Assessment Task: Lab 3

Qualification national code and title	22603VIC Certificate IV in Cybersecurity
Unit/s national code/s and title/s	**ICTPRG434 - Automate processes** **ICTPRG435 - Write script for software applications**

Assessment type (☑):

☐ Questioning (Oral/Written)

☐ Practical Demonstration

☐ 3rd Party Report

☒ Other – Lab

Assessment Resources:

The base requirements this assessment task include:

- IDE or editor for developing Python programs *(only IDLE and PyCharm supported by the college)*
- Access to Office 365 & Microsoft Word
- Virtual machine

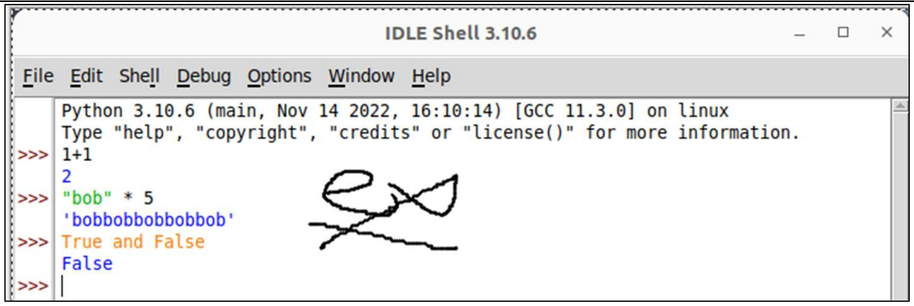
You may not need all these for every part in this assessment

Assessment Due:

This assessment is due after the weekly session, **Week 3, Friday 17:00**.

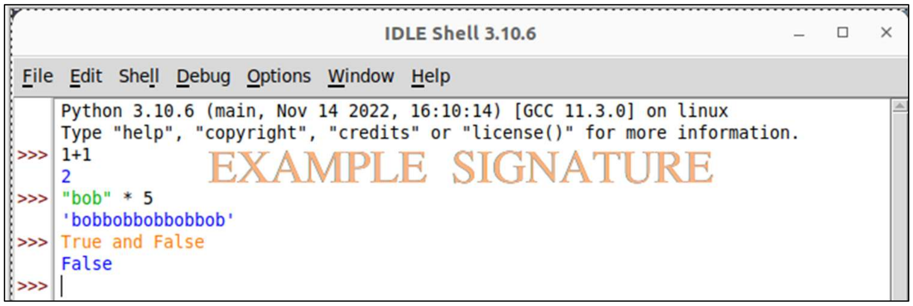
Assessment Instructions:

1. Your code must be written in IDLE or PyCharm IDEs. If you are using a different IDEs or a different structure for your application, then assistance from your lecturers may be limited (at best). Discuss with your lecturer before straying too far off the path!
2. All resources used should be referenced with the question. Answers may not be copied and pasted from any resource. All answers must be reworded to display your understanding.
3. You may only use Python functionality, methods and libraries which were taught in this unit.
4. First line of code in a program should have the student's name and number, as proof of authenticity.
5. Screenshots of all programs must be included **in** this document, with the appropriate question.
6. Screenshots of testing, showing your code works as intended, should be included with the relevant question.
7. Python programs should be named: XXX_Lab##_SYY_QZZ
   Replace XXX with your initials
   Replace ## with Lab number
   Replace YY with Section number,
   Replace ZZ with Question number
8. It is a submission requirement that all screen shots be signed in some way. Some acceptable examples of signed screen shots are shown below.



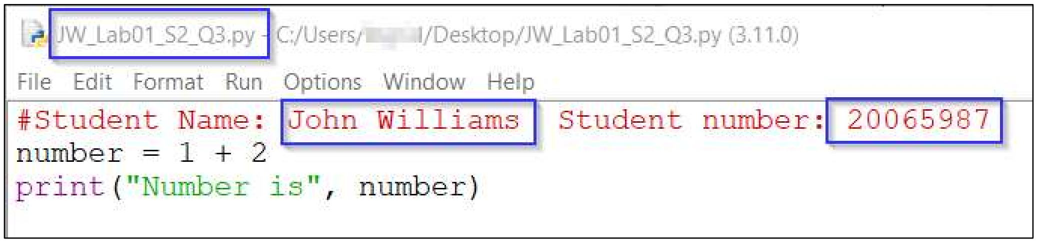
```
Python 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> 1+1
2
>>> "bob" * 5
'bobbobbobbobbob'
>>> True and False
False
>>> |
```

Example 1: Signed using a simple drawing tool.



```
Python 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> 1+1
2
>>> "bob" * 5
'bobbobbobbobbob'
>>> True and False
False
>>> |
```

Example 2: Water marked signature.


JW_Lab01_S2_Q3.py - C:/Users/[redacted]/Desktop/JW_Lab01_S2_Q3.py (3.11.0)
File Edit Format Run Options Window Help

```
#Student Name: John Williams  Student number: 20065987
number = 1 + 2
print("Number is", number)
```

Example 3: Program named as prescribed, as well as first line comment with student name and number. Program saved as pre-described.

9. All python programs must be included in the submission, as well as this document.

Assessment Instrument:

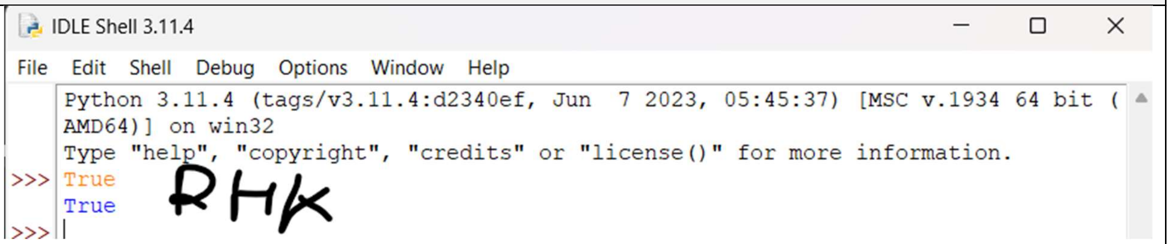
Qualification national code and title	22603VIC Certificate IV in Cybersecurity
Unit/s national code/s and title/s	**ICTPRG434 - Automate processes** **ICTPRG435 - Write script for software applications**

