墨学 FIT 9133 Programming Foundations in Python

--Searching and Sorting Algorithms 主讲：雪梨

Searching：在 list 当中，寻找一个特定的值(value)。

3	6	7	1	2	4	8

Where is 1? -> 3

两种不同的算法：

Linear 算法：从头到尾一次寻找

Worst case： n （n 是该 list 的长度）

Binary 算法：把 list 分开成两半，分别开始寻找

Worst case： log2（n）

Linear Search

```
def linear_search(the_list, target_item):

    # obtain the length of the_list
    n = len(the_list)

    for i in range(n):
        # if the target is found
        if the_list[i] == target_item:
            return True

    # search through the list
    # the target is not found
    return False
```

```
>>> char_list = ['p', 'y', 't', 'h', 'o', 'n']
>>> result = linear_search(char_list, 't')
>>> print(result)
>>>
>>> result = linear_search(char_list, 'T')
>>> print(result)
>>>
```

Binary Search

```
def binary_search(the_list, target_item):
    low = 0
    high = len(the_list)-1

    # repeatedly divide the list in half
    # as long as the target item is not found
    while low <= high:

        # find the mid position
        mid = (low + high) // 2

        if the_list[mid] == target_item:
            return True
        elif target_item < the_list[mid]:
            high = mid - 1 # search lower half
        else:
            low = mid + 1  # search upper half

    # the list cannot be further divided
    # the target is not found
    return False
```

Sorting 排序

1. Bubble sort

```
def bubble_sort(the_list):

    # obtain the length of the list
    n = len(the_list)

    # perform n-1 iterations
    for i in range(n-1, 0, -1):

        # for each iteration
        # move the next largest item to the end
        for j in range(i):

            # swap if two adjacent items are
            # out of order
            if the_list[j] > the_list[j+1]:
                temp = the_list[j]
                the_list[j] = the_list[j+1]
                the_list[j+1] = temp
```

2, Selection sort

```
def selection_sort(the_list):

    # obtain the length of the list
    n = len(the_list)

    # perform n-1 iterations
    for i in range(n-1):

        # assume item at index i as the smallest
        smallest = i

        # check if any other item is smaller
        for j in range(i+1, n):
            if the_list[j] < the_list[smallest]:
                # update the current smallest item
                smallest = j

        # place the current smallest item
        # in its correct position
        the_list[smallest], the_list[i] = \
            the_list[i], the_list[smallest]
```

2. insertion sort

```
def insertion_sort(the_list):

    # obtain the length of the list
    n = len(the_list)

    # begin with the first item in the list
    # assume as the only item in the sorted sublist
    for i in range(1, n):

        # indicate the current item to be positioned
        current = the_list[i]

        # find the correct position where the current
        # item should be placed in the sorted sublist
        pos = i
        while pos > 0 and the_list[pos-1] > current:
            # shift items in the sorted sublist
            # for those larger than the current item
            the_list[pos] = the_list[pos-1]
            pos -= 1

        # place the current item in its correct position
        the_list[pos] = current
```

墨学 FIT 9133 Programming Foundations in Python

--Input/Output File and Important Library　　主讲：雪梨

Input/Output File

Input file：把一个文件（本学期只关心于文本文件）放入 python script 中。放入之后，我们可以对这个文件进行操作。比如说读取 setting 文件，读取 data 文件。

代码：

```
open(file, mode)
```

注意：这里的 file 是文件的全路径名，如果 python 文件和读取目标文件在同一个文件夹里，则不需要全部路径。 mode 是指我们打开文件执行的 operation 是什么，比如说读写，只读等等。

Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	open for exclusive creation, failing if the file already exists
'a'	open for writing, appending to the end of the file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open a disk file for updating (reading and writing)

```
variableName = open(file, mode)
```

这个文件中的内容就会传入这个 variable 中，然后这个 variable 就可以调用一些 function 对文件进行一些处理。

Function Example:

file = open("hello.txt", "r")

读取一行: file.readline()

读取所有行: file.readlines() 结果放入一个 list

```
1  file = open("hello.txt",'r')
2
3  print(file.readline(), end=" ")
4
5  print(file.readlines())
```

```
1
 ['2\n', '3\n', '4\n', '5\n', '6\n', '7\n', '8\n', '9']

Process finished with exit code 0
```

对于每一行可以进行操作:

for line in file.readlines():

```
1
2
3
4
5
6
7
8
9
Process finished with exit code 0
```

```
1  file = open("hello.txt",'r')
2
3  for line in file.readlines():
4      print(line, end="")
```

QUESTION? - 如果把 print (line, end="") 中的 end="" 去掉，结果有什么变化。

关闭 file，每次用完了这个 file 我们需要关闭它 (concurrency problem)

file.close()

Output file

输出一个文件:

改写文件 (overwrite) - open(fileName, "w")

在文件末尾加数据 (add at the end) - open(fileName, "a")

Example:

```
s = ""

for a in range(10):
    s += str(a)

savefile = open("saveFile.txt", 'w')
savefile.write(s)
```

Question: 如果当前路径下没有"saveFile.txt"这个文件，我们运行以上代码会报错吗?

