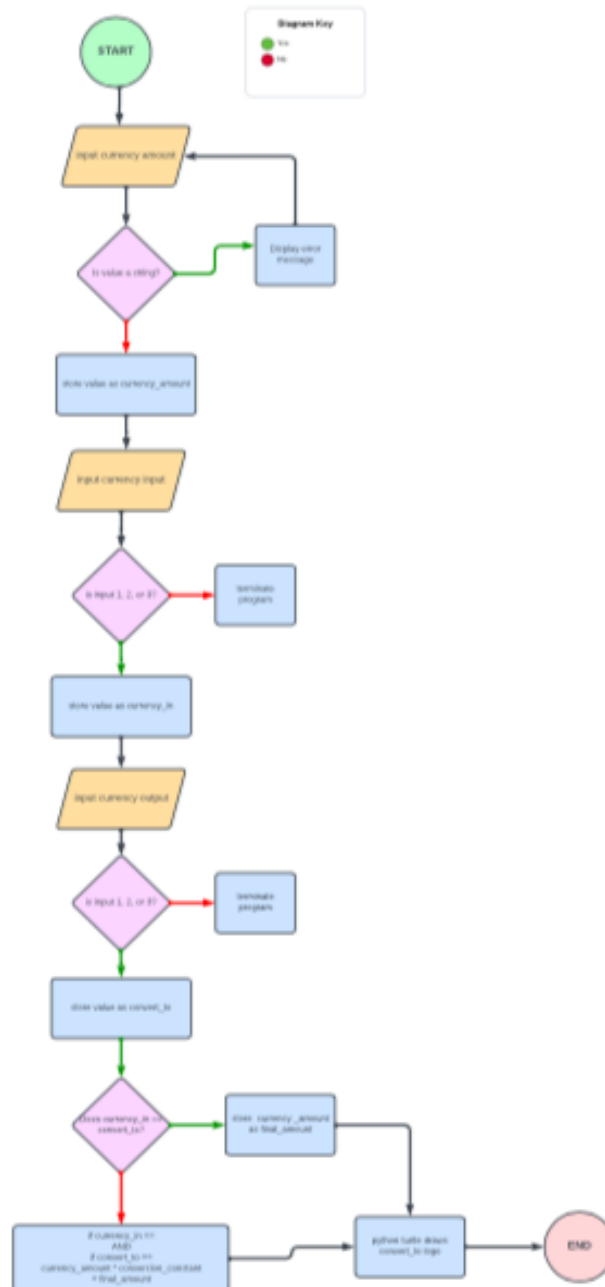


REPORT

1. Flow chart

A screenshot of my chart is below. See labTwoFlowChart.pdf in zip file for more detail.

