Προγραμματισμός και Εφαρμογές με την Python Turtle

Δαδαλιάρης Αντώνιος (dadaliaris@uth.gr)

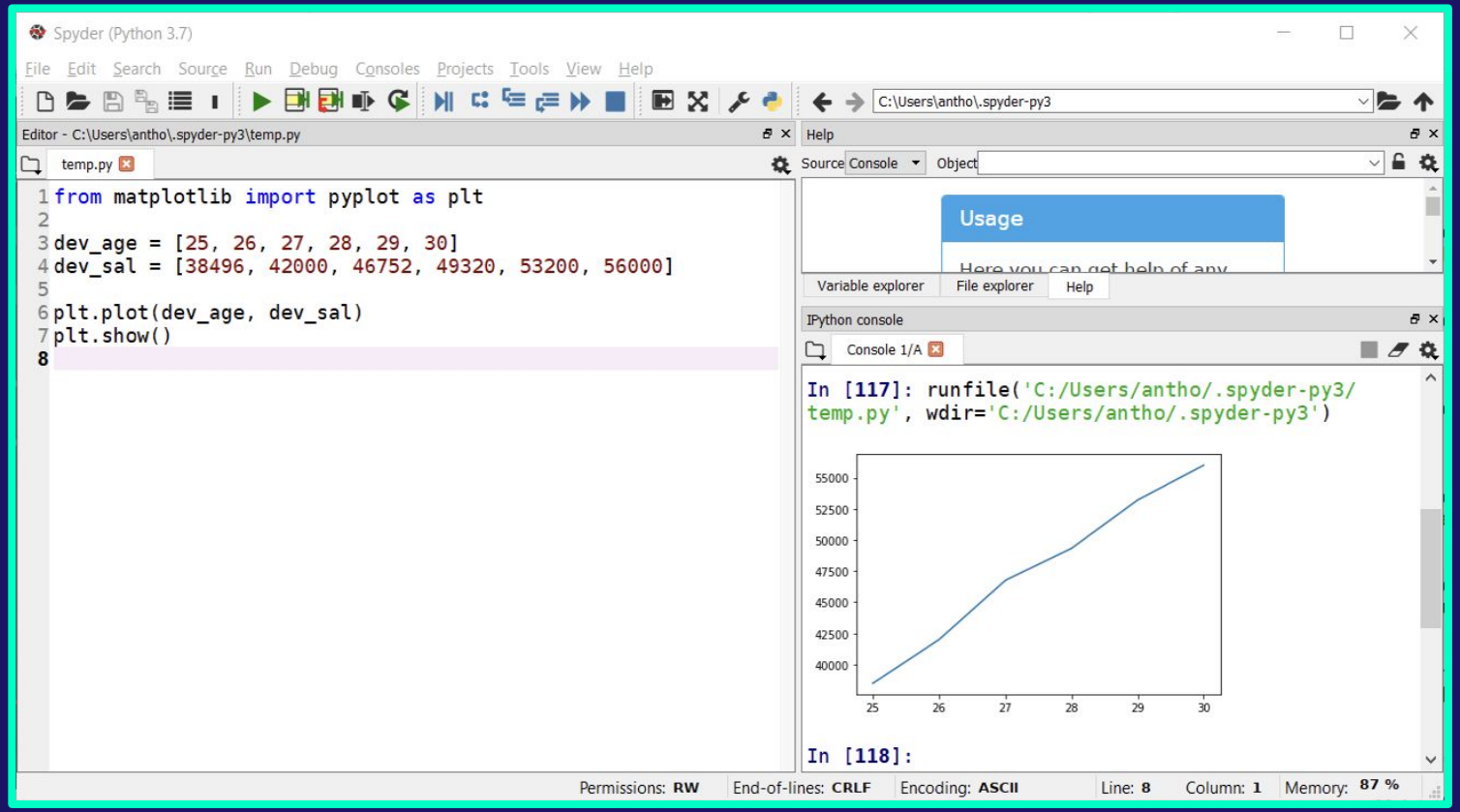
02

Matplotlib

Matplotlib (1)

- “Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and for graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.“ [https://matplotlib.org/]

Matplotlib (2)



Matplotlib (3)

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\antho\.spyder-py3

Editor - C:\Users\antho\.spyder-py3\temp.py

temp.py

```
1 """ Bar Charts """
2
3 # Simple Bar Chart
4 from matplotlib import pyplot as plt
5
6 dev_age = [25, 26, 27, 28, 29, 30]
7 dev_sal = [39000, 42000, 47000, 50000, 52000, 56000]
8 pydev_sal = [40000, 44000, 50000, 53000, 55000, 60000]
9
10 plt.bar(dev_age, dev_sal, color="r", label="Devs")
11
12 plt.legend()
13
14 plt.title("Developers")
15 plt.xlabel("Ages")
16 plt.ylabel("Salaries")
17
18 plt.show()
19
```

Help

Source Console Object

Usage

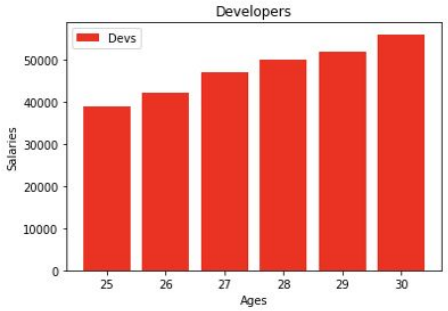
Here you can get help of any

Variable explorer File explorer Help

IPython console

Console 1/A

```
temp.py', wdir='C:/Users/antho/.spyder-py3')
```




```
In [143]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 19 Column: 1 Memory: 82 %

Matplotlib (4)

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\antho\.spyder-py3

Editor - C:\Users\antho\.spyder-py3\temp.py

temp.py

```
""" Bar Charts """

# Complex Bar Chart
import numpy as np
from matplotlib import pyplot as plt

dev_age = [25, 26, 27, 28, 29, 30]
dev_sal = [39000, 42000, 47000, 50000, 52000, 56000]
pydev_sal = [40000, 44000, 50000, 53000, 55000, 60000]
javadev_sal = [32000, 40000, 51000, 50000, 50000, 65000]

x_index = np.arange(len(dev_age))
width = 0.2

plt.bar(x_index - width, dev_sal, width=width, color="r", label="Devs")
plt.bar(x_index, pydev_sal, width=width)
plt.bar(x_index + width, javadev_sal, width=width)
plt.legend()

# Ticks should be used to correctly prompt x axis values
plt.xticks(ticks=x_index, labels=dev_age)
plt.title("Developers")
plt.xlabel("Ages")
plt.ylabel("Salaries")

plt.show()
```

Help

Source Console Object

Usage

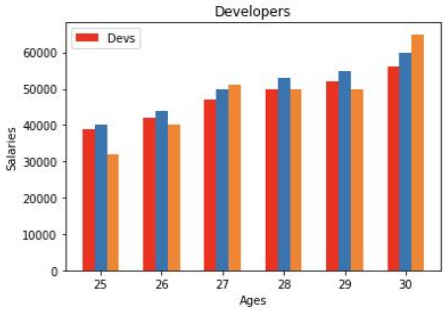
Here you can get help

Variable explorer File explorer Help

IPython console

Console 1/A

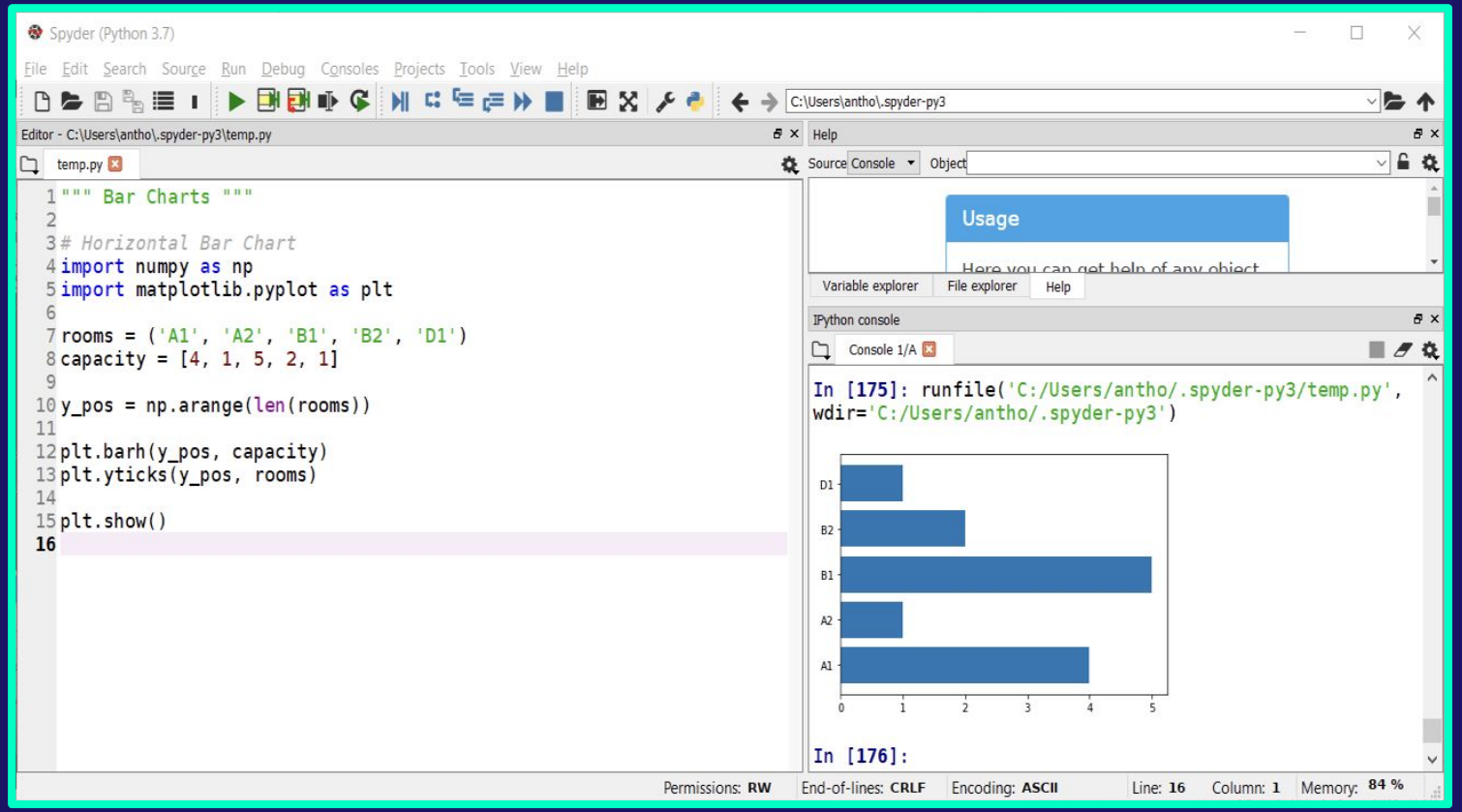
Users/antho/.spyder-py3')



In [152]:

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 27 Column: 1 Memory: 79 %

Matplotlib (5)



Matplotlib (6)

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\antho\.spyder-py3

Editor - C:\Users\antho\.spyder-py3\temp.py

temp.py

```
1 from matplotlib import pyplot as plt
2
3 dev_age = [25, 26, 27, 28, 29, 30]
4 dev_sal = [38496, 42000, 46752, 49320, 53200, 56000]
5
6 # Adding legend
7 plt.plot(dev_age, dev_sal, label="all developers")
8
9 py_dev_sal = [45372, 48876, 53850, 57287, 63016, 65998]
10
11 # Adding legend
12 plt.plot(dev_age, py_dev_sal, label="python developers")
13
14 # Define axes names
15 plt.xlabel("Ages")
16 plt.ylabel("Salaries")
17
18 # Define a title for the plot
19 plt.title("Developer salaries by age")
20
21 # Show legends
22 plt.legend()
23
24 plt.show()
25
```

Help

Source Console Object

Usage

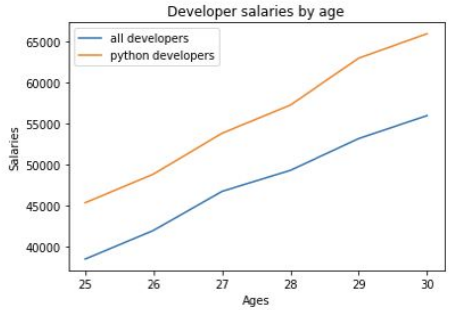
Here you can get help of any

Variable explorer File explorer Help

IPython console

Console 1/A

```
temp.py', wdir='C:/Users/antho/.spyder-py3')
```




```
In [123]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 25 Column: 1 Memory: 83 %

Matplotlib (7)

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\antho\.spyder-py3

Editor - C:\Users\antho\.spyder-py3\temp.py

temp.py

```
1 from matplotlib import pyplot as plt
2
3 dev_age = [25, 26, 27, 28, 29, 30]
4 dev_sal = [38496, 42000, 46752, 49320, 53200, 56000]
5
6 # Adding legend & formatting
7 plt.plot(dev_age, dev_sal, color = 'g', label="all developers")
8
9 py_dev_sal = [45372, 48876, 53850, 57287, 63016, 65998]
10
11 # Adding legend & formatting
12 plt.plot(dev_age, py_dev_sal, linestyle='--', label="python developers")
13
14 # Define axes names
15 plt.xlabel("Ages")
16 plt.ylabel("Salaries")
17
18 # Define a title for the plot
19 plt.title("Developer salaries by age")
20
21 # Show legends
22 plt.legend()
23
24 plt.show()
25
```

Help

Source Console Object

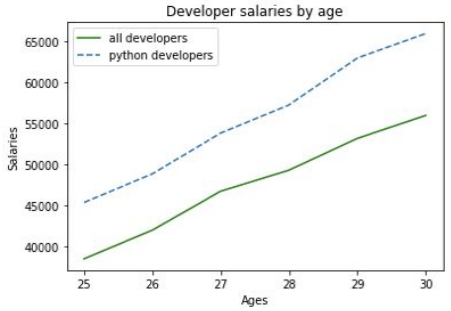
Usage

Here you can get help

Variable explorer File explorer Help

IPython console

Console 1/A



In [126]:

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 25 Column: 1 Memory: 87 %

Matplotlib (8)

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\antho\.spyder-py3

Editor - C:\Users\antho\.spyder-py3\temp.py

temp.py

```
1 from matplotlib import pyplot as plt
2
3 dev_age = [25, 26, 27, 28, 29, 30]
4 dev_sal = [38496, 42000, 46752, 49320, 53200, 56000]
5
6 # Adding legend & formatting
7 plt.plot(dev_age, dev_sal, color = 'g', label="all developers")
8
9 py_dev_sal = [45372, 48876, 53850, 57287, 63016, 65998]
10
11 # Adding legend & formatting
12 plt.plot(dev_age, py_dev_sal, linestyle='--', label="python developers")
13
14 # Define axes names
15 plt.xlabel("Ages")
16 plt.ylabel("Salaries")
17
18 # Define a title for the plot
19 plt.title("Developer salaries by age")
20
21 # Show legends & grid
22 plt.legend()
23 plt.grid(True)
24 plt.show()
25
```

