COMP1021 Large Turtle Module Summary

Here is a summary of the turtle module that we use on the course.

Turtle (the module)

The turtle module contains two types of functions: functions for the actual turtles, and functions for other things, particularly for the turtle window.

Actual turtles ➤ (one or more)

The following functions can be used for the default turtle and any user-created turtles which are created by *t = turtle.Turtle()*.

- `up`
- `down`
- `goto`
- `forward`
- `backward`
- `left`
- `right`
- `home`
- `dot`
- `circle`
- `setheading`
- `pencolor`
- `fillcolor`
- `color`
- `begin_fill`
- `end_fill`
- `write`
- `shape`
- `shapesize`
- `hideturtle`
- `showturtle`
- `xcor`
- `ycor`
- `pos`
- `heading`
- `width`
- `speed`
- `clear`
- `onclick`
- `ondrag`

} *Event handling*

Turtle screen (only one)

('Turtle screen' means 'turtle window').

This is a list of functions for other things, particularly for the turtle window.

- `colormode`
- `textinput`
- `numinput`
- `addshape`
- `tracer`
- `setworldcoordinates`
- `setup`
- `bgcolor`
- `bgpic`
- `bye`
- `done`
- `listen`
- `onkeypress`
- `onscreenclick`

} *Related to event handling*

COMP1021 Large Turtle Module Summary

Moving and Drawing

***turtle_name*.up()**
Pulls the pen up, then no drawing when moving.

***turtle_name*.down()**
Puts the pen down, then have drawing when moving.

***turtle_name*.goto(*X*, *Y*)**
Moves turtle to the absolute position (*X*, *Y*)
e.g. `0,0`

***turtle_name*.forward(*DISTANCE*)**
Moves the turtle forward by *DISTANCE*, in the direction of the turtle e.g. `100`

***turtle_name*.backward(*DISTANCE*)**
Moves the turtle backward by *DISTANCE*
e.g. `100`
Does not change the direction of the turtle.

***turtle_name*.left(*ANGLE*)**
Turns turtle left by *ANGLE* degrees e.g. `45`

***turtle_name*.right(*ANGLE*)**
Turns turtle right by *ANGLE* degrees e.g. `45`

***turtle_name*.setheading(*ANGLE*)**
Sets the turtle's exact direction to *ANGLE*. An angle of 0 means the turtle points right. An angle of 90 means the turtle points up. And so on.

***turtle_name*.home()**
This is the same as `turtle.goto(0, 0)`

***turtle_name*.dot(*SIZE*)**
Draws a solid circle with diameter *SIZE* e.g. `50`
The center is at the current position of the turtle.
The circle is always solid, using the pen colour (not the fill colour). Works even if the pen is off the page.

***turtle_name*.circle(*RADIUS*, *EXTENT*)**
Draws a circle with given *RADIUS*. If *RADIUS* is positive the circle is drawn to the left of the turtle. If *RADIUS* is negative it is drawn to the right.
EXTENT is optional. *EXTENT* is an angle that determines how many degrees are drawn.
An example pair of numbers: `200,90`

Handling Colour

***turtle_name*.pencolor(*PENCOLOR*)**
Sets the pen color to *PENCOLOR* e.g. `"red"`

***turtle_name*.fillcolor(*FILLCOLOR*)**
Sets the fill color to *FILLCOLOR* e.g. `"blue"`

***turtle_name*.color(*COLOR*)**
Sets both the pen color and the fill color to *COLOR.*
E.g. *turtle_name*`.color("red")`

***turtle_name*.color(*PENCOLOR*, *FILLCOLOR*)**
Sets the pen color to *PENCOLOR* and sets the fill color to *FILLCOLOR* e.g.
turtle_name`.color("red", "blue")`

We may or may not have time to consider the following this semester:

turtle.colormode(255)
Tells the turtle system to accept red (R), green (G), and blue (B) values to represent one colour. Each value is an integer in the range 0…255. For example, after executing `turtle.colormode(255)` the command *turtle_name*`.pencolor(165, 42, 42)` sets the turtle pen colour to brown, because brown is represented by the RGB values 165, 42, and 42.

COMP1021 Large Turtle Module Summary

Filling

***turtle_name*.begin_fill()**
Begins the color filling. Put this before the code which draws the shape you want to fill.

***turtle_name*.end_fill()**
Ends the color filling. Put this after the code which draws the shape you want to fill.

Text/Number Input & Output

```
turtle_name.write("TEXT",
  font=("FONTYPE",
        FONTSIZE, "FONTSTYLE"))
```
Writes *TEXT* at the current turtle position, using the **font** information, e.g. `font=("Arial", 20, "bold")`

```
turtle.textinput( "TITLE","PROMPT" )
```
Shows a small window which asks for a user's text input. *TITLE* specifies the text shown at the top of the small window, and *PROMPT* specifies the message shown.

```
turtle.numinput("TITLE", "PROMPT",
  DEFAULT, MIN, MAX)
```
Shows a small window which asks for a user's numerical input. *TITLE* is the text shown at the top of the small window, *PROMPT* is the message shown. The last three input values are optional: *DEFAULT* specifies the default value, while *MIN* and *MAX* specify the minimum value and maximum value allowed for the user's input. The function returns a float value.