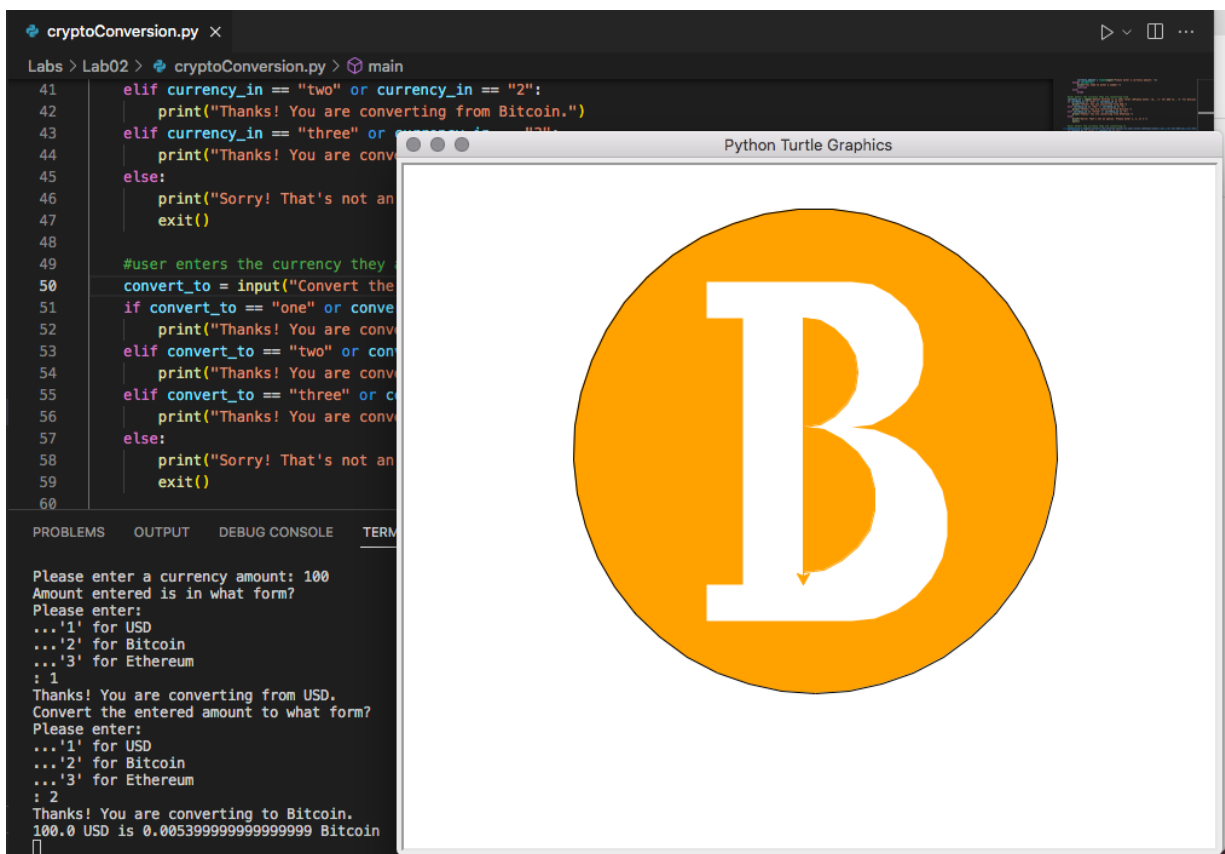


2. Required Task Elements

See below.

TRIAL 1 (USD to Bitcoin)

This shows my calculator converting 100 USD to Bitcoin, which turns out to be less than one cent. Once it converts, python draws the Bitcoin logo, which uses color. You can see some of my extension work here: the program will print a confirmation of the currency selections, then print a final statement in full which shows the currency_in, convert_to, and final_amount variables in action.

