Introduction to Programming/ Week 1: Python interactive shell / Turtle graphics

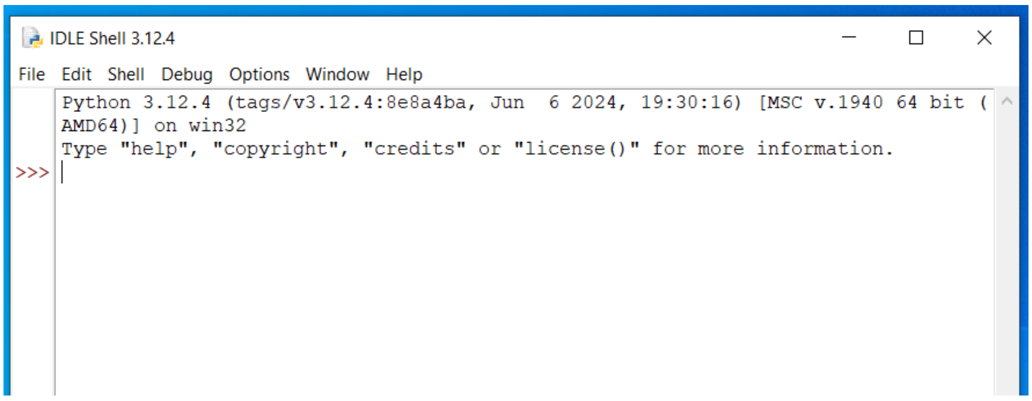
Turtle graphics

There are many ways to draw pictures by writing code. A simple one that we will use to get used to writing Python commands into the shell is Turtle graphics.

The idea is that you are controlling a turtle which walks around the screen. The turtle has a pen that draws a line behind it.

Load IDLE Shell

1. Open IDLE. The easiest way is to click the Start menu and search for 'IDLE'.
2. This should show you a IDLE Shell.



```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Setup

First, we import functions from the `turtle` package, which gives a set of commands for controlling the turtle.

```
from turtle import *
```

One command at a time

Sometimes on this page, I might give several commands. You must run these one at a time. (We'll deal with writing programs, where a sequences of commands are given all together, later.)

turtle.Terminator error

In the examples that follow, when you tell the turtle to move a window will appear showing that movement. Try not to close the window. If at any point in what follows you get a red error ending in `turtle.Terminator`, it is possible you have closed the window and are trying to move a turtle that isn't there any more. To get out of this situation, you can run

```
Screen()
home()
```

to make the window and the turtle reappear and try again.

Movement

Now you can tell your turtle to move around the screen. For example, to tell the turtle to move forwards 100 steps, use

```
forward(100)
```

You can tell the turtle to turn `left()` or `right()`. For example, to get it to turn 45 degrees left, use turn 45 degrees left, use

```
left(45)
```

The following code draws an equilateral triangle. Run one command at a time and don't close the drawing window in between. It is a good idea to run `reset()` before each new set of commands to clear what we have drawn previously.

```
forward(100)
left(120)
forward(100)
left(120)
forward(100)
```

Can you adapt the code to draw a square?

You can always send the turtle back to its starting position and orientation using `home()`.

```
forward(50)
home()
```

The amounts we ask the turtle to move can be randomised using the `random` package. If you haven't yet imported this, do so

```
from random import *
```

We can ask the turtle to move a random amount forwards using `randint`.

```
forward(randint(100,200))
```

We can ask it to turn a random amount and then walk forwards again.

```
right(randint(45,135))
forward(randint(100,200))
```

Coordinates

You can access the current position of the turtle using `pos()`. For example, here we ask the turtle to report its current position.

```
pos()
```

You can tell the turtle to go to a particular set of coordinates using `setpos`, for example

```
setpos(20,30)
```

You can set the x and y coordinates separately.

```
setx(100)
```

and

```
sety(-200)
```

You can also set the direction the turtle is pointing using `setheading`. For example, to make the turtle turn east (the original orientation), use

```
setheading(0)
```

From here we turn left, so for example north is `90`, west is `180` and south is

270. You can ask the turtle to report its current position using `heading()`.

Shapes

Draw a circle using `circle(r)`, where `r` is the radius.