How to import library?

import 包名 as 变量名

变量名.调用方法()

import 包名

包名.调用方法()

Important Library and Examples

Math

re (regular expression)

Numpy + Scipy

Matplotlib

Pandas

墨学 FIT 9133 Programming Foundations in Python

-- Important Library 主讲：雪梨

1. math library

常用来描述用符号无法表达出来的数学运算，或者一些常见的常数。

math.exp(x)	
math.log(x,base)	
math.pow(x,y)	
math.sin(x)	
math.sinh(x)	
math.pi	
math.e	

2. random library

常用来做随机值

random.seed(x) ->伪随机	
random.randint(a, b)	a <= N <= b
random.randrange(stop)	1: 0, 2: 0 或 1
random.randrange(start, stop[, step])	
random.random()	
choice(seq) shuffle(seq) sample(seq,k)	

3. Numpy library (import numpy as np)

常用来做矩阵-> matrix

一维: an_array = np.array([1,2,3])

二维:

an_array = np.array([[1,2],[3,4]])

an_array = np.array([[1,2,3],[4,5,6],[7,8,9]])

an_array = np.arange(1,10).reshape(3,3)

an_array = np.zeros(5)

an_array = np.ones(5)

an_array = np.zeros((3,2))

an_array = np.arange(10)

4. Matplotlib 作图用 (import matplotlib.pyplot as plt)

(refer assignment)

5. Pandas（处理数据用）

```
import pandas as pd

# 读取文件    pd.read_csv -> csv 和 txt

csv = pd.read_excel('PopulationEstimates.xls')



#将 dataframe 里面的每一条 row 转化成 arrary

csv.as_matirx()



#header

header = csv.as_matrix()[1]

csv.columns = header

#手动输入 header:

# header = ['string1', 'string2', 'string3' ....]

# csv.columns = header

#扔掉不需要的行数

csv.drop(index=csv.index[:2], inplace=True) #inplace=True 我确定要删掉  False

只显示结果但是不会修改原数据



#保存文件

csv.to_csv("newfile.csv")



#组建新的 dataframe
```

```
col1 = csv['POP_ESTIMATE_2010']

newDF = pd.DataFrame()    # 声明新的 dataframe

newDF['POP_ESTIMATE_2010'] = col1

newDF['POP_ESTIMATE_2011'] = csv['POP_ESTIMATE_2011']
```

墨学 FIT 9133 Programming Foundations in Python

--Testing&Error&Exception　　主讲：雪梨

Testing:

Unit Test: 只测试程序中的某一小部分

Integration Test: 多个 unit test 组成一个 group， 测试这个 group

System Test: 测试整个程序 (functionality)

Acceptance Test: 程序发布之前的测试(business requirements)

Test Strategy: 如何去测试，测试那些东西， 测试的流程

Test Cases: 具体测试值

Valid(positive) cases: 正确的 (ex: input an positive integer: 1 2 3)

Invalid(negative) cases 错误的 (ex: input an positive integer: -1，'abc')

Boundary cases:徘徊在边缘的 (range 1 -100， 测试 1，100)

Python 中用来 debug 的工具： print， assert

Assert: assert (condition), "error message"

```
>>> i = 3
>>> assert i == 3, "i is not equal to 3"
>>> i = 2
>>> assert i == 3, "i is not equal to 3"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AssertionError: i is not equal to 3
```

当 i 等于 3 的时候， assert 就不会报错

当 i 不等于 3 的时候，assert 就会引发报错，AssertionError exception

Python 中的 unittest 包：

```
def product_func(first_arg, second_arg):
    result = first_arg * second_arg
    return result
```

这是我们想要测试的 function

```
import unittest
class TestForProduct(unittest.TestCase):
    def test_product(self):
        self.assertEqual(product_func(2, 4), 8)
```

