Name_____ ( ) Class: Sec _____

SERANGOON SECONDARY SCHOOL

PRELIMINARY EXAMINATION 2019

SECONDARY 4 EXPRESS

COMPUTING **7155/02**

PAPER 2 **3 September 2019**

Name of Setter: Mr Adrial Tan **2 hour 30 minutes**

READ THESE INSTRUCTIONS FIRST

Write your name, class and index number on all the work you hand in.
Write in dark blue or black pen.
Answer **all** questions.

All tasks must be done in the computer laboratory.
Programs are to be written in Python. A Quick Reference Glossary for Python is provided.

The number of marks is given in brackets [ ] at the end of each question or part question.

Retrieve the template files from the **COMPEXAM** folder in the thumb drive.

Save your work inside the **COMPEXAM** folder in the thumb drive using the file names given in the question as and when necessary.

FOR EXAMINER'S USE
50

Quick Reference for Python

This quick reference shows some examples of the Python language constructs. The complete Python language is not limited to these examples.

1. Identifiers

When naming functions, variables and modules, the following rules must be observed:

- Names should begin with character 'a' - 'z' or 'A' - 'Z' or '_'
  and followed by alphanumeric characters or '_'
- Reserved words should not be used.
- User-defined identifiers are case sensitive.

2. Comments and Documentation Strings

```
# This is a comment

"""
   This is a documentation string over multiple
   lines
"""
```

3. Input/Output

```
print ("This is a string")

s = input ("Instructions to prompt for data entry.")
```

4. Import

import <module>

e.g. **import** math

5. Data Type

Data Type	Notes
int	integer
float	real number
bool	boolean
str	string (immutable)
list	series of values

6. Assignment

Assignment Statement	Notes
a = 1	integer
b = c	variable
d = "This is a string"	string
mylist = [1, 2, 3, 5, 5]	list or array

7. Arithmetic Operators

Operator	Notes
+ -	plus, subtract
* /	multiply, divide
%	remainder or modulus
**	exponential or power
//	quotient of floor division

8. Relational Operators

Operator	Notes
==	equality
!=	not equal to
> >=	greater than, greater than or equal to
< <=	less then, less than or equal to

9. Boolean Expression

Boolean Expression	Notes
a and b	logical and
a or b	logical or
not a	logical not

10. Iteration

while loop
while condition(s): <statement(s)>

for loop
for i **in** range(n): <statement(s)>
for record **in** records: <statement(s)>

11. Selection

Type 1	Type 2	Type 3
if condition(s): <statement(s)>	**if** condition(s): <statement(s)> **else**: <statement(s)>	**if** condition(s): <statement(s)> **elif** condition(s): <statement(s)> **else**: <statement(s)>

12. Built-in functions

(a) Basic functions

abs( )	chr( )	float( )	input( )	int( )
ord( )	print( )	range( )	round( )	str( )
format( )				

(b) Mathematical functions

ceil( )	exp( )	fabs( )	floor( )	log( )
max( )	min( )	pow( )	sqrt( )	trunc( )
format( )				

(c) String functions

endswith( )	find( )	isalnum( )	isalpha( )	isdigit( )
islower( )	isspace( )	isupper( )	len( )	lower( )
startswith( )	upper( )			

13. Reserved Words

Reserved words cannot be used as identifiers. They are part of the syntax of the language.

