



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)

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

```
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 plt.plot(dev_age, dev_sal)
7 plt.show()
8
```

Usage

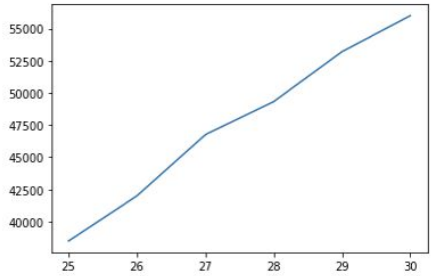
Here you can get help of any

Variable explorer File explorer Help

IPython console

Console 1/A

```
In [117]: runfile('C:/Users/antho/.spyder-py3/
temp.py', wdir='C:/Users/antho/.spyder-py3')
```



dev_age	dev_sal
25	38496
26	42000
27	46752
28	49320
29	53200
30	56000

```
In [118]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 8 Column: 1 Memory: 87%

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

```
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
```

Usage

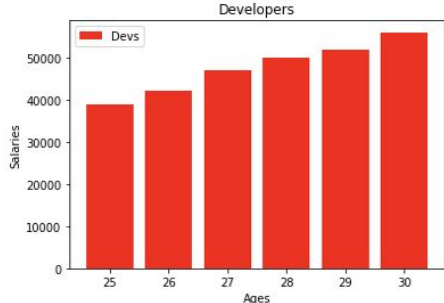
Here you can get help of any

Variable explorer File explorer Help

IPython console

Console 1/A

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



Age	Salary
25	39000
26	42000
27	47000
28	50000
29	52000
30	56000

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

```
1 """ Bar Charts """
2
3 # Complex Bar Chart
4 import numpy as np
5 from matplotlib import pyplot as plt
6
7 dev_age = [25, 26, 27, 28, 29, 30]
8 dev_sal = [39000, 42000, 47000, 50000, 52000, 56000]
9 pydev_sal = [40000, 44000, 50000, 53000, 55000, 60000]
10 javadev_sal = [32000, 40000, 51000, 50000, 50000, 65000]
11
12 x_index = np.arange(len(dev_age))
13 width = 0.2
14
15 plt.bar(x_index - width, dev_sal, width=width, color="r", label="Devs")
16 plt.bar(x_index, pydev_sal, width=width)
17 plt.bar(x_index + width, javadev_sal, width=width)
18 plt.legend()
19
20 # Ticks should be used to correctly prompt x axis values
21 plt.xticks(ticks=x_index, labels=dev_age)
22 plt.title("Developers")
23 plt.xlabel("Ages")
24 plt.ylabel("Salaries")
25
26 plt.show()
```

Help

Usage

Here you can get help

Variable explorer File explorer Help

IPython console

Console 1/A

Users/antho/.spyder-py3

Age	Devs	PyDevs	JavaDevs
25	39000	40000	32000
26	42000	44000	40000
27	47000	50000	51000
28	50000	53000	50000
29	52000	55000	50000
30	56000	60000	65000

In [152]:

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

Matplotlib (5)

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

```
1 """ Bar Charts """
2
3 # Horizontal Bar Chart
4 import numpy as np
5 import matplotlib.pyplot as plt
6
7 rooms = ('A1', 'A2', 'B1', 'B2', 'D1')
8 capacity = [4, 1, 5, 2, 1]
9
10 y_pos = np.arange(len(rooms))
11
12 plt.barh(y_pos, capacity)
13 plt.yticks(y_pos, rooms)
14
15 plt.show()
16
```

Usage

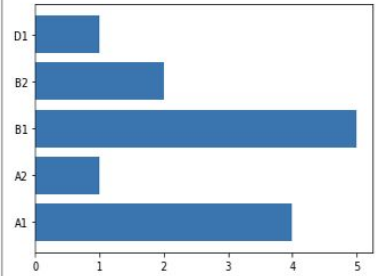
Here you can get help of any object

Variable explorer File explorer Help

Python console

Console 1/A

In [175]: runfile('C:/Users/antho/.spyder-py3/temp.py', wdir='C:/Users/antho/.spyder-py3')



Room	Capacity
A1	4
A2	1
B1	5
B2	2
D1	1

In [176]:

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 16 Column: 1 Memory: 84 %

Matplotlib (6)

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\antho\.spyder-py3\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
```