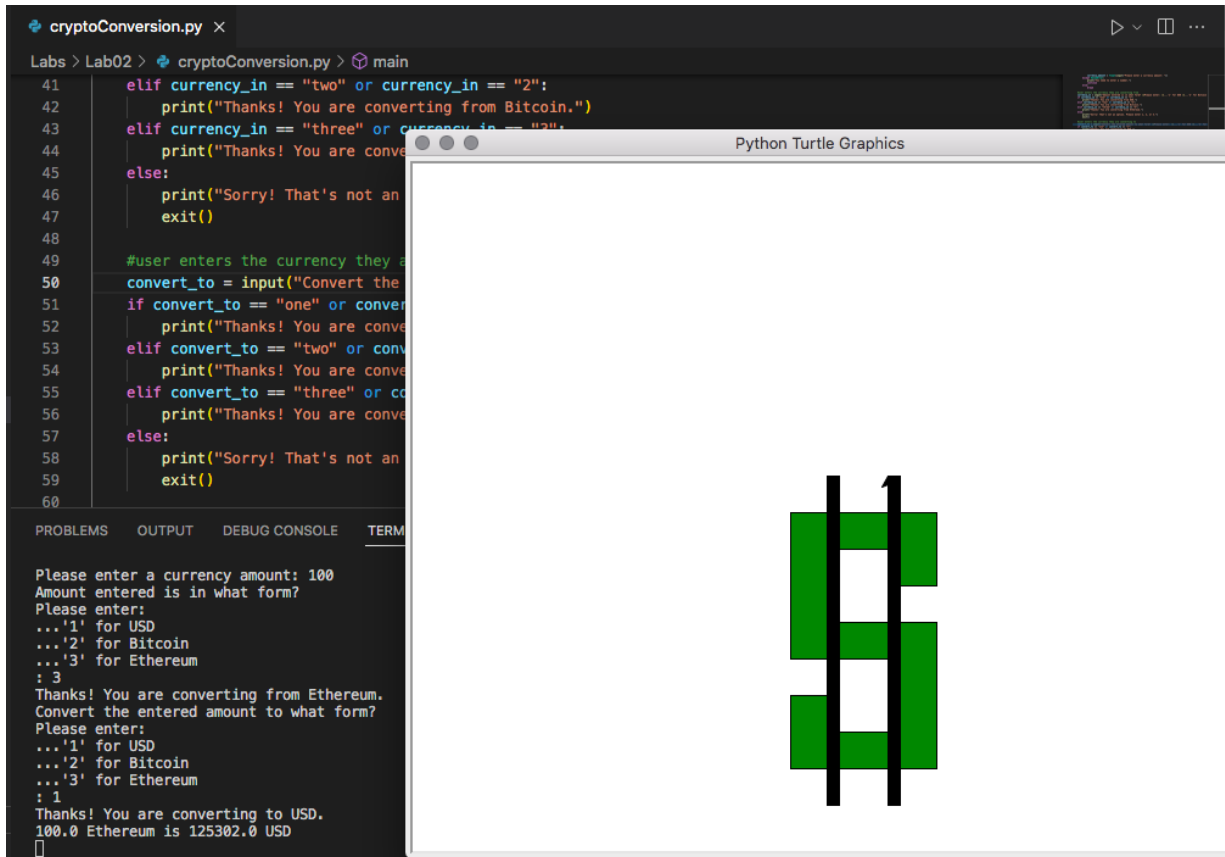


The screenshot displays a Python IDE with a file named `cryptoConversion.py`. The code is a script that prompts the user for a currency amount and conversion type. It includes conditional logic to handle different currencies and conversion directions. A `Python Turtle Graphics` window is open, showing a large orange circle with a white 'B' inside, representing the Bitcoin logo. The terminal window shows the program's execution, including user input and the final output: `100.0 USD is 0.005399999999999999 Bitcoin`.

```
cryptoConversion.py x
Labs > Lab02 > cryptoConversion.py > main
41 elif currency_in == "two" or currency_in == "2":
42     print("Thanks! You are converting from Bitcoin.")
43 elif currency_in == "three" or currency_in == "3":
44     print("Thanks! You are converting from Ethereum.")
45 else:
46     print("Sorry! That's not an option.")
47     exit()
48
49 #user enters the currency they want to convert to
50 convert_to = input("Convert the amount to what form? ")
51 if convert_to == "one" or convert_to == "1":
52     print("Thanks! You are converting to USD.")
53 elif convert_to == "two" or convert_to == "2":
54     print("Thanks! You are converting to Bitcoin.")
55 elif convert_to == "three" or convert_to == "3":
56     print("Thanks! You are converting to Ethereum.")
57 else:
58     print("Sorry! That's not an option.")
59     exit()
60
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Please enter a currency amount: 100
Amount entered is in what form?
Please enter:
... '1' for USD
... '2' for Bitcoin
... '3' for Ethereum
: 1
Thanks! You are converting from USD.
Convert the entered amount to what form?
Please enter:
... '1' for USD
... '2' for Bitcoin
... '3' for Ethereum
: 2
Thanks! You are converting to Bitcoin.
100.0 USD is 0.005399999999999999 Bitcoin
```

TRIAL 2 (Ethereum to USD)

This shows my calculator converting 100 Ethereum to USD, which turns out to be about 125,382 USD. Once it converts, python draws the US dollar sign. You can see some of my extension work here, which I detailed in my trial one statement.