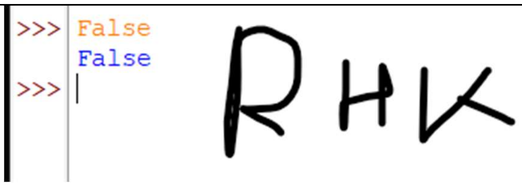
Section 1: Boolean expression

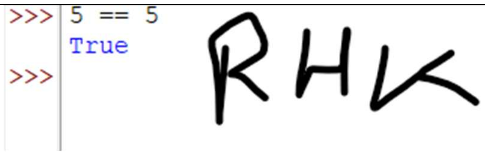
Booleans are either true or false, they are commonly used to in iteration and evaluation-based syntax. Run the following exercises in the Python shell/interpreter.

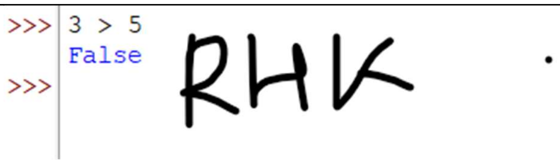
1. **Run** the following operations and
2. **evaluate** the output:

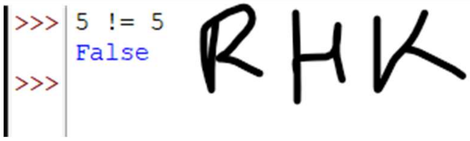
Code	Screenshot and evaluation

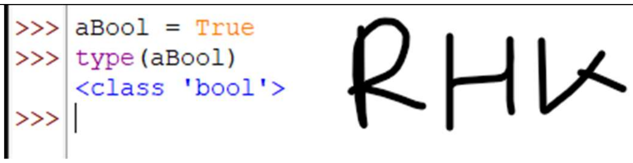
True	
	IDLE Shell 3.11.4
	`Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32`
	`Type "help", "copyright", "credits" or "license()" for more information.`
	`>>> True`
	`True`
	`>>>`
	RHK

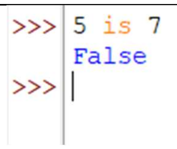
False	
	`>>> False`
	`False`
	`>>>`
	RHK

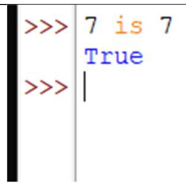
5 == 5	
	`>>> 5 == 5`
	`True`
	`>>>`
	RHK

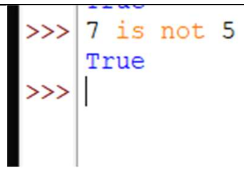
3 > 5	
	`>>> 3 > 5`
	`False`
	`>>>`
	RHK

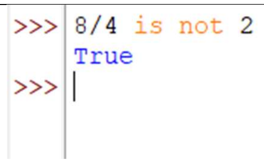
5 != 5	
	`>>> 5 != 5`
	`False`
	`>>>`
	RHK

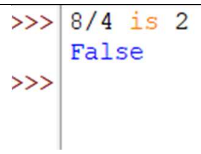
aBool = True
type(aBool)
	
	`>>> aBool = True`
	`>>> type(aBool)`
	`<class 'bool'>`
	`>>>`
	RHK

5 is 7	 `>>> 5 is 7`
	`False`
	`>>>`
	RHK

7 is 7	 `>>> 7 is 7`
	`True`
	`>>>`
	RHK

7 is not 5	 `>>> 7 is not 5`
	`True`
	`>>>`
	RHK

8/4 is not 2	 `>>> 8/4 is not 2`
	`True`
	`>>>`
	RHK

8//4 is 2	 `>>> 8/4 is 2`
	`False`
	`>>>`
	RHK .

3. What is the difference between the **/** and the **//** operators?

/ (Classic Division Operator): Regardless of whether the operands are floats or integers, this operator always returns a floating-point number after performing standard division.
// (Floor Division Operator): This operator rounds the result to the closest whole number (integer) after performing division.

4. What does the **modulus** operator do?

The modulus operator in Python is an arithmetic operator that yields the remainder of a division operation and is denoted by the percent sign (%).

5. Why does **10 == 2 + 2*4** in Python return the answer **True** and not **False**?

Qualification national code and title	22603VIC Certificate IV in Cybersecurity
Unit/s national code/s and title/s	**ICTPRG434 - Automate processes** **ICTPRG435 - Write script for software applications**

2 + 2 x 4
16
59 %
15
7 %
14
8 %
13
26 %
10

This is because of the operator precedence. So * has higher precedence than +, therefore at 2 * 4 = 8 and then 2 added to 8, which results in 10 and 10 == 10 is True

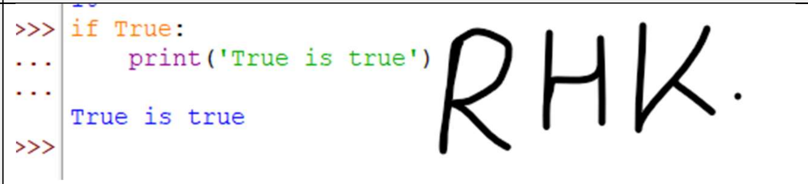
Section 2: Conditional execution

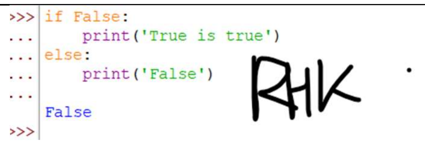
Understanding a conditional execution is key to decision making in coding. Run the following code in the Python shell/interpreter.

1. Run the following operations and evaluate the output:

   Remember, watch for your indenting here!