Help

Source Console Object

Usage

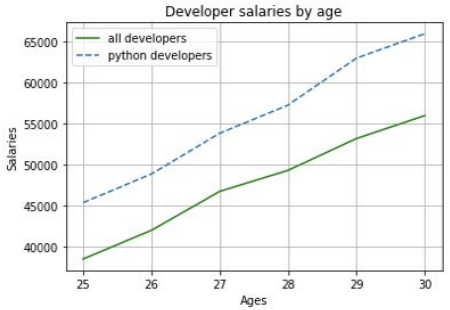
Here you can get help

Variable explorer File explorer Help

IPython console

Console 1/A

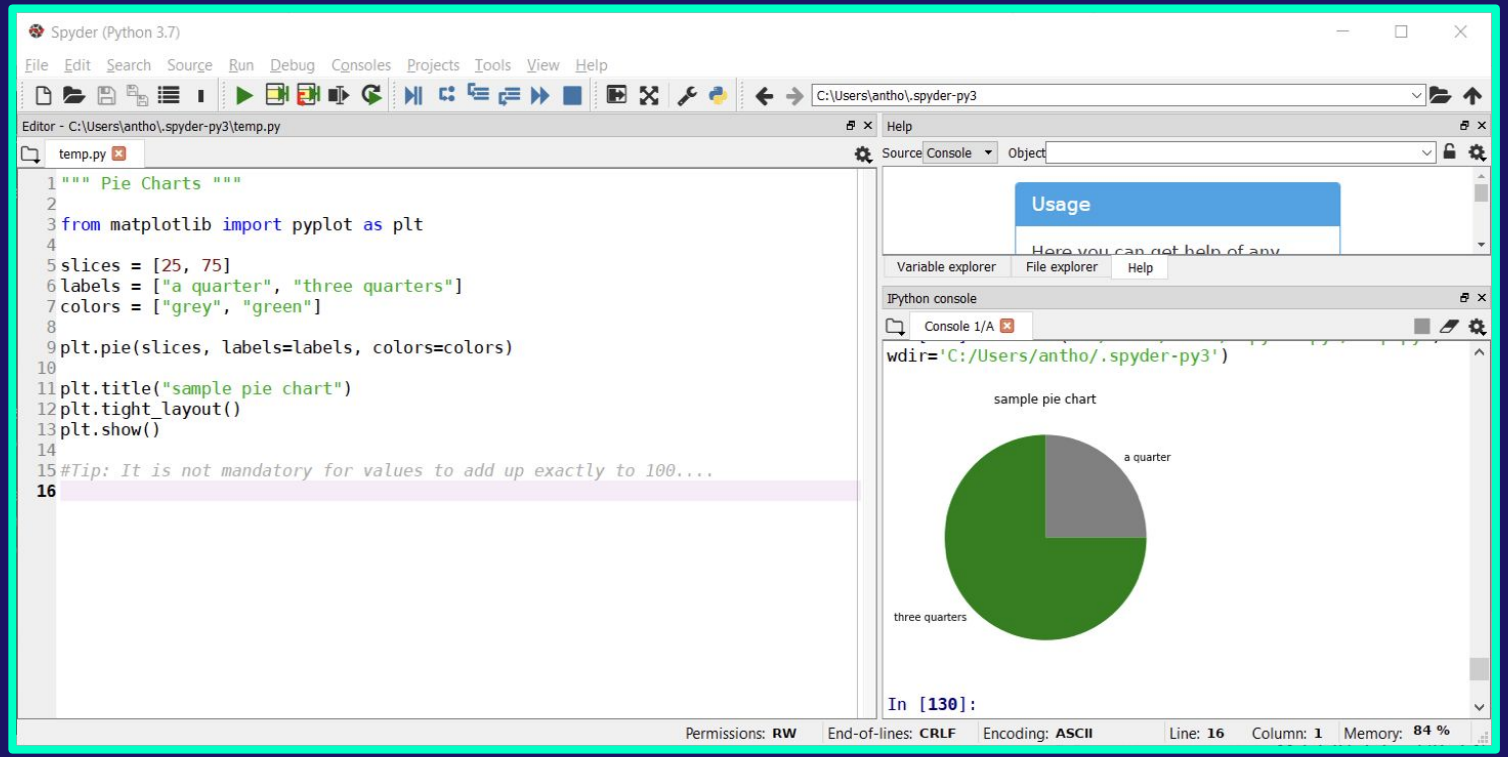
Users/antho/.spyder-py3')



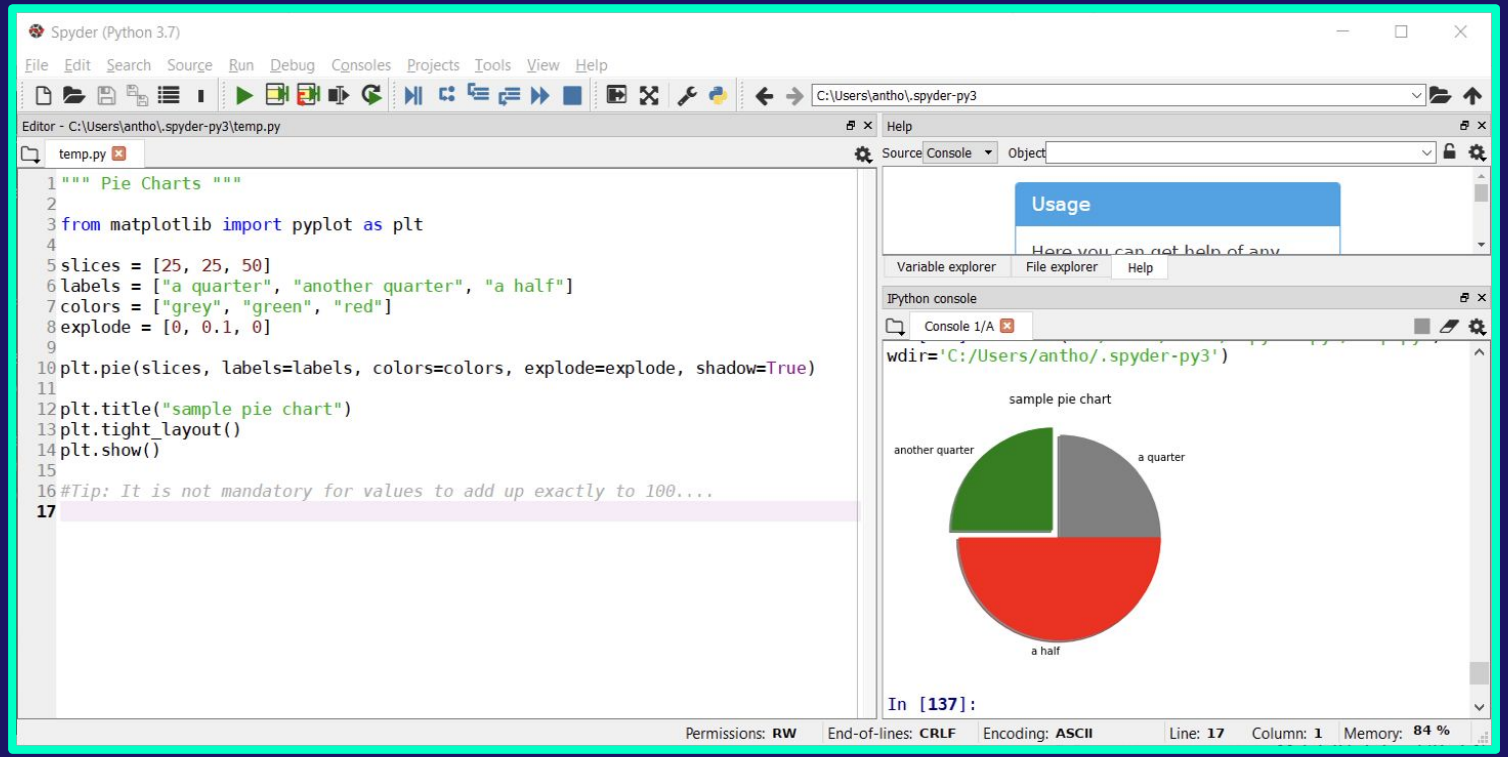
In [127]:

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 25 Column: 1 Memory: 83 %

Matplotlib (9)



Matplotlib (10)



Matplotlib (11)

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\antho\.spyder-py3

Editor - C:\Users\antho\.spyder-py3\temp.py

temp.py

```
""" Stack Plots """

from matplotlib import pyplot as plt

minute = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
pm =     [0, 1, 1, 2, 3, 4, 5, 1, 1, 8]
sg =     [1, 1, 2, 2, 3, 3, 4, 5, 1, 2]
sf =     [1, 1, 2, 2, 2, 2, 2, 4, 4, 2]
pf =     [1, 2, 2, 3, 1, 1, 1, 1, 2, 2]

labels = ["pm", "sg", "sf", "pf"]

plt.stackplot(minute, pm, sg, sf,pf, labels=labels)

plt.legend(loc="upper left")
plt.title("sample stack plot")
plt.tight_layout()
plt.show()

```

Help

Source Console Object

Usage

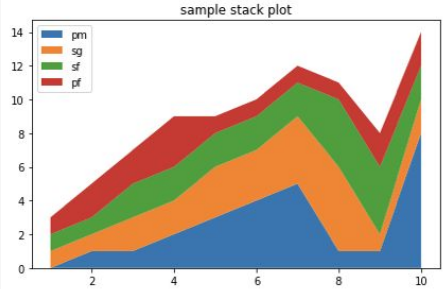
Here you can get help of any

Variable explorer File explorer Help

IPython console

Console 1/A

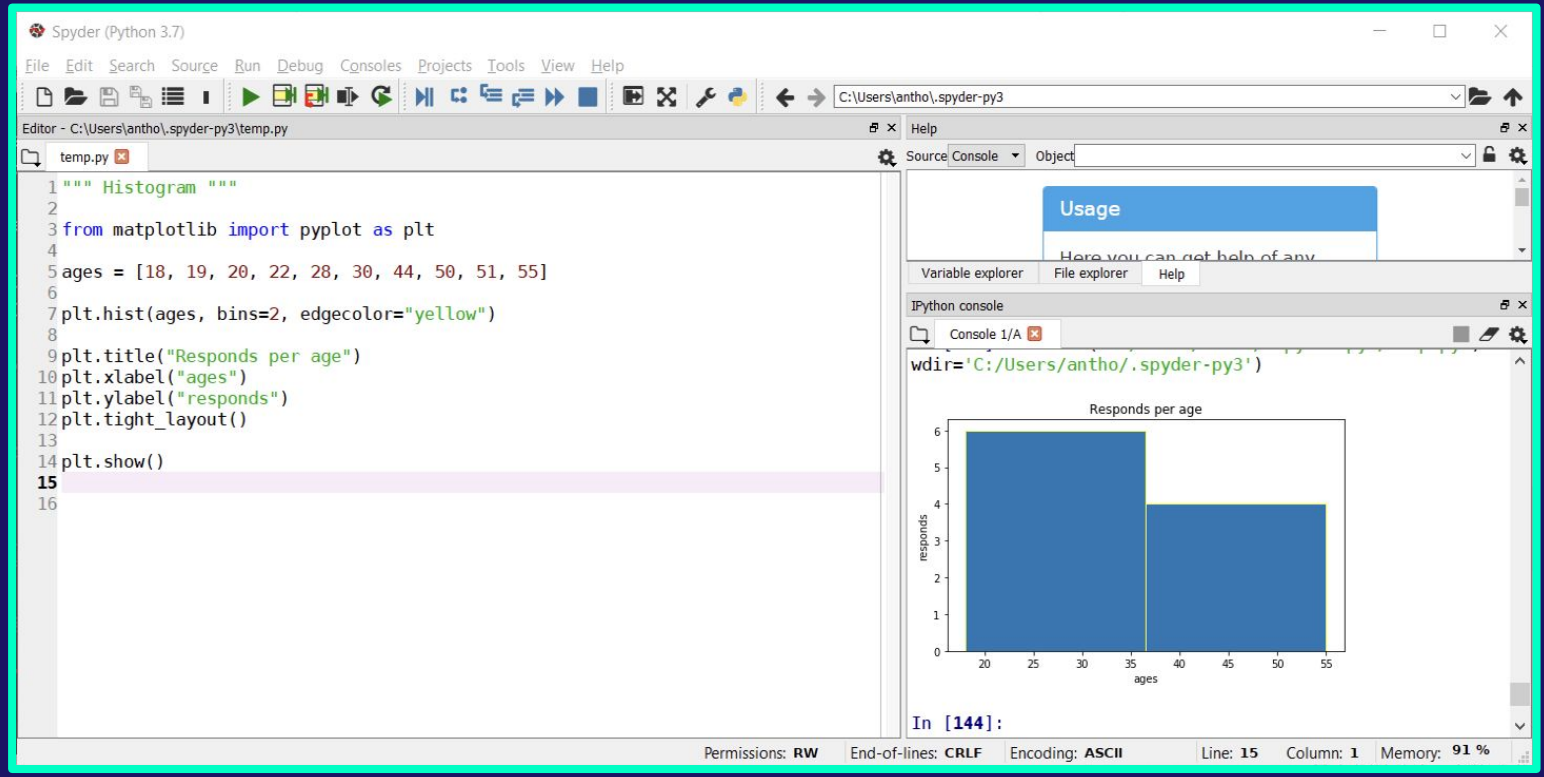
```
wdir='C:/Users/antho/.spyder-py3')
```



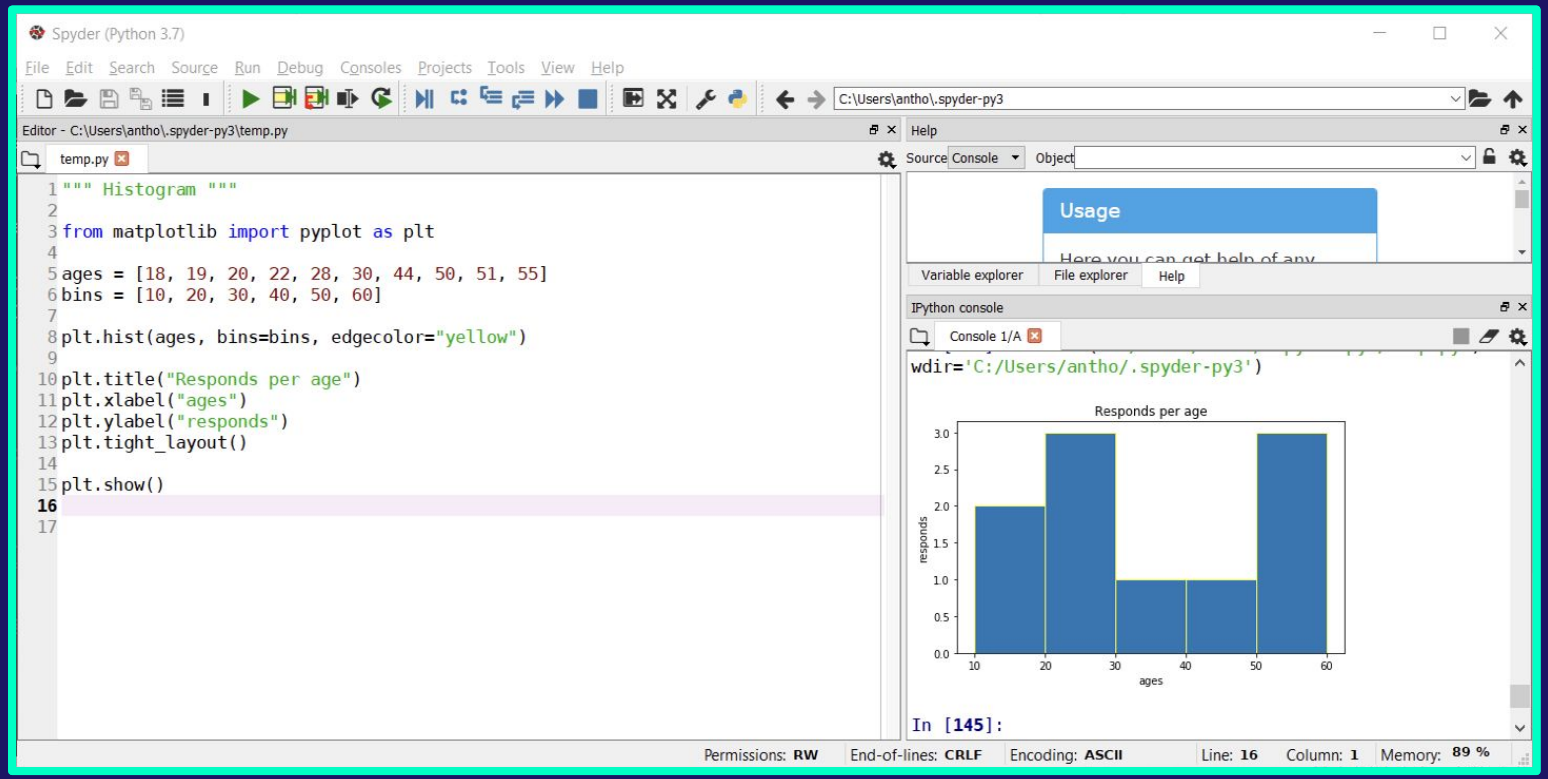

```
In [141]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 19 Column: 1 Memory: 85 %

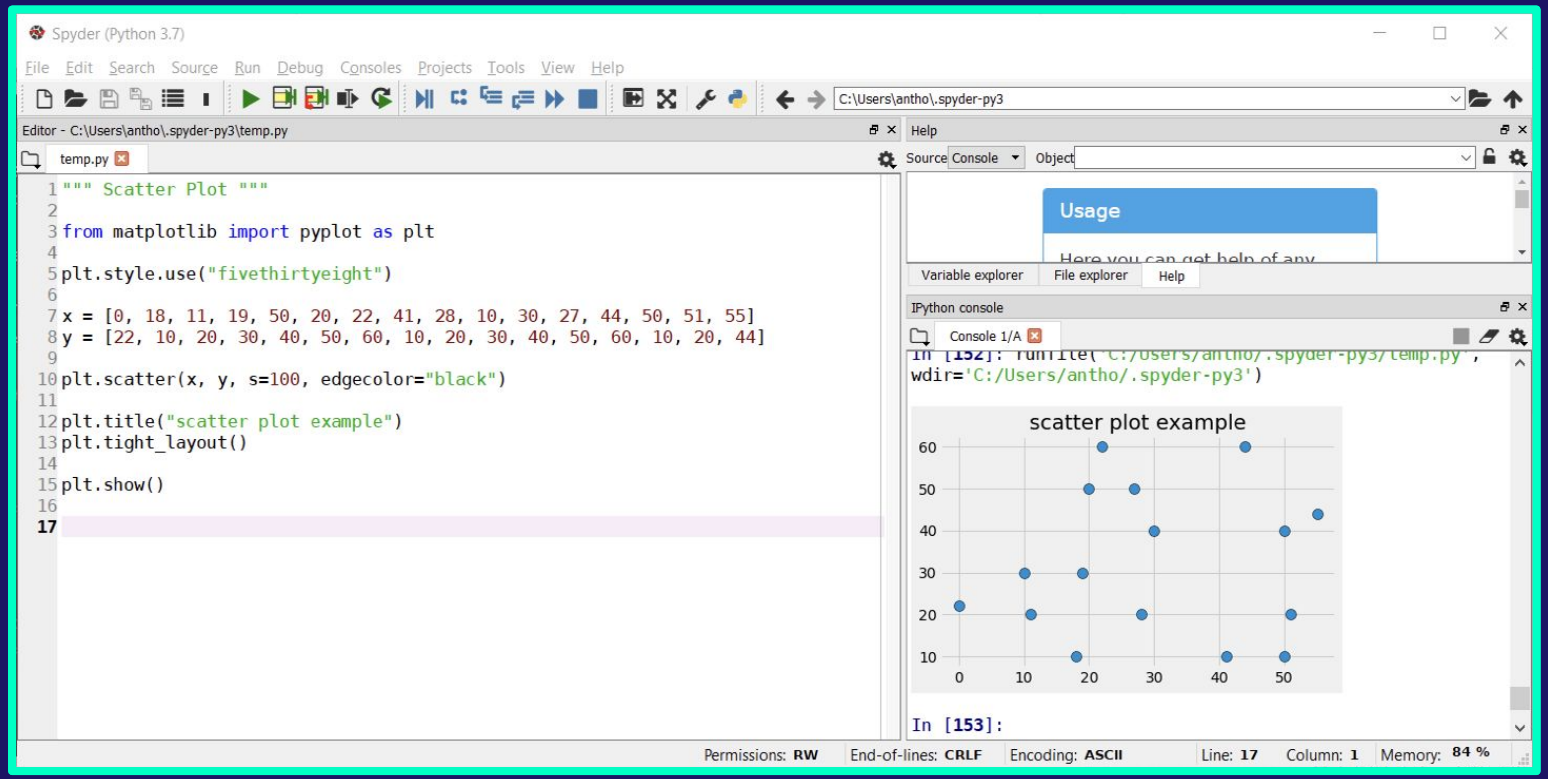
Matplotlib (12)



Matplotlib (13)



Matplotlib (14)



Matplotlib (15)

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\antho\.spyder-py3

Editor - C:\Users\antho\.spyder-py3\temp.py

temp.py

