Pyscripter issues when using Turtles

Pyscripter is a very good programming environment – but there can be some hangups using it with programs that require user input and that also use Turtle graphics. Here is some guidance that may ameliorate some of those issues.

Part 1 –

- In Pyscripter, select *Run* from the menu bar.
- Choose *Python Engine*; make sure that *Internal* is checked

Part 2 –

- If your program gets into a state where you can't close the graphics window, try this. Open the *Python Interpreter* window in Pyscripter. You should see a Python shell prompt ">>>". Type turtle.bye() and press enter. The graphics window should close.
- For some tasks you can have your program read its input data before the graphics window is created. This will reduce problems with the input dialog box being hidden behind the graphics window. You will see this approach in part III of this lab. However it doesn't apply to your current project (Connect4 board) because the program needs to work with both input dialog and the graphics window simultaneously.

Instructions

- Complete this lab report during the lab period on Wednesday or hand in your completed report by 5pm Friday. My mail box is in Manchester 233. The two programming exercises described in sections II and III of this handout should be submitted through Sakai's Assignment link by 5pm Friday.
- Program files submitted through the Sakai Assignment page must use the following naming format:
    - Lastname_Firstinitial_Lab_lab#_problem#.py
    - For example, for Lab #3, problem #2, my file would be named: *Thomas_S_Lab_3_2.py*
- All programming assignments must follow the Style Guide and include meaningful comments.

I. Look up the purpose of each of the following turtle methods either by reading the documentation starting at http://docs.python.org/3.3/library/ or by your own means. Describe the purpose of each method.

- turtle.stamp()

  ____

  ____

- turtle.dot()

  ____

  ____

- turtle.home()

  ____

- turtle.reset()

  ____

  ____

- turtle.bye()

  ____

II. The following program works correctly. It asks the user to type in the amount of a bill and the amount paid and it determines how to make change using the smallest possible number of dollars, quarters, dimes, nickels, and pennies. However, it violates several points regarding good programming style. Your task is to make it readable and understandable. In other words, you need to give it some style points. Specifically, you need to:

- replace the variable names `c, r, ll, q, d, n,` and `p` by meaningful variable names,
- add comments that explain what the program does in important steps
- add a well-structured preamble (introduction)

You can download the source code from Sakai (Lab_3_2) to get started. Submit your "improved" version through Sakai → Assignments using the correct file naming convention.

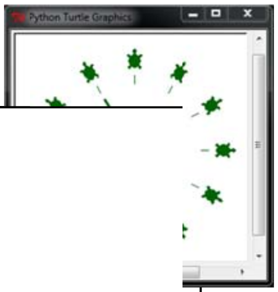
```
# Get the amount of the bill
amount_due = eval(input("How much was the bill? "))

# and the amount paid
amount_paid = eval(input("How much was paid? "))

c = amount_paid - amount_due
r = int(c * 100)
ll = r // 100
r = r % 100
q = r // 25
r = r % 25
d = r // 10
r = r % 10
n= r // 5
r = r % 5
p = r

print("Amount tendered: $", amount_paid)
print("           Bill: $", amount_due)
print("Give back:")
print(ll,"dollars")
print(q, "quarters")
print(d,  "dimes")
print(n, "nickels")
print(p, "pennies")
```

III. In this exercise you are asked to complete a program that will display a "turtle clock." Step 1 – display a clock face using turtle graphics (see example). Step 2 – Add an hour hand. Step 3 – Add a minute hand. You can download the beginning source code from Sakai as Lab_3_3. When finished, upload your program to Sakai → Assignments with the proper name.


Python Turtle Graphics