**CFRS 772: Forensic Artifact Extraction**

**Homework Project 1**

**Rachel B. Gully**

1. **Confirm Python on the lab system**

- Start:Search:python

or

- Install Python 3.x if needed (e.g., personal computer)

1. **Open terminal**

- run "IDLE (Python 3.x GUI)" or similar

- you do not need to run this as Administrator

1. **Check python version installed:**

- IDLE window (before the Python command prompt) should start with "Python 3.x.x"

or

>>> import sys

>>> print(sys.version)

1. **Set command history in IDLE:**

- Options:ConfigureIDLE:Keys

- Scroll down and select "history-next"

- Click on "Get new keys"

- Scroll down to "Down Arrow" and click OK

- Name the Custom Key Set "CFRS772"

- Repeat for "history-previous" with "Up Arrow"

- Apply, OK

1. **Basic interaction:**

- Which of the below work? (put y/n in the blanks)

>>> print("hello") \_**Y**\_\_

>>> print('hello') \_**Y**\_\_

>>> print("hello') \_**N**\_\_

>>> print(`hello`) \_**N**\_\_

>>> print "hello" \_**N**\_\_

>>> print 'hello' \_**N**\_\_

- Try the following commands: (put result in the blanks)

>>> print(x) \_\_**NameError: name ‘x’ is not defined**\_\_

>>> x=10

>>> print(x) \_\_**10**\_\_

>>> print(X) \_\_**NameError: name ‘X’ is not defined**\_\_

>>> y=3

>>> print(y) \_\_**3**\_\_

>>> print(x+y) \_**13**\_\_

>>> print(x\*y) \_**30**\_\_

>>> print(x/y) \_\_ **3.3333333333333335**\_\_

>>> print(x-y) \_\_**7**\_\_

>>> print(x//y) \_\_**3**\_\_

>>> print(x%y) \_\_**1**\_\_

>>> z=x+y

>>> print(z) \_**13**\_\_

1. **Basic script creation:**

- In IDLE, go to File:New File

- Enter the following in the Editor Window:

x=10

y=3

print(x+y)

- File:SaveAs

test1.py (note the location)

- Run:RunModule or F5

check IDLE console for output

- In console

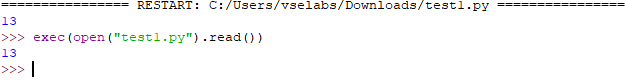
Try:

>>> test1.py

Did this work? **🡪 YES**

Try:

>>> exec(open("test1.py").read())



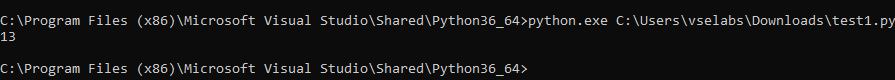
- At Windows command line

Start:Search:cmd

Run "cmd.exe" (not as Administrator)

Change to directory c:\python3x

C:\Python3x>python <path>/test1.py



1. **Comments and conditionals:**

- In IDLE, File:Open

test1.py

- Add comments:

# This is a one-line comment

''' This is a

multiline comment'''

- Add an IF statement to the end of test1.py:

// The if statement contents, i.e., the stuff after the

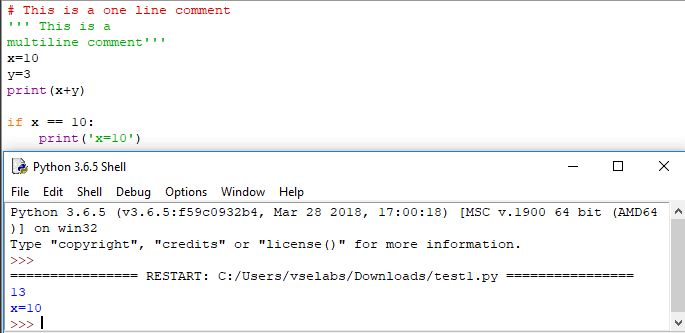
// "if" statement, \*must\* be indented (use tab key),

// like this:

if x==10:

print ('x=10')

- Run the script



- Edit the script to have only this content:

x=10

if x==10:

print ('x=10')

elif x==9:

print ('x=9')