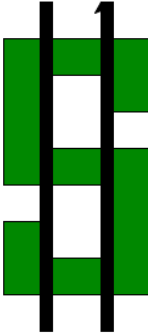
The screenshot displays a Python IDE with a file named `cryptoConversion.py`. The code is a script that prompts the user for a currency amount and its form (USD, Bitcoin, or Ethereum). It then prompts for the target currency (USD, Bitcoin, or Ethereum) and performs the conversion. The output shows the conversion of 100 Ethereum to USD, resulting in 125,382.0 USD. A Python Turtle Graphics window is also open, displaying a green dollar sign graphic.

```
cryptoConversion.py x
Labs > Lab02 > cryptoConversion.py > main
41 elif currency_in == "two" or currency_in == "2":
42     print("Thanks! You are converting from Bitcoin.")
43 elif currency_in == "three" or currency_in == "3":
44     print("Thanks! You are converting from Ethereum.")
45 else:
46     print("Sorry! That's not a valid currency form.")
47     exit()
48
49 #user enters the currency they want to convert to
50 convert_to = input("Convert the entered amount to what form? ")
51 if convert_to == "one" or convert_to == "1":
52     print("Thanks! You are converting to USD.")
53 elif convert_to == "two" or convert_to == "2":
54     print("Thanks! You are converting to Bitcoin.")
55 elif convert_to == "three" or convert_to == "3":
56     print("Thanks! You are converting to Ethereum.")
57 else:
58     print("Sorry! That's not a valid currency form.")
59     exit()
60
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL**

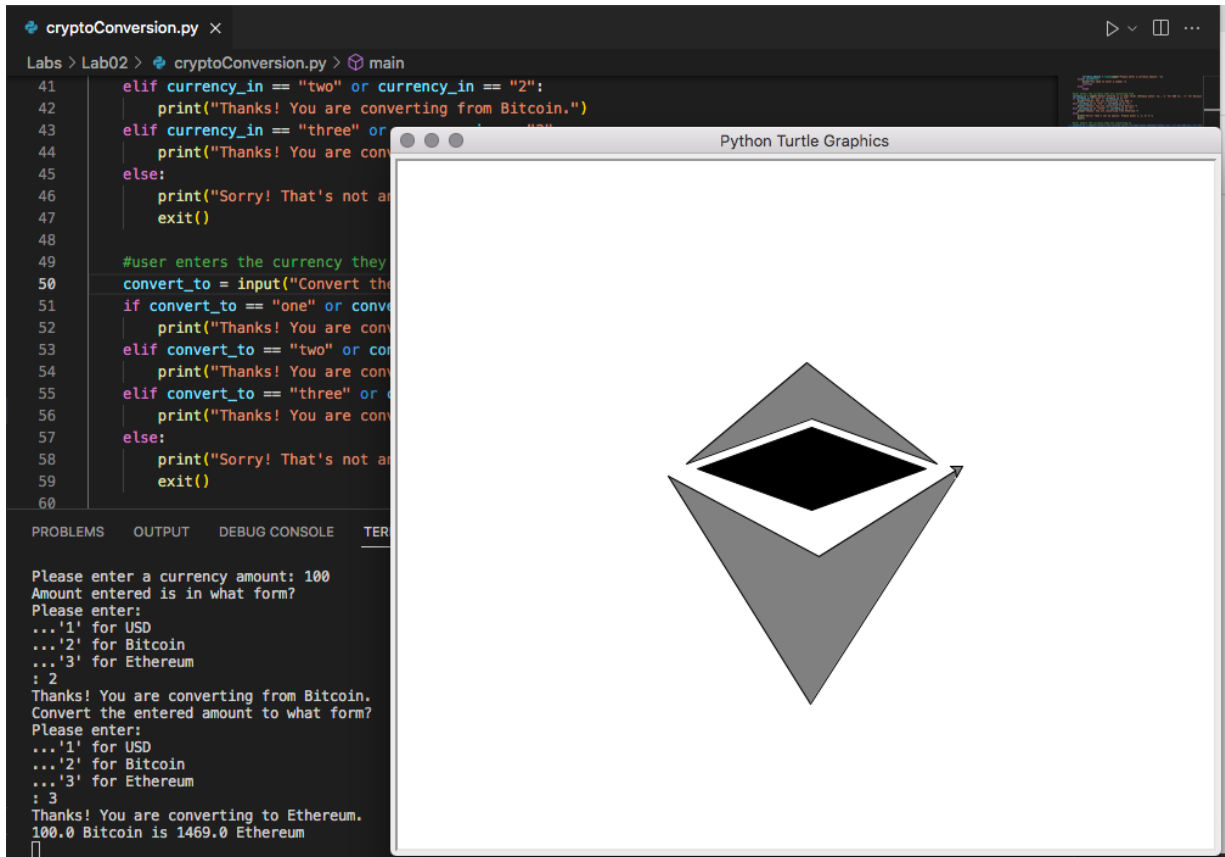
```
Please enter a currency amount: 100
Amount entered is in what form?
Please enter:
... '1' for USD
... '2' for Bitcoin
... '3' for Ethereum
: 3
Thanks! You are converting from Ethereum.
Convert the entered amount to what form?
Please enter:
... '1' for USD
... '2' for Bitcoin
... '3' for Ethereum
: 1
Thanks! You are converting to USD.
100.0 Ethereum is 125382.0 USD
█
```

Python Turtle Graphics



TRIAL 3 (Bitcoin to Ethereum)

This shows my calculator converting 100 Bitcoin to Ethereum, which turns out to be about 1469 Ethereum. Once it converts, python draws the Ethereum logo. You can see some of my extension work here, which I detailed in my trial one statement.

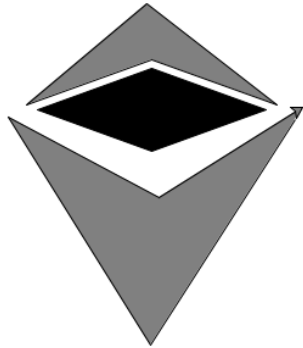