```
1 """ Scatter Plot """
2
3 from matplotlib import pyplot as plt
4
5 plt.style.use("fivethirtyeight")
6
7 x = [0, 18, 11, 19, 50, 20, 22, 41, 28, 10, 30, 27, 44, 50, 51, 55]
8 y = [22, 10, 20, 30, 40, 50, 60, 10, 20, 30, 40, 50, 60, 10, 20, 44]
9 sizes = [22, 210, 120, 230, 140, 450, 560, 110, 520, 130, 540, 150, 160, 110, 120, 144]
10
11 colors = [0, 8, 1, 9, 0, 0, 2, 1, 8, 10, 10, 7, 4, 10, 1, 5]
12
13 plt.scatter(x, y, s=sizes, c=colors, cmap="Spectral", edgecolor="black")
14
15 color_bar = plt.colorbar()
16 color_bar.set_label("answers")
17
18 plt.title("scatter plot example")
19 plt.tight_layout()
20
21 plt.show()
22
23
```

Help

Source Console Object

Usage

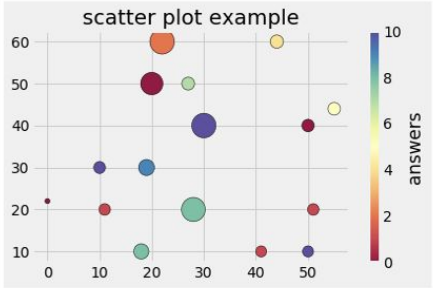
Here you can get help of

Variable explorer File explorer Help

IPython console

Console 1/A

```
In [159]: runfile('C:/Users/antho/.spyder-py3/temp.py', wdir='C:/Users/antho/.spyder-py3')
```




```
In [160]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 22 Column: 1 Memory: 82 %

Matplotlib (16)

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\antho\.spyder-py3

Editor - C:\Users\antho\.spyder-py3\temp.py

temp.py

```
""" Subplots """

from matplotlib import pyplot as plt

plt.style.use("fivethirtyeight")

ages = [20, 25, 30, 35, 40, 45, 50, 55, 60]
dev_wages = [2000, 2500, 3000, 3400, 3900, 4200, 4400, 5000, 5200]
py_wages = [2500, 2200, 3200, 3200, 3700, 4000, 4800, 5500, 5900]
ja_wages = [1700, 2500, 3000, 3400, 3900, 4200, 4400, 5000, 5200]

fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1)

ax1.plot(ages, dev_wages, label="all devs")
ax2.plot(ages, py_wages, label="python devs")
ax2.plot(ages, ja_wages, label="java devs")

ax1.legend()
ax1.set_ylabel("wage")

ax2.legend()
ax2.set_xlabel("age")
ax2.set_ylabel("wage")

#plt.tight_layout()
plt.show()
```

Help

Source Console Object

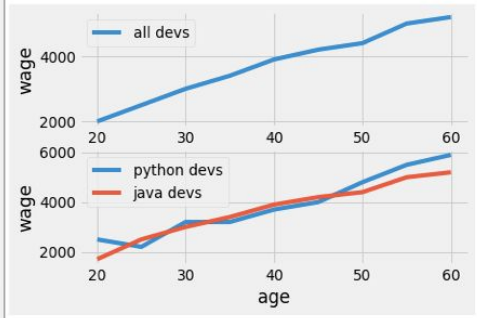
Usage

Here you can get help of any object

Variable explorer File explorer Help

IPython console

Console 1/A




```
In [176]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 27 Column: 1 Memory: 81 %

A 5-step Guide to Data Visualization (www.elsevier.com)

1. Be clear on the question
2. Know your data & start with basic visualizations
3. Identify messages of the visualization, and generate the most informative indicator
4. Choose the right chart type
5. Use colour, size, scale, shapes & legends to direct attention to the key messages

03

Python Turtle

Python Turtle: Introduction (1)

- **Logo:**
  - Designed in 1967 by W. Feurzeig, S.Papert, and C. Solomon
  - Name derived from “Λόγος”
  - General purpose programming language
  - Better known for turtle graphics
  - Deprecated

- **Python Turtle library is kind of a port (practically an extended reimplementation) of Turtle Graphics in Python.**

- Roadmap:
  1. Understand
  2. Learn
  3. Program
  4. Develop

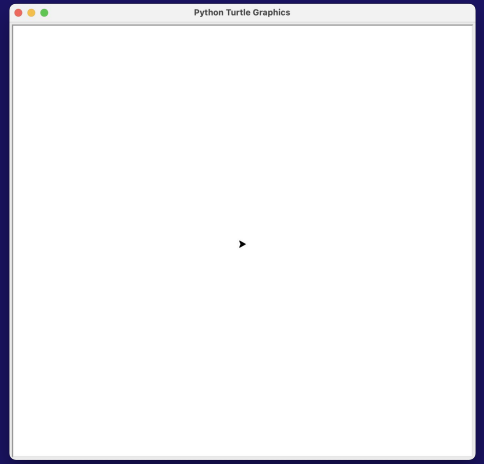
Python Turtle: Introduction (2)

- Turtle library is pre-installed and available in Python 3x versions.
- We just need to “import” it in our code (import turtle).
- It is used for creating pictures & shapes (or mini-games and animations) in a virtual canvas.
- The onscreen pen is “called” the turtle hence the library name.

Python Turtle: Introduction (3)

turtle_1.py | turtle_2.py

```
1  import turtle
2
3  # The first thing we have to do, is to create a window (that is called a screen).
4
5  # This can be done by calling the getscreen() function and assigning it to a
6  # variable.
7
8  # In this screen we can see the output of our code.
9  # get_screen() returns the TurtleScreen object the turtle is drawing on.
10 s = turtle.getscreen()
11
12 # Now by initializing another variable by calling the Turtle() method
13 # a black triangle is created in the middle of the screen, depicting
14 # our turtle.
15 t = turtle.Turtle()
16
17 # If the screen closes immediately after you run the code, include
18 # the following command.
19 turtle.mainloop()
```



Python Turtle: Introduction (4)

turtle_1.py | turtle_2.py

```
1   import turtle
2
3   t = turtle.Turtle()
4
5   # Let's begin by moving the turtle. The following example presents all the directions
6   # in which the turtle can move.
7   t.right(90)         # the turtle will turn right for 90 degrees
8   t.forward(200)      # the turtle will move forward (where it points) for 200 units
9   t.left(45)          # the turtle will turn left for 45 degrees
10  t.backward(100)     # the turtle will move backward (where it points) for 200 units
11
12  # For each of the aforementioned functions there is a "shorter" version considering
13  # their name.
14  #   t.rt()
15  #   t.fd()
16  #   t.lt()
17  #   t.bk()
18  turtle.mainloop()
```

Python Turtle: Introduction (5)

turtle_1.py | turtle_2.py | turtle_3.py

```
1  import turtle
2
3  t = turtle.Turtle()
4
5  # Based on the previous example we can draw a square using the
6  # following commands:
7  t.right(90)
8  t.forward(50)
9  t.right(90)
10 t.forward(50)
11 t.right(90)
12 t.forward(50)
13 t.right(90)
14 t.forward(50)
15
16 turtle.mainloop()
17
```

Python Turtle: Introduction (6)

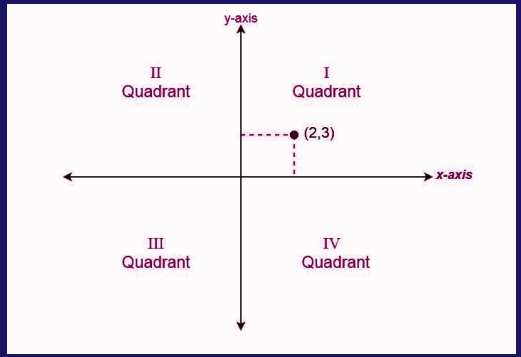
turtle_1.py | turtle_2.py | turtle_3.py | turtle_4.py | turtle_5.py

```
1  import turtle
2
3  t = turtle.Turtle()
4
5  # What should we change in order to draw a typical rectangle?
6  # following commands:
7  t.right(90)
8  t.forward(150)
9  t.right(90)
10 t.forward(50)
11 t.right(90)
12 t.forward(150)
13 t.right(90)
14 t.forward(50)
15
16 turtle.mainloop()
17
```

Python Turtle: Introduction (7)

turtle_1.py | turtle_2.py | turtle_3.py | turtle_4.py | turtle_5.py

```
1  import turtle
2
3  # An alternative way to draw in turtle is by using coordinates in combination
4  # with the goto() function.
5
6  # The coordinates are based in a typical four quadrant 2d Cartesian representation
7  # of the screen.
8
9  # The turtle is placed at (0, 0) at the beginning of a default run (which is called Home)
10 # and can return there at any time by calling the home() function.
11 t = turtle.Turtle()
12 t.goto(50, 100)
13 t.goto(-50, 100)
14 t.home()
15
16 turtle.mainloop()
17
```



?

Python Turtle: Introduction (8)

turtle_1.py | turtle_2.py | turtle_3.py | turtle_4.py | turtle_5.py | turtle_6.py

```
1  import turtle
2
3  # Let's try and draw the quadrants (kind of)
4  t = turtle.Turtle()
5
6  t.goto(0, 200)
7  t.home()
8  t.goto(200, 0)
9  t.home()
10 t.goto(0, -200)
11 t.home()
12 t.goto(-200, 0)
13 t.home()
14
15 turtle.mainloop()
16
```

Python Turtle: Introduction (9)

turtle_7.py

```
1   import turtle
2
3   t = turtle.Turtle()
4
5   # The library provides some additional "preset" figures.
6
7   # example: create a circle invoking the circle() method.
8   t.circle(100)    # circle radius = 100
9   t.circle(50)    # circle radius = 100
10
11  # example: create a dot invoking the dot() method.
12  t.dot(50)        # dot diameter = 50
13
14  turtle.mainloop()
15
```

Python Turtle: Modifying Characteristics (1)

turtle_7.py | turtle_8.py

```
1  import turtle
2
3  # We can modify the canvas background colour at any time of a program's execution
4  # using the bgcolor() command.
5  t = turtle.Turtle()
6
7  turtle.bgcolor("green")
8  t.circle(100)
9
10 turtle.bgcolor("blue")
11 t.circle(50)
12
13 turtle.bgcolor("#800000")
14
15 turtle.mainloop()
16
```

Python Turtle: Modifying Characteristics (2)

turtle_7.py | turtle_8.py | turtle_9.py

```
1   import turtle
2
3   # We can modify the window title with the turtle() command.
4   t = turtle.Turtle()
5
6   turtle.title("Demo Green Screen")
7   turtle.bgcolor("green")
8   t.circle(100)
9
10  turtle.title("Demo Blue Screen")
11  turtle.bgcolor("blue")
12  t.circle(50)
13
14  turtle.title("Demo Hex-defined Screen")
15  turtle.bgcolor("#800000")
16
17  turtle.mainloop()
```

Python Turtle: Modifying Characteristics (3)

turtle_7.py | turtle_8.py | turtle_9.py | turtle_10.py

```
1  import turtle
2
3  # Modifying the turtle size with shapesize().
4  # TIP: These changes do not affect the lines drawn.
5  # TIP: The parameters concern stretch length, stretch width and outline width
6  #      (in this order).
7  turtle.title('shepesize()')
8
9  t = turtle.Turtle()
10 t.shapesize(1,5,10)
11 t.circle(100)
12
13 t.shapesize(5,5,10)
14 t.circle(120)
15
16 t.shapesize(5,5,50)
17 t.circle(150)
18
19 turtle.mainloop()
```

Python Turtle: Modifying Characteristics (4)

turtle_7.py | turtle_8.py | turtle_9.py | turtle_10.py | turtle_11.py

```
1  import turtle
2
3  # Modifying a line's thickness with pensize().
4  t = turtle.Turtle()
5
6  t.pensize(1)
7  t.circle(10)
8  t.pensize(5)
9  t.circle(50)
10 t.pensize(10)
11 t.circle(100)
12
13 turtle.mainloop()
14
```

Python Turtle: Modifying Characteristics (5)

turtle_7.py | turtle_8.py | turtle_9.py | turtle_10.py | turtle_11.py | turtle_12.py

```
1  import turtle
2
3  # Modifying turtle and/or pen colour.
4  t = turtle.Turtle()
5
6  t.shapesize(5, 5, 5)
7  t.fillcolor("blue")
8  t.pencolor("blue")
9  t.circle(50)
10
11 t.color("green", "green") # pen color, turtle fill
12 t.circle(100)
13
14 turtle.mainloop()
15
```

Python Turtle: Modifying Characteristics (6)

turtle_10.py | turtle_11.py | turtle_12.py | turtle_13.py

```
1  import turtle
2
3  # Filling an image with color (by default black).
4  t = turtle.Turtle()
5
6  t.begin_fill()
7  t.circle(100)
8  t.end_fill()
9
10 t.fillcolor('green')
11 t.begin_fill()
12 t.circle(50)
13 t.end_fill()
14
15 turtle.mainloop()
16
```

Python Turtle: Modifying Characteristics (7)

turtle_10.py | turtle_11.py | turtle_12.py | turtle_13.py | turtle_14.py