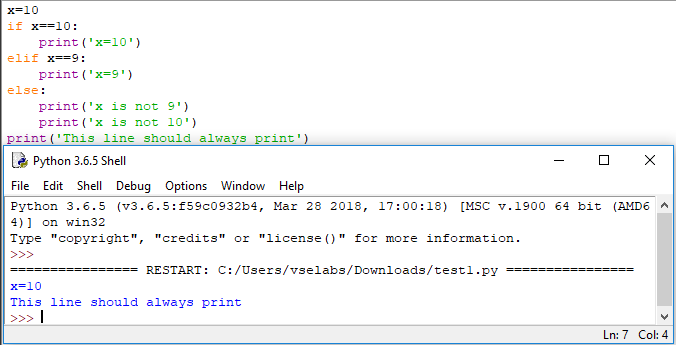
else:

print ('x is not 9')

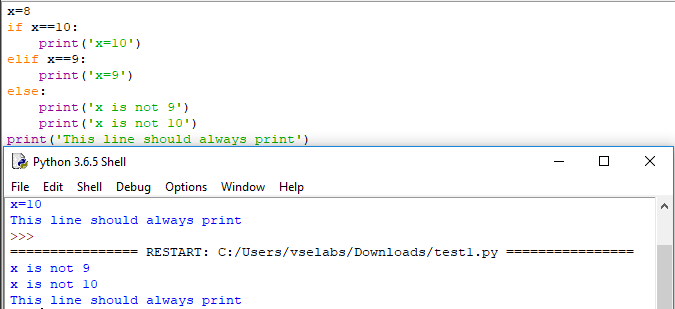
print ('x is not 10')

print ('This line should always print')

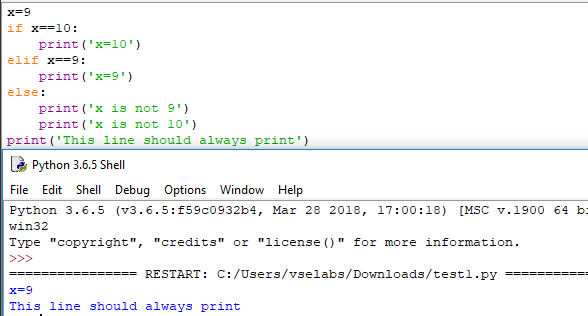
- Run the script



- Change x=8 and run the script



- Change x=9 and run the script



1. **Explore context:**

- Start a new IDLE session

>>> print(x)

>>> x=7

>>> print(x)

Run the test1.py script (it should have "x=9" in it)

>>> print(x)

- What does this result tell you? \_\_\_\_**9**\_\_\_\_\_\_\_\_\_\_\_\_\_

>>> x=6

>>> print(x)

Remove the "x=..." line from your test1.py script

Run the test1.py script

Why didn't that work? \_\_\_\_**x isn’t defined**\_\_\_\_\_\_\_\_\_\_\_\_\_

>>> x=10

>>> print(x)

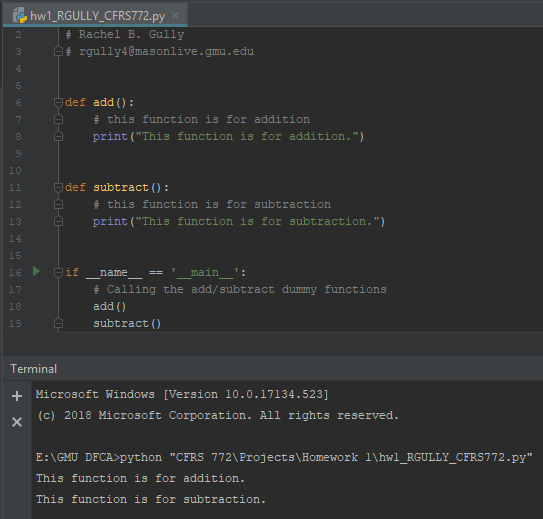
Run the test1.py script

- Why didn't that work? **\_It does work because x is now defined, and assigned a value of 10\_**

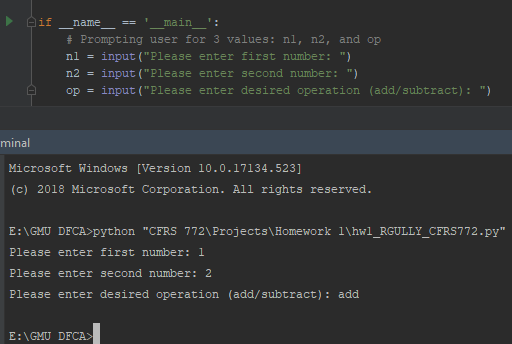
1. **Create a Python script**

Create a new file (script) in IDLE; name the file hw1\_yourname.py

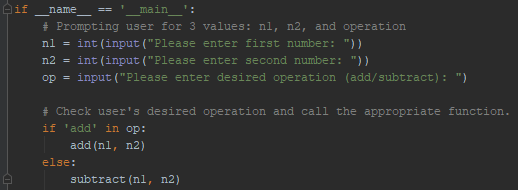
Create a simple program with a main routine that calls two functions, add() and subtract(); for each function, include comments about its purpose but just print a short message and return