```
cryptoConversion.py x
Labs > Lab02 > cryptoConversion.py > main
41 elif currency_in == "two" or currency_in == "2":
42     print("Thanks! You are converting from Bitcoin.")
43 elif currency_in == "three" or currency_in == "3":
44     print("Thanks! You are converting from Ethereum.")
45 else:
46     print("Sorry! That's not a valid currency.")
47     exit()
48
49 #user enters the currency they want to convert to
50 convert_to = input("Convert the entered amount to what form? ")
51 if convert_to == "one" or convert_to == "1":
52     print("Thanks! You are converting to USD.")
53 elif convert_to == "two" or convert_to == "2":
54     print("Thanks! You are converting to Bitcoin.")
55 elif convert_to == "three" or convert_to == "3":
56     print("Thanks! You are converting to Ethereum.")
57 else:
58     print("Sorry! That's not a valid currency.")
59     exit()
60
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Please enter a currency amount: 100
Amount entered is in what form?
Please enter:
... '1' for USD
... '2' for Bitcoin
... '3' for Ethereum
: 2
Thanks! You are converting from Bitcoin.
Convert the entered amount to what form?
Please enter:
... '1' for USD
... '2' for Bitcoin
... '3' for Ethereum
: 3
Thanks! You are converting to Ethereum.
100.0 Bitcoin is 1469.0 Ethereum

Python Turtle Graphics



3. Extensions

In addition to the extensions described above (colored logos, confirmation message, final calculation, etc.) I created a while statement for the first part of the program. If the user inputs a value that cannot be converted to a float value, this will trigger a Value Error. If that happens, the program will run a message to the user reminding them that the input must be a number. The program will continue asking for a number until it receives one, creating a loop until the correct value is entered.

```
#user enters $$$ to convert from one currency to another
#if an invalid value is entered the program will keep asking until we get a number
while True:
    try:
        currency_amount = float(input("Please enter a currency amount: "))
    except ValueError:
        print("You need to enter a number.")
        continue
    else:
        break
```

```
Please enter a currency amount: hi
You need to enter a number.
Please enter a currency amount: what is numbr
You need to enter a number.
Please enter a currency amount: oh ok
You need to enter a number.
Please enter a currency amount: 1
Amount entered is in what form? Enter '1' for USD, '2' for Bitcoin and '3' for Ethereum: 2
Thanks! You are converting from Bitcoin.
Amount entered is in what form? Enter '1' for USD, '2' for Bitcoin and '3' for Ethereum: 3
Thanks! You are converting to Ethereum.
1.0 Bitcoin is 14.69 Ethereum
```

