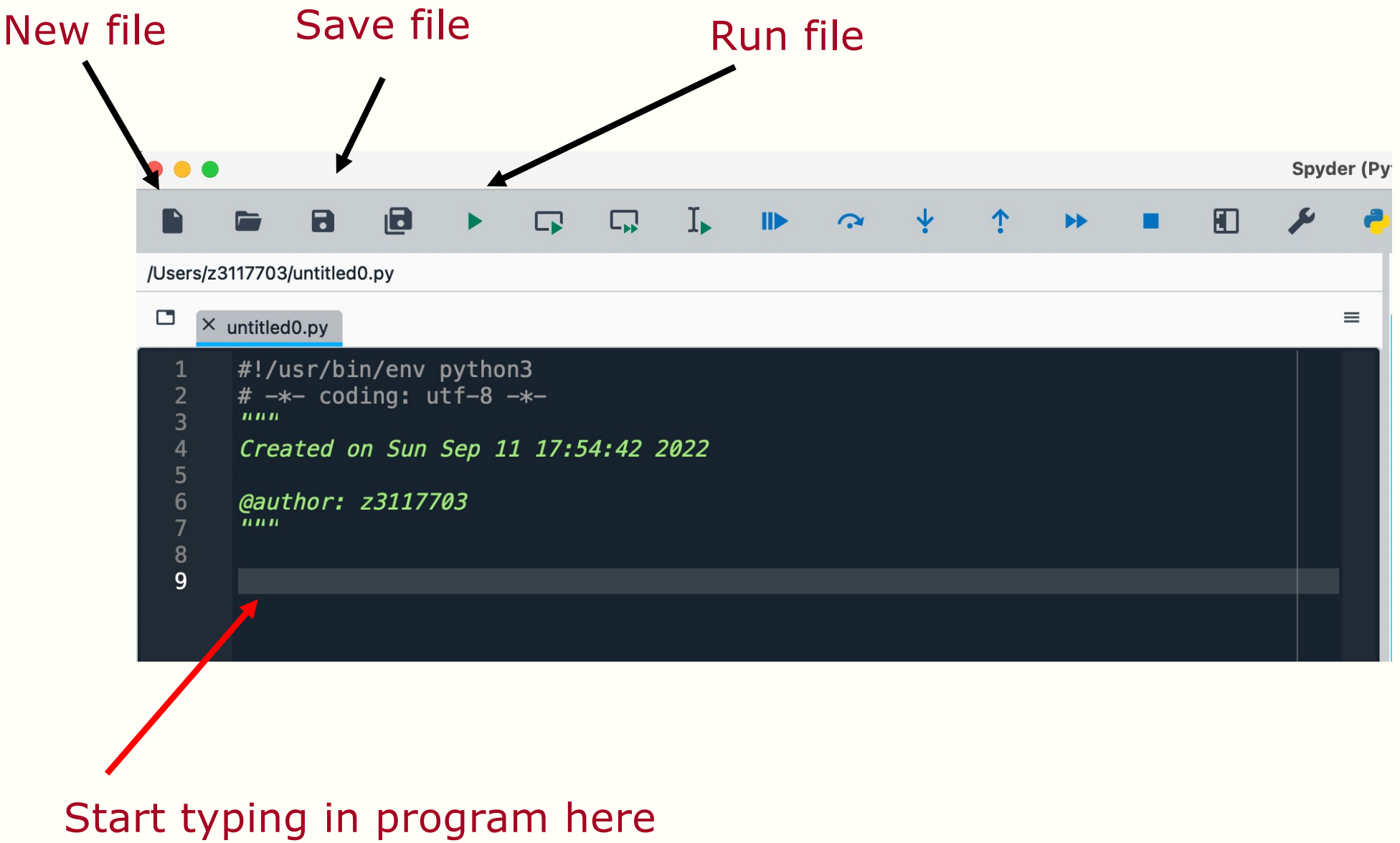
ENGG1811 Computing for Engineers

Day 1 Getting started

The Spyder editor



Draw a triangle (version 1)

```
 9  import draw
10
11  draw.start()
12  draw.draw_line(0, 0, 1, 0)
13  draw.draw_line(1, 0, 0.8, 0.9)
14  draw.draw_line(0.8, 0.9, 0, 0)
```

