CP1404 Prac 4

list\_comprehensions.py:

*"""  
CP1404/CP5632 Practical  
List comprehensions  
"""*names = ["Bob", "Angel", "Jimi", "Alan", "Ada"]  
full\_names = ["Bob Martin", "Angel Harlem", "Jimi Hendrix", "Alan Turing",  
 "Ada Lovelace"]  
  
# for loop that creates a new list containing the first letter of each name  
first\_initials = []  
for name in names:  
 first\_initials.append(name[0])  
print(first\_initials)  
  
# list comprehension that does the same thing as the loop above  
first\_initials = [name[0] for name in names]  
print(first\_initials)  
  
# list comprehension that creates a list containing the initials  
# splits each name and adds the first letters of each part to a string  
full\_initials = [name.split()[0][0] + name.split()[1][0] for name in  
 full\_names]  
print(full\_initials)  
  
# one more example, using filtering to select only the names that start with A  
a\_names = [name for name in names if name.startswith('A')]  
print(a\_names)  
  
# *TODO: use a list comprehension to create a list of all of the full\_names*# in lowercase format  
# lowercase\_full\_names =  
lowercase\_full\_names = [name.lower() for name in full\_names]  
print(lowercase\_full\_names)  
  
  
almost\_numbers = ['0', '10', '21', '3', '-7', '88', '9']  
# *TODO: use a list comprehension to create a list of integers*# from the above list of strings  
# numbers =  
numbers = [int(almost\_numbers) for almost\_numbers in almost\_numbers]  
print(numbers)  
  
# *TODO: use a list comprehension to create a list of only the numbers that are*# greater than 9 from the numbers (not strings) you just created  
numbers\_greater\_than\_9 = [number for number in numbers if number > 9]  
print(numbers\_greater\_than\_9)

list\_exercises.py:

def main():  
 usernames = ['jimbo', 'giltson98', 'derekf', 'WhatSup', 'NicolEye', 'swei45', 'BaseInterpreterInterface',  
 'BaseStdIn', 'Command', 'ExecState', 'InteractiveConsole', 'InterpreterInterface', 'StartServer',  
 'bob']  
  
 numbers = []  
 name = str(input("What is your name: "))  
 if name in usernames:  
 print("Access granted")  
 else:  
 print("Access denied.")  
  
 for number in range(1,6):  
 number\_1 = int(input("Number {} :".format(number)))  
 while number\_1 < 0:  
 number\_1 = int(input("Number {} :".format(number)))  
  
 numbers.append(number\_1)  
  
 print("The first number is {}".format(numbers[0]))  
 print("The last number is {}".format(numbers[-1]))  
 print("The smallest number is {}".format(min(numbers)))  
 print("The largest number is {}".format(max(numbers)))  
 print("The average of the numbers is {}".format(sum(numbers)/5))  
  
  
  
  
  
  
  
  
main()

lists\_warmup.py:

numbers = ["10", 1, 4, 1, 5, 9, 1]  
  
# Answers:  
# numbers[0] : 3  
# numbers[-1] : 2  
# numbers[3] : 1  
# numbers[:-1] : 3,1,4,1,5,9,2  
# numbers[3:4] : 1  
# 5 in numbers : True  
# 7 in numbers : False  
# "3" in numbers : False  
# numbers + [6, 5, 3] : 3,1,4,1,5,9,2,6,5,3  
  
print(numbers[2:])  
print(3 in numbers)

Memberwise.py:

def main():  
  
 print(add\_memberwise([1, 2, 3], [1, 2, 3, 4]))  
  
  
  
def add\_memberwise(l1, l2):  
 a = l1[0] + l2[0]  
 b = l1[1] + l2[1]  
 c = l1[2] + l2[2]  
 d = 0 + l2[3]  
  
 return a, b, c, d  
  
  
  
  
  
  
  
  
  
  
main()

quick\_picks.py:

import random  
  
  
def main():  
  
 row = int(input("How many quick picks?"))  
 for i in range(row):  
 CONSTANTS = []  
 for a in range(6):  
 random\_num = random.randint(1, 45)  
 CONSTANTS.append(random\_num)  
 CONSTANTS.sort()  
  
 print(" ".join("{}".format(number) for number in CONSTANTS))  
  
  
main()

repeat.py:

def main():  
 str\_list = []  
 new\_list = []  
  
 strings = str(input("Enter your string:"))  
 str\_list.append(strings)  
 print(str\_list)  
 for i in str\_list:  
 if i not in new\_list:  
 new\_list.append(i)  
 else:  
 print(i, end=" ")  
  
  
main()

scores.py:

*"""  
CP1404/CP5632 Practical  
Debugging exercise: almost-working version of a CSV scores file program.  
The scores.csv file stores scores for each subject for 10 people.  
This code reads the lines into lists, saves the first line as a list of subject codes and  
converts the rest of the lines from a list of strings into a list of numbers,  
which it then prints with the maximum value for that subject.  
Nice. Except, it’s broken! It reads the lists per user not per subject so the results are incorrect.  
Use the debugger to follow what it's doing... then fix it.  
"""*def main():  
 *"""Read and display student scores from scores file."""* scores\_file = open("scores.csv")  
 scores\_data = scores\_file.readlines()  
 print(scores\_data)  
 subjects = scores\_data[0].strip().split(",")  
 score\_values = []  
 for score\_line in scores\_data[1:]:  
 score\_strings = score\_line.strip().split(",")  
 score\_numbers = [int(value) for value in score\_strings]  
 score\_values.append(score\_numbers)  
 scores\_file.close()  
 for i in range(len(subjects)):  
 print(subjects[i], "Scores:")  
 for score in score\_values[i]:  
 print(score)  
 print("Max:", max(score\_values[i]))  
 print()  
  
  
main()

sorting.py:

*"""  
CP1404/CP5632 Practical  
Demo of sorting lists of 'compound' values  
"""*from operator import itemgetter  
  
names\_with\_ages = [['Derek', 7], ['Carrie', 8], ['Bob', 6], ['Aaron', 9]]  
  
# This sorts by the first value (name) in the element  
names\_with\_ages.sort()  
print(names\_with\_ages)  
  
# This tells sort() to use the second (index 1, age) value in the element  
names\_with\_ages.sort(key=itemgetter(1))  
print(names\_with\_ages)  
  
data = [['Derek', 7], ['Carrie', 7], ['Bob', 6], ['Aaron', 6]]  
# This tells sort() to use the second (index 1) value...  
# then the first (index 0) in the element  
data.sort(key=itemgetter(1, 0))  
print(data)  
  
# The following items are in the form: id, first name, last name, age  
# *TODO: sort the following list of tuples by last name then first name*items = [('123', 'Derek', 'Smith', 7), ('124', 'Carrie', 'Brown', 7),  
 ('125', 'Bob', 'Smith', 6), ('126', 'Aaron', 'Hewitt', 6)]  
# ??  
items.sort(key=itemgetter(1, 0))  
print(items)

subject\_reader.py:

*"""  
CP1404/CP5632 Practical  
Data file -> lists program  
"""*FILENAME = "subject\_data.txt"  
  
  
def main():  
 data = get\_data()  
 print(data)  
  
 display\_details(data)  
  
def get\_data():  
 *"""Read data from file formatted like: subject,lecturer,number of students."""* input\_file = open(FILENAME)  
 data = [] # creat a list called data  
 for line in input\_file:  
 line = line.strip() # Remove the \n  
 parts = line.split(',') # Separate the data into its parts  
 parts[2] = int(parts[2]) # Make the number an integer (ignore PyCharm's warning)  
 data.append(parts) # add list to data  
  
 input\_file.close()  
 return data  
  
  
def display\_details(data):  
 *"""Display subject details"""* for subject\_details in data:  
 print("{} is taught by {} and has {} students. ".format(\*subject\_details))  
  
  
main()

total\_income.py:

*"""  
CP1404/CP5632 Practical  
Starter code for cumulative total income program  
"""*def main():  
 *"""Display income report for incomes over a given number of months."""* incomes = []  
 num\_of\_months = int(input("How many months? "))  
  
 for month in range(1, num\_of\_months + 1):  
 income = float(input("Enter income for month {}: ".format(month)))  
 incomes.append(income)  
  
 enumerate(incomes, num\_of\_months)  
  
  
def enumerate(incomes, num\_of\_months):  
 *"""This function will print the monthly income and total report."""* print("\nIncome Report\n-------------")  
 total = 0  
 for month in range(1, num\_of\_months + 1):  
 income = incomes[month - 1]  
 total += income  
 print("Month {:2} - Income: ${:10.2f} Total: ${:10.2f}".format(month, income, total))  
  
  
main()