```
1  import turtle
2
3  # We can change the turtle's shape by using the shape() method.
4  # There are 6 available options:
5  # - 'arrow'
6  # - 'circle'
7  # - 'classic' (the default option)
8  # - 'square'
9  # - 'triangle'
10 # - 'turtle'
11
12 t = turtle.Turtle()
13 t.shape("turtle")
14 t.circle(50)
15 t.forward(200)
16
17 turtle.mainloop()
18
```

Python Turtle: Modifying Characteristics (8)

ex_22.py | turtle_14.py | turtle_15.py

```
1  import turtle
2
3  # Changing the drawing speed
4  t = turtle.Turtle()
5
6  t.speed(1)      # lowest speed
7  t.circle(20)
8
9  t.speed(10)     # highest speed
10 t.circle(120)
11
12 # One-liners (suitable in the presence of several turtles)
13 t.pen(pencolor="purple", fillcolor="orange", pensize=10, speed=9)
14 t.circle(144)
15 turtle.mainloop()
16
```

Python Turtle: Modifying Characteristics (9)

turtle_15.py | turtle_16.py

```
1  import turtle
2
3  # Putting the pen down/up with penup() and pendown()
4
5  # The following program will not draw anything. It will
6  # just 'move' the pen.
7  t = turtle.Turtle()
8
9  t.penup()
10 t.fd(10)
11 t.rt(45)
12 t.fd(100)
13 t.rt(90)
14 t.fd(100)
15
16 turtle.mainloop()
17
```

Python Turtle: Modifying Characteristics (10)

turtle_15.py | turtle_16.py | turtle_17.py

```
1  import turtle
2
3  # Putting the pen down/up with penup() and pendown()
4
5  # The following program will draw two parallel lines
6  t = turtle.Turtle()
7
8  t.fd(100)
9
10 t.penup()
11 t.rt(90)
12 t.fd(100)
13
14 t.pendown()
15 t.rt(90)
16 t.fd(100)
17
18 turtle.mainloop()
```

Python Turtle: Modifying Characteristics (11)

turtle_15.py | turtle_16.py | turtle_17.py | turtle_18.py

```
1  import turtle
2
3  # undo() and clear()
4  # undo():   targets only the last command that was executed, if we want to undo more commands
5  #           we can call undo more than one time (each call goes one command further back)
6  # clear():  cleans up everything from our screen
7
8  t = turtle.Turtle()
9
10 t.circle(10)
11 t.circle(100)
12 t.undo()
13
14 turtle.mainloop()
15
16
17
```

Python Turtle: Modifying Characteristics (12)

turtle_18.py | turtle_19.py

```
1  import turtle
2
3  # undo() and clear()
4  # undo():   targets only the last command that was executed, if we want to undo more commands
5  #           we can call undo more than one time (each call goes one command further back)
6  # clear():  cleans up everything from our screen
7
8  t = turtle.Turtle()
9
10 t.circle(10)
11 t.circle(100)
12 t.clear()
13
14 turtle.mainloop()
15
16
17
```

Python Turtle: Modifying Characteristics (13)

turtle_18.py | turtle_19.py | turtle_20.py

```
1   import turtle
2
3   # reset(): clearing everything in our screen & code
4   t = turtle.Turtle()
5
6   turtle.title("Demo")
7
8   t.pensize(20)
9   t.pencolor('green')
10  t.fd(100)
11  t.lt(50)
12  t.dot(250)
13
14  t.reset()
15
16  t.fd(200)
17
18  turtle.mainloop()
```

Python Turtle: Modifying Characteristics (14)

turtle_21.py turtle_22.py

```
1  import turtle
2
3  # Creating a copy/clone of your turtle with clone()
4  t = turtle.Turtle()
5  t_c = t.clone()
6
7  t.color("magenta")
8
9  t.fd(100)
10
11 t_c.rt(180)
12 t_c.fd(60)
13
14 turtle.mainloop()
15
```

Python Turtle: Modifying Characteristics (15)

turtle_21.py turtle_22.py

```
1  import turtle
2
3  # Same outcome, avoiding clone()
4  t = turtle.Turtle()
5  c = turtle.Turtle()
6
7  t.color("magenta")
8
9  t.fd(100)
10
11 c.rt(180)
12 c.fd(60)
13
14 turtle.mainloop()
15
```

Python Turtle: Encompassing Basic Python Concepts (1)

turtle_21.py | turtle_22.py | turtle_23.py | turtle_24.py

```
1   import turtle
2
3   # Drawing a square, utilizing a for loop
4
5   # The 'typical/classic' way
6   t = turtle.Turtle()
7
8   t.fd(200)
9   t.rt(90)
10  t.fd(200)
11  t.rt(90)
12  t.fd(200)
13  t.rt(90)
14  t.fd(200)
15  t.rt(90)
16
17  turtle.mainloop()
18
```

Python Turtle: Encompassing Basic Python Concepts (2)

turtle_21.py | turtle_22.py | turtle_23.py | turtle_24.py | turtle_25.py

```
1  import turtle
2
3  # Drawing a square, utilizing a for loop
4
5  t = turtle.Turtle()
6
7  for i in range(4):
8      t.fd(200)
9      t.rt(90)
10
11 turtle.mainloop()
12
13
14
15
```

Python Turtle: Encompassing Basic Python Concepts (3)

turtle_21.py | turtle_22.py | turtle_23.py | turtle_24.py | turtle_25.py

```
1   import turtle
2
3   # Drawing a square, utilizing a for loop
4
5   t = turtle.Turtle()
6
7   for i in range(4):
8       t.fd(200)
9       t.rt(90)
10
11  # Which parts of the drawing will be undone?
12  for i in range(3):
13      t.undo()
14
15  turtle.mainloop()
```

Python Turtle: Encompassing Basic Python Concepts (4)

turtle_25.py turtle_26.py