- After typing the program, you should save it:
  - Do give the program a meaningful name.
  - Organise files using folders
  - Note that Python programs have the extension .py
  - Don’t forget to save the file regularly when you work on Spyder
- You can run the program using the run button 
- Results will be displayed in a plot under the “Plots” tab

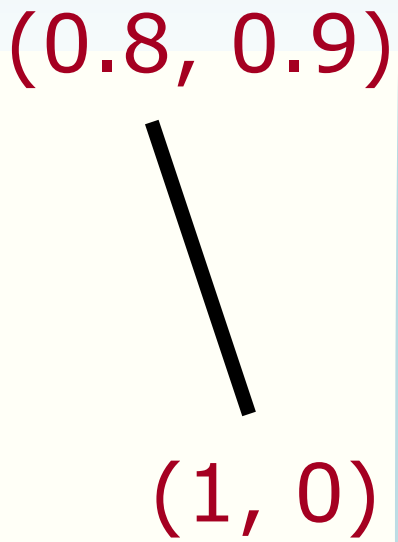
Program execution (1)

```
 9  import draw
10
11  draw.start()
12  draw.draw_line(0, 0, 1, 0)
13  draw.draw_line(1, 0, 0.8, 0.9)
14  draw.draw_line(0.8, 0.9, 0, 0)
```

- This program consists of 5 statements
  - At lines 9, 11 and 12-14
- The statements are executed in the order that they appear
- Line 9 enables us to use the draw facility
- You always need to use draw.start() before drawing

Program execution (2)

```
 9   import draw
10
11   draw.start()
12   draw.draw_line(0, 0, 1, 0)
13   draw.draw_line(1, 0, 0.8, 0.9)
14   draw.draw_line(0.8, 0.9, 0, 0)
```



- Each code in Line 12-14 draws a line in the picture
- E.g., draw.draw_line(1, 0, 0.8, 0.9) draws a line segment between (1, 0) and (0.8, 0.9)
  - First two numbers are the x- and y-coordinates of one end
  - Last two numbers are the x- and y-coordinates of the other end

Draw a triangle (Version 2)

```
 9  import draw
10
11  draw.start()
12  # Draw a triangle with the vertices at (0,0), (1,0) and (0.8, 0.9)
13  draw.draw_line(0, 0, 1, 0)
14  draw.draw_line(1, 0, 0.8, 0.9)
15  draw.draw_line(0.8, 0.9, 0, 0)
```

- Comments are added to explain what a program does
  - All text after the # symbol is comment
- Comments are ignored when a program is executed
- Comments are for people to read

Change a vertex

```
 9  import draw
10
11  draw.start()
12  # Draw a triangle with the vertices at (0,0), (1,0) and (0.8, 0.9)
13  draw.draw_line(0, 0, 1, 0)
14  draw.draw_line(1, 0, 0.8, 0.9)
15  draw.draw_line(0.8, 0.9, 0, 0)
```

- Let us say we want to change the vertex from (1,0) to (0.2, 0.5), and I will purposely make a mistake

```
 9  import draw
10
11  draw.start()
12  # Draw a triangle with the vertices at (0,0), (0.2,0.5) and (0.8, 0.9)
13  draw.draw_line(0, 0, 0.2, 0.5)
14  draw.draw_line(0.2, 0, 0.8, 0.9)
15  draw.draw_line(0.8, 0.9, 0, 0)
```

Draw a triangle (Version 3)

- Use variables to specify the coordinates
- Complete the code in sample_draw_a_triangle.py
  - Fill in Lines 21, 26 using the sample code below

```
13  import draw
14
15  # Draw a triangle with vertices (x0, y0), (x1, y1) and (x2, y2)
16  x0 = 0
17  y0 = 0
18  x1 = 0.2
19  y1 = 0.5
20  x2 = 0.8
21  y2 = 0.9
22
23  draw.start()
24  draw.draw_line(x0, y0, x2, y2)
25  draw.draw_line(x2, y2, x1, y1)
26  draw.draw_line(x1, y1, x0, y0)
```

Summary and next step

- Skills acquired by now:
  - Python skill
    - Arithmetic operators
    - Define and use variables
    - Repetition using “for” and “range”
  - Drawing commands:
    - draw_start()
    - draw.drawline() draws a line connecting 2 points
    - draw.move()
    - draw.turn()
- Work on Exercises 1 to 3.

