The exercises are designed for students to finish in an individual capacity. The exercises are not designed to be completed in tutorial sessions but rather to give you some tasks and a starting point to continue and complete on your own.

Lab Execises

1. **Environment Setup** Please follow instructions in `FIT3031-Lab1-Environment-Setup.pdf` to install VirtualBox for your Windows or Mac machine and finish the environment setup for the provided virtual machine instance.

2. **Using pip to install packages** pip is the most popular standard package-management system used to install and manage software packages written in Python. In this part, we will install Python Cryptography Toolkit (**Pycrypto**), a Python package where a set of secure hash functions and encryption algorithms were implemented for developers. You can use one of the workstations in GNS3 project for this lab. To get it installed, right click on one of the workstations and select `console`, please run the following two commands separately.

   ```
   apt update
   apt install python3-pip
   pip3 install pycryptodome
   ```

   The first line is to install `pip` for your system, the second line is to use `pip` install `pycryptodome` where `pip3` is to specify this installation is for Python 3.

3. **Introduction to Python** In this section, we will provide a quick introduction to basic python programming. The objective of this part is to provide a starting point for students without any prior experiences in Python programming. It is well-known that Python 2 and Python 3 are both widely employed by the coding community. However, we choose Python 3 for the following tutorials because not only python 3 provides much more awesome features than Python 2 does but also Python 2 has been officially abandoned. As a high-level, interpreted and general-purpose dynamic programming language, Python focuses on solving real-world problems in a easy and readable way. With Python, **indentation** is **mandated** to structure its logic blocks instead of using braces like it in other languages such as Java. In this part, we will go through how basic elements in Python language and how they are used. Now please open the terminal and type `python3` followed by pressing **Enter** to enter into interactive Python console. After that, please finish the following example coding tasks on your own. Feel free to turn to your tutor if you have any questions.

   (a) **Arithmetic Operators**

   ```
   >>>x = 10  # define a variable
   >>>print(x)       # Prints the value of x
   10
   >>>print(type(x)) # Prints type of value
   <type 'int'>
   >>>x + 1   # Addition
   11
   >>>x - 1   # Subtraction
   9
   >>>x * 2   # Multiplication
   20
   >>>x ** 2  # Exponentiation
   100
   >>>x += 1  # Argumented assignment
   >>>print(x)
   11
   >>>y = 0.5
   >>>print(type(y))
   <class 'float'>
   ```

   (b) **Booleans**

```
>>>T = True
>>>F = False
>>>print(type(T))
<class 'bool'>
>>>print(T and F) # AND
False
>>>print(T or F)  # OR
True
>>>print(not T)   # NOT
False
```

(c) **Strings:** Python provides various operations for strings

```
>>>first_word = 'Hello'    # String literals can use single quotes
>>>second_word = "FIT5037" # or double quotes
>>>print(first_word)
Hello
>>>print(len(first_word))  # String length;
5
>>>s = first_word + ' ' + second_word  # String concatenation
>>>print(s)
Hello FIT5037
>>>print(s.upper())        # Convert a string to uppercase
HELLO FIT5037
>>>print(s.lower())        # Convert a string to lowercase
hello fit5037
>>>print(s.replace('FIT', 'COMP'))  # Replace all instances of
Hello COMP5037                      # one substring with another;
```

(d) **Containers**

There are several built-in container types in Python: lists, dictionaries, sets, and tuples.

Lists

A list is the Python equivalent of an array, but can be re-sized and can contain different types of value. It worth pointing out that there is not type for a variable in Python, but there is a type for a value in a variable. This is different from C++/Java where a variable is restricted by a certain type of value once its type is declared explicitly.

```
>>>x = [1, 2, 3]     # Create a list
>>>print(x[2])
3
>>>print(x[-1])      # Negative indices
3
>>>x[0] = True      # Lists can contain elements of different types
>>>print(x)
[True, 2, 3]
>>>x.append('FIT5037')  # Add a new element to the end of the list
>>>print(x)
[True, 2, 3, 4, 'FIT5037']
>>>print(x.pop())  # Remove and return the last element of the list
'FIT5037'
```

Loops

```
>>>colors = ['red', 'blue', 'yellow']
>>>for c in colors:
>>>    print(c)
```

