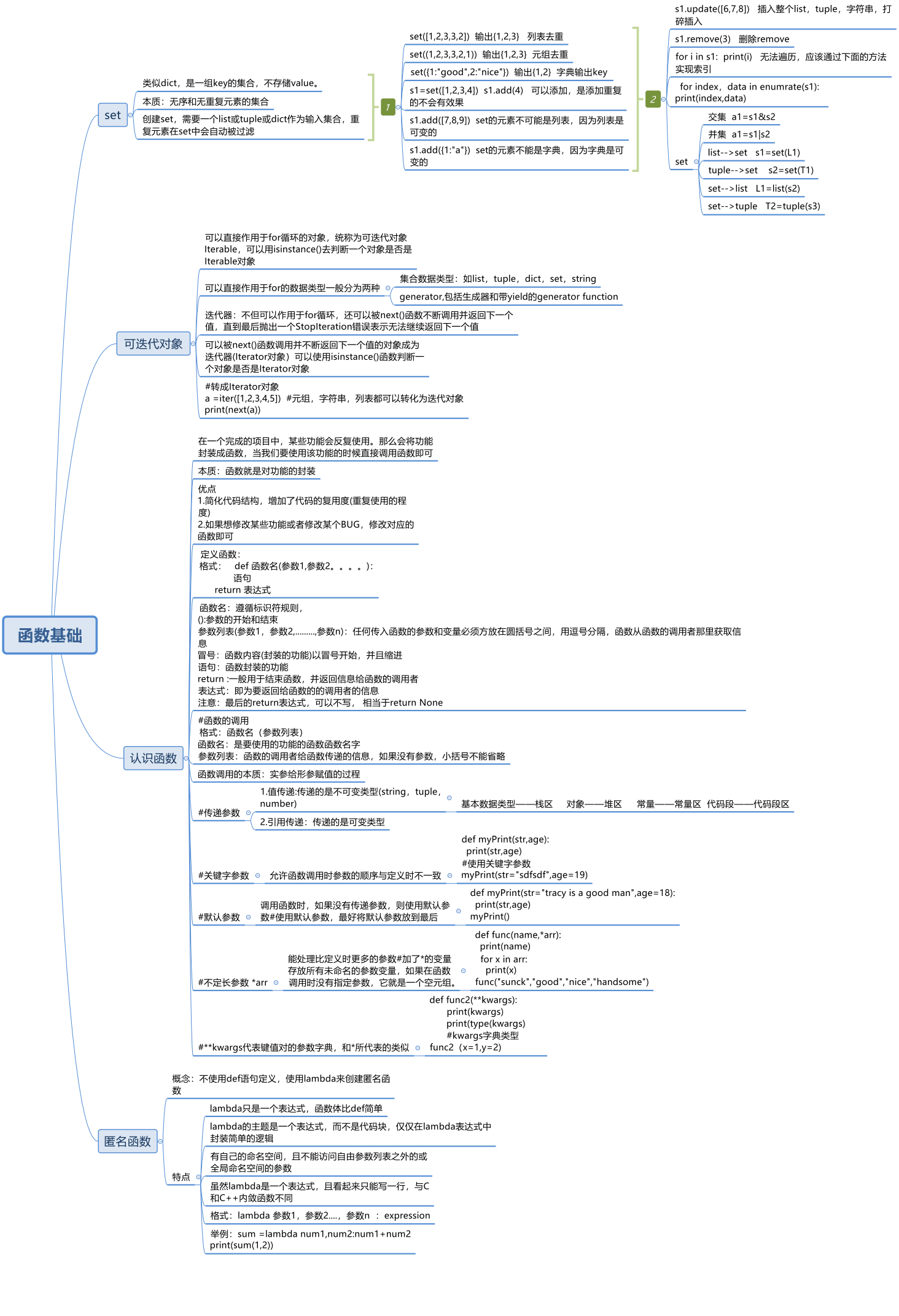
函数基础
set
类似dict，是一组key的集合，不存储value。
本质：无序和无重复元素的集合
创建set，需要一个list或tuple或dict作为输入集合，重复元素在set中会自动被过滤
1
set([1,2,3,3,2])输出{1,2,3} 列表去重
set((1,2,3,3,2,1))输出{1,2,3} 元组去重
set({1:"good",2:"nice"})输出{1,2} 字典输出key
s1=set([1,2,3,4]) s1.add(4) 可以添加，是添加重复的不会有效果
s1.add([7,8,9]) set的元素不可能是列表，因为列表是可变的
s1.add({1:"a"}) set的元素不能是字典，因为字典是可变的
2
s1.update([6,7,8]) 插入整个list，tuple，字符串，打碎插入
s1.remove(3) 删除remove
for i in s1：print(i) 无法遍历，应该通过下面的方法实现索引
for index，data in enumrate(s1):
print(index,data)
set
交集 a1=s1&s2
并集 a1=s1|s2
list-->set s1=set(L1)
tuple-->set s2=set(T1)
set-->list L1=list(s2)
set-->tuple T2=tuple(s3)
可迭代对象
可以直接作用于for循环的对象，统称为可迭代对象
Iterable，可以用isinstance()去判断一个对象是否是Iterable对象
可以直接作用于for的数据类型一般分为两种
集合数据类型：如list，tuple，dict，set，string
generator,包括生成器和带yield的generator function
迭代器：不但可以作用于for循环，还可以被next()函数不断调用并返回下一个值，直到最后抛出一个StopIteration错误表示无法继续返回下一个值
可以被next()函数调用并不断返回下一个值的对象成为迭代器(Iterator对象）可以使用isinstance()函数判断一个对象是否是Iterator对象
#转成Iterator对象
a =iter([1,2,3,4,5]) #元组，字符串，列表都可以转化为迭代对象
print(next(a))
认识函数
在一个完成的项目中，某些功能会反复使用。那么会将功能封装成函数，当我们要使用该功能的时候直接调用函数即可
本质：函数就是对功能的封装
优点
1.简化代码结构，增加了代码的复用度(重复使用的程度)
2.如果想修改某些功能或者修改某个BUG，修改对应的函数即可
定义函数：