```
1  import turtle
2
3  # Draw ten circles with different radiuses starting from
4  # the same point utilizing a while loop
5  t = turtle.Turtle()
6
7  circ_num = 1
8  while circ_num <= 10:
9      t.circle(10 * circ_num)
10     circ_num = circ_num + 1
11
12 turtle.mainloop()
13
14
15
16
17
```

Python Turtle: Encompassing Basic Python Concepts (5)

turtle_25.py | turtle_26.py | turtle_27.py

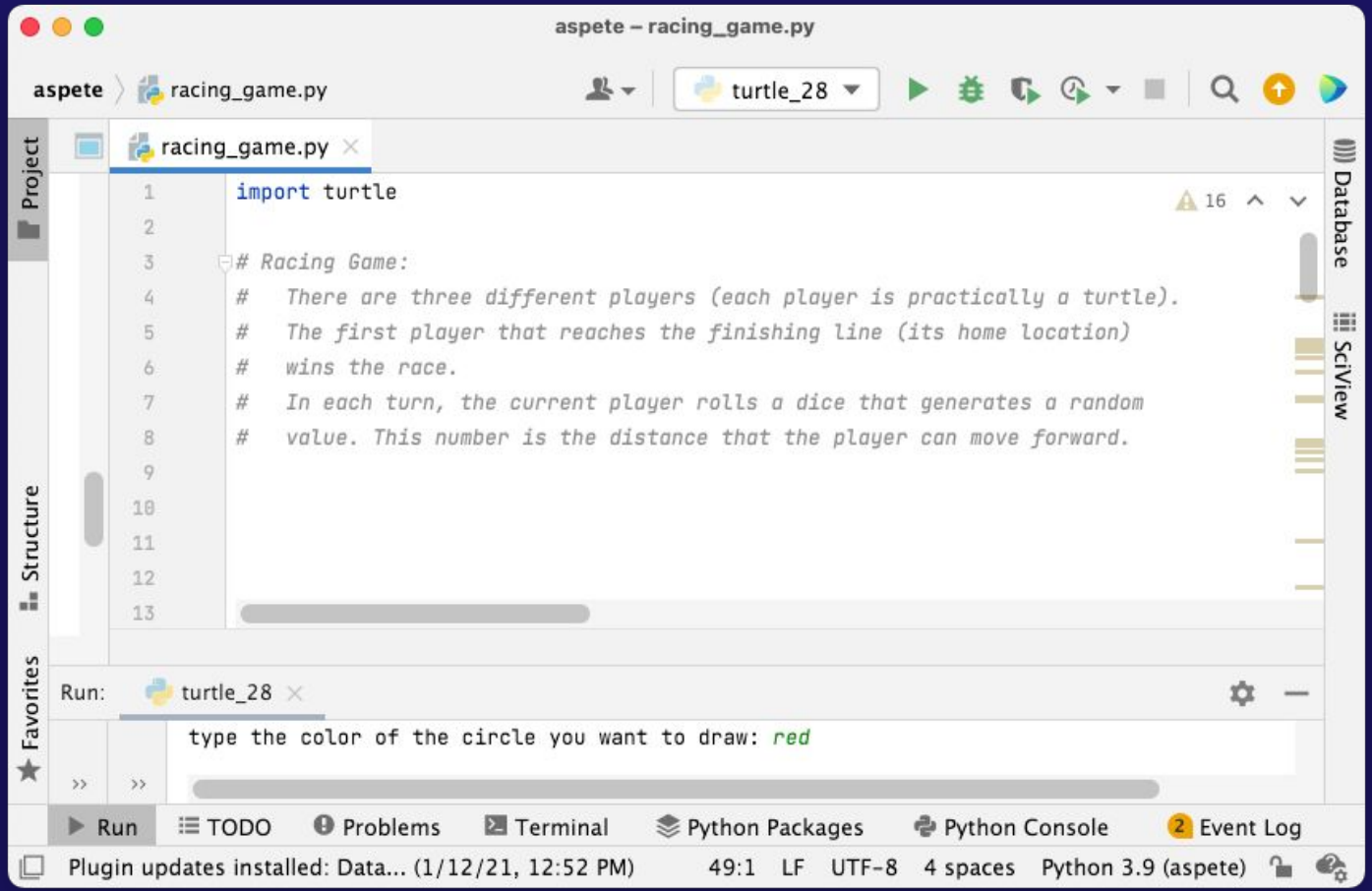
```
1   import turtle
2   import random
3
4   # Draw ten circles with different radiuses and colors starting from
5   # the same point utilizing a while loop
6   t = turtle.Turtle()
7
8   colors = ['green', 'orange', 'red', 'blue', 'magenta']
9
10  circ_num = 1
11  while circ_num <= 10:
12      t.pencolor(colors[random.randint(0, 4)])
13      t.circle(10 * circ_num)
14      circ_num = circ_num + 1
15
16  turtle.mainloop()
17
```

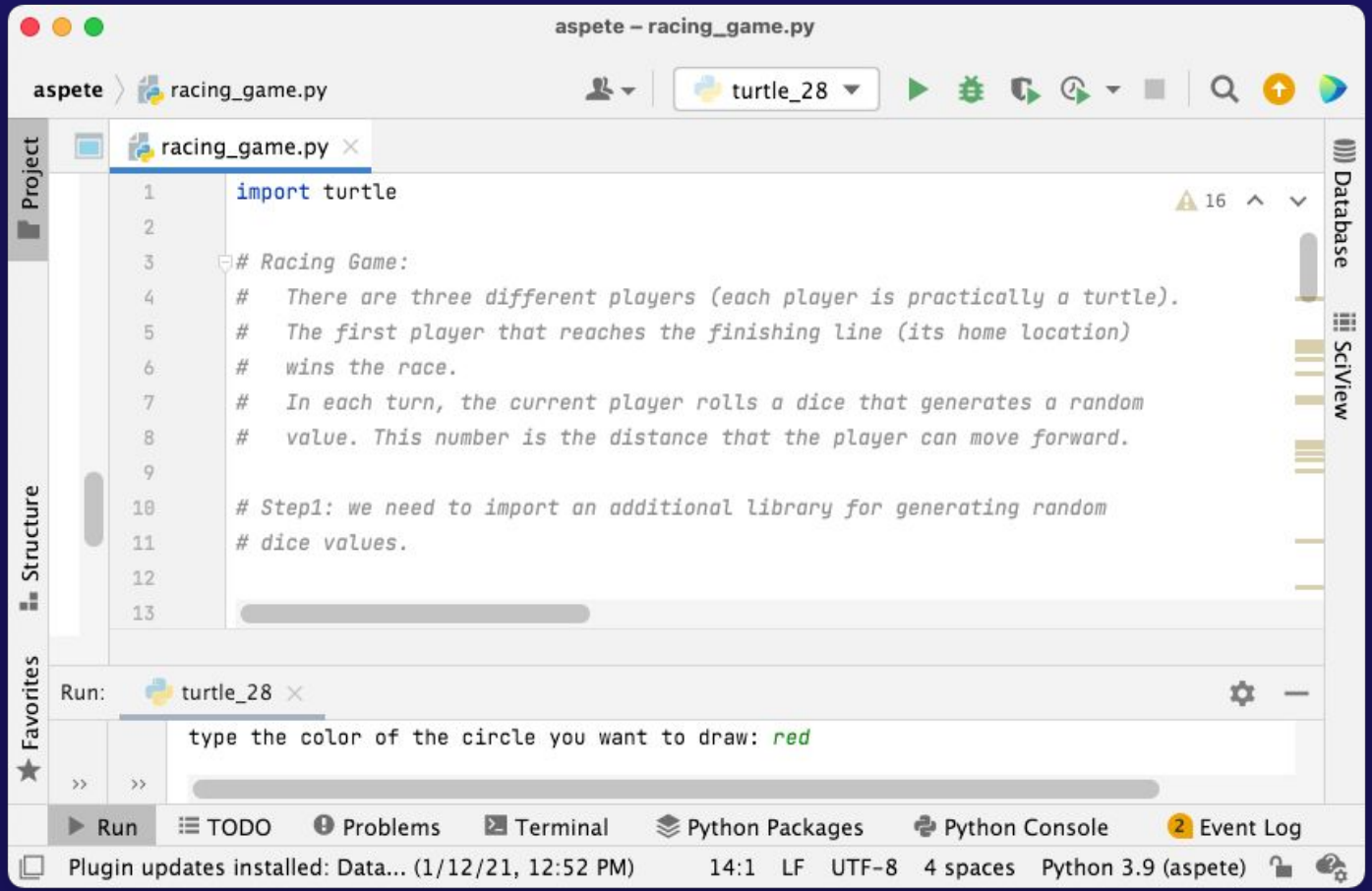
Python Turtle: Encompassing Basic Python Concepts (6)

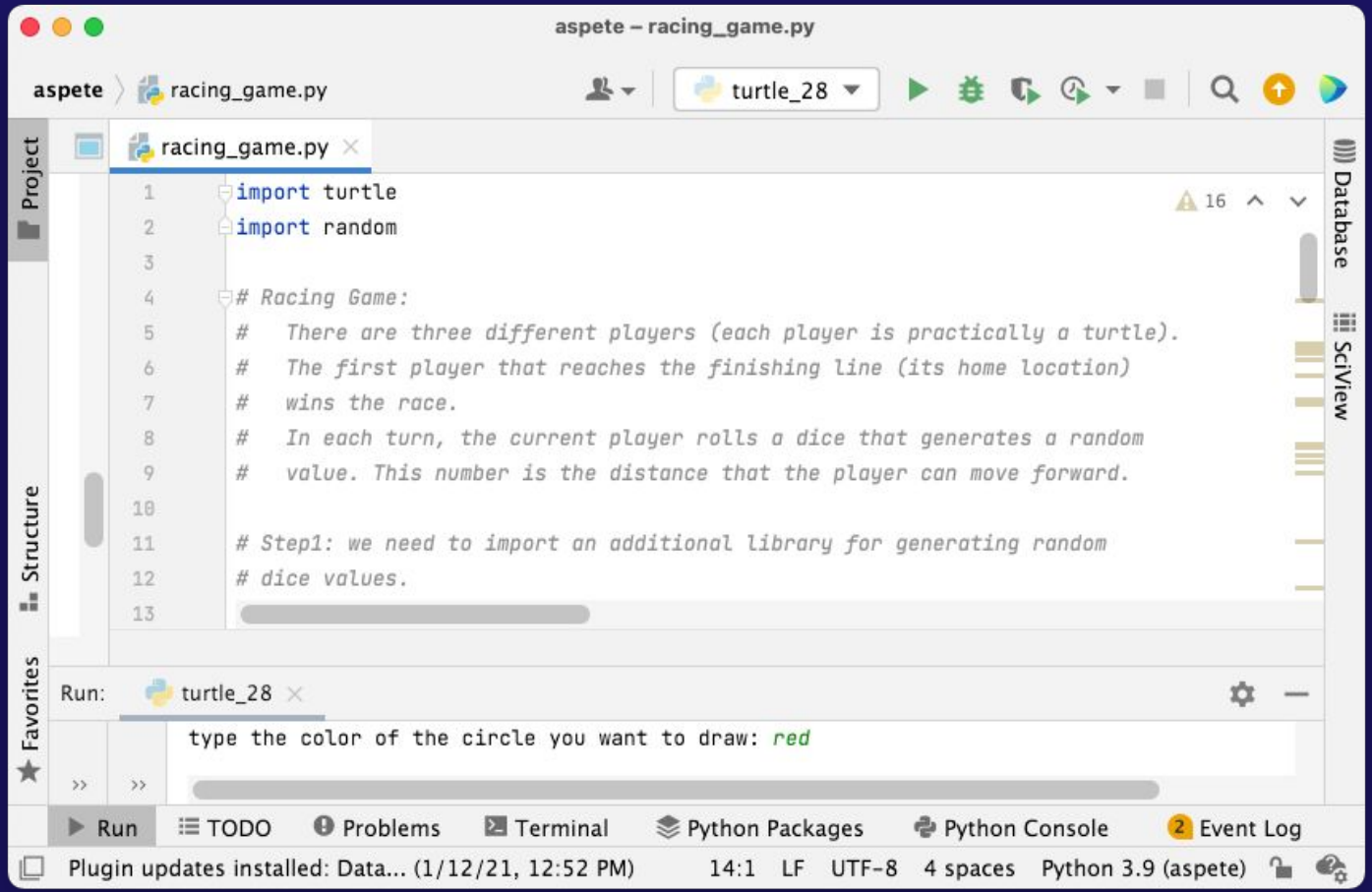
turtle_25.py | turtle_26.py | turtle_27.py | turtle_28.py

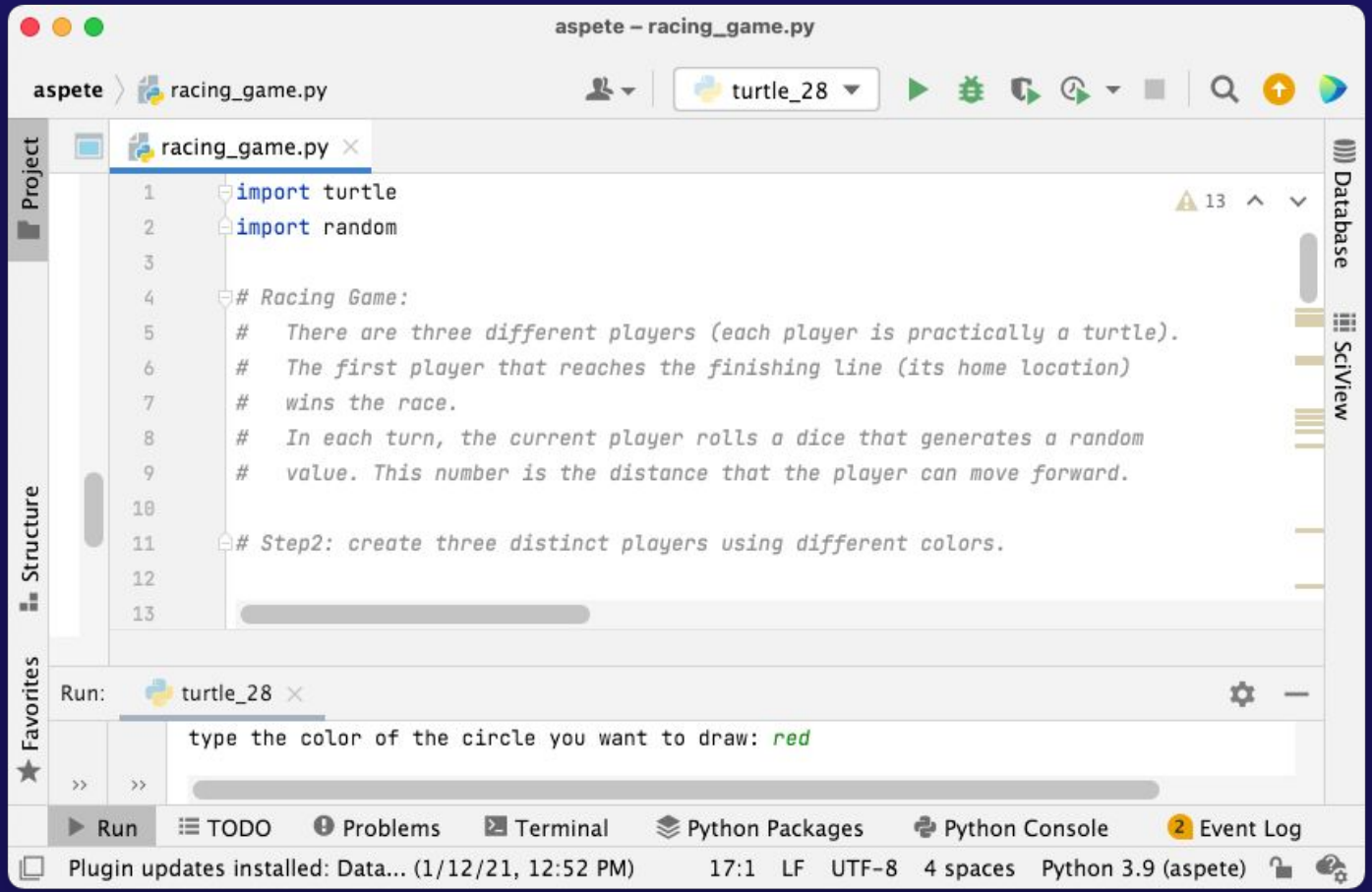
```
1   import turtle
2
3   # User defined circle color drawing with conditional statements
4   t = turtle.Turtle()
5
6   colors = ['green', 'orange', 'red', 'blue', 'magenta']
7   color = input("type the color of the circle you want to draw: ")
8
9   if color not in colors:
10      print("the selected color is not supported")
11  else:
12      t.pencolor(color)
13      t.circle(100)
14
15  turtle.mainloop()
16
17  # What would be different if the circle radius should be also
18  # user-defined?
19
```

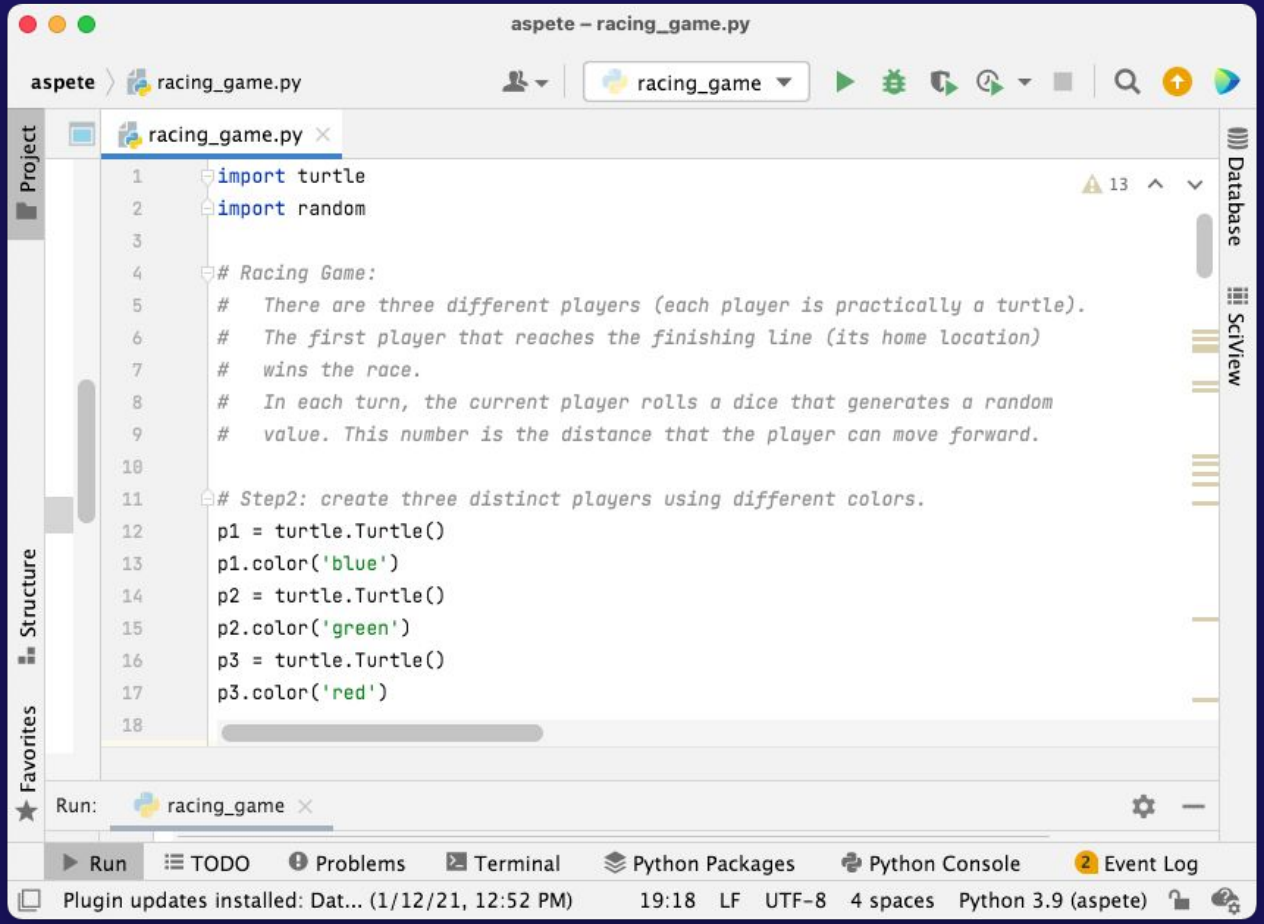
Python Demo: Racing Game

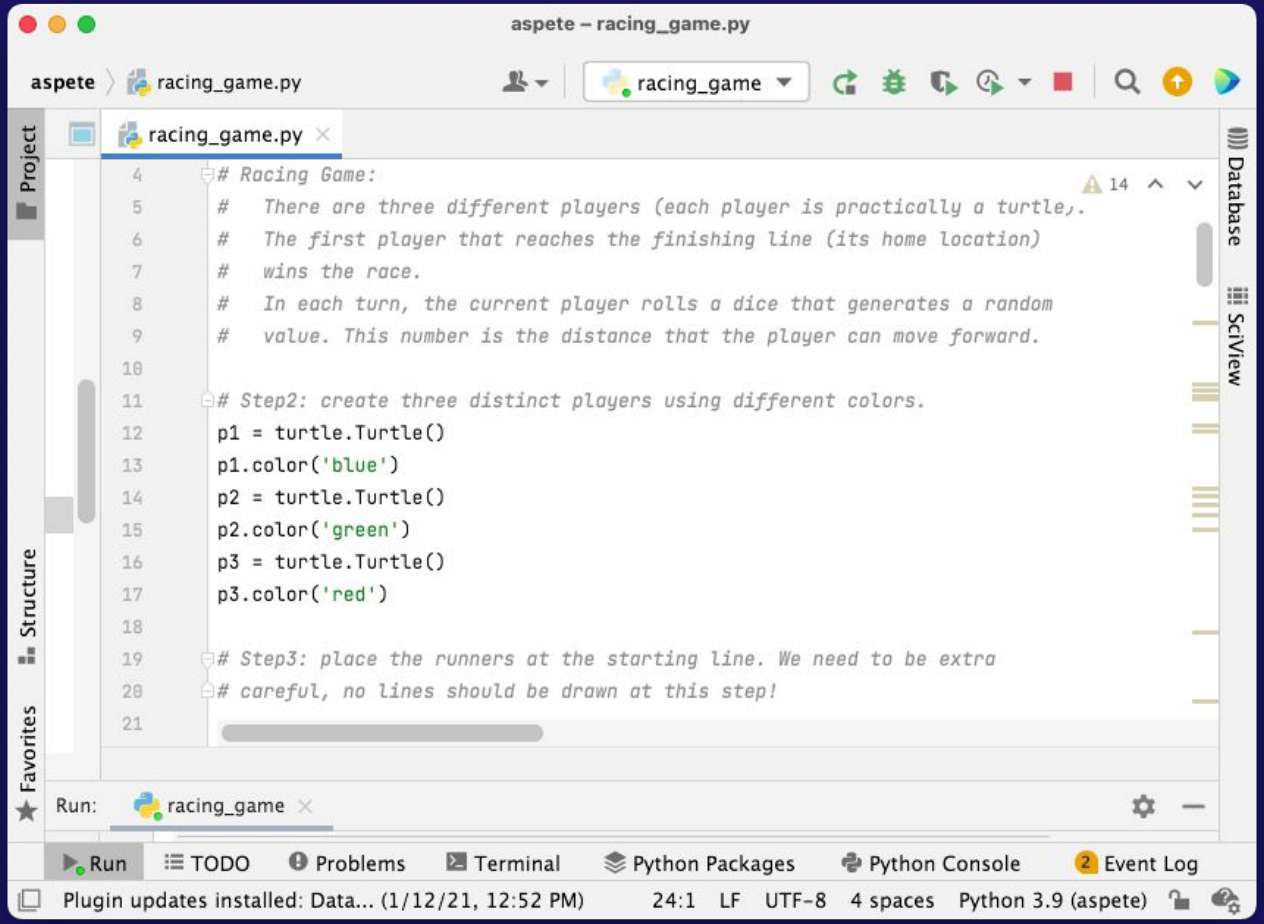

aspete – racing_game.py
aspete
racing_game.py
turtle_28
racing_game.py
Project
Structure
Favorites
Database
SciView
1 import turtle
2
3 # Racing Game:
4 # There are three different players (each player is practically a turtle).
5 # The first player that reaches the finishing line (its home location)
6 # wins the race.
7 # In each turn, the current player rolls a dice that generates a random
8 # value. This number is the distance that the player can move forward.
9
10
11
12
13
16
Run:
turtle_28
type the color of the circle you want to draw: red
Run
TODO
Problems
Terminal
Python Packages
Python Console
2 Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
49:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)

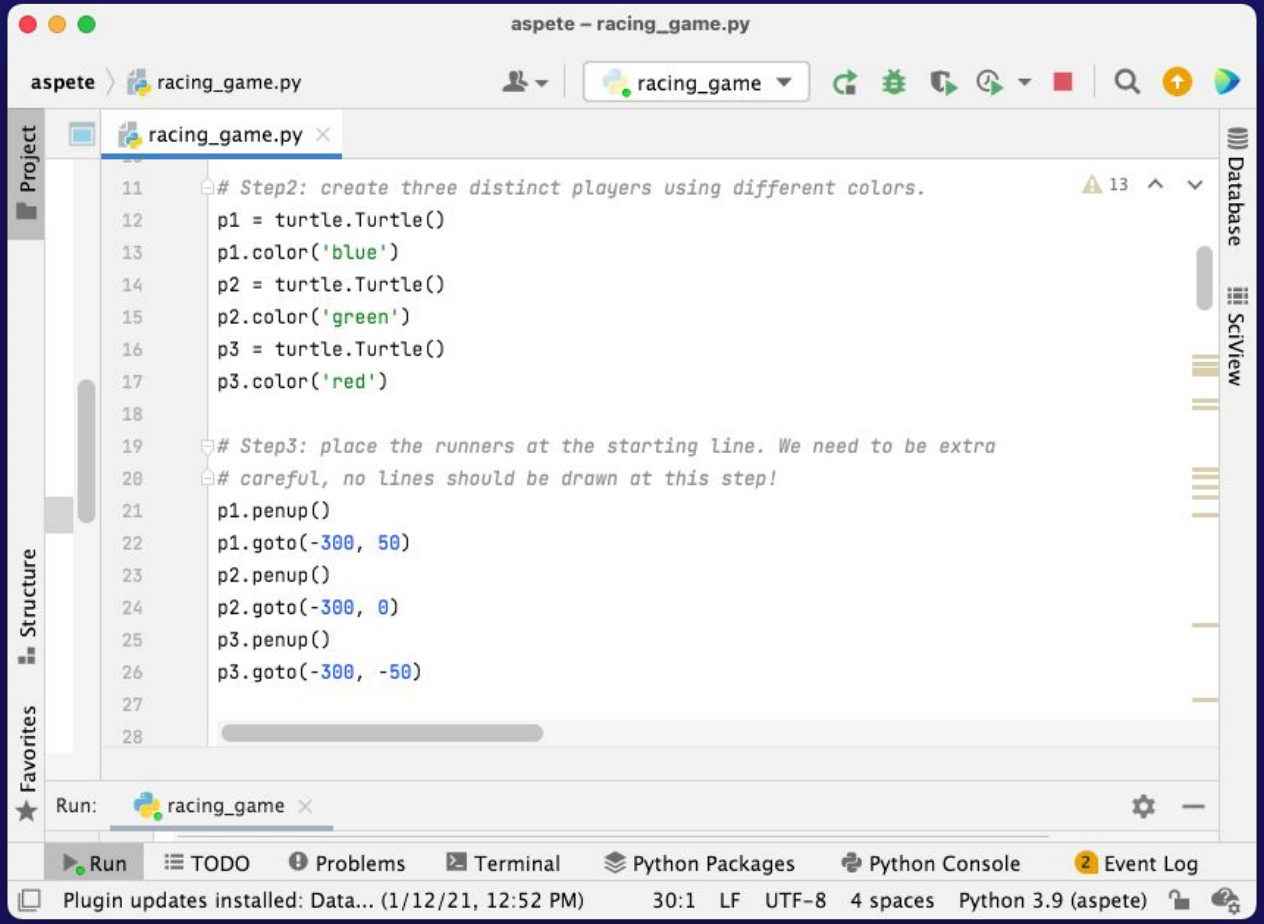

aspete – racing_game.py
aspete
racing_game.py
turtle_28
racing_game.py
Project
Structure
Favorites
Database
SciView
1 import turtle
2
3 # Racing Game:
4 # There are three different players (each player is practically a turtle).
5 # The first player that reaches the finishing line (its home location)
6 # wins the race.
7 # In each turn, the current player rolls a dice that generates a random
8 # value. This number is the distance that the player can move forward.
9
10 # Step1: we need to import an additional library for generating random
11 # dice values.
12
13
16
Run:
turtle_28
type the color of the circle you want to draw: red
Run
TODO
Problems
Terminal
Python Packages
Python Console
2 Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
14:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)

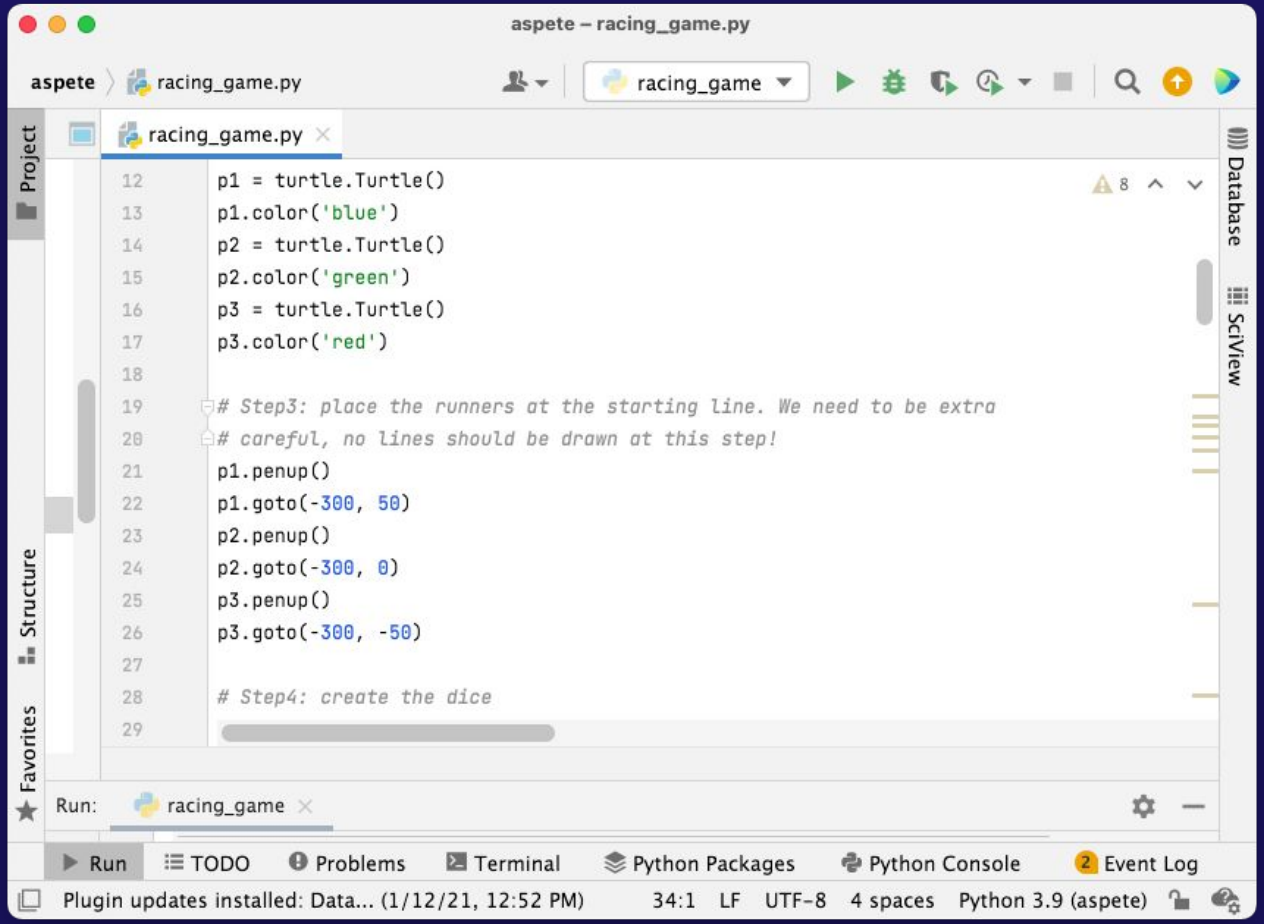

aspete – racing_game.py
aspete
racing_game.py
turtle_28
racing_game.py
Project
Structure
Favorites
Database
SciView
16
1 import turtle
2 import random
3
4 # Racing Game:
5 # There are three different players (each player is practically a turtle).
6 # The first player that reaches the finishing line (its home location)
7 # wins the race.
8 # In each turn, the current player rolls a dice that generates a random