False	None	True	and	as
assert	break	class	continue	def
del	elif	else	except	finally
for	from	global	if	import
in	is	lambda	nonlocal	not
or	pass	raise	return	try
while	with	yield		

Task 1

Serangoon Public Bank launched a promotional Merdeka Deposit scheme recently. This promotional Merdeka Deposit scheme allows Singapore Citizens from the Merdeka Generation to deposit money with the bank for a fixed period of time with special interest rates. The scheme has the following features:

- Deposit amounts from $ 5,000.00 to $ 200,000.00.
- Interest rates vary according to deposit amount as follow:
    - $ 5,000 to $ 10,000, 1.00 % per annum
    - $ 10,001 to $ 30,000, 1.02 % per annum
    - $ 30,001 to $ 50,000, 1.05 % per annum
    - $ 50,001 to $ 75,000, 1.10 % per annum
    - $ 75,001 to $ 100,000, 1.20 % per annum
    - $ 100,001 to $ 150,000, 1.35 % per annum
    - $ 150,001 to $ 200,000, 1.50 % per annum
- Flexible deposit tenures ranging from 1 year to 6 years.
- Deposit accounts with tenures of 3 years and more are entitled to additional bonus interest as follow:
    - 3 years, additional 0.1 % per annum throughout entire tenure
    - 4 years, additional 0.2 % per annum throughout entire tenure
    - 5 years, additional 0.3 % per annum throughout entire tenure
    - 6 years, additional 0.4 % per annum throughout entire tenure
- Interest payments are compounded yearly and paid up in full only at the end of the deposit tenure.

The bank uses spreadsheet software to record the deposits made. You are required to finish setting up the spreadsheet to record the details.

Open the file **DEPOSIT.XLSX**. You will see the following data.

Save the file as **FIXEDDEPOSIT**_<Class>_<Index_Number>_<Your_Name>**.xlsx**

The first two records in the spreadsheet were calculated manually without using formulae and entered into the spreadsheet as the set up of the spreadsheet was not completed. Those two records have been verified to be correct.

	A	B	C	D	E	F	G
1	**Serangoon Public Bank**						
2	**Merdeka Deposit Accounts**						
3							
4	**Account Number**	**Amount Deposited ($)**	**Number of Years Of Deposit (years)**	**Base Merdeka Deposit Interest Rate (%)**	**Additional Bonus Interest (%)**	**Total Interest Rate (%)**	**Amount Received At Maturity ($)**
5	SPB-19001	$ 101,000.00	6	1.35%	0.4%	1.75%	$ (112,079.94)
6	SPB-19002	$ 7,000.00	2	1.00%	0.0%	1.00%	$ (7,140.70)
7	SPB-19003	$ 55,000.00	5				
8	SPB-19004	$ 61,000.00	3				
9	SPB-19005	$ 27,000.00	4				
10	SPB-19006	$ 96,000.00	1				
11	SPB-19007	$ 23,000.00	4				
12	SPB-19008	$ 112,000.00	1				
13	SPB-19009	$ 152,000.00	4				
14	SPB-19010	$ 102,000.00	2				
15	SPB-19011	$ 199,000.00	6				
16	SPB-19012	$ 8,000.00	4				
17	SPB-19013	$ 93,000.00	3				
18	SPB-19014	$ 87,000.00	6				
19	SPB-19015	$ 6,000.00	3				
20	SPB-19016	$ 35,000.00	2				
21	SPB-19017	$ 60,000.00	2				
22	SPB-19018	$ 144,000.00	2				
23	SPB-19019	$ 132,000.00	6				
24	SPB-19020	$ 51,000.00	1				
25							
26	**Total Deposits:**						
27	**No. of deposits ≥ $100,000:**						

Rates			
Deposit Amount			**Interest Rate (per year)**
From		**To**	
$ 5,000.00		$ 10,000.00	1.00%
$ 10,001.00		$ 30,000.00	1.02%
$ 30,001.00		$ 50,000.00	1.05%
$ 50,001.00		$ 75,000.00	1.10%
$ 75,001.00		$ 100,000.00	1.20%
$ 100,001.00		$ 150,000.00	1.35%
$ 150,001.00		$ 200,000.00	1.50%

1 Use an appropriate function to search for the **Interest Rate** in the **Rates Table** and use it to complete the **Base Merdeka Deposit Interest Rate** column. [2]

2 Use a conditional statement to complete the **Additional Bonus Interest** column using the features described for the Merdeka Deposit scheme. [2]

3 Enter a formula to calculate and complete the **Total Interest Rate** column. The value in the **Total Interest Rate** column can be found by adding the values in the **Base Merdeka Deposit Interest Rate** and **Additional Bonus Interest** columns. [2]

4 Enter a formula using yearly compounded interest to complete the **Amount Received At Maturity** column. [2]

5 In cell **C26**, enter a formula to calculate the total deposits collected by the bank. [1]

6 In cell **C27**, enter a formula to count the number of deposits that are worth at least $ 100,000.00. [1]

Save and close your file.