引入 unittest 包

需要建一个新的 class， 传入值是一个 unittest.TestCase

这个测试 class 中的每一个 function，对应需要测试的 function，和测试的 case

```
suite = unittest.TestLoader().loadTestsFromTestCase(TestForProduct)
unittest.TextTestRunner().run(suite)
```

以上代码的意思是集成一个测试 class 为一个 suite。

Errors:

Syntax errors：代码格式写错了

Run-time errors：代码运行时产生的报错，比如 I/O（可以用 exception 避免）

Logic errors：：代码能运行完，可是不是我们想要的结果 (test case)

SyntaxError:

```
if a_number > 2
    print(a_number, "is greater than 2")
```

NameError:

```
a_number = random.random()
```

TypeError:

```
if a_number > 2:
    print(a_number + "is greater than 2")
```

ValueError:

```
sum_of_two = int('1') + int('b')
```

try：

这里的代码会产生 exception

可能抛出多个异常

except 什么 type error：

怎么处理这个 error

except 另外的一个 type error： -》如果不写 type error 则是接住所有异常

怎么处理这个 error

else: -》else 是 optional 的

some code should be executed if the try block does not raise any exception.

finally： -》finally 是 optional 的

无论有没有抛出异常，这里都要走一遍

两个例子：

```
try:
    num1 = int(input("Enter first number: "))
    num2 = int(input("Enter second number: "))
    result = num1 // num2
    print("Result of division:", result)
except ValueError:
    print("Invalid input value")
except ZeroDivisionError:
    print("Cannot divide by zero")
```

```
file_name = "another_input_file.txt"

try:
    file_handle = open(file_name, 'r')
except IOError:
    print("Cannot open", file_name)
except RuntimeError:
    print("A run-time error has occurred")
else:
    print(file_name, "has",
          len(file_handle.readlines()), "lines")
    file_handle.close()
finally:
    print("Exiting file reading")
```

墨学 FIT 9133 Programming Foundations in Python

-- Divide&Conquer Recursion　　主讲：雪梨

Divide&Conquer：

计算机中很重要的抽象概念，当一问题非常难于解决之时，我们把问题分成两半去分别求解。

Divide&Conquer + Recursion：

两半解决+循环：当一问题非常难于解决之时，我们把问题分成两半去分别求解，分成的两半再继续分解成两半，直至 base case。

Binary Search review

```
def binary_search(the_list, target_item):
    low = 0
    high = len(the_list)-1

    # repeatedly divide the list into two halves
    # as long as the target item is not found
    while low <= high:

        # find the mid position
        mid = (low + high) // 2

        if the_list[mid] == target_item:
            return True
        elif target_item < the_list[mid]:
            high = mid - 1 # search lower half
        else:
            low = mid + 1  # search upper half

    # the list cannot be further divided
    # the target is not found
    return False
```

Binary Recursion

```
def rec_binary_search(the_list, target_item):
    # repeatedly divide the list into two halves
    # as long as the target item is not found

    # the list cannot be further divided i.e. item is not found
    if len(the_list) == 0:
        return False
    else:
        # find the mid position
        mid = len(the_list) // 2

        # check if target item is equal to middle item
        if the_list[mid] == target_item:
            return True
        # check if target item is less than middle item
        # search lower half
        elif target_item < the_list[mid]:
            smaller_list = the_list[:mid]
            return rec_binary_search(smaller_list, target_item)
        # check if target item is greater than middle item
        # search upper half
        else:
            smaller_list = the_list[mid+1:]
            return rec_binary_search(smaller_list, target_item)
```

Rec
Fun
A fu
that ca
repe

Recursion 三大组成部分：

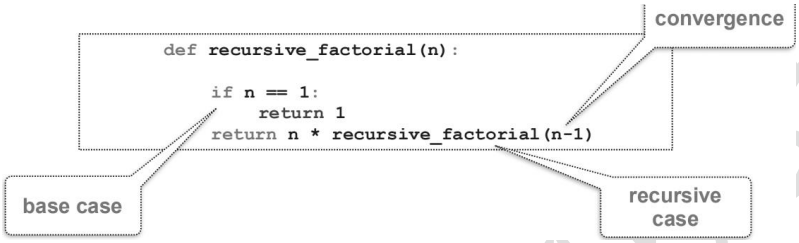
Base case：运行到这里停止

Convergence：必须包含 base case，并且会把问题缩小。

Recursive case：function 必须 call 自己

简单的例子：

```
def recursive_addition(a, b):
    if a == 0:
        return b
    return recursive_addition(a-1, b+1)
```



