Turtle Oxford

Ioana Vasile

Jan 24, 2024

CONTENTS:

CHAPTER

ONE

USAGE

1.1 Installing Python

A good place to start is to consult the official Python starter documentation here: https://wiki.python.org/moin/BeginnersGuide.

First, you need to have both Python and pip installed. Some operating systems come with the former already pre-installed. To check if it is indeed installed, you can run the following command in a shell (this is a terminal window on a Unix machine or a Command Prompt or PowerShell window on a Windows machine):

```
python3 --version
```

If the version is too old (starts with 2.X), then you need to follow the installation steps anyway.

To install Python, please go to https://www.python.org/downloads/ where you can download a Python installer. If you're having any issues or questions related to the installer, you should consult this installation guide which has information for most operating systems: https://docs.python.org/3/using/index.html.

1.2 Installing Pip

The official guide to installing Pip can be found here: https://pip.pypa.io/en/stable/installation/.

Pip is a python package manager which you can use to install python libraries or applications.

First, you may already have pip installed, especially if you've just downloaded and installed Python. You can check by running the following command in a shell:

```
pip3 --version
```

If you do not have it installed, you can install it by running the following commands

For Max or Linux:

```
python3 -m ensurepip --upgrade
```

For Windows:

```
py -m ensurepip --upgrade
```

1.3 Installing an IDE

You can, of course, write your code in any text editor you want, then save it with a *.py* extension and run it from a shell like so

For Mac or Linux:

```
python3 example.py
```

For Windows:

```
py example.py
```

But certain text editors provide a bit more functionality for writing code, such as syntax highlighting, auto-formatting and a shell within the same window. A very popular such editor (also called an IDE) is Visual Studio Code. Although this is a Microsoft product, there are free versions of it available for most operating systems, so you do not need to own a Windows machine to be able to install and use it. You can download an installer from https://code.visualstudio.com/.

1.4 Installing Turtle Oxford for Python

The code for Turtle Oxford is hosted entirely on GitHub at the following address: https://github.com/turtleoxford/turtle-python.

You can install the package in a shell, using pip with the following command:

```
pip3 install git+https://github.com/turtleoxford/turtle-python.git
```

1.5 Executing one of the examples

You can execute one of the Oxford Turtle Python examples from https://github.com/turtleoxford/turtle-python/tree/main/examples by downloading them and then running the following command in a shell (*draw_pause.py* is just one of the file names, it could be any of them):

On Mac or Linux:

```
python3 draw_pause.py
```

Or on Windows:

```
py draw_pause.py
```

1.6 Writing your own programme

The structure of one of Oxford Turtle Python programmes is as follows:

```
from turtle_oxford import *  # This tells the interpreter to import all symbols from the␣
↪Turtle module
from math import *           # Optional: import one or more other packages and their␣
↪symbols, math is a useful one\
```

(continues on next page)

(continued from previous page)

```
with turtle_canvas(0, 0, 500, 500) as t: # Create the canvas with the desired dimensions,
↪ you can omit the parameters to use the default size
    <Turtle Commands>
```

You can execute this like any other python file, like in the previous section.

CHAPTER

TWO

API

turtle_oxford	Turtle Oxford - a python library for the Oxford Turtle System

2.1 turtle_oxford

Turtle Oxford - a python library for the Oxford Turtle System

Functions

angles(degrees)	Change the number of degrees in a circle.
antilog(a, b, mult)	
back(distance)	Move back.
blank(*args, **kwargs)	
blot(*args, **kwargs)	
box(*args, **kwargs)	
circle(*args, **kwargs)	
colour(new_colour)	Set the new colour of the turtle.
colour_to_int(colour)	Convert the colour parameter from any acceptable format to an integer (from 0 to 255).
colour_to_str(colour)	Convert the colour parameter form any acceptable format to a string.
delete(s, idx, l)	
detect(key_sym, timeout)	
direction(degrees)	Turtle changes direction to face this number of degrees.
display(*args, **kwargs)	
divmult(a, b, c)	

continues on next page

Table 1 – continued from previous page

draw(func)	Private.
drawxy(*args, **kwargs)	
ellblot(*args, **kwargs)	
ellipse(*args, **kwargs)	
fill(*args, **kwargs)	
forget(n)	Forget the last n positions of the turtle
forward(distance)	Move forward.
get_key_code()	
get_key_sym()	
halt([e])	
home()	Move the turtle to the center of the canvas
intdef(s, default)	
left(degrees)	Turn left.
maxint()	
mixcols(col1, col2, prop1, prop2)	
move(func)	Private.
movexy(*args, **kwargs)	
new_turtle(arr)	
noupdate()	Refrain from updating the Canvas when executing all subsequent drawing commands, until update() is called.
old_turtle()	
on_press(event)	
on_release(event)	
pad(s, padding, length)	
pause(duration)	Pause *duration* milliseconds.
pendown()	Put down the pen, all movement functions now produce drawings.
penup()	Pick up the pen, stop drawing.
pixcol(x, y)	
pixset(*args, **kwargs)	
polygon(*args, **kwargs)	
polyline(*args, **kwargs)	

continues on next page

Table 1 – continued from previous page

qint(s, mult, default)	
qstr(a, b, decplaces)	
randcol(n)	
remember()	Add the current coordinates to the history of the turtle
reset(key_sym)	
resolution(x, y)	Set the resolution of the canvas to x by y
rgb(n)	
right(degrees)	Turn right.
setx(*args, **kwargs)	
setxy(*args, **kwargs)	
sety(*args, **kwargs)	
status(key_sym)	
thickness(new_thickness)	Set the thickness of the pen
turnxy(x, y)	Turn to face the point (x, y) on the canvas.
turtle_canvas([origin_x, origin_y, width, ...])	Context manager that creates a canvas at the start and halts at the end.
update()	Update the Canvas, and continue updating with all subsequent drawing commands.

Classes

TurtleCanvas()	Class with mostly static member describing the turtle and the canvas.

PYTHON MODULE INDEX

t

INDEX