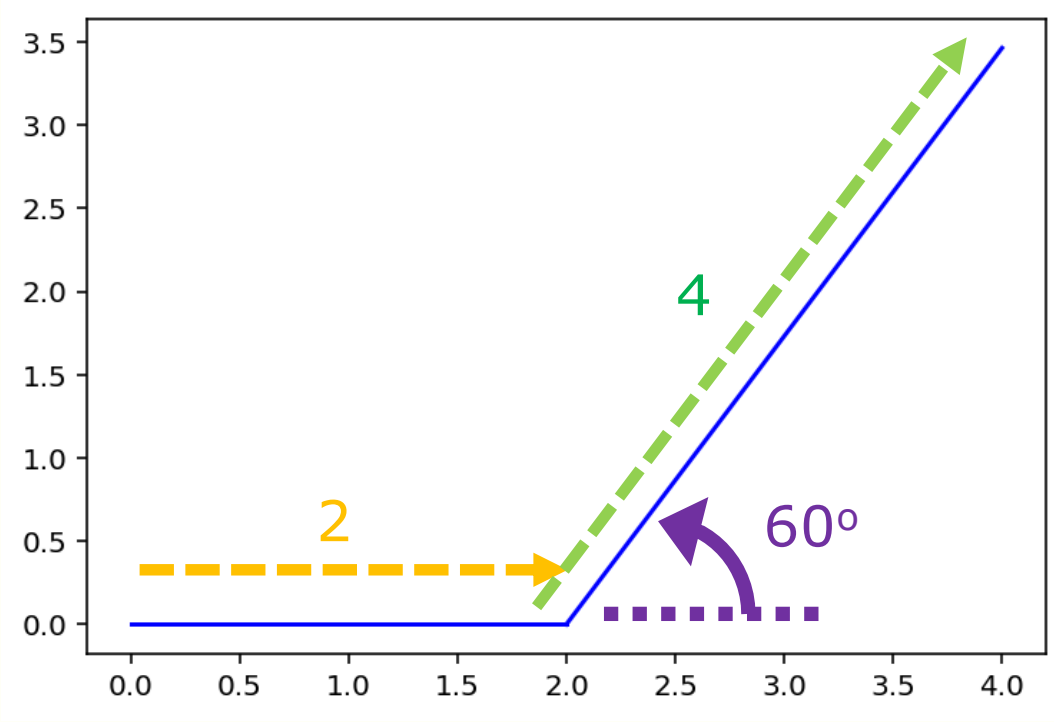
Moving paint brush: draw.move and draw.turn

- File: sample_start_move_turn.py

```
draw.start()     # starting position (0,0) pointing at +x axis
draw.move(2)     # move by a distance of 2
draw.turn(60)    # turn anti-clockwise an angle of 60 degrees
draw.move(4)     # move by a distance of 4
```

Initial position(0,0)

Initial direction



Summary and next step

- Skills acquired by now:
  - Python skill
    - Arithmetic operators
    - Define and use variables
    - Repetition using “for” and “range”
  - Drawing commands:
    - draw_start()
    - draw.drawline() draws a line connecting 2 points
    - draw.move()
    - draw.turn()

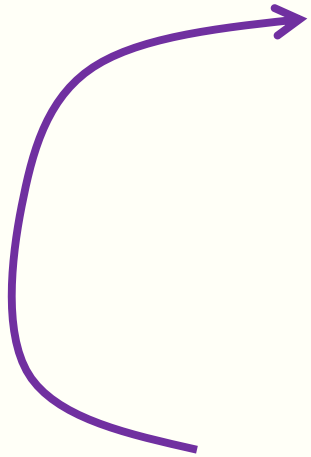
- Work on Exercises 4 and 5.

Reducing repetition

- On the left: code to draw a square with edge length 2
- Code repeated 4 times. New trick: for

```
import draw

draw.start()

draw.move(2)
draw.turn(90)
draw.move(2)
draw.turn(90)
draw.move(2)
draw.turn(90)
draw.move(2)
draw.turn(90)
```

```
import draw

draw.start()

for i in range(4):
    draw.move(2)
    draw.turn(90)
```



Repeat the indented code four times

Indentation e.g., tab

Summary and next step

- Skills acquired by now:
  - Python skill
    - Arithmetic operators
    - Define and use variables
    - Repetition using “for” and “range”
  - Drawing commands:
    - draw_start()
    - draw.drawline() draws a line connecting 2 points
    - draw.move()
    - draw.turn()

- Work on Exercises 6 and 7.