Usage

Here you can get help of any

Variable explorer File explorer Help

IPython console

Console 1/A wdir='C:\Users\antho\.spyder-py3'

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

Age	all developers	python developers
25	38496	45372
26	42000	48876
27	46752	53850
28	49320	57287
29	53200	63016
30	56000	65998

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

```
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
```

Usage

Here you can get help

Variable explorer File explorer Help

Python console

Console 1/A

Developer salaries by age

Ages	all developers	python developers
25	38496	45372
26	42000	48876
27	46752	53850
28	49320	57287
29	53200	63016
30	56000	65998

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

```
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
```

Usage

Here you can get help

Variable explorer File explorer Help

IPython console

Console 1/A

Users/antho/.spyder-py3'

Developer salaries by age

Age	all developers	python developers
25	38496	45372
26	42000	48876
27	46752	53850
28	49320	57287
29	53200	63016
30	56000	65998

In [127]:

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

Matplotlib (9)

The screenshot shows the Spyder Python IDE interface. The main editor window displays a Python script for creating a pie chart. The code is as follows:

```
1 """ Pie Charts """
2
3 from matplotlib import pyplot as plt
4
5 slices = [25, 75]
6 labels = ["a quarter", "three quarters"]
7 colors = ["grey", "green"]
8
9 plt.pie(slices, labels=labels, colors=colors)
10
11 plt.title("sample pie chart")
12 plt.tight_layout()
13 plt.show()
14
15 #Tip: It is not mandatory for values to add up exactly to 100...
16
```

The Python console on the right shows the execution of the script, resulting in a pie chart titled "sample pie chart". The chart has two slices: a grey slice representing "a quarter" (25%) and a green slice representing "three quarters" (75%).

Usage

Here you can get help of any

Variable explorer File explorer Help

Python console

Console 1/A

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

sample pie chart

a quarter

three quarters

In [130]:

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 16 Column: 1 Memory: 84 %

Matplotlib (10)

The screenshot shows the Spyder Python IDE interface. The main editor window displays a Python script named 'temp.py' with the following code:

```
1 """ Pie Charts """
2
3 from matplotlib import pyplot as plt
4
5 slices = [25, 25, 50]
6 labels = ["a quarter", "another quarter", "a half"]
7 colors = ["grey", "green", "red"]
8 explode = [0, 0.1, 0]
9
10 plt.pie(slices, labels=labels, colors=colors, explode=explode, shadow=True)
11
12 plt.title("sample pie chart")
13 plt.tight_layout()
14 plt.show()
15
16 #Tip: It is not mandatory for values to add up exactly to 100...
17
```

The right-hand side of the IDE contains several panels. At the top is a 'Usage' panel. Below it are the 'Variable explorer', 'File explorer', and 'Help' panels. The 'IPython console' panel is active, showing the command prompt 'wdir='C:/Users/antho/.spyder-py3'' and a rendered pie chart. The pie chart is titled 'sample pie chart' and consists of three slices: a grey slice labeled 'a quarter' (exploded), a green slice labeled 'another quarter', and a red slice labeled 'a half'. The console output below the chart reads 'In [137]:'.

At the bottom of the IDE, the status bar shows: Permissions: RW, End-of-lines: CRLF, Encoding: ASCII, Line: 17, Column: 1, Memory: 84 %.

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

```
1 """ Stack Plots """
2
3 from matplotlib import pyplot as plt
4
5 minute = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
6 pm = [0, 1, 1, 2, 3, 4, 5, 1, 1, 8]
7 sg = [1, 1, 2, 2, 3, 3, 4, 5, 1, 2]
8 sf = [1, 1, 2, 2, 2, 2, 4, 4, 4, 2]
9 pf = [1, 2, 2, 3, 1, 1, 1, 1, 2, 2]
10
11 labels = ["pm", "sg", "sf", "pf"]
12
13 plt.stackplot(minute, pm, sg, sf, pf, labels=labels)
14
15 plt.legend(loc="upper left")
16 plt.title("sample stack plot")
17 plt.tight_layout()
18 plt.show()
19
20
```

Usage

Here you can get help of any

Variable explorer File explorer Help

IPython console

Console 1/A

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

sample stack plot

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

In [141]:

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

Matplotlib (12)

