**Python Development**

Python Exercises

© FDM Group Ltd 2020. All Rights Reserved.

Any unauthorised reproduction or distribution in part  
or in whole will constitute an infringement of copyright.

**Table of Contents**

Overview 3

01 Introduction 3

04 Syntax and Operators 4

05 Control Structures 5

06 Lists and Tuples 6

07 Set and Dict 7

08 A) Data Storage 1 - File IO 7

08 E) Data Storage 5 - SQLite.pptx 7

09 Functions 7

10 Classes 8

01 Introduction 8

# Overview

The following are the exercises from the PowerPoint.

# 01 Introduction

1. Use text/ASCII art to print out your name.

**02 PEP 8**

None

**03 Data Types**

1. Write a program that will get the first name and last name of the user

and return the first name in uppercase and capitalized the first letter of the last name of the user. (You can also add a message).

1. Write a program that will get the name of the user and 2 numerical values,

then return the name in uppercase and the **sum** of the numbers with a message.

1. Write a program that will get the following information from the user:

* Full name
* Date of birth

Return a greeting to the user, with the following information:

* User’s current age
* User’s age on their next birthday
* Number of days to the next birthday
* If year of birth is a leap year

# 04 Syntax and Operators

1. Write a program to ask the user to type in their Username and Password.

Then display a welcome message for the user.

1. Update your program to ask the user to type in their Username and Password.

Check if the Username is one of the following (display appropriate message):

allUserList = ['UK\_User1', 'US\_User2', 'Africa\_User3',   
 'Canada\_User4' , 'Australia\_User6']

1. Update your program to ask the user to type in their Username and Password. Display error message or a welcome message if the password has:

* a number of characters greater than 8
* at least one numerical character
* at least one uppercase
* at least one lowercase
* at least one non alphanumerical character

# 05 Control Structures

1. Invite user to enter a number:

* If the user number is a multiple of 3: Display "**FIZZ**"
* if it is a multiple of 5: Display "**BUZZ**"
* and if it is a multiple of both 3 and 5: Display "**FIZZ BUZZ**“
* \* If none of the above, display the user number.

1. Write a program which will find all the numbers which are divisible by 7 but are not a multiple of 5, between 100 and 203 (both included).

The numbers obtained should be printed in a comma-separated sequence on a single line.

1. Display a randomly generated number between 1 and 13.

* Show the number
* Invite user to guess if next card/number is higher (H) or lower (L)
* Then generated another random number between 1 - 13
* Player loses if they guess wrong. Player wins if they guess right
* Show the number and the appropriate message
* **Loop: the game should default to replay but also offer an exit condition**

1. Modify your code from 1) but instead do the following:

* **Generate a list of every number between 1 and 100**
* if a number is a multiple of 3 it is replaced with "**FIZZ**"
* if it is a multiple of 5 it is replaced by "**BUZZ**"
* if it is a multiple of both 3 and 5, it is replaced by " **FIZZ BUZZ** "

Print out the list of results with a space between each item.

1. Modify the code created for 3) to **use exception handling to validate the user inputs.**

If invalid responses are given the code should:

* display error message to the user
* not crash the program
* repeat the prompt till a valid response is received

# 06 Lists and Tuples

1. Write a program to:

* store a list of first name, last name and favourite colour for everyone in the class using a nested tuple
* display all the favourite colours
* prompt for a name in the list to display the unpacked tuple (each value on separate line)

1. Create a list of your favourite modes of transportation, for example motorcycles.

Use your list to print a series of statements about these items, such as “I dream to ride a Honda motorcycle.”

1. Changing an ingredient’s list:

* create a list of ingredients for a recipe
* you need to remove one ingredient that is not adequate
* add a new ingredient to the list
* show the changes in the list after each operation

1. Create the square2020 list, to hold the square of numbers 1 to 2020 (using - List Comprehension). Display the list.
2. Using the squares list created in 4):

* display the list
* display the square of numbers 1 to 10
* display the first and the last item in the list
* display the square of 5, 6 and 7

1. Generate a long list of random integers.

* find the biggest number in the list
* use other aggregate functions to describe the list as well

1. Use the favourite modes of transportation list created in 2):

* display the list
* prompt the user to select one of the modes shown or add something new
* search and display a message if the mode is in the list  
  if not add it to the list

1. Using this Customerinfo:

Customerinfo = [ [100, "Jill", 1001, 2001],

                 [109, "Jack", 1002, 2002],

                 [119, "Frances", 1007, 2007],

                 [110, "Matt", 1008, 2008] ]

* create another list of the customer’s names
* display the list in descending order

# 07 Set and Dict

1. Write a program to generate a dictionary that contains (i, i\*i) for each integral number between 1 and n (both included).