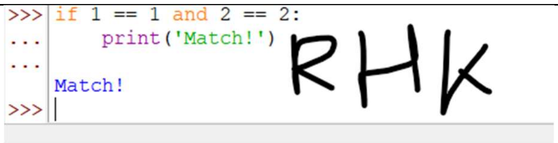
Code	Screen shot

if True:
    print("True is true")
	
	>>> if True:
	... print('True is true')
	...
	True is true
	>>>
	RHK.

if False:
    print("True is true")
else:
    print("False")
	
	>>> if False:
	... print('True is true')
	... else:
	... print('False')
	...
	False
	>>>
	RHK

Qualification national code and title	22603VIC Certificate IV in Cybersecurity
Unit/s national code/s and title/s	**ICTPRG434 - Automate processes** **ICTPRG435 - Write script for software applications**

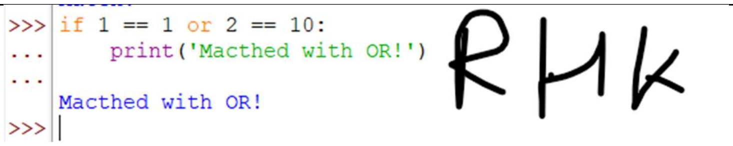
if 1==1 and 2==2:
    print(“Match!”)



```
>>> if 1 == 1 and 2 == 2:
...     print('Match!')
...
Match!
>>> |
```

RHK

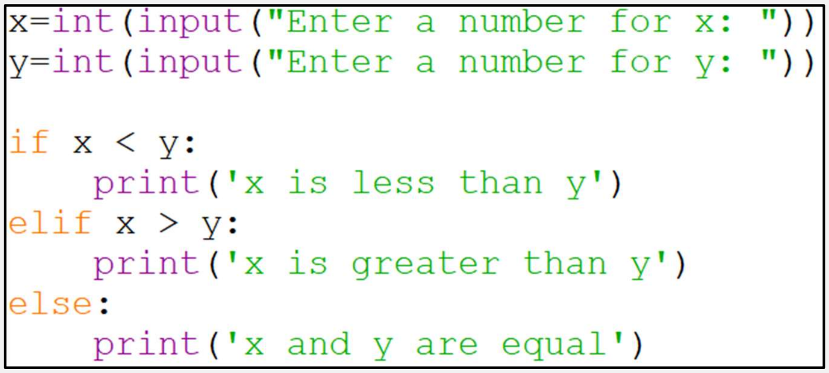
if 1==1 or 2==10:
    print(“Matched with OR!”)



```
>>> if 1 == 1 or 2 == 10:
...     print('Macthed with OR!')
...
Macthed with OR!
>>> |
```

RHK

2. Run the following code in a code file to demonstrate a chained conditional execution statement:

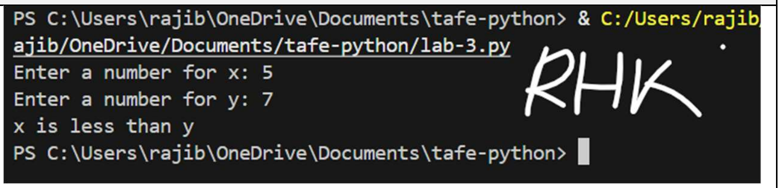


```
x=int(input("Enter a number for x: "))
y=int(input("Enter a number for y: "))

if x < y:
    print('x is less than y')
elif x > y:
    print('x is greater than y')
else:
    print('x and y are equal')
```

Input

Screenshot of output

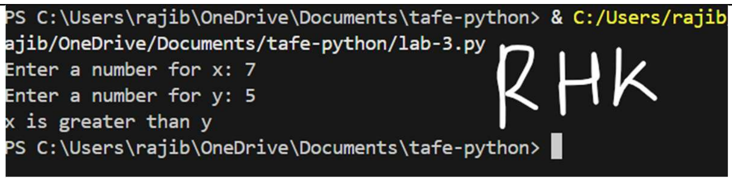
Enter a number for x: 5
Enter a number for y: 7



```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter a number for x: 5
Enter a number for y: 7
x is less than y
PS C:\Users\rajib\OneDrive\Documents\tafe-python>
```

RHK

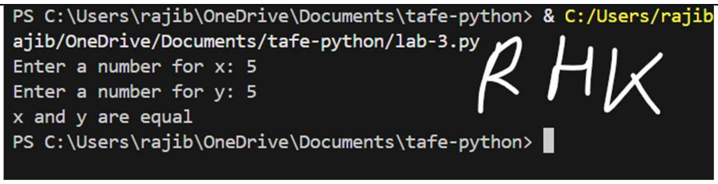
Enter a number for x: 7
Enter a number for y: 5



```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter a number for x: 7
Enter a number for y: 5
x is greater than y
PS C:\Users\rajib\OneDrive\Documents\tafe-python>
```

RHK

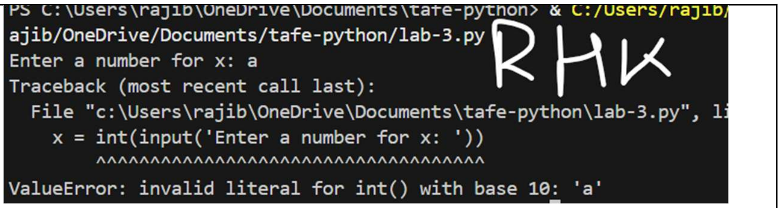
Enter a number for x: 5
Enter a number for y: 5



```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter a number for x: 5
Enter a number for y: 5
x and y are equal
PS C:\Users\rajib\OneDrive\Documents\tafe-python>
```

RHK

Qualification national code and title	22603VIC Certificate IV in Cybersecurity
Unit/s national code/s and title/s	**ICTPRG434 - Automate processes** **ICTPRG435 - Write script for software applications**

Enter a number for x: a **Enter a number for y:** 5 *Your program will crash*	
Explain why the program crashed when x is a, and y is 5 were entered.	Because a cannot be converted to a number

```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib/
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter a number for x: a
Traceback (most recent call last):
  File "c:\Users\rajib\OneDrive\Documents\tafe-python\lab-3.py", li
    x = int(input('Enter a number for x: '))
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: 'a'
```

RHK

Assessment Task: Lab 3

Qualification national code and title	22603VIC Certificate IV in Cybersecurity
Unit/s national code/s and title/s	**ICTPRG434 - Automate processes** **ICTPRG435 - Write script for software applications**

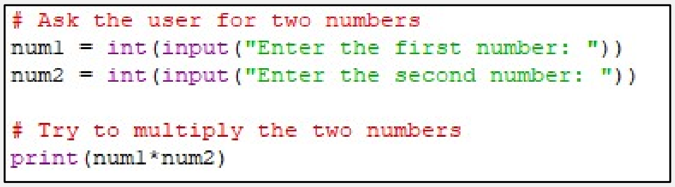
Section 3: Try and except

Try and except are another form of conditional checking (based on syntax errors). It's important to understand exceptions and how they are handled. You may find some of the online documentation to be useful for this topic.
Using code files, complete the following tasks.

