COMP1021 Turtle Command Summary V2

Moving and Drawing

`turtle.up()`

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

`turtle.down()`

Puts the pen down, so drawing when moving.
Alternative command names: `pendown()`, `pd()`

`turtle.goto(` *X, Y* **`)`**

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

`turtle.forward(` *DISTANCE* **`)`**

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

`turtle.backward(` *DISTANCE* **`)`**

Moves the turtle backward by *DISTANCE*, opposite to the direction the turtle is headed. Does not change the direction of the turtle.
Alternative command names: `bk()`, `back()`

`turtle.left(` *ANGLE* **`)`**

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

`turtle.right(` *ANGLE* **`)`**

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

`turtle.setheading(` *ANGLE* **`)`**

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

`turtle.home()`

Turns turtle to the origin, i.e. coordinate (0, 0)

`turtle.dot(` *SIZE* **`)`**

Draws a filled circle with diameter *SIZE*. The center is at the current position of the turtle. The circle is always filled and cannot be made hollow. The dot is drawn no matter whether the pen is up or down.

`turtle.circle(` *RADIUS, EXTENT* **`)`**

Draws a circle with given *RADIUS*. The center is *RADIUS* units left of the turtle; *EXTENT* is an angle that determines how many degrees of the circle is drawn. *EXTENT* can be omitted to draw the entire circle.
If *RADIUS* has a negative value, the turtle turns clockwise instead of anticlockwise. If *EXTENT* is given with a negative value, the turtle goes backward when drawing the arc.

`turtle.stamp()`

Stamps an image of the turtle onto the screen at the current position.

`turtle.clearstamps(` *N* **`)`**

Clears the first *N* stamps, if *N* is a positive number. If *N* is a negative number, the last -*N* stamps will be cleared. If *N* is not given, all stamps will be cleared.

Handling Colour

`turtle.pencolor(` *PENCOLOR* **`)`**

Sets the pen colour to *PENCOLOR*.

`turtle.fillcolor(` *FILLCOLOR* **`)`**

Sets the fill colour to *FILLCOLOR*.

`turtle.color(` *PENCOLOR, FILLCOLOR* **`)`**

Sets the pen colour to *PENCOLOR* and sets the fill colour to *FILLCOLOR* at the same time.

`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, `turtle.pencolor(240,160,80)` works.

`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, `turtle.pencolor(1.0,0.6,0.8)` works.

COMP1021 Turtle Command Summary V2

Filling

`turtle.begin_fill()`
Begins the colour filling. It should be put before the drawing code of the shape(s) that you want to fill.

`turtle.end_fill()`
Ends the colour filling. It should be put after the drawing code of the shape(s) that you want to fill.

Text Input and Output

`turtle.write("`*TEXT*`", font=("`*FONTYPE*`",`
FONTSIZE`, "`*FONTSTYLE*`"))`
Writes *TEXT* with the specified font, for example, `font=("Arial", 8, "bold")`

`turtle.textinput("`*TITLE*`","`*PROMPT*`")`
Shows a window dialog to ask for a user's string input. *TITLE* specifies the dialog title, and *PROMPT* specifies the dialog message.

`turtle.numinput("`*TITLE*`", "`*PROMPT*`",`
DEFAULT`,` *MIN*`,` *MAX*`)`
Shows a window dialog to ask for a user's numerical input. *TITLE* specifies the dialog title, and *PROMPT* specifies the dialog message. 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 input respectively.

Visibility Control

`turtle.hideturtle()`
Hides the turtle from the screen.
Alternative command name: `ht()`

`turtle.showturtle()`
Shows the turtle on the screen.
Alternative command name: `st()`

Shape Control

`turtle.shape("`*SHAPE*`")`
Sets the turtle's shape to *SHAPE*. Initially, there are the following polygon shapes: "arrow", "turtle", "circle", "square", "triangle", "classic".