格式： def 函数名(参数1,参数2。。。。):
语句
return 表达式
函数名：遵循标识符规则，
():参数的开始和结束
参数列表(参数1，参数2,.....,参数n)：任何传入函数的参数和变量必须方放在圆括号之间，用逗号分隔，函数从函数的调用者那里获取信息
冒号：函数内容(封装的功能)以冒号开始，并且缩进
语句：函数封装的功能
return :一般用于结束函数，并返回信息给函数的调用者
表达式：即为要返回给函数的的调用者的信息
注意：最后的return表达式，可以不写， 相当于return None
#函数的调用
格式：函数名（参数列表）
函数名：是要使用的功能的函数函数名字
参数列表：函数的调用者给函数传递的信息，如果没有参数，小括号不能省略
函数调用的本质：实参给形参赋值的过程
#传递参数
1.值传递:传递的是不可变类型(string，tuple，number)
基本数据类型——栈区 对象——堆区 常量——常量区 代码段——代码段区
2.引用传递：传递的是可变类型
#关键字参数
允许函数调用时参数的顺序与定义时不一致
def myPrint(str,age):
print(str,age)
#使用关键字参数
myPrint(str="sdfsdf",age=19)
#默认参数
调用函数时，如果没有传递参数，则使用默认参数#使用默认参数，最好将默认参数放到最后
def myPrint(str="tracy is a good man",age=18):
print(str,age)
myPrint()
#不定长参数 *arr
能处理比定义时更多的参数#加了*的变量存放所有未命名的参数变量，如果在函数调用时没有指定参数，它就是一个空元组。
def func(name,*arr):
print(name)
for x in arr:
print(x)
func("sunck","good","nice","handsome")
#**kwargs代表键值对的参数字典，和*所代表的类似
def func2(**kwargs):
print(kwargs)
print(type(kwargs)
#kwargs字典类型
func2（x=1,y=2)
匿名函数
概念：不使用def语句定义，使用lambda来创建匿名函数
特点
lambda只是一个表达式，函数体比def简单
lambda的主题是一个表达式，而不是代码块，仅仅在lambda表达式中封装简单的逻辑
有自己的命名空间，且不能访问自由参数列表之外的或全局命名空间的参数
虽然lambda是一个表达式，且看起来只能写一行，与C和C++内敛函数不同
格式：lambda 参数1，参数2....，参数n ： expression
举例：sum =lambda num1,num2:num1+num2
print(sum(1,2))