Merge Sort：

```
def merge_sort(the_list):
    # obtain the length of the list
    n = len(the_list)

    if n > 1: # check for base case
        # find the middle of the list
        mid = n // 2

        # based upon the middle index, two sublists are then created
        # from the 0 index until the mid-1 index
        left_sublist = the_list[:mid]
        # from the mid index until the n - 1 index
        right_sublist = the_list[mid:]
        print("Splitting: " + str(left_sublist) + " and " + str(right_sublist))

        # merge sort is called again on the two new created sublists
        merge_sort(left_sublist)
        merge_sort(right_sublist)

        # sort and merge
        print("Merging " + str(left_sublist) + " with " + str(right_sublist))
        i = 0 # index for left sublist
        j = 0 # index for right sublist
        k = 0 # index for main list


        while i < len(left_sublist) and j < len(right_sublist):
            if left_sublist[i] <= right_sublist[j]:
                the_list[k] = left_sublist[i]
                i += 1
            else:
                the_list[k] = right_sublist[j]
                j += 1
            k += 1

        # insert the remaining elements into main list
        while i < len(left_sublist):
            the_list[k] = left_sublist[i]
            i += 1
            k += 1

        while j < len(right_sublist):
            the_list[k] = right_sublist[j]
            j += 1
            k += 1
```

Quick Sort

```
def quick_sort(the_list):
    # pass the indices of first and last elements of the list
    first = 0
    last  = len(the_list) - 1
    quick_sort_aux(the_list, first, last)

def quick_sort_aux(the_list, first, last):
    # if it is not the base case
    if first < last:
        # find the partition point
        partition_point = partitioning(the_list, first, last)

        print("partition at index: ", partition_point)
        print(the_list)
        print ("after partitioning: " + str(the_list[first:partition_point]) +
               " and " + str(the_list[partition_point+1:last+1]))

        # call the quick sort function again on the new sublists
        quick_sort_aux(the_list, first, partition_point - 1)
        quick_sort_aux(the_list, partition_point + 1, last)
```

```
def partitioning(the_list, first, last):
    # take first element of the list is the pivot
    pivot_value = the_list[first]
    print("pivot: ", pivot_value)

    # these two indices will help us in locating the index point
    # where the list will be partitioned
    left_index = first + 1
    right_index = last

    complete = False

    while not complete:
        # start with the left index and keep on incrementing it
        # until a value greater than the pivot value is found
        while left_index <= right_index and
              the_list[left_index] <= pivot_value:
            left_index += 1

        # now look for element from the right of the list
        # which is smaller than the pivot value
        while right_index >= left_index and
              the_list[right_index] >= pivot_value:
            right_index -= 1

        # check whether left and right indices have crossed each other
        # if that is the case exit the while loop
        if right_index < left_index:
            complete = True
        else:
        # otherwise swap the two elements
            the_list[left_index], the_list[right_index]
                 = the_list[right_index], the_list[left_index]

    # swap the pivot element with the element of the right index
    the_list[first], the_list[right_index]
         = the_list[right_index], the_list[first]

    # return right index which is the partition point
    return right_index
```

墨学 FIT 9133 Programming Foundations in Python

-- Final Sample　　主讲：雪梨

Question1

Taking a list of tuples as argument, write a function in Python, **list_to_dictionary**, which converts the list of tuples into a dictionary and return the dictionary. Each tuple in the list will become an entry in the dictionary where the second element of each tuple serves as the key for each dictionary entry. Note that you should not use the built-in dict() function in your implementation.

```
def list_to_dictionary(tuple_list):
```

For example, given that **tuple_list = [(1, 'one'),(2, 'two'),(3, 'three')]**, the function should return a dictionary, **new_dict = {'one':1,'two':2,'three':3}**. If the given **tuple_list** is empty, the function should return an empty dictionary. (You can assume that the key in each of the tuples in the given set is unique.)

Input：list of tuple　　Output：dictionary

考点：collective data type 如何做转换

复习重点：理解所有的 collective data type 是什么样子的，list，set，dictionary，tuple

```
dic ={}
for t in tuple_list:
    dic[t[0]] = t[1]
return dic
```

Question2

Consider the partially defined Account class below, complete the implementation of the three methods:

1) __init__(),

2) deposit(),

3) withdraw().

```
class Account :
    def __init__ ( self, holder_name, acct_number, initial_balance) :
    def deposit ( self, deposit_fund ) :
    def withdraw ( self, withdraw_fund ) :
```

Note that for depositing, the amount to be deposited must be greater than 0. As for withdrawing, it should only be allowed if the remaining balance after withdrawal is not less than 0.