`turtle.addshape("`*FILENAME*`")`
Adds a new turtle shape for the `turtle.shape()` command. *FILENAME* is the file name of a GIF image file in the same directory of the python program.
Alternative command name: `register_shape()`

`turtle.shapesize(`
WIDTH_STRETCH_FACTOR`,`
HEIGHT_STRETCH_FACTOR`,`
OUTLINE_WIDTH **`)`**
Sets the turtle's shape size. Stretches the 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. Here are some examples with different numbers of input values:

- One input value: `turtle.shapesize(2)`
  - The only input value (2 in this case) controls both the width and length of the turtle, i.e. both the width and length are doubled.
- Two input values: `turtle.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.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 Turtle Command Summary V2

Screen Update Control

`turtle.tracer(True)`
Enables the automatic screen update. The new drawings will automatically appear on the screen without calling `turtle.update()`.

`turtle.tracer(False)`
Disables the automatic screen update completely, `turtle.update()` is then needed to put the new drawings on the screen at once. No new drawings will appear until `turtle.update()` is called.

`turtle.update()`
Shows all the accumulated drawings on the screen. It is used to manually update the screen when `turtle.tracer(False)` is used.

Window Setup

`turtle.mode("world")`
Enables the use of a customized coordinate system.

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

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

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

`turtle.bgpic("`*FILENAME***`")`**
Sets the window's background picture. *FILENAME* is the file name of a GIF image file in the same directory of the python program.

Getting the Properties

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

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

`p = turtle.position()`
Returns the (x position, y position) of the turtle.
Alternative command name: *`pos()`*

`h = turtle.heading()`
Returns the direction of the turtle in degrees.

The following turtle commands also appear in some other sections of this document. As well as changing properties, the commands can also be used to retrieve the value of the properties when no input values are provided.

`w = turtle.width()`
Alternative command name: *`pensize()`*

`pc = turtle.pencolor()`

`fc = turtle.fillcolor()`

`pc, fc = turtle.color()`

`bg = turtle.bgpic()`

`sp = turtle.speed()`

`sh = turtle.shape()`

`w, h, ow = turtle.shapesize()`

COMP1021 Turtle Command Summary V2

Event Handling

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

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

`turtle.ontimer(` *EVENT_HANDLER*`,` *DELAY* **`)`**
Sets up a timer that calls *EVENT_HANDLER* after *DELAY* milliseconds.

`turtle.onscreenclick(` *EVENT_HANDLER* **`)`**
After this, the function *EVENT_HANDLER* will be executed when the turtle window screen is clicked.

`turtle.onkeypress(` *EVENT_HANDLER*`,` **`"`***KEY*`"` **`)`**
After this, the function *EVENT_HANDLER* will be executed when *KEY* is pressed.
If *KEY* is omitted, key press events for all keys will be handled by *EVENT_HANDLER*.

`turtle.listen()`
Tells Python to start handling key events.

Other Turtle Functions

`turtle.width(` *WIDTH* **`)`**
Sets the line thickness to *WIDTH*.
Alternative command name: `pensize()`

`turtle.speed(` *SPEED* **`)`**
Sets the turtle's animation speed to *SPEED*.
1 is the slowest, 10 is the fastest. 0 means no animation, i.e. instant drawing.

`t = turtle.Turtle()`
Creates a new turtle.

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

`turtle.distance(` *ANOTHER_TURTLE* **`)`**
Returns the distance from the center of the turtle to the center of *ANOTHER_TURTLE*.

`turtle.clear()`
Deletes the turtle's drawings from the screen.

`turtle.reset()`
Deletes the turtle's *drawings* from the screen, re-centers the turtle, and sets all settings to their initial values. Shows a blank turtle window.

`turtle.undo()`
Undoes the last turtle action.

`turtle.bye()`
Closes the turtle window.

`turtle.done()`
Finishes the turtle graphics drawing procedure. It must be written at the end of a turtle program. Otherwise the turtle graphics window will not behave properly, and the event handling will not work properly.
Alternative command name: `mainloop()`

COMP1021 Turtle Command Summary V2

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`
- `stamp`
- `setheading`
- `pencolor`
- `fillcolor`
- `color`
- `begin_fill`
- `end_fill`
- `write`
- `shape`
- `shapesize`
- `hideturtle`
- `showturtle`
- `xcor`
- `ycor`
- `pos`
- `heading`
- `width`
- `speed`
- `distance`
- `clear`
- `reset`
- `undo`
- `onclick`
- `ondrag`

} *Event handling*

Turtle screen (only one)

The following functions can be used as turtle screen functions only.

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

} *Event handling*