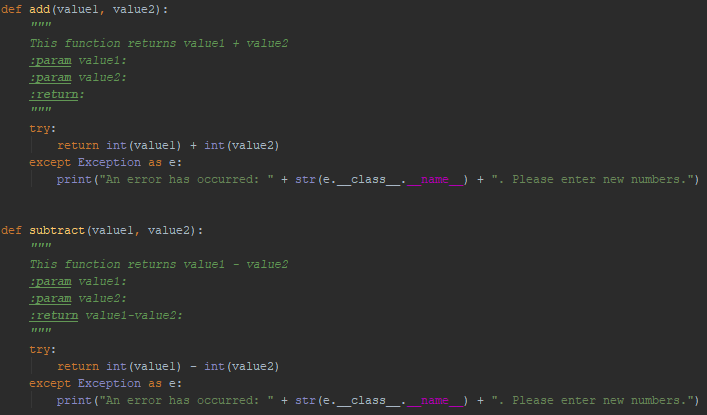
Prompt for user input in the main routine; create three prompts: number 1, number 2, and operation



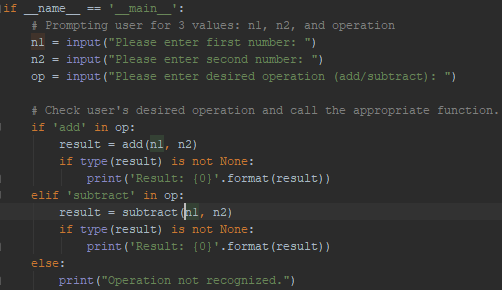
Add code (IF block) to the main routine to check the value of the 3rd user argument and call the appropriate function with the first two user arguments



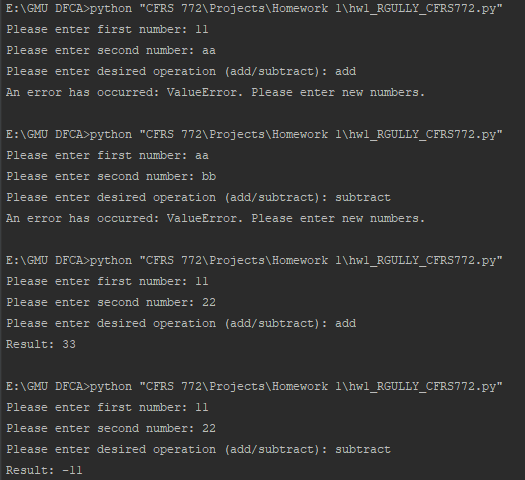
Modify the two functions so they take two arguments (the two numbers entered by the user) and compute and return the sum or difference; add try-except structures to catch errors



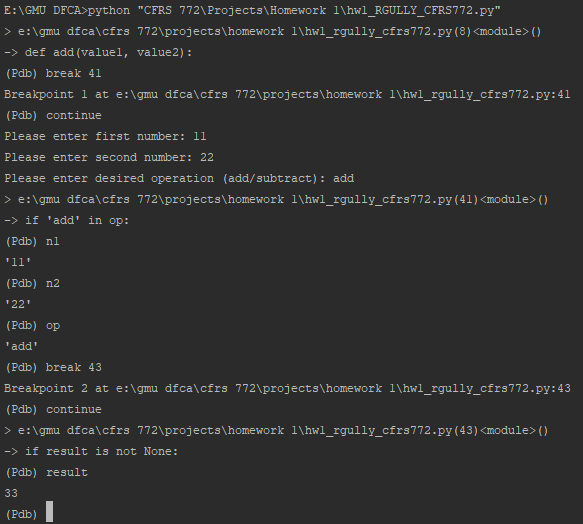
Modify the main routine to print out the user input and result



Run a few different examples, including ones that generate errors



Add code to use the debugger; run your script in debugging mode, step through your program, and check values at various points



**On BlackBoard, submit (1) a single PDF document with all screenshots (numbered) and answers to questions (all the blanks, also numbered; you can insert answers into this document and save it as PDF for submission), and (2) your final code for part 9 with debugging disabled.**