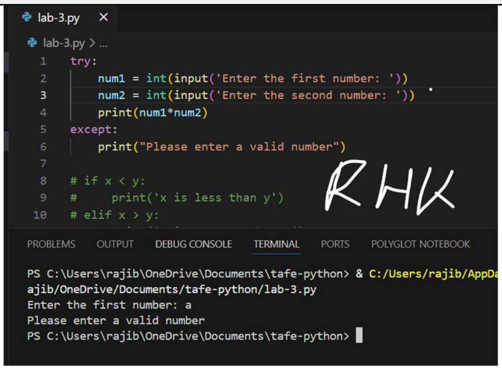
1. What is meant by (and what could possibly cause) the following exceptions?

Exception	**Description and cause**
TypeError	When an operation or function is applied to an object of an improper or incompatible data type, a built-in exception known as TypeError is raised.
NameError	When a program tries to use a name (variable, function, class, module, etc.) that has not been defined or is not accessible in the current scope, it generates a NameError
ValueError	When an argument with the right data type but an incorrect value is passed to a function or operation, a built-in exception known as ValueError is raised.
IndentationError	In Python, an IndentationError is a built-in exception that arises when code is not properly or consistently indented.

2. Examine the code in the screen shot. Improve this code by using a try and except statement to handle any incorrect user input gracefully (remember to demonstrate that it works):



```
# Ask the user for two numbers
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

# Try to multiply the two numbers
print(num1*num2)
```

Input	**Screenshot of output**
Enter the first number: a **Enter the second number:** 7	

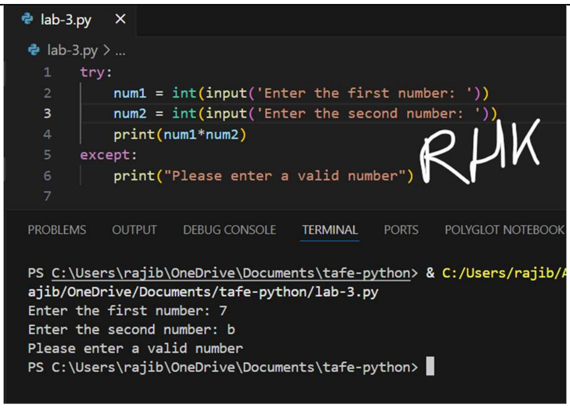
```
try:
    num1 = int(input('Enter the first number: '))
    num2 = int(input('Enter the second number: '))
    print(num1*num2)
except:
    print("Please enter a valid number")

# if x < y:
#     print('x is less than y')
# elif x > y:
```

```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib/AppDa
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter the first number: a
Please enter a valid number
PS C:\Users\rajib\OneDrive\Documents\tafe-python> 
```

Qualification national code and title	22603VIC Certificate IV in Cybersecurity
Unit/s national code/s and title/s	**ICTPRG434 - Automate processes** **ICTPRG435 - Write script for software applications**

Enter the first number: 7

Enter the second number: b



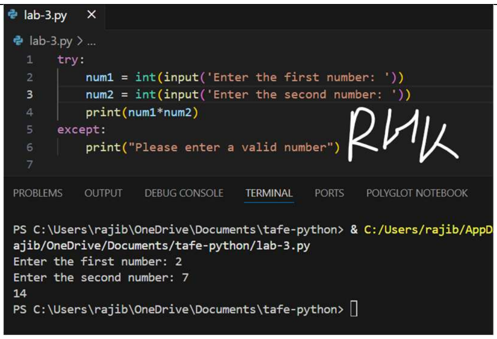
```
lab-3.py > ...
try:
    num1 = int(input('Enter the first number: '))
    num2 = int(input('Enter the second number: '))
    print(num1*num2)
except:
    print("Please enter a valid number")
```

RHK

```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib/A
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter the first number: 7
Enter the second number: b
Please enter a valid number
PS C:\Users\rajib\OneDrive\Documents\tafe-python> 
```

Enter the first number: 2

Enter the second number: 7

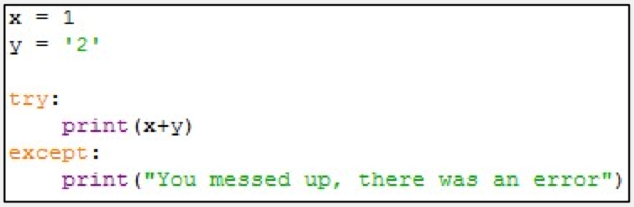


```
lab-3.py > ...
try:
    num1 = int(input('Enter the first number: '))
    num2 = int(input('Enter the second number: '))
    print(num1*num2)
except:
    print("Please enter a valid number")
```

RHK

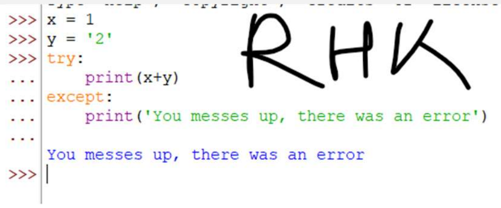
```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib/AppD
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter the first number: 2
Enter the second number: 7
14
PS C:\Users\rajib\OneDrive\Documents\tafe-python> 
```

3. Write the following code and execute it, it's a good example how a failure can be handled.



```
x = 1
y = '2'

try:
    print(x+y)
except:
    print("You messed up, there was an error")
```