9 # value. This number is the distance that the player can move forward.
10
11 # Step1: we need to import an additional library for generating random
12 # dice values.
13
Run:
turtle_28
type the color of the circle you want to draw: red
Run
TODO
Problems
Terminal
Python Packages
Python Console
2 Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
14:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)

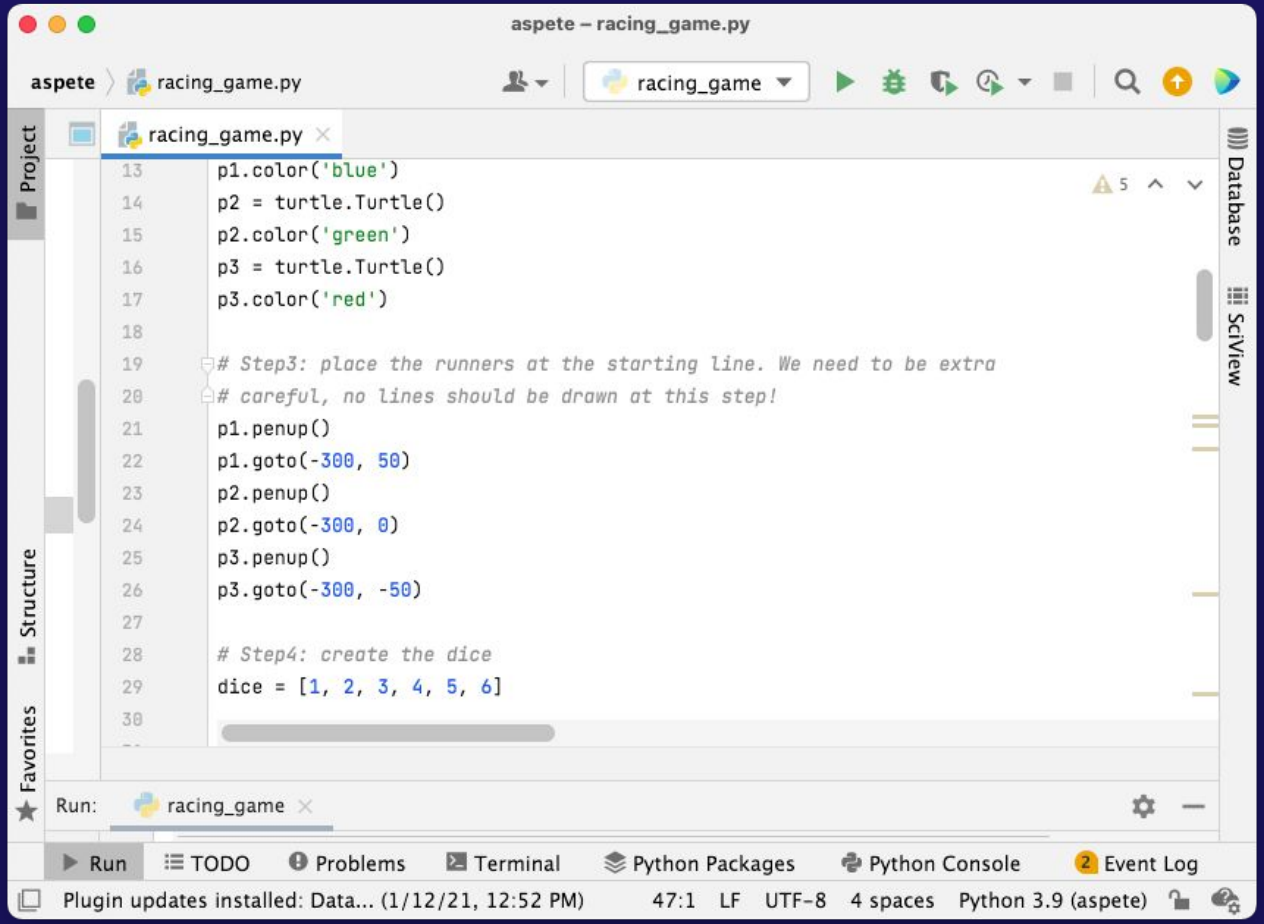

aspete – racing_game.py
aspete
racing_game.py
turtle_28
racing_game.py
import turtle
import random
# Racing Game:
# There are three different players (each player is practically a turtle).
# The first player that reaches the finishing line (its home location)
# wins the race.
# In each turn, the current player rolls a dice that generates a random
# value. This number is the distance that the player can move forward.
# Step2: create three distinct players using different colors.
13
Project
Structure
Favorites
Database
SciView
Run:
turtle_28
type the color of the circle you want to draw: red
Run
TODO
Problems
Terminal
Python Packages
Python Console
2 Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
17:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)

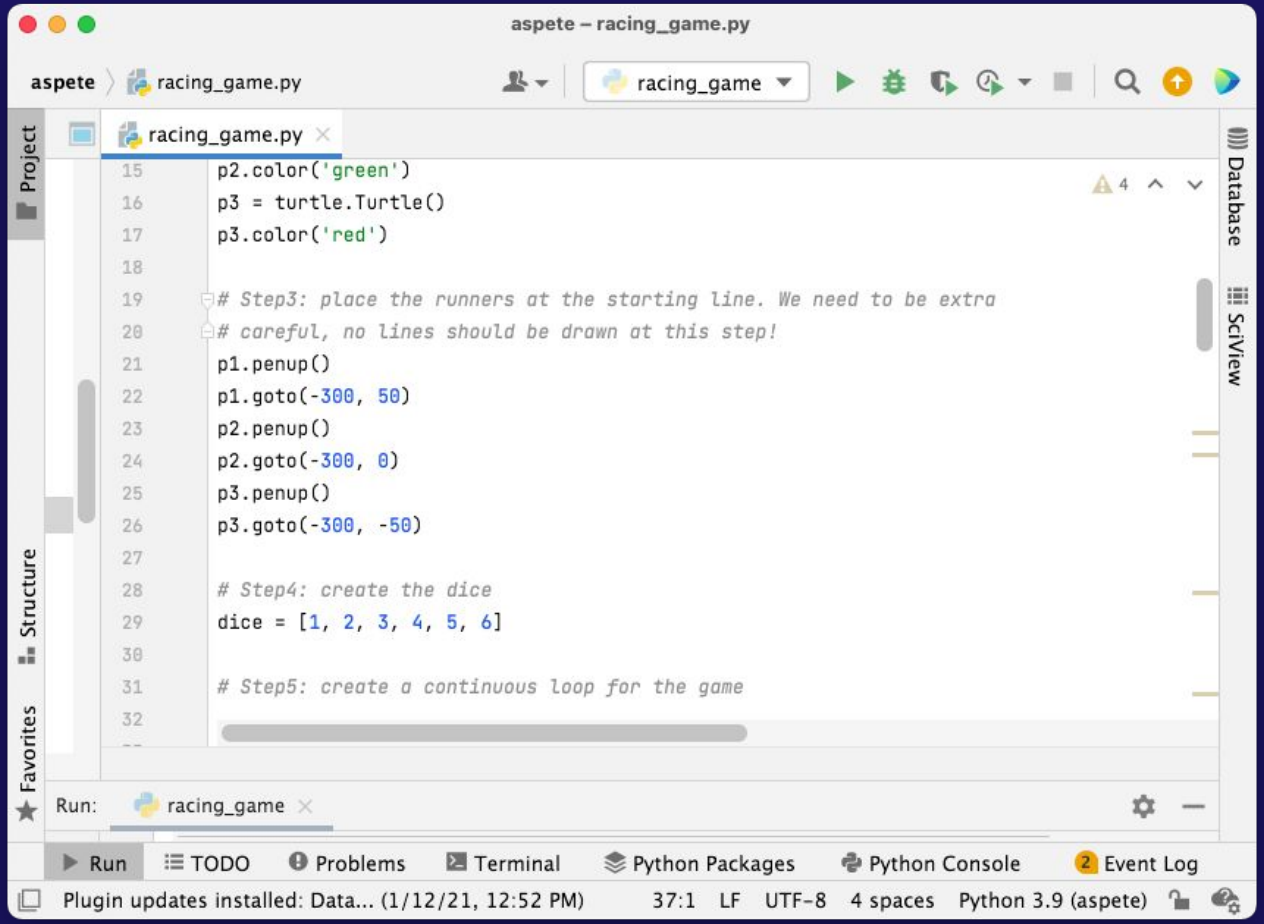

aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
Project
Database
SciView
Structure
Favorites
13
import turtle
import random
# Racing Game:
# There are three different players (each player is practically a turtle).
# The first player that reaches the finishing line (its home location)
# wins the race.
# In each turn, the current player rolls a dice that generates a random
# value. This number is the distance that the player can move forward.
# Step2: create three distinct players using different colors.
p1 = turtle.Turtle()
p1.color('blue')
p2 = turtle.Turtle()
p2.color('green')
p3 = turtle.Turtle()
p3.color('red')
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
2 Event Log
Plugin updates installed: Dat... (1/12/21, 12:52 PM)
19:18
LF
UTF-8
4 spaces
Python 3.9 (aspete)

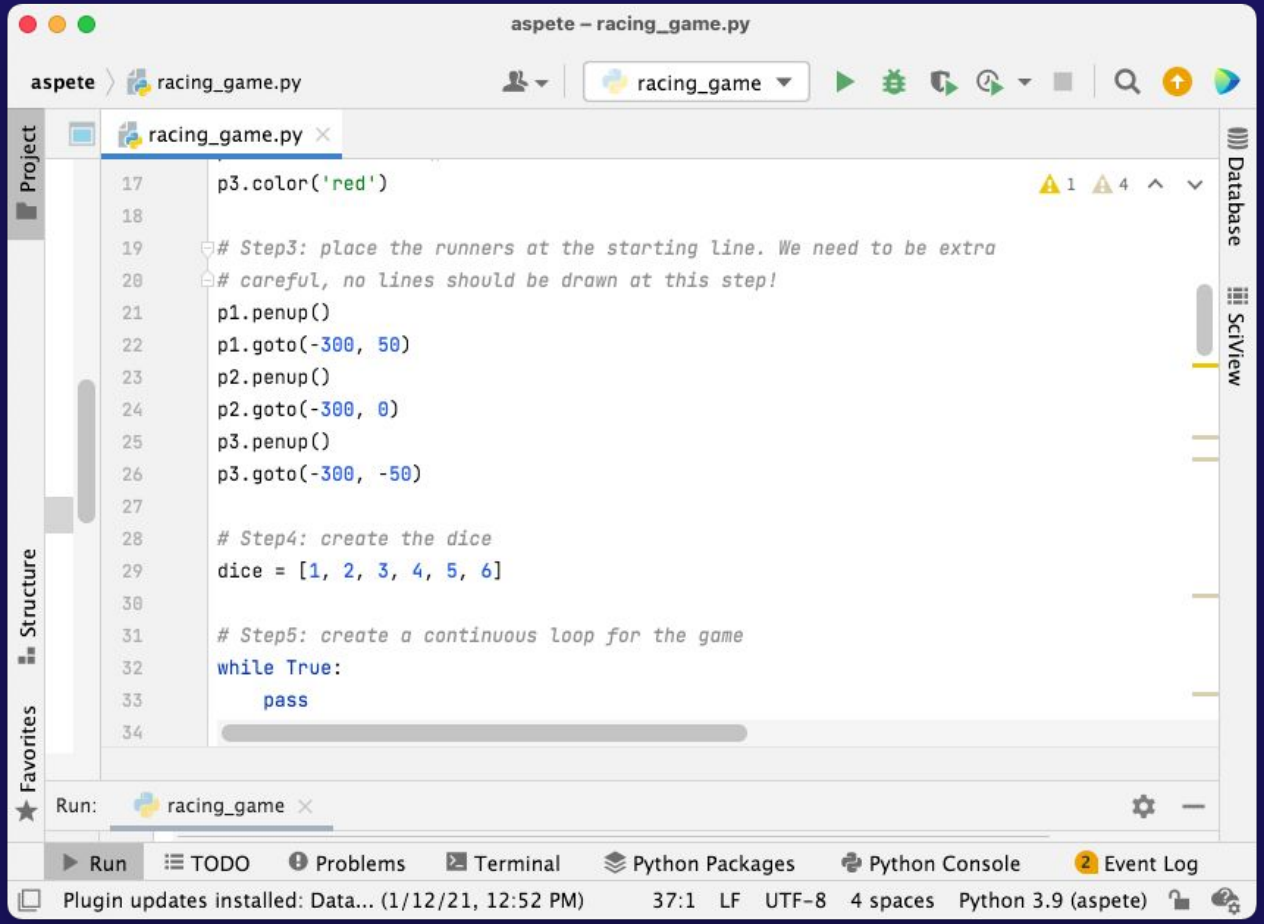

aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
Project
Database
SciView
Structure
Favorites
14
# Racing Game:
#   There are three different players (each player is practically a turtle,.
#   The first player that reaches the finishing line (its home location)
#   wins the race.
#   In each turn, the current player rolls a dice that generates a random
#   value. This number is the distance that the player can move forward.
# Step2: create three distinct players using different colors.
p1 = turtle.Turtle()
p1.color('blue')
p2 = turtle.Turtle()
p2.color('green')
p3 = turtle.Turtle()
p3.color('red')
# Step3: place the runners at the starting line. We need to be extra
# careful, no lines should be drawn at this step!
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
2 Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
24:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)

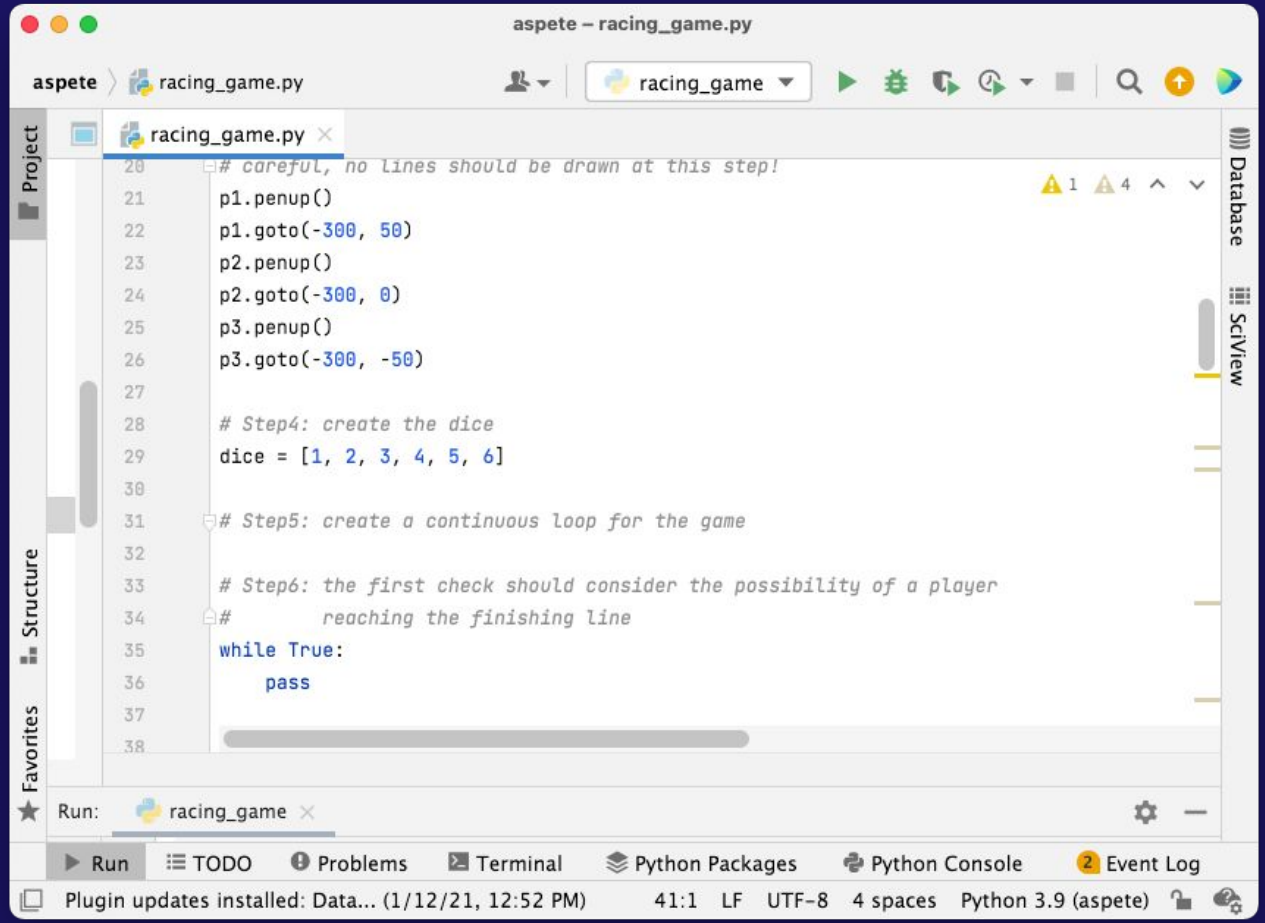

aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
Project
Database
SciView
Structure
Favorites
13
# Step2: create three distinct players using different colors.
p1 = turtle.Turtle()
p1.color('blue')
p2 = turtle.Turtle()
p2.color('green')
p3 = turtle.Turtle()
p3.color('red')
# Step3: place the runners at the starting line. We need to be extra
# careful, no lines should be drawn at this step!
p1.penup()
p1.goto(-300, 50)
p2.penup()
p2.goto(-300, 0)
p3.penup()
p3.goto(-300, -50)
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
2 Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
30:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)

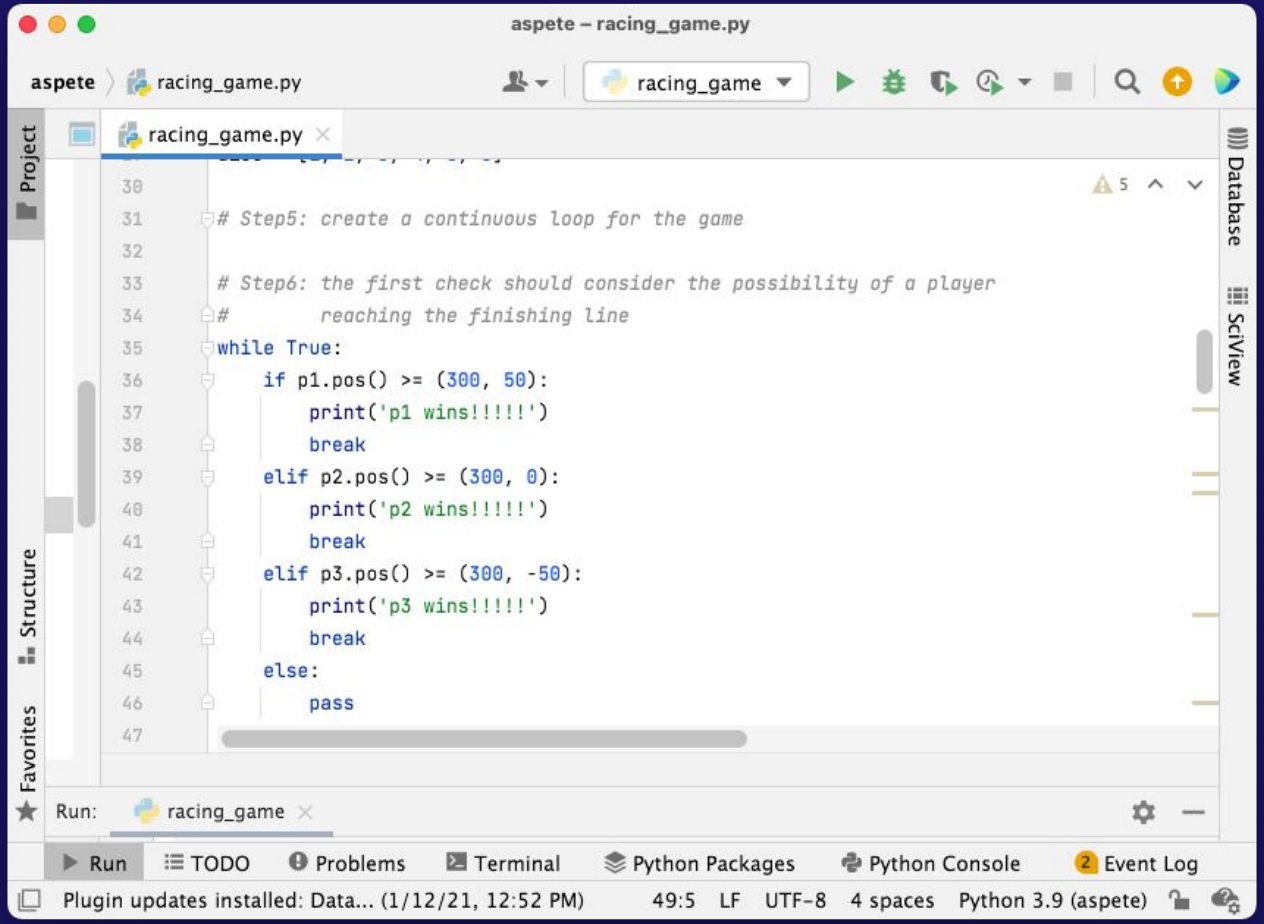

aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