* a given integral number ‘n’ is a constant in the program
* use it to generate the dictionary
* print the dictionary

Suppose the following input is supplied to the program: 8

Then, the output should be:

{1:1, 2:4, 3:9, 4:16, 5:25, 6:36, 7:49, 8:64}

1. Use the zip() function to join the supplied lists as a dictionary of exam results and then find the key that has the maximum score.  
     
   Supplied lists:

keys = ["Adam", "Betty", "Cathy", "Donald"]

results = [75,95,80,80]

1. Create a program called shoppingCart:
2. Define a function computeBill that takes arguments for groceries and prices as input and returns the item total
3. In the program create a variable total with an initial value of zero
4. For each item in the food list, add the price of that item to total
5. Finally, return the total
6. Ignore whether or not the item you're billing for is in stock

Supplied data:

groceries = ["banana","orange","apple"]

stock = {

"banana": 16,

"apple": 0,

"orange": 32,

"pear": 15

}

prices = {

"banana": 4,

"apple": 6,

"orange": 7,

"pear": 7

}

1. Make the following changes to your shoppingCart program:  
     
   While you loop through each item of food, only add the price of the item to total if the stock count is greater than zero.  
     
   If the item is in stock and after you add the price to the total, subtract one from the item's stock count.
2. **Unique words-** Consider and use the best Python data collection object to implement the following with minimum coding:

Your program should print out every word from a string in alphabetical order without duplicates.

Use a long string of text scrapped from the internet and save it to a variable in your program.

Use data cleaning methods to remove differences in the letter case and any punctuation.

1. Create a dictionary, in which the key will the letters in the string ‘abracadabra’, and the value will be the number of occurrences of each letter.
2. Create an empty dictionary e.g.

lettersCount = dict()

1. Using a for loop iterate, through the string ‘abracadabra’
2. Check to see whether each letter is already in your dictionary using in

if char in lettersCount

1. Increment the count by 1, and add it back to the dictionary
2. Print out the dictionary, which will show you all of the letters and the counts

# 08 A) Data Storage 1 - File IO

1. Modify the unique words code to use a text file as the source of the string
2. create a loop that prompts for a text file
3. use error handling to get a valid file
4. display a list of unique words in that file before prompting for another file
5. test the program with larger text files
6. Improve the unique words program again to remove **stopwords**

# 08 E) Data Storage 5 - SQLite.pptx

1. Create a sqlite3 database to manage an automated car park

On entering the car park, the car’s number plate is automatically scanned and recorded.

When the driver wants to leave, they enter their number plate into the pay machine.

The payment machine tells them how much they need to pay:

* up to 1 hour £2.50
* up to 2 hours £4.00
* up to 3 hours £6.00
* over 3 hours £15

On leaving, number plate is automatically scanned again. Leaving without paying results in an automatic fine. Generate the list of cars that should be fined.

The database table should have the following columns:

car\_rego\_number TEXT,

entry\_time   date and time,

exit\_time   date and time,

amount\_due   INTEGER

amount\_paid   Yes / No

# 09 Functions

1. Write a function that will greet a person if the person’s name is given or display a generic greeting if no name is given.
2. Add other calculator functions (- , / , \* etc.)

Add get input from user and print the result on the screen

1. Create a function that will get username and password from the user.

You can then encrypt and decrypt the password.

NOTE: You will need to install additional module:

pip install cryptography

1. Use function(s) to create a menu system for your selected project.

You can create/call placeholder functions of simply print out a message to verify user selection.

# 10 Classes

1. Create a CurrencyManagement class that will manage all database connectivity.

Add the following 3 methods:

def createTable\_CustomerInfo(self)

To create a database table: CustomerInfoManager

CREATE TABLE CustomerInfoManager (

Created\_Date  TEXT,

username  TEXT,

userRole  TEXT,

Password  TEXT,

encryptionKey  BLOB )

def create\_AddNew\_Customer(self, user\_name, Pword,   
 ennKey , user\_Role=None)

To add specified data to the database

def validate\_Customer\_Account(self, user\_name, Pword )

To validate the username and password

1. Create/Update a CurrencyManagement class with CRUD functionalities that can be called from another application using a menu option and will manage all database connectivity.

NOTE: CRUD – Create, Read, Update, Delete

1. Create a new class in your project: stringparsing

Create a module in the package with 2 functions:

password\_validation()

username\_validation()

Optionally a third function named:

email\_validation()

Create a main program outside of your class that will use the module’s functions.