Task 2 begins on the next page.

Task 2

The following program accepts the shuttle run timings of 40 students, calculates the average timing and prints out the average timing.

```
total = 0
for count in range(40):
    timing = float(input("Enter shuttle run timing (s) : "))
    total = total + timing
average = total/40
print("Average shuttle run timing is {} s".format(average))
```

Open the file **TIMINGS.py**

Save the file as **TIMINGS**_<Class>_<Index_Number>_<Your_Name>**.py**

7 Edit your program so that it:

(a) accepts timing inputs for 20 students. [1]

(b) whenever a timing that is being input is faster than any previous timing entered, the system will output the message "`This is a new record timing!`". [4]

[Note: the message should be displayed when the first timing in entered.]

(c) tests if the timing input is between 0 and 30 inclusive, and if not, asks the user for input again as necessary. [3]

Save your program.

8 Save your program as **VARTIMINGS**_<Class>_<Index_Number>_<Your_Name>**.py**

Edit your program so that it works for any number of timings. [2]

Save your program.

Task 3

The following program should check which student is eligible for the job attachment programme using the following rules:

- age must be at least sixteen years
- age not more than eighteen years
- proficiency test score must be at least 60.

The program calculates the number of students who are eligible for the job attachment programme and the number rejected. When a student is rejected, all the reason(s) for rejection will be printed. The program finishes when an age of zero or a score of zero is input. The number of students who are eligible for the job attachment and the number rejected are then printed out.

There are several syntax errors and logical errors in the program.

```
age = 0
rejected = 100
Eligible = 0
age = int(input("Please enter student's age: " )
result = float(input("Please enter proficiency test score: "))
while age != 0 or result != 0:
    if age < 16  or age > 18 or result < 60
        if age < 16:
            print("Age must be at least sixteen years.")
        elif age < 18:
            print("Age must not be more than eighteen years.")
        if result < 60:
            print("Proficiency test score must be at least 60.")
        rejected = rejected - 1
        print("Student is NOT eligible for job attachment programme.")
else:
        print("Student is eligible for job attachment programme.")
        eligible = eligible + 1

        age = int(input("Please enter student's age: " ))
    result = float(input("Please enter proficiency test score: "))

print("Number of students eligible: " eligible)
print("Number of students rejected: ", rejected)
```

Open the file **JOB.py**

Save the file as **JOBERROR**_<your name>_<class>_<index_number>

9 Identify and correct the errors in the program so that it works correctly according to the rules above.

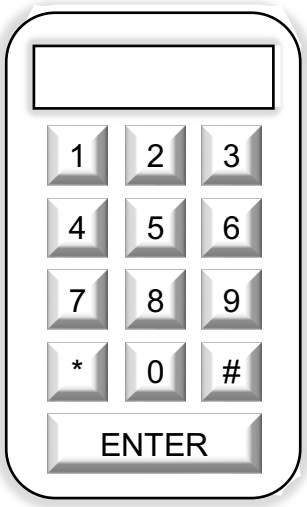
[10]

Save your program.

Task 4

You have been tasked to write a program for an in-room Electronic Safe System for a hotel group. The Electronic Safe System and the accompanying user instructions are shown in the following diagram. The Electronic Safe System consists of the following input and output components:

- LCD display panel;
- standard numeric keypad; and
- "ENTER" key.