```
circle(10)
```

Note that the circle is drawn from the current position on the perimeter. To see what this means, try drawing a bigger circle on top of that one.

```
circle(50)
```

Text

You can write text at the current position using `write()`. For example

```
write("Hello, World!")
```

Pen options

By default, the turtle draws a line behind it. You can tell it to stop drawing a line by lifting the pen `up()` before moving, and tell it to start drawing again using `down()`.

For example, this code moves the turtle forwards without drawing, then puts the pen down and draws a line.

```
up()
forward(50)
down()
forward(50)
```

There is a version of `setpos()` that moves the turtle to a coordinate without drawing a line, which is `teleport()`. Here we move the turtle with and without lines to show this in action.

```
setpos(100,100)
teleport(100,-100)
setpos(0,0)
```

You can change the thickness of the pen using `width()`.

For example, here we draw a thin line then a thick line.

```
forward(100)
width(5)
forward(100)
```

Colo(u)r

You can change the colour of the pen using `color()` (note the American spelling!). For example, here we draw a green line instead of the default black one.

```
color('green')
forward(100)
```

As well as `color()` to control the line colour, there is also `fillcolor()` to fill in the shapes you draw. You start filling using `begin_fill()` and once you have completed the shape to fill `end_fill()`.

```
color("blue")
fillcolor("yellow")
```

```
begin_fill()


setpos(100,0)
setpos(100,50)
setpos(0,0)

end_fill()
```

You can draw dots as you go (for example, as vertices of a polygon) using `dot()`.

```
forward(100)
dot()
left(120)
forward(100)
dot()
left(120)
forward(100)
dot()
```

You can change the background colour of the screen using `bgcolor()`. For example

```
bgcolor("yellow")
```

Hiding the turtle

You can hide the turtle but leave its tracks behind.

```
hideturtle()
```

You can show the turtle again.

```
showturtle()
showturtle()
```

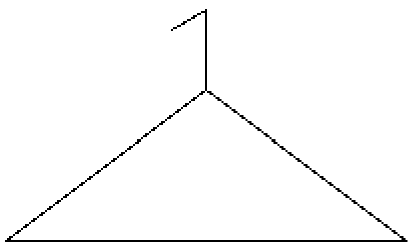
Undo

If at any point you want to take back what you just did, use `undo()`. For example

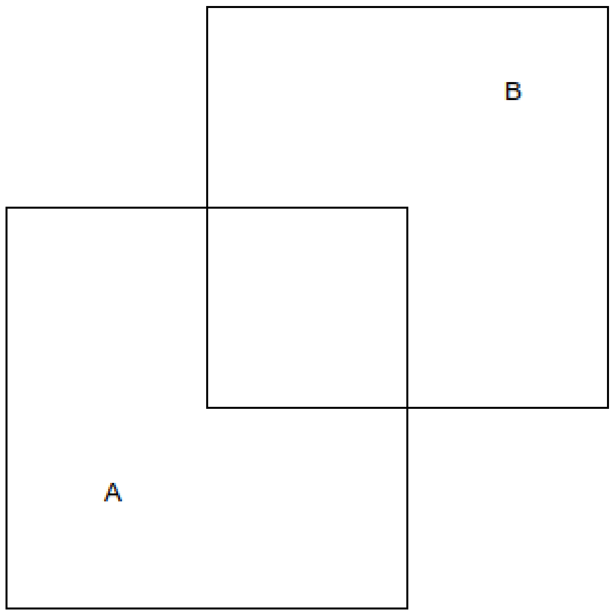
```
forward(100)
undo()
```

Exercises

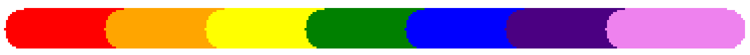
1. Can you make the following polygons?
    a. Pentagon.
    b. Hexagon.
    c. Heptagon.
    d. Octagon.
2. Can you draw a star?
3. Can you draw a basic coat hanger like the one pictured?



4. Can you draw a diagram like this?



5. Can you draw a multi-coloured line like this one?



6. What else can you draw? Perhaps you know some other programming techniques you could use here?