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

```
1 """ Histogram """
2
3 from matplotlib import pyplot as plt
4
5 ages = [18, 19, 20, 22, 28, 30, 44, 50, 51, 55]
6
7 plt.hist(ages, bins=2, edgecolor="yellow")
8
9 plt.title("Responds per age")
10 plt.xlabel("ages")
11 plt.ylabel("responds")
12 plt.tight_layout()
13
14 plt.show()
15
16
```

Usage

Here you can get help of any

Variable explorer File explorer Help

IPython console

Console 1/A

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

ages	responds
20-35	6
35-55	4

In [144]:

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 15 Column: 1 Memory: 91 %

Matplotlib (13)

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

```
1 """ Histogram """
2
3 from matplotlib import pyplot as plt
4
5 ages = [18, 19, 20, 22, 28, 30, 44, 50, 51, 55]
6 bins = [10, 20, 30, 40, 50, 60]
7
8 plt.hist(ages, bins=bins, edgecolor="yellow")
9
10 plt.title("Responds per age")
11 plt.xlabel("ages")
12 plt.ylabel("responds")
13 plt.tight_layout()
14
15 plt.show()
16
17
```

Usage

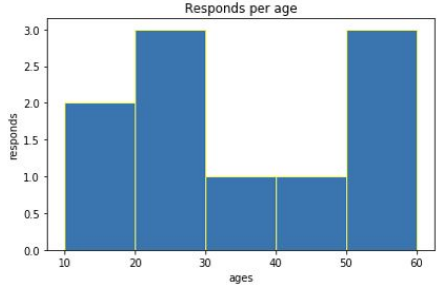
Here you can get help of any

Variable explorer File explorer Help

IPython console

Console 1/A

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



ages	responds
10-20	2.0
20-30	3.0
30-40	1.0
40-50	1.0
50-60	3.0

In [145]:

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 16 Column: 1 Memory: 89 %

Matplotlib (14)

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

```
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
10 plt.scatter(x, y, s=100, edgecolor="black")
11
12 plt.title("scatter plot example")
13 plt.tight_layout()
14
15 plt.show()
16
17
```

Usage

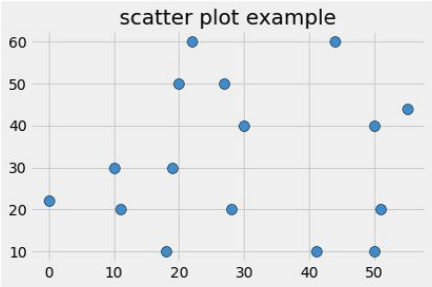
Here you can get help of any

Variable explorer File explorer Help

IPython console

Console 1/A

In [152]: runfile('C:/Users/antho/.spyder-py3/temp.py', wdir='C:/Users/antho/.spyder-py3')



In [153]:

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 17 Column: 1 Memory: 84 %

Matplotlib (15)

The image shows the Spyder Python IDE interface. The main editor window displays a Python script named `temp.py` with the following code:

```
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
```

The IPython console on the right shows the execution of the script:

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

The console also displays a scatter plot titled "scatter plot example". The plot shows data points where the x-axis ranges from 0 to 50 and the y-axis ranges from 10 to 60. The size and color of each point correspond to the `sizes` and `colors` arrays in the code. A color bar on the right of the plot is labeled "answers" and ranges from 0 to 10. The plot uses the "fivethirtyeight" style and "Spectral" colormap.

At the bottom of the console, the following command is shown:

```
In [160]:
```

The status bar at the bottom of the IDE indicates: 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

```
1 """ Subplots """
2
3 from matplotlib import pyplot as plt
4
5 plt.style.use("fivethirtyeight")
6
7 ages = [20, 25, 30, 35, 40, 45, 50, 55, 60]
8 dev_wages = [2000, 2500, 3000, 3400, 3900, 4200, 4400, 5000, 5200]
9 py_wages = [2500, 2200, 3200, 3200, 3700, 4000, 4800, 5500, 5900]
10 ja_wages = [1700, 2500, 3000, 3400, 3900, 4200, 4400, 5000, 5200]
11
12 fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1)
13
14 ax1.plot(ages, dev_wages, label="all devs")
15 ax2.plot(ages, py_wages, label="python devs")
16 ax2.plot(ages, ja_wages, label="java devs")
17
18 ax1.legend()
19 ax1.set_ylabel("wage")
20
21 ax2.legend()
22 ax2.set_xlabel("age")
23 ax2.set_ylabel("wage")
24
25 #plt.tight_layout()
    plt.show()
```