Similarly, the second and third part of the program will not accept a value outside of the range 1-3. Entering a value that does not fit these qualifications will prompt a message letting the user know they need to input either 1, 2, or 3 and terminate the program. Additionally, if the user was to input a qualifying string value (a “one” instead of a 1, for example) the program will accept this value.

```
#user enters the currency they are converting from
currency_in = input("Amount entered is in what form? \nPlease enter: \n...'1' for USD \n...'2' for Bitcoin\n...'3' for Ethereum\n:")
if currency_in == "one" or currency_in == "1":
    print("Thanks! You are converting from USD.")
elif currency_in == "two" or currency_in == "2":
    print("Thanks! You are converting from Bitcoin.")
elif currency_in == "three" or currency_in == "3":
    print("Thanks! You are converting from Ethereum.")
else:
    print("Sorry! That's not an option. Please enter 1, 2, or 3.")
    exit()

#user enters the currency they are converting to
convert_to = input("Convert the entered amount to what form? \nPlease enter: \n...'1' for USD \n...'2' for Bitcoin\n...'3' for Ethereum\n:")
if convert_to == "one" or convert_to == "1":
    print("Thanks! You are converting to USD.")
elif convert_to == "two" or convert_to == "2":
    print("Thanks! You are converting to Bitcoin.")
elif convert_to == "three" or convert_to == "3":
    print("Thanks! You are converting to Ethereum.")
else:
    print("Sorry! That's not an option. Please enter 1, 2, or 3.")
    exit()
```

```
Please enter a currency amount: 60
Amount entered is in what form?
Please enter:
...'1' for USD
...'2' for Bitcoin
...'3' for Ethereum
: one
Thanks! You are converting from USD.
Convert the entered amount to what form?
Please enter:
...'1' for USD
...'2' for Bitcoin
...'3' for Ethereum
: 4
Sorry! That's not an option. Please enter 1, 2, or 3.
```

4. Tests

actual input	currency_in	convert_to	CONSTANT	expected output	Pass/Fail?
100	USD	USD	1	100	P
100	Bitcoin	Bitcoin	1	100	P
100	Ethereum	Ethereum	1	100	P
100	USD	Bitcoin	0.000054	.0054	P*
100	USD	Ethereum	0.00080	.08	P
100	Bitcoin	USD	18459.50	1,845,950	P
100	Bitcoin	Ethereum	14.69	1469	P
100	Ethereum	USD	1253.02	125,302	P
100	Ethereum	Bitcoin	0.068	6.8	P*

*These have expected values that are extremely close to the actual output, and when rounded they are the same. I'm not sure why the program would return 0.005399999999999999 instead of .0054 and 6.8000000000000001 instead of just 6.8. It only happened to the values that were converting to Bitcoin, so I wonder if it's because Bitcoin conversion rates were either so much smaller or larger than the others? I did a [little research](#) on this and saw that this is a common issue with how python stores float numbers and how computers use binary approximation to create values.

5. Grading Statement

I would give myself a 28 for this lab. Overall, this was a time consuming lab that I worked hard to complete, but there were still a few things I would have changed if I had more time. I completed the main objectives, and my program works, but I would have liked to figure out how to import my python turtle programs into the main currencyConverter.py file rather than inputting the code directly. I realize that the way I achieved the logo part of the program isn't ideal because it creates a lot of repetitive code that makes the program as a whole difficult to read. Additionally, I would have liked to find a workaround for the decimal issue described in my testing above. I assume python has a rounding function, but I'll save that for another time. Admittedly, my logos also could have been drawn a little better. Despite this, I believe my code is organized and commented to show what each part of the program does. I also included many extensions for this lab that I think make my program a little more sophisticated. I've included the rubric below with my own assessment of my work.

	Possible	Estimated
Main Objectives		
Gather user input correctly	4	4
Uses menus as requested	4	4
Converts as requested	3	2
Prints logos	3	2
Terminates on incorrect input	3	3
Misc		
Report (all or nothing)	5	5
Code Quality (correct indentation, comment blocks, variable naming, etc)	4	3
Not included in total possible:		
Extensions (Not calculated without report)	4	3
Creative or went above and beyond	4	2
Code does not compile	-30	0
Late penalty	-6	0
Not implemented as requested	-30	0
TOTAL POINTS POSSIBLE out of 30	26	2