Caerbannog Documentation

Release 0.1.1

AlexV

Feb 10, 2019

Contents:

Follow the white rabbit... at your own peril.

CHAPTER 1

Object Oriented Turtle

Data and behavior are combined into one object.

Client calls a turtle class instance. Turtle class needs to keep track of the Turtle 'State'. The turtle state is defined here for simplicity as (position, angle, pen's state - up or down). The turtle class holds the turtle state and mutates it.

The turtle class has a few methods : - move(distance) - left(angle) - right(angle) - penup() - pendown()

Note : we use pint for unit of measure (radians / degrees)

```
import turtle


import enum
import pint
ureg = pint.UnitRegistry()


# Side Note: Yes, Python has enums !
class PenState(enum.Enum):
    UP = -1
    DOWN = 1


# The usual OO inheritance interface
# taking turtle.Turtle as an unknown black box.
# We provide shortcuts to some of its methods and attributes here for interfacing␣
↪with turtle,
# while trying to keep the turtle API small.
class Tootle(turtle.Turtle):
    """
    Inheriting from provided Turtle class, OO style.
    """
    @property
    def position(self):
        return super().position()  # TODO : mutable
```

(continues on next page)

(continued from previous page)

```
    @property
    def angle(self):
        return super().heading()  # TODO : mutable

    @property
    def penState(self):
        return super().pen().get('pendown')  # TODO mutable

    def move(self, distance: int):

        # TMP HACK
        distance = int(distance)

        super().forward(distance=distance)

    def right(self, angle: int):

        # TMP HACK
        angle = int(angle * ureg.degrees)

        super().right(angle)

    def left(self, angle: int):

        # TMP HACK
        angle = int(angle)

        super().left(angle)

    def penup(self):
        super().penup()

    def pendown(self):
        super().pendown()
```

Pros:

- Familiar

Cons :

- Stateful/Blackbox/hard to test
- CAnt compose
- Hard coded dependencies

1.1 API

Tootle API

CHAPTER 2

Various API documentation

2.1 Tootle API

class tootle.**PenState**
An enumeration.

class tootle.**Tootle**(*shape='classic'*, *undobuffersize=1000*, *visible=True*)
Inheriting from provided Turtle class, OO style.

left(*angle: int*)
Turn turtle left by angle units.

Aliases: left | lt

Argument: angle – a number (integer or float)

Turn turtle left by angle units. (Units are by default degrees, but can be set via the degrees() and radians() functions.) Angle orientation depends on mode. (See this.)

Example (for a Turtle instance named turtle): >>> turtle.heading() 22.0 >>> turtle.left(45) >>> turtle.heading() 67.0

pendown()
Pull the pen down – drawing when moving.

Aliases: pendown | pd | down

No argument.

Example (for a Turtle instance named turtle): >>> turtle.pendown()

penup()
Pull the pen up – no drawing when moving.

Aliases: penup | pu | up

No argument

Example (for a Turtle instance named turtle): >>> turtle.penup()

position
Return the turtle's current location (x,y), as a Vec2D-vector.

Aliases: pos | position

No arguments.

Example (for a Turtle instance named turtle): >>> turtle.pos() (0.00, 240.00)

right (*angle: int*)
Turn turtle right by angle units.

Aliases: right | rt

Argument: angle – a number (integer or float)

Turn turtle right by angle units. (Units are by default degrees, but can be set via the degrees() and radians() functions.) Angle orientation depends on mode. (See this.)

Example (for a Turtle instance named turtle): >>> turtle.heading() 22.0 >>> turtle.right(45) >>> turtle.heading() 337.0

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index