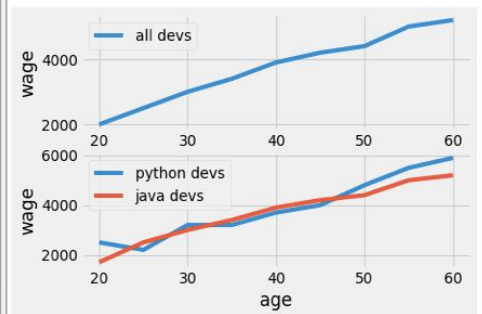
Usage

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

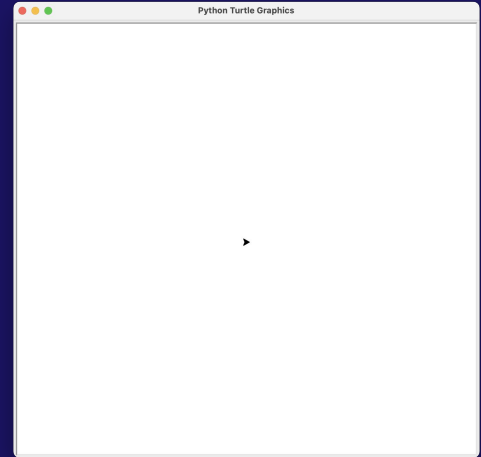
Python Turtle: Introduction (2)

- 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 x turtle_2.py x
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
```



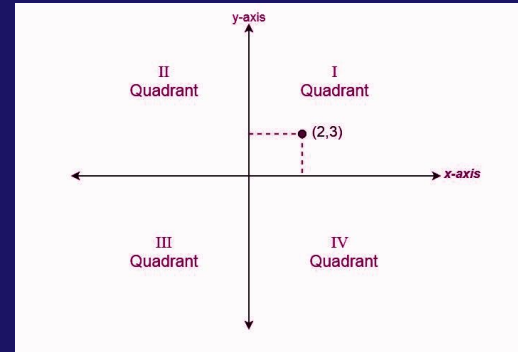
Python Turtle: Introduction (6)