INSTRUCTIONS

To Lock:
1. Place items in the safe.
2. Close door.
3. Enter a 6-digit password.
4. Press ENTER to lock.

To Unlock:
1. Enter the same 6-digit password used to lock.
2. Press ENTER to unlock.
3. Open door.

Call extension 100 or contact concierge service for help.

The Electronic Safe System is to be designed for hotel guests to securely store items in it. Guests could easily use the safe by setting a password to lock the safe and open the safe using the same password.

The algorithm of the program should be as follows:

Step 1: Display the message "Enter password to lock: ", and wait for the user to enter a 6-digit password. If the password is not exactly 6-digit long, display the message "Invalid password." and repeat Step 1. Otherwise, go to Step 2.

Step 2: Display the message "Safe is locked. Unlock with the same password." and go to Step 3.

Step 3: Display the message "Enter password to unlock safe: ", and wait for the user to enter the 6-digit password to unlock the safe. Users have 3 attempts to unlock the safe. Go to Step 4.

Step 4: If the user enters the correct password, display the message "Safe is unlocked." Whenever a wrong password is entered, display the message "Wrong Password. Enter password to unlock safe: ", and wait for user to enter the password again. Repeat Step 4 until user enters the wrong password the 3rd time, go to Step 5.

Step 5: Display the message "Exceeded maximum tries. Please contact concierge to unlock."

10 Write your program and test that it works. [10]

Save your program as **ELECTRONICSAFE**_<Class>_<Index_Number>_<Your_Name>**.py**

11 When your program is complete, use the following test data to show your test results:

Test 1:

```
Enter password to lock: 53423
:
```

Test 2:

```
Enter password to lock: 534237
:
Enter password to unlock safe: 534237
:
```

Test 3:

```
Enter password to lock: 534237
:
Enter password to unlock safe: 732435
Wrong password. Enter password to unlock safe: 534237
:
```

Test 4:

```
Enter password to lock: 534237
:
Enter password to unlock safe: 732435
Wrong password. Enter password to unlock safe: 732435
Wrong password. Enter password to unlock safe: 732435
:
```

Take a screen shot of: [4]

- Test 1. Save this screenshot as:
  TEST1_<Class>_<Index_Number>_<Your_Name>
- Test 2. Save this screenshot as:
  TEST2_<Class>_<Index_Number>_<Your_Name>
- Test 3. Save this screenshot as:
  TEST3_<Class>_<Index_Number>_<Your_Name>
- Test 4. Save this screenshot as:
  TEST4_<Class>_<Index_Number>_<Your_Name>

Save your files in either **.png** or **.jpg** format.

12 Save your program as **ELECTRONICSAFE2**_<Class>_<Index_Number>_<Your_Name>**.py**

It was noticed that many hotel guests had entered wrong passwords by mistake when locking the Electronic Safe and were unable to unlock them subsequently. The hotel management would like to change the algorithm of the Electronic Safe System such that users were required to re-enter the password prior to locking to verify that the intended password was being entered. The designers of the system suggested to change Step 2 of the algorithm to the following:

Step 2: Display the message "Re-enter password to confirm: ", and wait for the user to re-enter the 6-digit password entered in Step 1. If the re-entered password does not match the password entered in Step 1, display the message "Passwords do not match. Locking cancelled. " and go to Step 1. Otherwise, display the message "Safe is locked. Unlock with the same password." and go to Step 3.

Extend your program to accommodate the new changes to the system as specified in Step 2 of the amended algorithm. [4]

Save your program.

13 Save your program as **ELECTRONICSAFE3**_<Class>_<Index_Number>_<Your_Name>**.py**

Extend your program to continuously allow the next user to enter a new password to lock the electronic safe immediately after it is being unlocked (i.e. return to Step 1 of the program algorithm after it is being unlocked). The program will only stop running when a user enters a wrong password three times when trying to unlock the safe. [2]

Save your program.

END OF PAPER