Run the code. This code should fail because you are trying to perform a mathematical operation on a string and integer. The script does not crash because **try** and **except** are used to handle any errors. This won't tell you why there was an error, only that there was one.



```
>>> x = 1
>>> y = '2'
>>> try:
...     print(x+y)
... except:
...     print('You messes up, there was an error')
...
You messes up, there was an error
>>> 
```

RHK

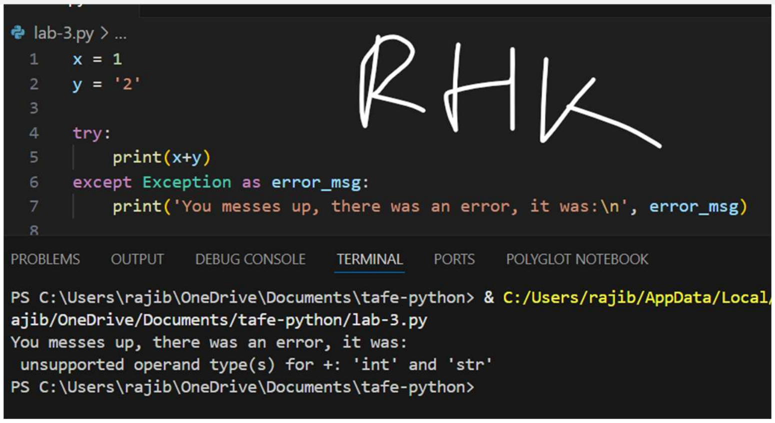
4. Modify your code to capture the error message and print it out by replacing:

```
except:
    print("You messed up, there was an error")
```

with:

```
except Exception as error_msg:
    print("You messed up, there was an error. It was:\n",error_msg)
```

Run the script. What is the error message presented on screen?



```
x = 1
y = '2'

try:
    print(x+y)
except Exception as error_msg:
    print('You messes up, there was an error, it was:\n', error_msg)
```

```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib/AppData/Local/
ajib/OneDrive/Documents/tafe-python/lab-3.py
You messes up, there was an error, it was:
 unsupported operand type(s) for +: 'int' and 'str'
PS C:\Users\rajib\OneDrive\Documents\tafe-python>
```

RHK

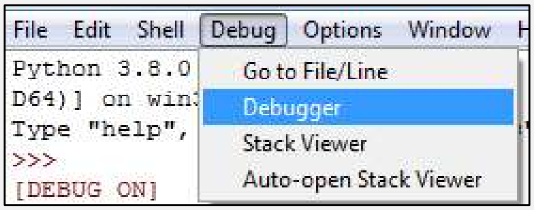
Qualification national code and title	22603VIC Certificate IV in Cybersecurity
Unit/s national code/s and title/s	**ICTPRG434 - Automate processes** **ICTPRG435 - Write script for software applications**

Section 4: Debugging

How it works:

It can get complicated when several things are happening inside of a script. You may need to use the debugger to assist you in working out what is happening in your script.

1. Open the debugger

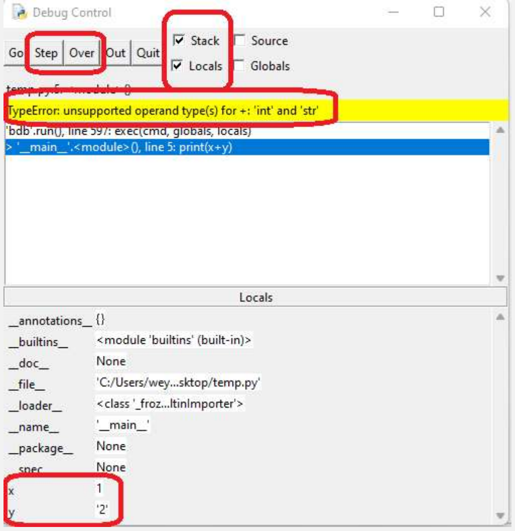


```
Python 3.8.0
D64)] on win
Type "help",
>>>
[DEBUG ON]
```

2. In the debugger, make sure the **Stack** and **Locals** boxes are ticked (the two other boxes provide you with more information than needed in this course). Leave the debugger open and run your script from the previous section more time.
3. The Debugger will run your code one line at a time and tell you each variable as it is seen by the interpreter. There are two buttons to do this:
   a. **Step**: The step button will execute each line of code in your entire script one line at a time (this includes all the lines of code in all the functions you use (including Python's prebuilt functions like *print()* and *input()*)).
   b. **Over**: the over button will process the script line by line without going into the code of each individual function. This is more useful when you just want to see what variables are changing without going through all the functions in your code (it's safe to assume that the Python prebuilt functions are error free).

   Eventually the error should appear in yellow. You should be able to quickly identify the issue by looking at the variables (see the screen shot example).

Qualification national code and title	22603VIC Certificate IV in Cybersecurity
Unit/s national code/s and title/s	**ICTPRG434 - Automate processes** **ICTPRG435 - Write script for software applications**



```
temp.py:5: <module>()
TypeError: unsupported operand type(s) for +: 'int' and 'str'
'bdb'.run(), line 597: exec(cmd, globals, locals)
> '__main__'.<module>(), line 5: print(x+y)
```

```
__annotations__ {}
__builtins__    <module 'builtins' (built-in)>
__doc__         None
__file__        'C:/Users/wey...sktop/temp.py'
__loader__      <class '_froz...ltinImporter'>
__name__        '__main__'
__package__     None
__spec__        None
x               1
y               '2'
```