考点：class 如何定义

复习重点： 如何写 class 和 class 里面的 method

Question3

Write a Python function, which takes as argument a Python list of characters in upper- and lower-cases, **char_list**, to determine the frequency of each unique uppercase character in the list and returns the counts as a dictionary with each uppercase character as the key and its frequency as the corresponding value. Note that you should not use the built-in **count()** method in your implementation.

```
def find_frequency(char_list) :
```

For example, given that **char_list = ['a','A','b','C','c','b','D']**, the function should return a dictionary, **uppercase_dict = {'A':1,'C':1,'D':1}**. If the given char_list is empty, the function should return an empty dictionary.

Input： list of string　　Output： dictionary

考点： collective data type 如何做转换； string 的 buildin function

复习重点： 理解所有的 collective data type 是什么样子的， list，set，dictionary，tuple

String 的一些 build in function

```
dic ={}
for s in char_list:
    if (s >= 'A') and (s <= 'Z'):
        if s in dic.key():
            dic[s] += 1
        else:
            dic[s] = 1
return dic
```

Question4

Consider a **Queue** class which implements a queue abstract data type using an array-based structure with unbounded capacity, and with only the following methods defined:

```
class Queue:
    def __init__(self):
        self.the_queue = []
        self.count = 0
        self.front = 0
        self.rear = -1
    def __len__(self):
        return self.count
    def is_empty(self):
        return len(self) == 0
```

(a) Define the method that appends a new item to the rear (end) of the queue.

考点： queue 的 implementation

复习重点： 理解所有的ADT 以及其 code

```
def append(self, item):
    self.the_queue.append(item)
    self.rear += 1
    self.count += 1
```

(b) Define the method that *serves* and returns the item which is at the front of the queue.

```
def serve(self):
    assert not self.is_empty(), "Cannot serve an empty queue"
    item = self.the_queue[self.front]
    self.front += 1
    self.count -= 1
    return item
```

Question5

Consider a Stack class which implements a stack abstract data type using an array-based structure with unbounded capacity, and defines the following methods:

```
class Stack:
    __init__()
    push(item)
    pop()
    is_empty()
```

Write a Python function (outside this class), which takes an ordered list of strings as an argument. The function returns a new list with the same elements from the original list but in the reverse order.

```
def reverse_function(str_list):
```

For example, given **str_list** that holds a sequence of strings **['abc','def','ghi']**, the function should return a new list **new_list** with the sequence of strings in the reverse order as **['ghi', 'def','abc']**.

Note that your implementation should use the Stack methods defined above; however you do not need to know how the **Stack** class was implemented.

考点： stack 的 implementation

复习重点： 理解所有的 ADT 以及其 code

```
s = Stack()
for str in str_list:
    s.push(str)

new_list = []
while s.is_empty() == False:
    new_list.append(s.pop())
```

Question6

Write a function, **selection_sort**, in Python which takes as argument a Python list of strings, **str_list**; and sorts this list into descending order using selection sort.

考点： selection_sort 的 implementation

复习重点： 理解所有的算法以及其 code

Question7

Given the following Python code for the linear search algorithm, what is the output for each of the following search? You should also describe the sequence of search steps (i.e. how many elements are compared before the search terminates).

```
def linear_search(the_list, target_item):
    # obtain the length of the_list
    n = len(the_list)
    for i in range(n):
        # if the target is found
        if the_list[i] == target_item:
            return True
    # the target is not found
    return False
```

(a) Given **the_list = ['c','e','a','f','b','h']** and **target_item = 'b'**.

考点： linear search 的 implementation

复习重点： 理解所有的算法以及其 code

True 5 comparisons

(b) Given **the_list = ['c','e','a','f','b','h']** and **target_item = 'd'**.

False 6 comparisons

Question8

Given the following recursive function, **some_func**, answer each of the following questions.

```
def some_func(a_list):
    if len(a_list) == 0:
        return 0
    else:
        return 1 + some_func(a_list[1:])
```

(a) What does the above recursive function do? Describe.

考点： Recursion 基础概念

如果 some_func ([1,2,3,4,5]) 结果 5

Count how many element in this list

(b) Identify the **base case**, the **recursive case**, and the **convergence** component in the given recursive function.

Base case:

Recursive case:

Convergence:

Base case: if len(a_list) == 0: return 0

Recursive case: 最后一行的 some_func ( )

Convergence: a_list[1:]

(c) Suggest two examples of **a_list**, where one will return the result as a zero value (0) and one with a non-zero value.

some_func([])

some_func([1,2,3,4,5])