Dictionaries

Dictionaries are to store (key, value) pairs which is similar to a Map/map in Java or C++ STL.

```
>>>d = {'Darcy': 'D', 'Elliot': 'Pass'}  # Create a dictionary
>>>print(d['Darcy'])  # Get an entry from a dictionary;
'D'
>>>print('Tom' in d)  # If a dictionary has a given key
False
>>>d['Tom'] = 'HD'    # Add an entry in a dictionary
>>>print(d.get('Jim', 'N/A'))  # Get an element with a default;
"N/A"
>>>del d['Tom']      # Remove an element from a dictionary
```

Sets

A set is an unordered collection of distinct elements.

```
>>>students = {'Darcy', 'Elliot', 'Tom'}
>>>print('Tom' in students)    # Check if an element is in a set;
True
>>>students.add('Jim')        # Add an element to a set
>>>print(len(students))        # Number of elements in a set;
4
>>>students.remove('Jim')      # Remove an element from a set
```

Tuples

A tuple is an (**immutable**) ordered list of values. The the most important difference between tuples and lists is that tuples can be used as keys in dictionaries and as elements of sets while lists cannot.

```
>>>d = {(x, x+1): x for x in range(5)} #Tuple keys for a dictionary
>>>t = (1, 2)          # Create a tuple
>>>print(d[t])
1
```

(e) **Importing Modules**

Other Python source files can be used as a module via an import statement in another Python script. The following statements demonstrate how to import a module from Pycrypto to compute a message digest for the string "FIT5037". If you have no idea what message digest means, do not feel worried and just finish this example as we will explain afterwards.

```
>>> from Crypto.Hash import SHA256
>>> hash = SHA256.new()
>>> hash.update(b'FIT5037')
>>> hash.digest()
b'\xfb4\x19\xe7\x0b/\x07r\x18\x82dB>p\x9d\x1c\x0eWW\xcct\xaf\xb9]\xca7\xc9\
    xfer\x8d\xe27'
```

4. **File I/O, Functions & Classes in Python**

(a) **File I/O**

Python has great support for File I/O. One of the most commonly used is the built-in module **open()**. The function returns a file object (or called handle). Please read the following common usages of Python file I/O.

```
f = open("data.txt")       # open a file with read mode
f = open("data.txt",'w')   # open a file with write mode
f = open("data.txt",'r+b') # read and write in binary mode
f.read() # read entire file content into memory
f.readline()# read line by line
f.write()# write to a file
f.writeline() # write a line to a file
```

(b) **Functions**

Python functions are defined using the **def** keyword. Same as other languages, return statement is not a necessity in a function though the given example does include a return statement to close the function definition.

```
def add(a, b):
    return a+b
```

(c) **Classes**

Python is one of the most representative Object-oriented programming language. It allows developers to define classes with a very straightforward syntax while all the standard features of OOP are provided such as inheritance mechanism. See the following example of how a class is defined.

Please note that class members including data and methods are normally seen as **public** in C++/Java's terminology. This does not mean private members cannot be declared. Please refer to this to explore how to use private members.

```
class Student:
    # Constructor
    def __init__(self, student_id):
        self.student_id = student_id

    def set_name(name):
        self.name = name

    def set_email_addr(email_addr):
        self.email_addr = email_addr

s = Student(12345678)  # Construct an instance
g.set_name('Jordan Thompson')
g.set_email_addr('jordan.thompson@monash.edu')
```

5. **Coding Challenge**
   You may have noticed we did not use console code in task 4. In most cases, we write python code into scripts and run it through command line or IDEs. You may also find there is a file named merge `sort.py` in the Moodle. This is an example python script of incomplete implementation of the merge sort algorithm.

   Base on the above tasks, please modify this script to make it output the correct answer? To understand the code, you are expected to understand Functions, Classes and File I/O in Python. Please note that the merge sort algorithms has been already implemented for you.

   You can use any editors if you like. We strongly recommend PyCharm if you prefer coding via an IDE. PyCharm is the most popular and powerful IDE for Python development. More importantly, it is free for students. See further reading part to explore how to install PyCharm for Education for your machine.