COMP1021 Large Turtle Graphics Summary

Moving and Drawing

***turtle_name*.up()**
Pulls the pen up, then no drawing when moving.
Alternative command names: `penup()`, `pu()`

***turtle_name*.down()**
Puts the pen down, then have drawing when moving.
Alternative command names: `pendown()`, `pd()`

***turtle_name*.goto(*X*, *Y*)**
Moves turtle to the absolute position (*X*, *Y*)
e.g. `0,0` *Alternative command names:* `setpos()`, `setposition()`

***turtle_name*.forward(*DISTANCE*)**
Moves the turtle forward by *DISTANCE*, in the direction of the turtle e.g. `100`
Alternative command name: `fd()`

***turtle_name*.backward(*DISTANCE*)**
Moves the turtle backward by *DISTANCE*
e.g. `100`
Does not change the direction of the turtle.
Alternative command names: `bk()`, `back()`

***turtle_name*.left(*ANGLE*)**
Turns turtle left by *ANGLE* degrees e.g. `45`
Alternative command name: `lt()`

***turtle_name*.right(*ANGLE*)**
Turns turtle right by *ANGLE* degrees e.g. `45`
Alternative command name: `rt()`

***turtle_name*.setheading(*ANGLE*)**
Sets the turtle's direction to *ANGLE*.
Alternative command name: `seth()`

***turtle_name*.home()**
Moves turtle to the origin, i.e. coordinate (0, 0)

***turtle_name*.dot(*SIZE*)**
Draws a filled circle with diameter *SIZE* e.g. `50`
The center is at the current position of the turtle.
The circle is always filled. Works even if the pen has been taken up, 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 it 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(**
PENCOLOR, *FILLCOLOR* **)**
Sets the pen color to *PENCOLOR* and optionally sets the fill color to *FILLCOLOR* at the same time
e.g. `"red", "blue"`

We may or may not have time to consider the following two commands and concepts this semester:

turtle.colormode(255)
Tells the turtle system to accept red (R), green (G), and blue (B) values, which are integers in the range 0…255, for the colour values. For example, after `turtle.colormode(255)` the command `turtle.pencolor(165, 42, 42)` sets the pen colour to brown.

turtle.colormode(1.0)
Tells the turtle system to accept red (R), green (G), and blue (B) values, which are floats in the range 0.0…1.0, for the colour values. For example, after `turtle.colormode(1.0)` the command `turtle.pencolor(0.647, 0.165, 0.165)` sets the pen colour to brown.

COMP1021 Large Turtle Graphics 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 Input and 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 string 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* specifies the text shown at the top of the small window, and *PROMPT* specifies 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.

Visibility Control

***turtle_name*.hideturtle()**
Hides the turtle, then you cannot see it in the turtle window. *Alternative command name:* `ht()`

***turtle_name*.showturtle()**
Shows the turtle, then you can see it in the turtle window. *Alternative command name:* `st()`

Shape Control

turtle.addshape("*FILENAME*")
Adds a new turtle shape to the turtle system which can then be used by `turtle_name.shape()`. *FILENAME* is the file name of a GIF image file, usually in the same directory as the Python program. *Alternative name:* `register_shape()`

***turtle_name*.shape("*SHAPE*")**
Sets the turtle's shape to *SHAPE*. There are the possible shapes: "arrow", "turtle", "circle", "square", "triangle", and "classic". A GIF image can be selected if it has been added to the turtle system.

***turtle_name*.shapesize(**
WIDTH_STRETCH_FACTOR,
HEIGHT_STRETCH_FACTOR,
OUTLINE_WIDTH **)**
Stretches the turtle's shape according to the *WIDTH_STRETCH_FACTOR* and *HEIGHT_STRETCH_FACTOR*. The third input value *OUTLINE_WIDTH* is optional. It determines the shape's outline width. (This command doesn't work if an image is being used for the turtle shape). Here are some examples:

- One input value:
  `turtle_name.shapesize(2)`
  - The only input value (2 in this case) controls both the width and length of the turtle, i.e. both the turtle width and the length are doubled.
- Two input values:
  `turtle_name.shapesize(2, 3)`
  - The first input value (2) controls the width of the turtle, i.e. the width is doubled.
  - The second input value (3) controls the length of the turtle, i.e. the length is tripled.
- Three input values:
  `turtle_name.shapesize(2, 3, 5)`
  - The first input value (2) controls the width of the turtle.
  - The second input value (3) controls the length of the turtle.
  - The third input value (5) controls the thickness of the turtle's outline, i.e. the turtle outline becomes 5 pixels wide.

COMP1021 Large Turtle Graphics Summary

Screen Update Control

`turtle.tracer(True)`
Enables the automatic screen update. The new drawings will automatically appear on the screen.

`turtle.tracer(False)`
Disables the automatic screen update completely.

`turtle.update()` – *not used this semester*
Shows all the accumulated drawings on the screen. This can be used to trigger an update of the screen when `turtle.tracer(False)` is used.

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 is usually in the same directory as the Python program.

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

`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 and Using Turtles

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

`turtle_name`**`.distance(`** *X, Y* **`)`**
Returns the distance from the center of the turtle to the point at (*X*,*Y*).

`turtle_name`**`.distance(`**
NAME_OF_ANOTHER_TURTLE **`)`**
Returns the distance from the center of the turtle called *turtle_name* to the center of another turtle called *NAME_OF_ANOTHER_TURTLE*.

Getting Turtle Properties

Here are some examples of extracting information out of a turtle. The variable names shown 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.

`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.

COMP1021 Large Turtle Graphics 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.ontimer(
EVENT_HANDLER, DELAY **)**
This sets up a timer in the turtle system that calls the function *EVENT_HANDLER* later, after *DELAY* milliseconds e.g.
`turtle.ontimer(display, 2000)`
means run the function *display* after 2 seconds.

turtle.onscreenclick(*EVENT_HANDLER*)
After this, the function *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 another window.

turtle.done()
This must be included at the end of a turtle program for event handling e.g. dragging turtles and key presses to work properly.
Alternative name: `turtle.mainloop()`

Other Turtle Functions

***turtle_name*.width(*WIDTH*)**
Sets the line thickness to *WIDTH* e.g. `5`
Alternative command name: `pensize()`

***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 the turtle called *turtle_name* has drawn.

***turtle_name*.undo()**
Undoes the last turtle action.

turtle.bye()
Closes the turtle window.

COMP1021 Large Turtle Graphics Summary

Turtle (the module)

The turtle module contains two types of functions – one for the actual turtles, one for the turtle screen.

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`
- `distance`
- `clear`
- `undo`
- `onclick`
- `ondrag`

} *Event handling*

Turtle screen (only one)

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

These are the turtle screen functions you can use.

- `colormode`
- `textinput`
- `numinput`
- `addshape`
- `tracer`
- `update` – *not used this semester*
- `setworldcoordinates`
- `setup`
- `bgcolor`
- `bgpic`
- `bye`
- `done`
- `listen`
- `onkeypress`
- `ontimer`
- `onscreenclick`

} *Event handling*