```
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 x turtle_8.py x
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 shapsize().
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('shapsize()')
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 x turtle_16.py x
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 x turtle_16.py x turtle_17.py x
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 x turtle_19.py x
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 x  turtle_22.py x
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 x turtle_22.py x
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 x turtle_22.py x turtle_23.py x turtle_24.py x
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 x turtle_26.py x
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 x turtle_26.py x turtle_27.py x
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
```

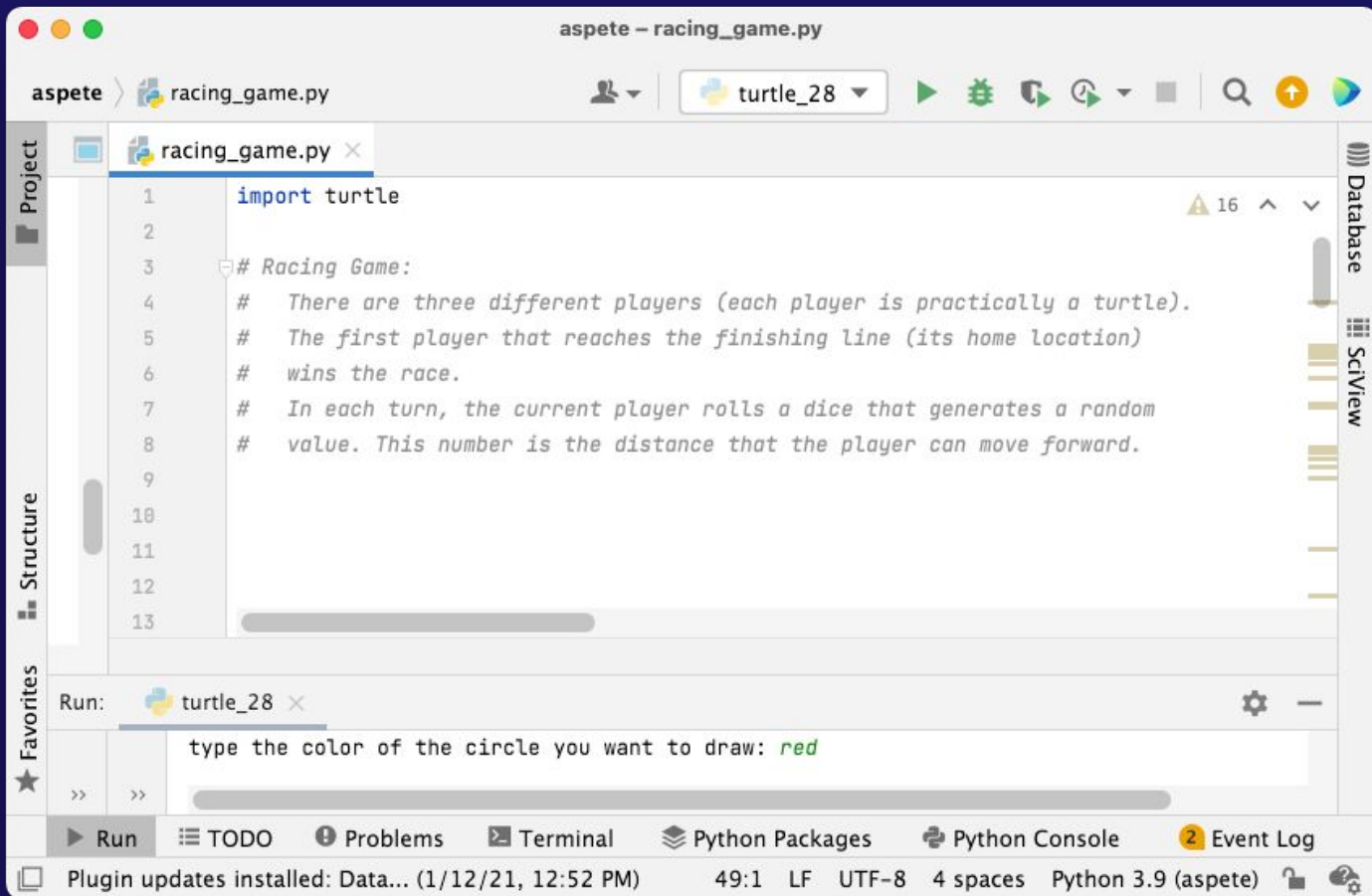


Python Turtle: Encompassing Basic Python Concepts (6)

```
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

Project: racing_game.py x

```
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
```

Run: turtle_28 x

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 x

```
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 x

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 x

```
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 # Step2: create three distinct players using different colors.
12
13
```

Run: turtle_28 x

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)

The screenshot shows an IDE window titled "aspete - racing_game.py". The main editor area displays the following Python code:

```
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 # Step2: create three distinct players using different colors.
12 p1 = turtle.Turtle()
13 p1.color('blue')
14 p2 = turtle.Turtle()
15 p2.color('green')
16 p3 = turtle.Turtle()
17 p3.color('red')
18
```

The IDE interface includes a Project sidebar on the left, a Run console at the bottom, and a status bar at the very bottom. The status bar shows "Plugin updates installed: Dat... (1/12/21, 12:52 PM)", "19:18 LF UTF-8 4 spaces Python 3.9 (aspete)", and a notification for "2 Event Log".

aspete – racing_game.py

aspete > racing_game.py

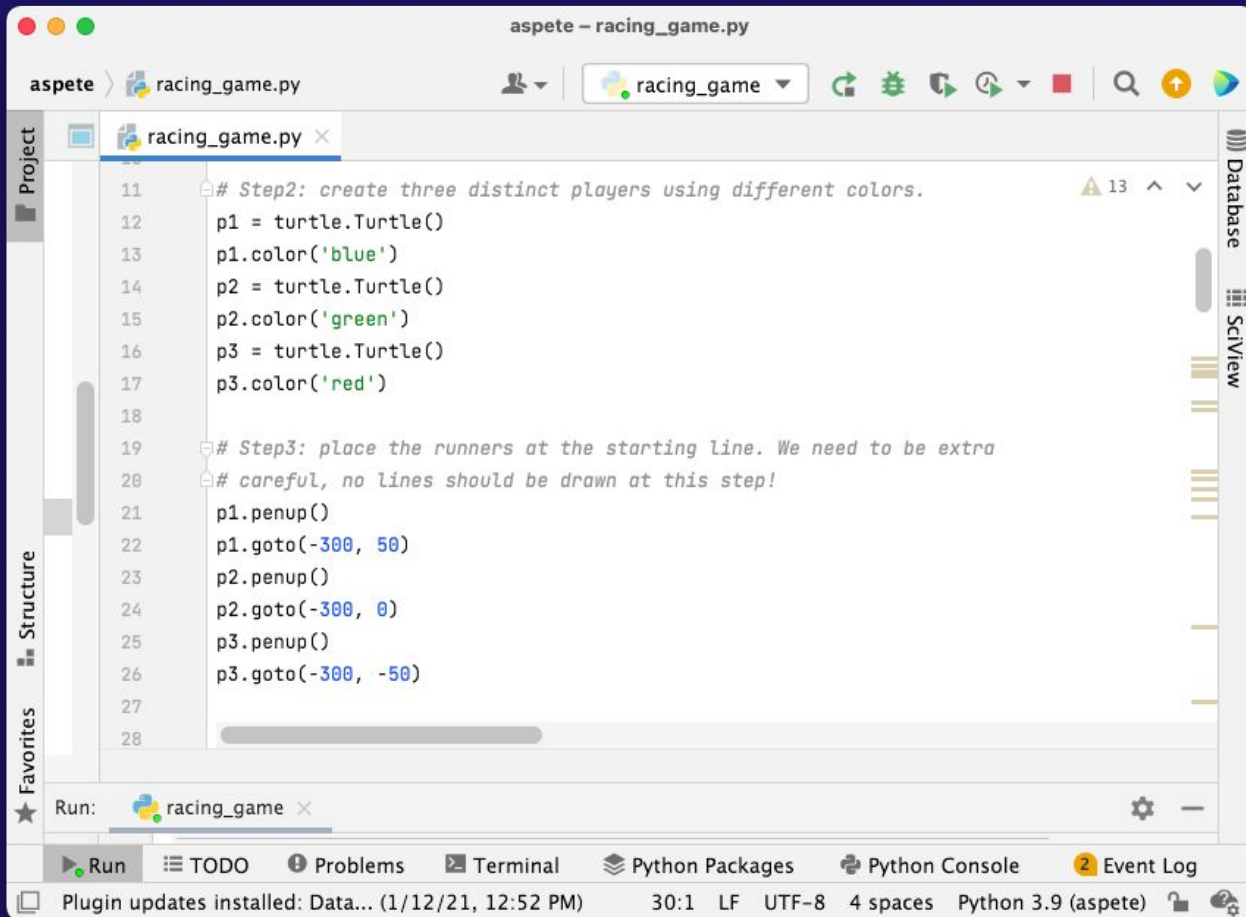
Project: racing_game.py

```
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 # Step2: create three distinct players using different colors.
12 p1 = turtle.Turtle()
13 p1.color('blue')
14 p2 = turtle.Turtle()
15 p2.color('green')
16 p3 = turtle.Turtle()
17 p3.color('red')
18
19 # Step3: place the runners at the starting line. We need to be extra
20 # careful, no lines should be drawn at this step!
21
```

Run: racing_game

Run | TODO | Problems | Terminal | Python Packages | Python Console | 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.py x
12 p1 = turtle.Turtle()
13 p1.color('blue')
14 p2 = turtle.Turtle()
15 p2.color('green')
16 p3 = turtle.Turtle()
17 p3.color('red')
18
19 # Step3: place the runners at the starting line. We need to be extra
20 # careful, no lines should be drawn at this step!
21 p1.penup()
22 p1.goto(-300, 50)
23 p2.penup()
24 p2.goto(-300, 0)
25 p3.penup()
26 p3.goto(-300, -50)
27
28 # Step4: create the dice
29
```

Run: racing_game x

Run | TODO | Problems | Terminal | Python Packages | Python Console | 2 Event Log

Plugin updates installed: Data... (1/12/21, 12:52 PM) 34:1 LF UTF-8 4 spaces Python 3.9 (aspete)

The screenshot shows an IDE window titled "aspete - racing_game.py". The main editor area displays the following Python code:

```
13 p1.color('blue')
14 p2 = turtle.Turtle()
15 p2.color('green')
16 p3 = turtle.Turtle()
17 p3.color('red')
18
19 # Step3: place the runners at the starting line. We need to be extra
20 # careful, no lines should be drawn at this step!
21 p1.penup()
22 p1.goto(-300, 50)
23 p2.penup()
24 p2.goto(-300, 0)
25 p3.penup()
26 p3.goto(-300, -50)
27
28 # Step4: create the dice
29 dice = [1, 2, 3, 4, 5, 6]
30
```

