行为函数设计中，只需要self作为形参，就可以进行对函数的设定

    #接下来，我们要定义这一ADT所需要的行为
    #前置知识：python的所有类都提供了一套标准方法，但是可能没有正常工作

    #首先是fraction的表示方法，或者说，让其能够以可视化方法打印出来
    #有两种做法：
    #1. 定义show方法，使得fraction对象能够将自己作为字符串来进行打印：
    def show(self):
        print(self.num,"/",self.den)
    #2. 重置类中标准函数str（将对象转换成字符串的方法）
    def __str__(self):
```

```
        return str(self.num)+"/"+str(self.den)

    #其次是分数的加法。__add__也是类的标准方法，我们可以对其进行重置
    def __add__(self,otherfraction):
        newnum = self.num*otherfraction.den+self.den*otherfraction.num
        newden = self.den*otherfraction.den
        return fraction(newnum,newden)

    #改进：考虑约分的问题。该方法需要寻找分子和分母的最大公因数GCD（greatest common divisor）
    def gcd(m,n):
        while m % n != 0:
            oldm = m
            oldn = n
            m = oldn
            n = oldm%oldn
        return n

    #从而可以对add进行改进
    #def __add__(self,otherfraction):
        #newnum = self.num*otherfraction.den+self.den*otherfraction.num
        #newden = self.den*otherfraction.den
        #common = god(newnum,newden)
        #return fraction(newnum//common,newden//common)

    #判断相等：通过重写__eq__方法，可以进行深相等（根据值来判断相等）
    def __eq__(self,other):
        firstnum = self.num*other.den
        secondnum = self.den*other.num
        return firstnum == secondnum
```

实际编程中其他需要注意的事项：

内置函数的使用：add用+/xxx1.**add**(xxx2)，equal用==，str直接用str符号判断/xxx.**str**()

最初定义类不加参数，后续函数才需要

运用创设在类中的函数时，需要带上类的名字，如fraction.show()

**2 继承Inheritance**

面向对象编程的另一个重要方面就是**继承**。

继承使一个类与另一个类相关联。python中的子类可以从父类继承特征数据和行为。父类也称为超类。我们将这样的关系结构称为继承层次结构。

在内建的python集合类中：有序集合类包括列表、字符串和元组，无序集合类包括字典；有序集合与无序集合共同构成了python内建的集合类。