Question

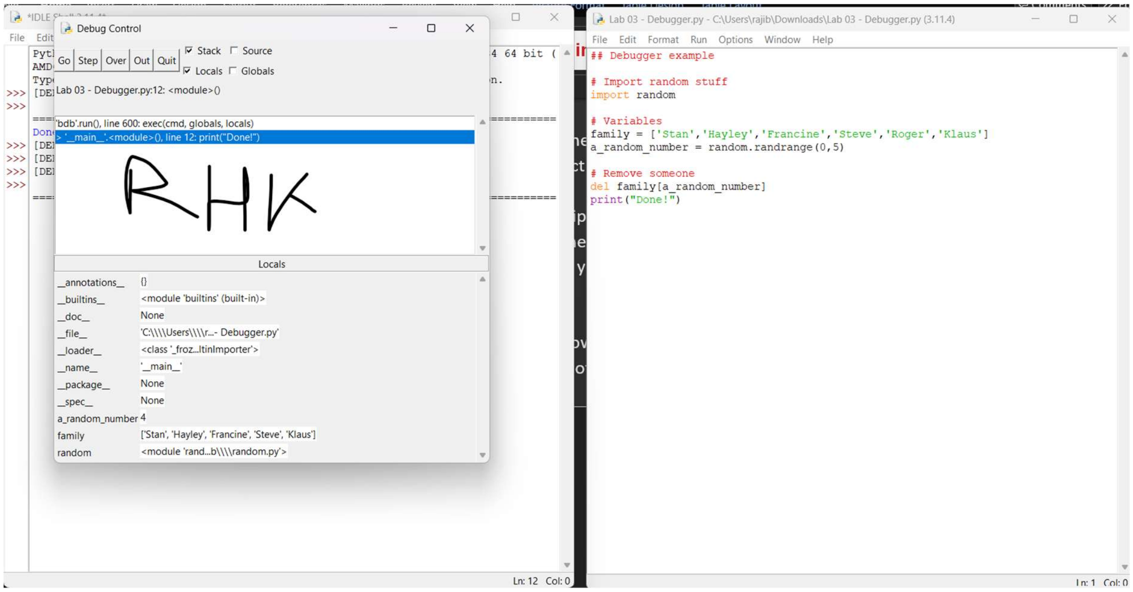
From BlackBoard, download the “Lab 3 debugger” file and open it in Python. This simple program creates a list of people and randomly removes one from it.

Use the debugger and the script (like the previous steps) to identify which family member was removed at random and what the random number was.

Family member that was removed: Roger

Random number was: 4

Qualification national code and title	22603VIC Certificate IV in Cybersecurity
Unit/s national code/s and title/s	**ICTPRG434 - Automate processes** **ICTPRG435 - Write script for software applications**



```
Pyt
AMD
Typ
>>> [DE
>>>
===
Don
>>> [DE
>>> [DE
>>> [DE
>>>
===
```

```
Lab 03 - Debugger.py:12: <module>()

'bdb'.run(), line 600: exec(cmd, globals, locals)
> '__main__'.<module>(), line 12: print("Done!")
```

RHK

__annotations__	{}
__builtins__	<module 'builtins' (built-in)>
__doc__	None
__file__	'C:\\\\Users\\\\r...- Debugger.py'
__loader__	<class '_froz...ltinImporter'>
__name__	'__main__'
__package__	None
__spec__	None
a_random_number	4
family	['Stan', 'Hayley', 'Francine', 'Steve', 'Klaus']
random	<module 'rand...b\\\\random.py'>

```
## Debugger example

# Import random stuff
import random

# Variables
family = ['Stan','Hayley','Francine','Steve','Roger','Klaus']
a_random_number = random.randrange(0,5)

# Remove someone
del family[a_random_number]
print("Done!")
```

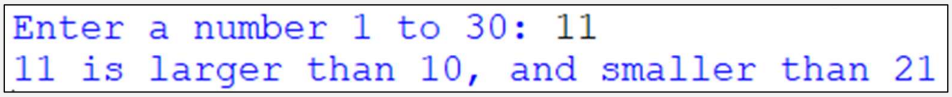
Qualification national code and title	22603VIC Certificate IV in Cybersecurity
Unit/s national code/s and title/s	**ICTPRG434 - Automate processes** **ICTPRG435 - Write script for software applications**

Section 5: A challenge using chained and nested conditionals.

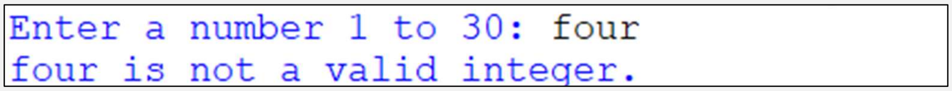
You will often need to use several if statements in a row to determine the action that should be taken based on an input of some description. Combine your skills to complete the following challenge using a code file (use the flow chart in appendix A to help you if required):

1. Create a simple input checking script that will ask a user to input a valid integer. The script must:
   a. Ensure the number input is an integer *between* 0 and 31. Be mindful: 0 and 31 are invalid values.
   b. Determine if the number is less than 1.
   c. Determine if the number is
      - smaller than 11,
      - 11 to 20 or
      - greater than 20

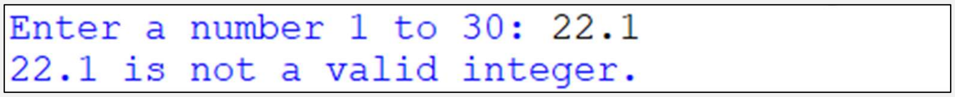
   Example screenshots:



```
Enter a number 1 to 30: 11
11 is larger than 10, and smaller than 21
```



```
Enter a number 1 to 30: four
four is not a valid integer.
```



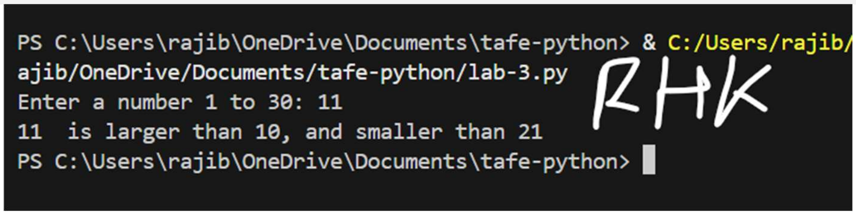
```
Enter a number 1 to 30: 22.1
22.1 is not a valid integer.
```

2. The output must be formatted like the provided screen shots, appropriate messages should also be provided for other scenarios.

3. Run the script several times to demonstrate it works as intended.

Note: Uploading your code file is a submission requirement for all code challenges.