Controlling Visibility

***turtle_name*.hideturtle()**
Hides the turtle, then you cannot see it in the turtle window.

***turtle_name*.showturtle()**
Shows the turtle, then you can see it in the turtle window.

COMP1021 Large Turtle Module Summary

Shape Control

`turtle.addshape("`*FILENAME***`")`**
Adds a new turtle image to the turtle system which can then be used by *`turtle_name`*`.shape()`. *FILENAME* is the file name of an image file. The image file must use the GIF format e.g. `monster.gif`. Put the image file in the same directory as the Python program.

`turtle_name`*`.shape("`SHAPE***`")`**
Sets the turtle's shape to *SHAPE*. These are the possible shapes: "arrow", "turtle", "circle", "square", "triangle", and "classic". Alternatively, a GIF image can be selected, if it is first added to the turtle system using `turtle.addshape()`.

***`turtle_name`*`.shapesize(`**
WIDTH_MULTIPLIER,
HEIGHT_MULTIPLIER **)**
Multiplies the turtle's width by the number *WIDTH_MULTIPLIER* and multiplies the turtle's height by the number *HEIGHT_MULTIPLIER*. (This command doesn't work if an image is being used for the turtle shape). An example:

```
turtle_name.shapesize(2, 3)
```

- The first value (2) multiplies the width of the turtle, i.e. the turtle width is doubled
- The second value (3) multiplies the length of the turtle, i.e. the turtle length is tripled

COMP1021 Large Turtle Module Summary

Screen Update Control

`turtle.tracer(True)`
After this line of code, new turtle drawings will appear on the screen as they are drawn.

`turtle.tracer(False)`
After this line of code, any turtle drawing are not shown - until you do `turtle.tracer(True)`, then everything drawn after `turtle.tracer(False)` is suddenly shown all at once.

Turtle Window Setup

`turtle.setup(` *WIDTH, HEIGHT* **`)`**
Resizes the turtle window to *WIDTH* x *HEIGHT*.

`turtle.bgcolor(` *BGCOLOR* **`)`**
Sets the turtle window's background colour to *BGCOLOR* e.g. `"blue"`.

`turtle.bgpic("`*FILENAME***`")`**
Sets the turtle window's background picture. *FILENAME* is the filename of a GIF image file, which usually has to be in the same directory as the Python program.

`turtle.setworldcoordinates(`
LEFT, BOTTOM, RIGHT, TOP **`)`**
Creates a customized coordinate system for the turtle window. *LEFT* is the x coordinate of the left side of the turtle window. *RIGHT* is the x coordinate of the right side of the window. *BOTTOM* is the y coordinate of the bottom of the turtle window. *TOP* is the y coordinate of the top of the window.

Creating Turtles

`turtle_name` **`= turtle.Turtle()`**
Creates a new turtle object called *turtle_name*.

Getting Turtle Properties

Here are some examples of extracting information out of a turtle. The variable names shown here on the left (`x`, `y`, etc) can be any variable name.

`x =` ***`turtle_name`*****`.xcor()`**
Returns the x position of the turtle.

`y =` ***`turtle_name`*****`.ycor()`**
Returns the y position of the turtle.

`x, y =` ***`turtle_name`*****`.position()`**
Returns the *(x position, y position)* of the turtle – two things are returned at the same time.

`h =` ***`turtle_name`*****`.heading()`**
Returns the angle of the turtle, in degrees.

`pc =` ***`turtle_name`*****`.pencolor()`**
Returns the pen colour that the turtle is using.

`fc =` ***`turtle_name`*****`.fillcolor()`**
Returns the fill colour that the turtle is using.

`w =` ***`turtle_name`*****`.width()`**
Returns the width of the turtle line.

And so on. Other commands not shown here can be used to extract almost any information out of a turtle object.

COMP1021 Large Turtle Module Summary

Event Handling

***turtle_name*.onclick(** *EVENT_HANDLER* **)**
After this, the function *EVENT_HANDLER* will be executed when the turtle called *turtle_name* is clicked.

***turtle_name*.ondrag(** *EVENT_HANDLER* **)**
After this, the function *EVENT_HANDLER* will be executed when the turtle called *turtle_name* is dragged.

turtle.onscreenclick(*EVENT_HANDLER* **)**
After this, the function called *EVENT_HANDLER* will be executed when the turtle window (not a turtle in the window) is clicked by the user.

turtle.onkeypress(
EVENT_HANDLER, **"***KEY***")**
After this, the function *EVENT_HANDLER* will be executed when *KEY* is pressed e.g. **"a"**.

turtle.listen()
Tells Windows to switch the focus to the turtle graphics window, so that any key presses go to the turtle window, and not to any other window.

turtle.done()
This must be included at the end of a turtle program which uses event handling e.g. to make sure clicking/dragging and key presses work properly.

Other Turtle Functions

***turtle_name*.width(** *WIDTH* **)**
Sets the line thickness to *WIDTH* e.g. `5`.
`turtle_name.pensize()` is the same as *`turtle_name.width()`*

***turtle_name*.speed(** *SPEED* **)**
Sets the turtle's animation speed to *SPEED* e.g. `5`
1 is slow, 10 is fast. 0 means very fast drawing.

***turtle_name*.clear()**
Deletes everything that the turtle called *turtle_name* has drawn.

turtle.bye()
Closes the turtle window.