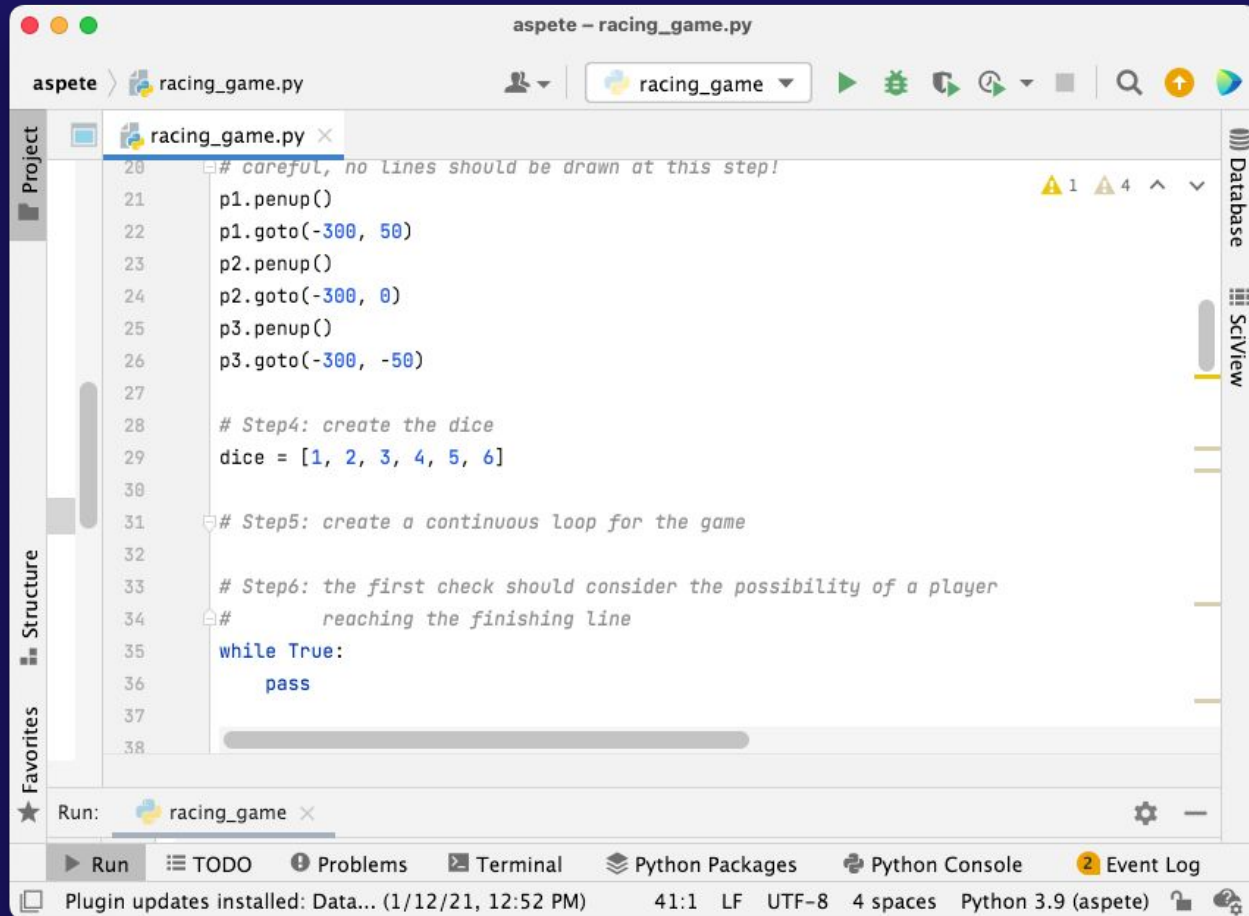
The IDE interface includes a top toolbar with icons for file operations, a search bar, and a run button. On the left, there are panels for "Project" (showing "racing_game.py"), "Structure", and "Favorites". On the right, there are panels for "Database" and "SciView". At the bottom, there is a "Run" panel with a "racing_game" tab, and a status bar showing "Plugin updates installed: Data... (1/12/21, 12:52 PM)", "47:1 LF UTF-8 4 spaces Python 3.9 (aspete)", and system icons.

```
aspete -- racing_game.py
aspete > racing_game.py
racing_game.py x
15 p2.color('green')
16 p3 = turtle.Turtle()
17 p3.color('red')
18
19 # Step3: place the runners at the starting line. We need to be extra
20 # careful, no lines should be drawn at this step!
21 p1.penup()
22 p1.goto(-300, 50)
23 p2.penup()
24 p2.goto(-300, 0)
25 p3.penup()
26 p3.goto(-300, -50)
27
28 # Step4: create the dice
29 dice = [1, 2, 3, 4, 5, 6]
30
31 # Step5: create a continuous loop for the game
32
--
Run: racing_game x
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)
```