Screenshot of output when *11* is entered.
Should display: 11 is between 10 and 20

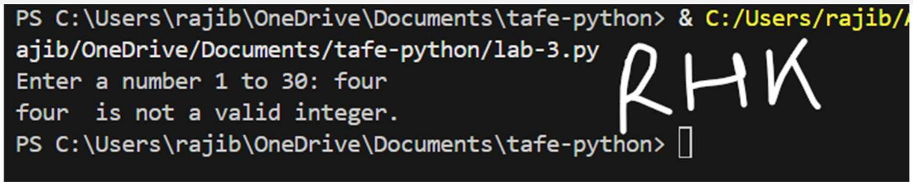


```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib/
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter a number 1 to 30: 11
11  is larger than 10, and smaller than 21
PS C:\Users\rajib\OneDrive\Documents\tafe-python> 
```

RHK

Screenshot of output when *four* is entered.

Should display: four is not a valid integer

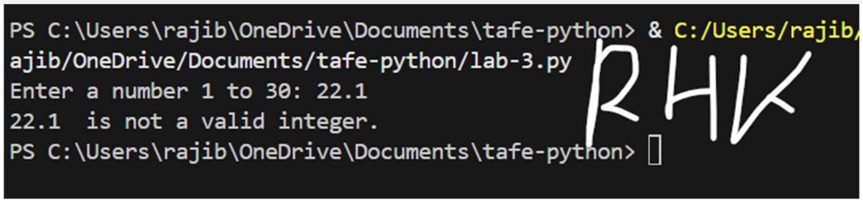


```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib/
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter a number 1 to 30: four
four  is not a valid integer.
PS C:\Users\rajib\OneDrive\Documents\tafe-python> 
```

RHK

Screenshot of output when *22.1* is entered.

Should display: 22.1 is not a valid integer

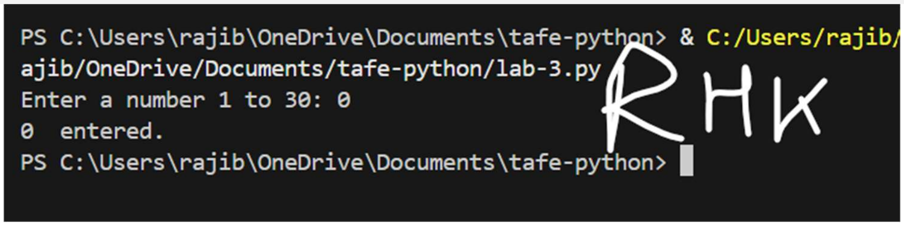


```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib/
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter a number 1 to 30: 22.1
22.1  is not a valid integer.
PS C:\Users\rajib\OneDrive\Documents\tafe-python> 
```

RHK

Screenshot of output when 0 is entered.

Your program should give an appropriate error message.

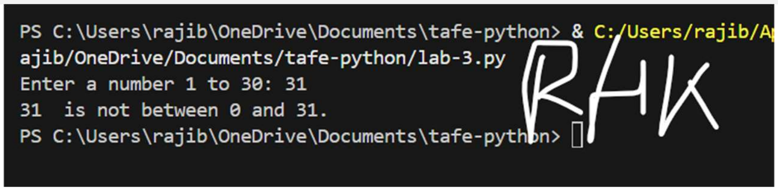


```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib/
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter a number 1 to 30: 0
0  entered.
PS C:\Users\rajib\OneDrive\Documents\tafe-python> 
```

RHK

Screenshot of output when *31* is entered.

Your program should give an appropriate error message.

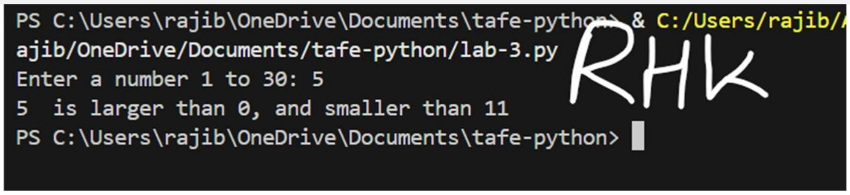


```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib/A
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter a number 1 to 30: 31
31  is not between 0 and 31.
PS C:\Users\rajib\OneDrive\Documents\tafe-python> 
```

RHK

Screenshot of output when *5* is entered.

Your program should give an appropriate message.

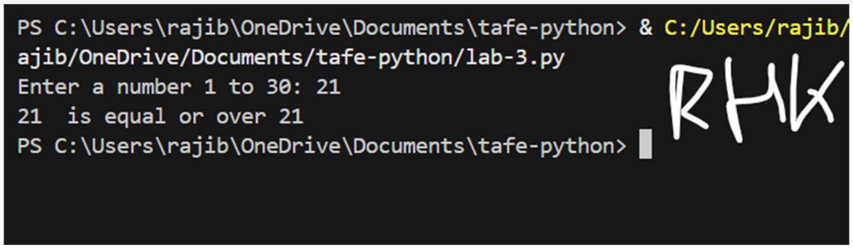


```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib/
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter a number 1 to 30: 5
5  is larger than 0, and smaller than 11
PS C:\Users\rajib\OneDrive\Documents\tafe-python> 
```

RHK

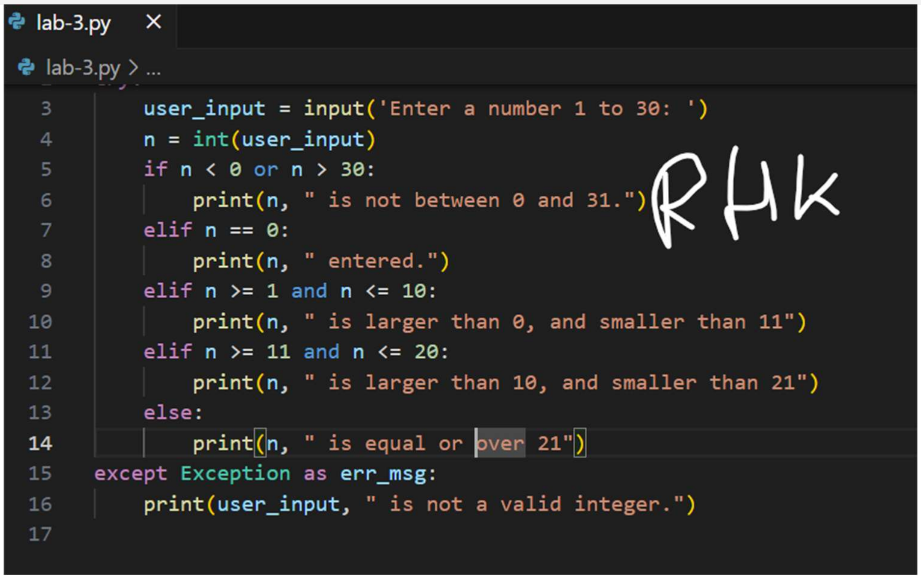
Screenshot of output when *21* is entered.