Project
Database
SciView
Structure
Favorites
p1 = turtle.Turtle()
p1.color('blue')
p2 = turtle.Turtle()
p2.color('green')
p3 = turtle.Turtle()
p3.color('red')
# Step3: place the runners at the starting line. We need to be extra
# careful, no lines should be drawn at this step!
p1.penup()
p1.goto(-300, 50)
p2.penup()
p2.goto(-300, 0)
p3.penup()
p3.goto(-300, -50)
# Step4: create the dice
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
34:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)

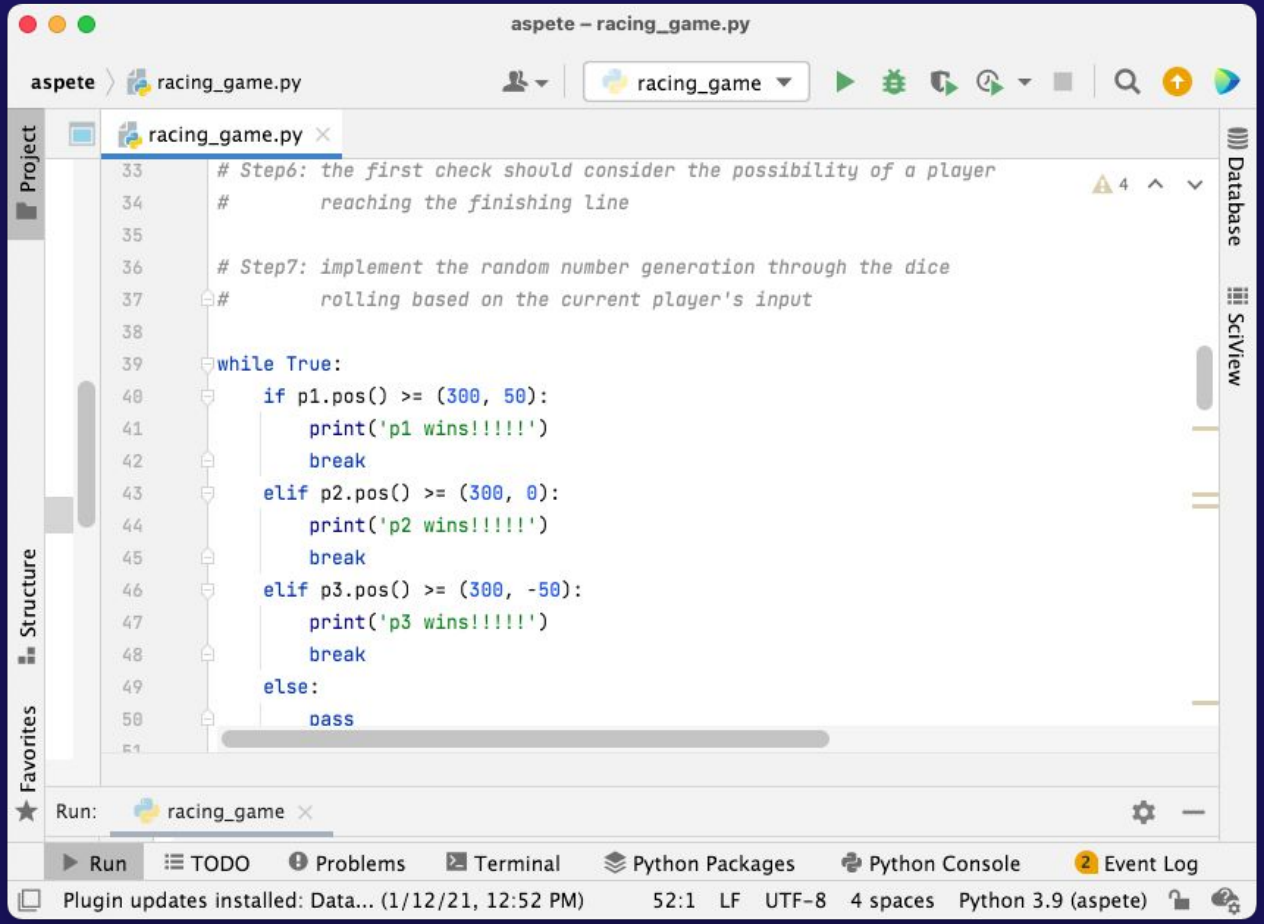

aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
Project
Database
SciView
Structure
Favorites
p1.color('blue')
p2 = turtle.Turtle()
p2.color('green')
p3 = turtle.Turtle()
p3.color('red')
# Step3: place the runners at the starting line. We need to be extra
# careful, no lines should be drawn at this step!
p1.penup()
p1.goto(-300, 50)
p2.penup()
p2.goto(-300, 0)
p3.penup()
p3.goto(-300, -50)
# Step4: create the dice
dice = [1, 2, 3, 4, 5, 6]
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
47:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)

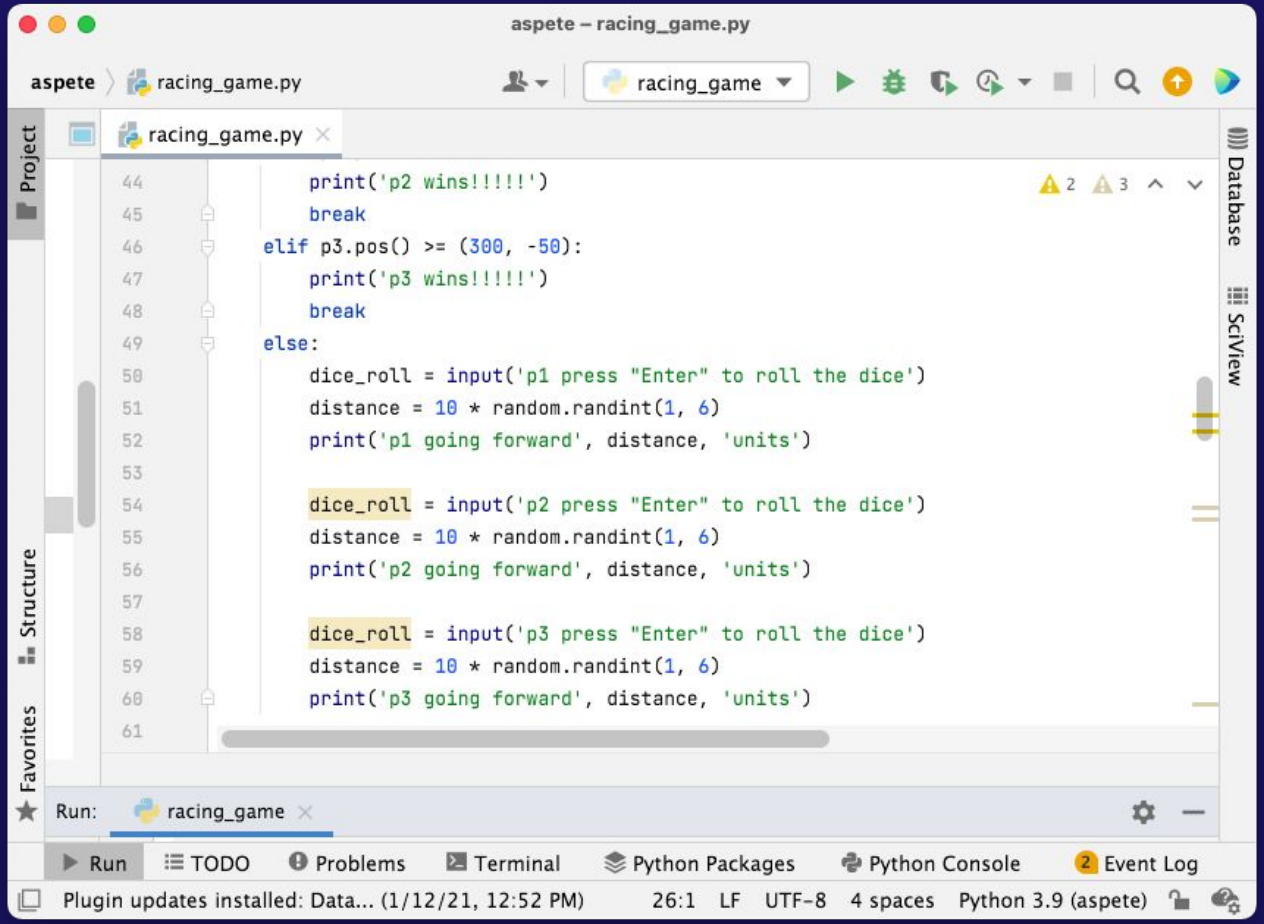

aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
p2.color('green')
p3 = turtle.Turtle()
p3.color('red')
# Step3: place the runners at the starting line. We need to be extra
# careful, no lines should be drawn at this step!
p1.penup()
p1.goto(-300, 50)
p2.penup()
p2.goto(-300, 0)
p3.penup()
p3.goto(-300, -50)
# Step4: create the dice
dice = [1, 2, 3, 4, 5, 6]
# Step5: create a continuous loop for the game
Project
Database
SciView
Structure
Favorites
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
37:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)

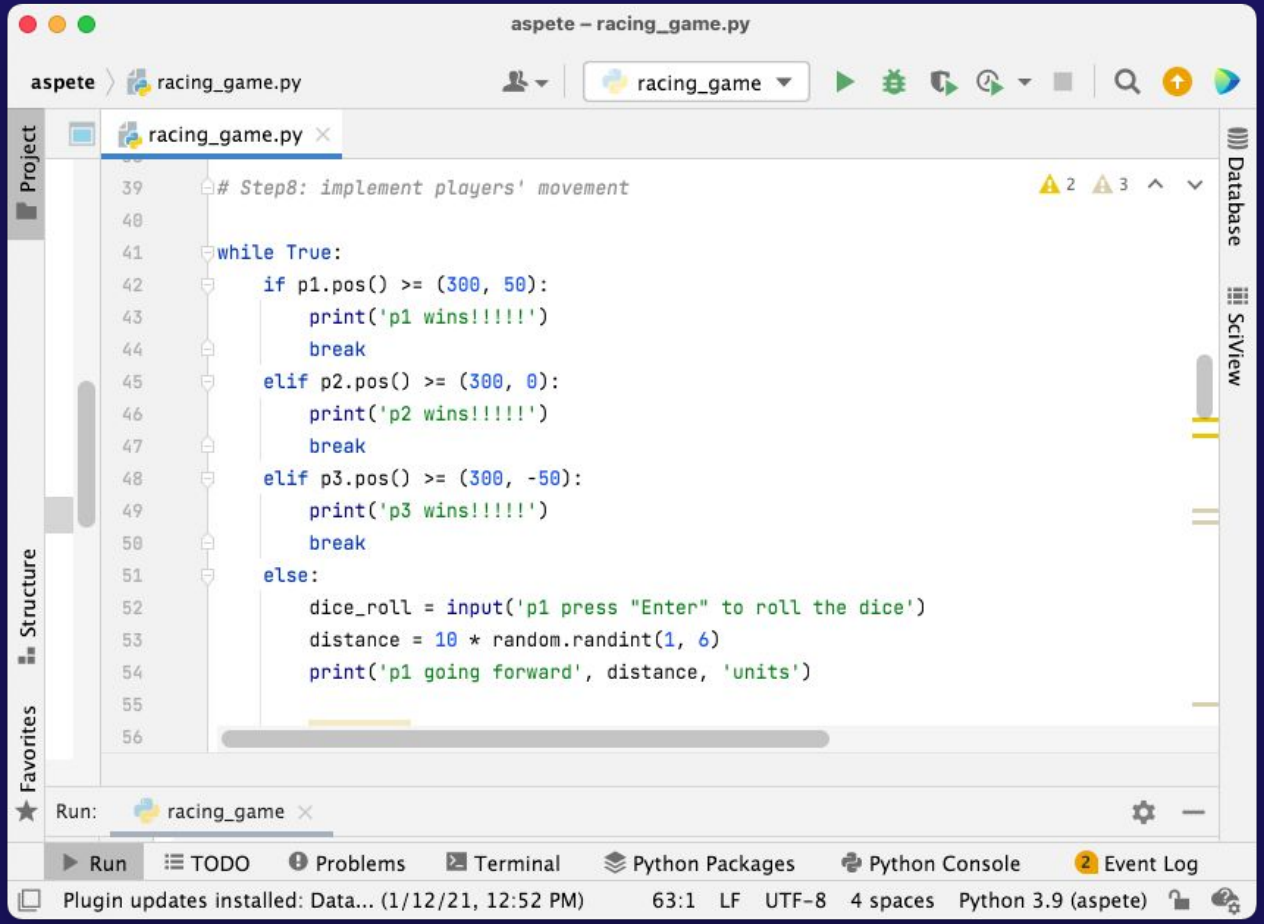

aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
Project
Database
SciView
Structure
Favorites
p3.color('red')
# Step3: place the runners at the starting line. We need to be extra
# careful, no lines should be drawn at this step!
p1.penup()
p1.goto(-300, 50)
p2.penup()
p2.goto(-300, 0)
p3.penup()
p3.goto(-300, -50)
# Step4: create the dice
dice = [1, 2, 3, 4, 5, 6]
# Step5: create a continuous loop for the game
while True:
pass
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
37:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)

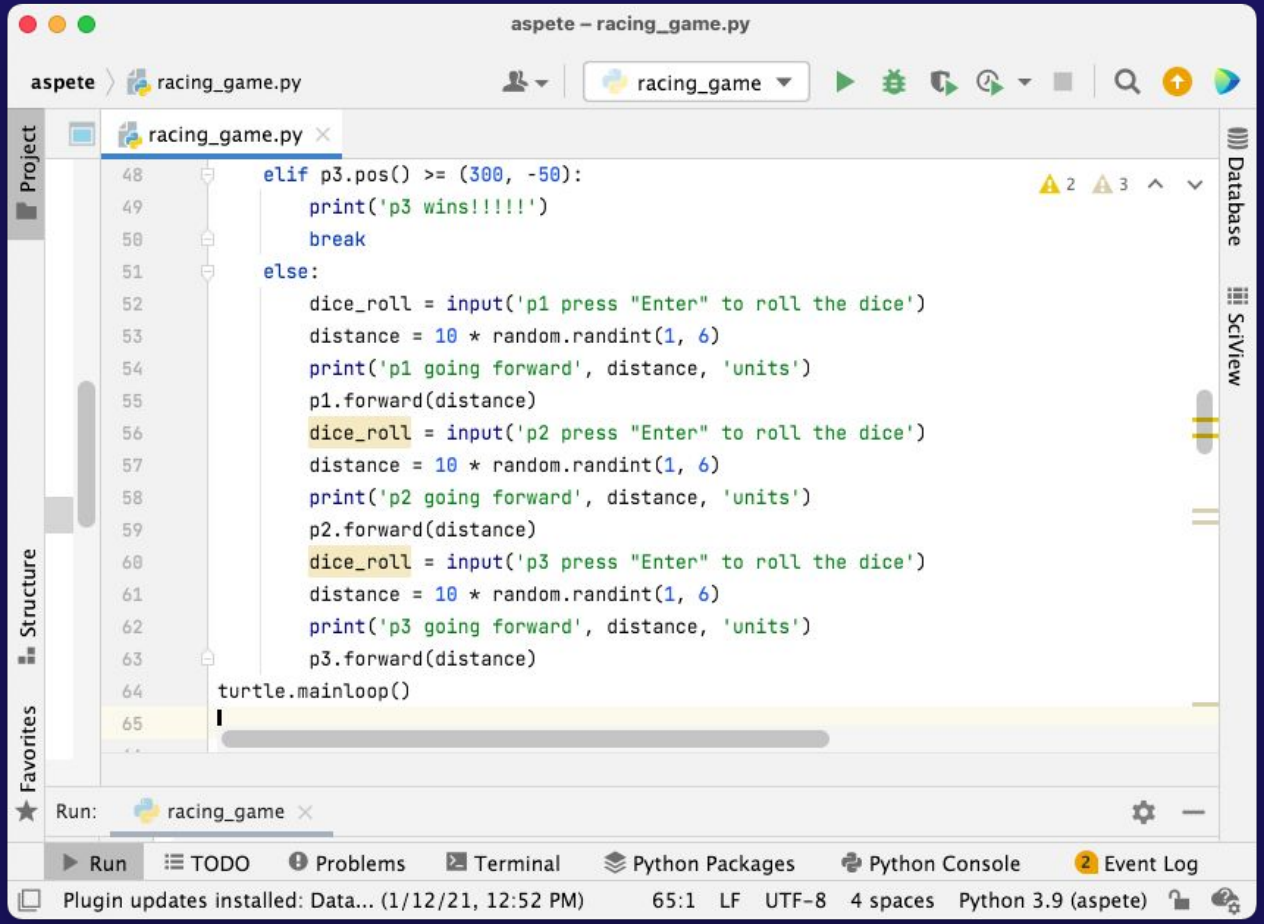

aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
Project
Structure
Favorites
Database
SciView
# careful, no lines should be drawn at this step!
p1.penup()
p1.goto(-300, 50)
p2.penup()
p2.goto(-300, 0)
p3.penup()
p3.goto(-300, -50)
# Step4: create the dice
dice = [1, 2, 3, 4, 5, 6]
# Step5: create a continuous loop for the game
# Step6: the first check should consider the possibility of a player
#   reaching the finishing line
while True:
pass
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
41:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)


aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
Project
Database
SciView
Structure
Favorites
30
31 # Step5: create a continuous loop for the game
32
33 # Step6: the first check should consider the possibility of a player
34 #   reaching the finishing line
35 while True:
36   if p1.pos() >= (300, 50):
37   print('p1 wins!!!!!')
38   break
39   elif p2.pos() >= (300, 0):
40   print('p2 wins!!!!!')
41   break
42   elif p3.pos() >= (300, -50):
43   print('p3 wins!!!!!')
44   break
45   else:
46   pass
47
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
2 Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
49:5
LF
UTF-8
4 spaces
Python 3.9 (aspete)


aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
Project
Database
SciView
Structure
Favorites