The image shows a screenshot of an IDE window titled "aspete - racing_game.py". The window contains a Python script for a racing game. The code is as follows:

```
17     p3.color('red')
18
19     # Step3: place the runners at the starting line. We need to be extra
20     # careful, no lines should be drawn at this step!
21     p1.penup()
22     p1.goto(-300, 50)
23     p2.penup()
24     p2.goto(-300, 0)
25     p3.penup()
26     p3.goto(-300, -50)
27
28     # Step4: create the dice
29     dice = [1, 2, 3, 4, 5, 6]
30
31     # Step5: create a continuous loop for the game
32     while True:
33         pass
34
```

The IDE interface includes a Project sidebar on the left, a Run console at the bottom, and a status bar at the very bottom. The status bar shows "Plugin updates installed: Data... (1/12/21, 12:52 PM)", "37:1 LF UTF-8 4 spaces Python 3.9 (aspete)", and a lock icon.



```
aspete -- racing_game.py
aspete > racing_game.py
racing_game.py x
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 x

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.py x
33 # Step6: the first check should consider the possibility of a player
34 #     reaching the finishing line
35
36 # Step7: implement the random number generation through the dice
37 #     rolling based on the current player's input
38
39 while True:
40     if p1.pos() >= (300, 50):
41         print('p1 wins!!!!')
42         break
43     elif p2.pos() >= (300, 0):
44         print('p2 wins!!!!')
45         break
46     elif p3.pos() >= (300, -50):
47         print('p3 wins!!!!')
48         break
49     else:
50         pass
51
```

Run: racing_game x

Run | TODO | Problems | Terminal | Python Packages | Python Console | 2 Event Log

Plugin updates installed: Data... (1/12/21, 12:52 PM) 52:1 LF UTF-8 4 spaces Python 3.9 (aspete)

The screenshot shows an IDE window titled "aspete - racing_game.py". The main editor displays the following Python code:

```
44     print('p2 wins!!!!')
45     break
46     elif p3.pos() >= (300, -50):
47         print('p3 wins!!!!')
48         break
49     else:
50         dice_roll = input('p1 press "Enter" to roll the dice')
51         distance = 10 * random.randint(1, 6)
52         print('p1 going forward', distance, 'units')
53
54         dice_roll = input('p2 press "Enter" to roll the dice')
55         distance = 10 * random.randint(1, 6)
56         print('p2 going forward', distance, 'units')
57
58         dice_roll = input('p3 press "Enter" to roll the dice')
59         distance = 10 * random.randint(1, 6)
60         print('p3 going forward', distance, 'units')
61
```

The IDE interface includes a Project sidebar on the left, a Run console at the bottom with a "Run" button, and a status bar at the very bottom showing "Plugin updates installed: Data... (1/12/21, 12:52 PM)", "26:1 LF UTF-8 4 spaces Python 3.9 (aspete)", and an "Event Log" icon.

aspete – racing_game.py

aspete > racing_game.py

racing_game.py

```
39 # Step8: implement players' movement
40
41 while True:
42     if p1.pos() >= (300, 50):
43         print('p1 wins!!!!')
44         break
45     elif p2.pos() >= (300, 0):
46         print('p2 wins!!!!')
47         break
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
56
```

Run: racing_game

Run | TODO | Problems | Terminal | Python Packages | Python Console | 2 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

```
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
```

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 'αναταρ' για κάθε έναν από τους χρήστες
 - c. να υπάρχει ένα ψευδοτυχαίο γεγονός πτώσης το οποίο θα ακινητοποιεί κάποιων από τους παίκτες