Your program should give an appropriate message.



```
PS C:\Users\rajib\OneDrive\Documents\tafe-python> & C:/Users/rajib/
ajib/OneDrive/Documents/tafe-python/lab-3.py
Enter a number 1 to 30: 21
21  is equal or over 21
PS C:\Users\rajib\OneDrive\Documents\tafe-python> █
```

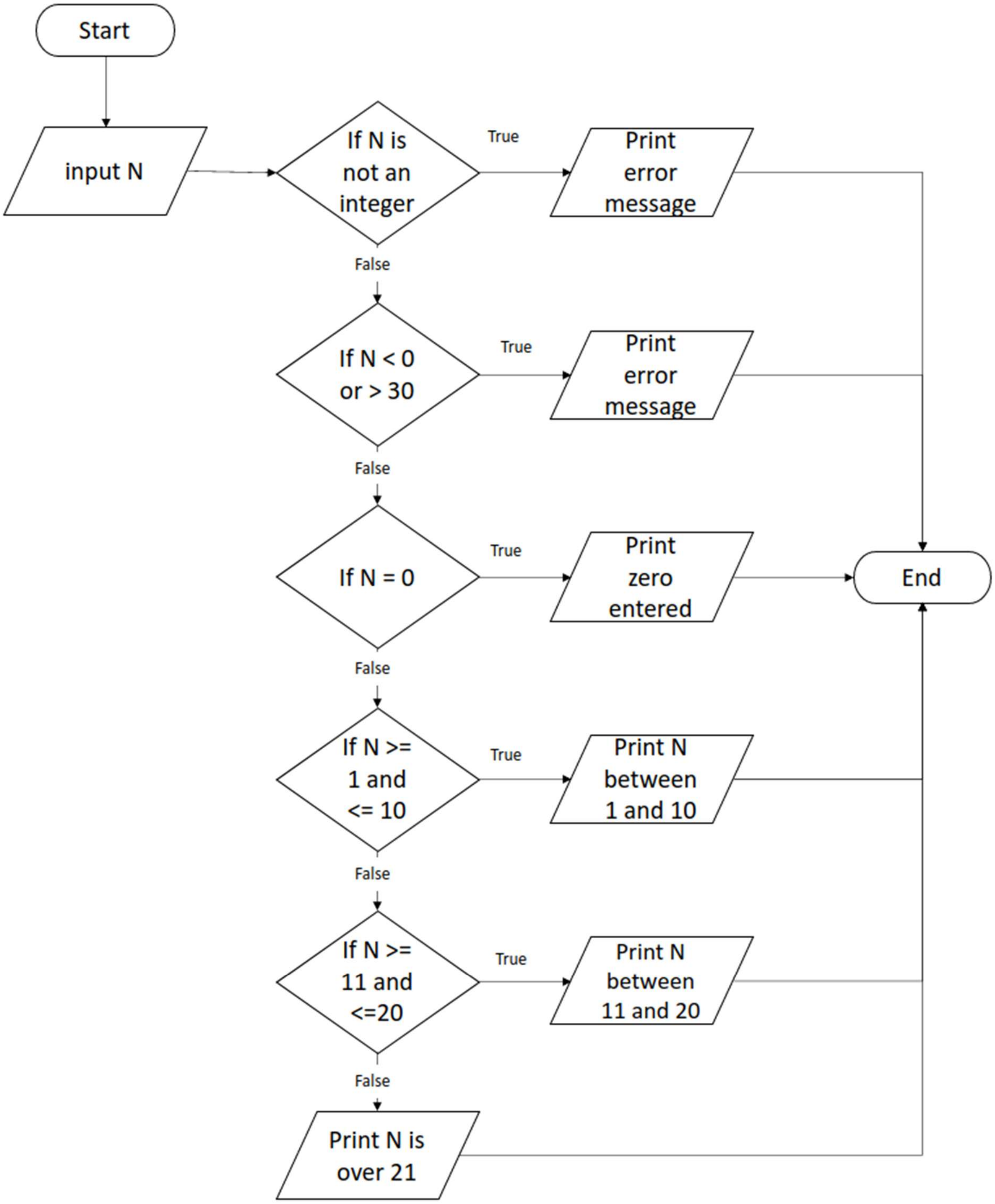
Screenshot of code:



```
    user_input = input('Enter a number 1 to 30: ')
    n = int(user_input)
    if n < 0 or n > 30:
        print(n, " is not between 0 and 31.")
    elif n == 0:
        print(n, " entered.")
    elif n >= 1 and n <= 10:
        print(n, " is larger than 0, and smaller than 11")
    elif n >= 11 and n <= 20:
        print(n, " is larger than 10, and smaller than 21")
    else:
        print(n, " is equal or over 21")
except Exception as err_msg:
    print(user_input, " is not a valid integer.")
```

Qualification national code and title	22603VIC Certificate IV in Cybersecurity
Unit/s national code/s and title/s	**ICTPRG434 - Automate processes** **ICTPRG435 - Write script for software applications**

Appendix A – Challenge flow chart


Start
input N
If N is not an integer
True
Print error message
False
If N < 0 or > 30
True
Print error message
False
If N = 0
True
Print zero entered
End
False
If N >= 1 and <= 10
True
Print N between 1 and 10
False
If N >= 11 and <=20
True
Print N between 11 and 20
False
Print N is over 21