# Step6: the first check should consider the possibility of a player
#   reaching the finishing line
# Step7: implement the random number generation through the dice
#   rolling based on the current player's input
while True:
if p1.pos() >= (300, 50):
print('p1 wins!!!!!')
break
elif p2.pos() >= (300, 0):
print('p2 wins!!!!!')
break
elif p3.pos() >= (300, -50):
print('p3 wins!!!!!')
break
else:
pass
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
52:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)


aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
Project
Database
SciView
Structure
Favorites
print('p2 wins!!!!!')
break
elif p3.pos() >= (300, -50):
print('p3 wins!!!!!')
break
else:
dice_roll = input('p1 press "Enter" to roll the dice')
distance = 10 * random.randint(1, 6)
print('p1 going forward', distance, 'units')
dice_roll = input('p2 press "Enter" to roll the dice')
distance = 10 * random.randint(1, 6)
print('p2 going forward', distance, 'units')
dice_roll = input('p3 press "Enter" to roll the dice')
distance = 10 * random.randint(1, 6)
print('p3 going forward', distance, 'units')
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
26:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)


aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
Project
Database
SciView
Structure
Favorites
# Step8: implement players' movement
while True:
if p1.pos() >= (300, 50):
print('p1 wins!!!!!')
break
elif p2.pos() >= (300, 0):
print('p2 wins!!!!!')
break
elif p3.pos() >= (300, -50):
print('p3 wins!!!!!')
break
else:
dice_roll = input('p1 press "Enter" to roll the dice')
distance = 10 * random.randint(1, 6)
print('p1 going forward', distance, 'units')
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
63:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)


aspete – racing_game.py
aspete
racing_game.py
racing_game
racing_game.py
Project
Structure
Favorites
Database
SciView
48 elif p3.pos() >= (300, -50):
49 print('p3 wins!!!!!')
50 break
51 else:
52 dice_roll = input('p1 press "Enter" to roll the dice')
53 distance = 10 * random.randint(1, 6)
54 print('p1 going forward', distance, 'units')
55 p1.forward(distance)
56 dice_roll = input('p2 press "Enter" to roll the dice')
57 distance = 10 * random.randint(1, 6)
58 print('p2 going forward', distance, 'units')
59 p2.forward(distance)
60 dice_roll = input('p3 press "Enter" to roll the dice')
61 distance = 10 * random.randint(1, 6)
62 print('p3 going forward', distance, 'units')
63 p3.forward(distance)
64 turtle.mainloop()
65
2
3
Run:
racing_game
Run
TODO
Problems
Terminal
Python Packages
Python Console
2 Event Log
Plugin updates installed: Data... (1/12/21, 12:52 PM)
65:1
LF
UTF-8
4 spaces
Python 3.9 (aspete)

Εργασία (παράδοση 10/2 (23:59))

1. Κάνοντας χρήση της βιβλιοθήκης Turtle να υλοποιηθούν τα κατάλληλα προγράμματα για την δημιουργία:
   a. μιας σκακιέρας
   b. του σήματος των Ολυμπιακών αγώνων
   c. ενός κίτρινου smiley face
   d. ενός φωτεινού σηματοδότη του οποίου τα σήματα εναλλάσσονται σε συγκεκριμένα (προκαθορισμένα) χρονικά διαστήματα
   e. 5 ομόκεντρων κύκλων, χρησιμοποιώντας κατάλληλη δομή επανάληψης
   f. δύο τετραγώνων (πλευράς ίσης με 110) των οποίων τα κέντρα απέχουν 50 μονάδες
   g. ενός τριγώνου του οποίου η μία πλευρά έχει μήκος 200, η δεύτερη πλευρά έχει μήκος 100 και η γωνία που σχηματίζεται μεταξύ τους είναι 75 μοιρών.
   h. ένα bar chart με διαφορετικά χρώματα και ορατούς άξονες
   i. ενός πολυγώνου του οποίου ο αριθμός των πλευρών καθορίζεται από το χρήστη

2. Τροποποιήστε το racing game που παρουσιάστηκε έτσι ώστε:
   a. να περιλαμβάνει μια γραμμή τερματισμού και μια σημαία τερματισμού παραπλήσια με αυτή της F1
   b. να υπάρχει δυνατότητα χρήσης διαφορετικών ‘avatar’ για κάθε έναν από τους χρήστες
   c. να υπάρχει ένα ψευδοτυχαίο γεγονός πτώσης το οποίο θα ακινητοποιεί κάποιων από τους παίκτες