

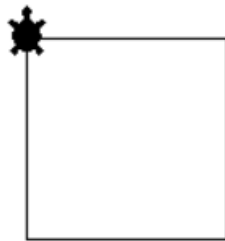
タートルグラフィックス

<https://docs.python.org/ja/3/library/turtle.html>

正方形の描画

turtle_1.py

```
1  from turtle import *
2
3  shape('turtle')
4
5  forward(100)
6  right(90)
7  forward(100)
8  right(90)
9  forward(100)
10 right(90)
11 forward(100)
12
13 mainloop()
14
```



turtle : タートルグラフィックス

Python » Japanese » 3.12.2 » 3.12.2 Documentation » Python 標準ライブラリ » プログラムのフレームワーク » turtle --- タートルグラフィックス

目次

turtle --- タートルグラフィックス

- イントロダクション
- はじめに
- チュートリアル
 - Starting a turtle environment
 - Basic drawing
 - Pen の制御
 - The turtle's position
 - Making algorithmic patterns
- How to...
 - Get started as quickly as possible
 - Use the turtle module namespace
 - Use turtle graphics in a script
 - Use object-oriented turtle graphics
- Turtle graphics reference
 - Turtle のメソッド
 - TurtleScreen/Screen のメソッド
- RawTurtle/Turtle のメソッドと対応する関数
 - Turtle の動き
 - Turtle の状態を知る
 - 設定と計画
 - Pen の制御
 - 描画制御
 - 色の制御
 - 塗りつぶし
 - さらなる描画の制御
- タートルの状態
 - 可視性
 - 見た目

turtle --- タートルグラフィックス

ソースコード: [Lib/turtle.py](#)

はじめに

Turtle graphics is an implementation of [the popular geometric drawing tools introduced in Logo](#), developed by Wally Feurzeig, Seymour Papert and Cynthia Solomon in 1967.

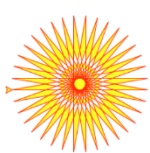
In Python, turtle graphics provides a representation of a physical "turtle" (a little robot with a pen) that draws on a sheet of paper on the floor.

It's an effective and well-proven way for learners to encounter programming concepts and interaction with software, as it provides instant, visible feedback. It also provides convenient access to graphical output in general.

Turtle drawing was originally created as an educational tool, to be used by teachers in the classroom. For the programmer who needs to produce some graphical output it can be a way to do that without the overhead of introducing more complex or external libraries into their work.

Turtle star

Turtle can draw intricate shapes using programs that repeat simple moves.



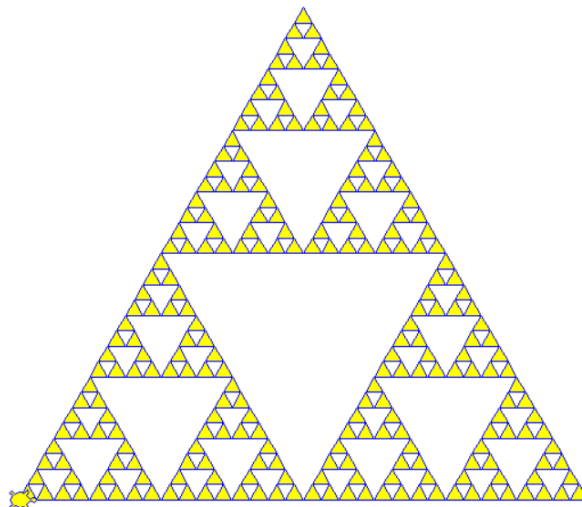
チュートリアル

New users should start here. In this tutorial we'll explore some of the basics of turtle drawing.

Starting a turtle environment

In a Python shell, import all the objects of the `turtle` module:

```
from turtle import *
```



<https://docs.python.org/ja/3/library/turtle.html>

タートルグラフィックスの主な関数

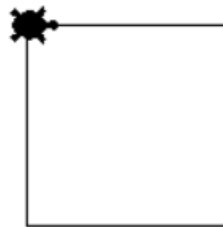
関数	短縮形（別表記）	説明
forward(units)	fd(units)	亀を units（ピクセル）前進
backward(units)	bk(units)	亀を units（ピクセル）後進
right(degrees)	rt(degrees)	亀を degrees 度 右に向ける（回転）
left(degrees)	lt(degrees)	亀を degrees 度 左に向ける（回転）
speed(s)		亀のスピード s, 0が最速
penup()	up()	ペンを上げる（以後、描画しない）
pendown()	pd()	ペンを下げる（以後、描画をはじめる）
setpos(x,y)	goto(x,y)	座標 (x, y) へ亀を移動
bgcolor()		背景の色を指定 (https://www.w3schools.com/colors/colors_names.asp)
pencolor()	color()	ペンの色を指定 (https://www.w3schools.com/colors/colors_names.asp)
pensize(w)	width(w)	ペンの大きさ w を指定
position()	pos()	亀の現在の座標 (x,y) を取得
heading()	getheading()	亀が向いている方向（角度）を取得

正方形の描画(2)

turtle_2.py

```
1  from turtle import *
2
3  shape('turtle')
4
5  for i in range(4):
6      forward(100)
7      right(90)
8
9  mainloop()
10
```

繰り返しは
forループで



正N角形を描く関数

turtle_3.py

最初にこのプログラムに関する
説明コメントを記述

関数のコメントはdocstring で書く.

""" 関数の説明 (1行で)

(空行を開ける)

場合によっては詳細な関数の説明

Args: (関数の引数の説明)

n: n角形

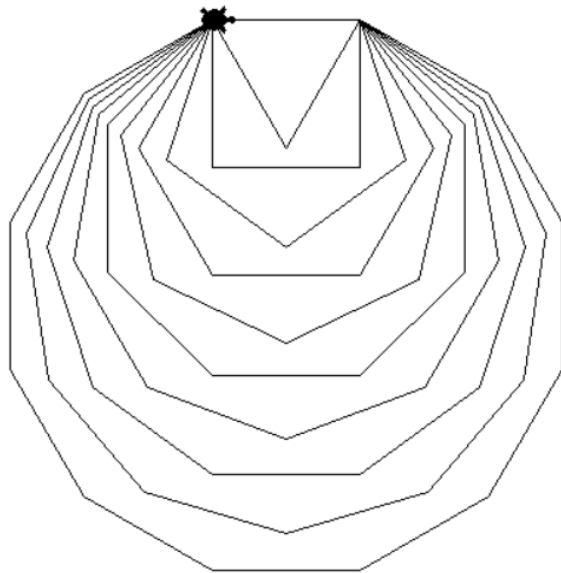
.....

Returns: (関数の戻り値の説明)

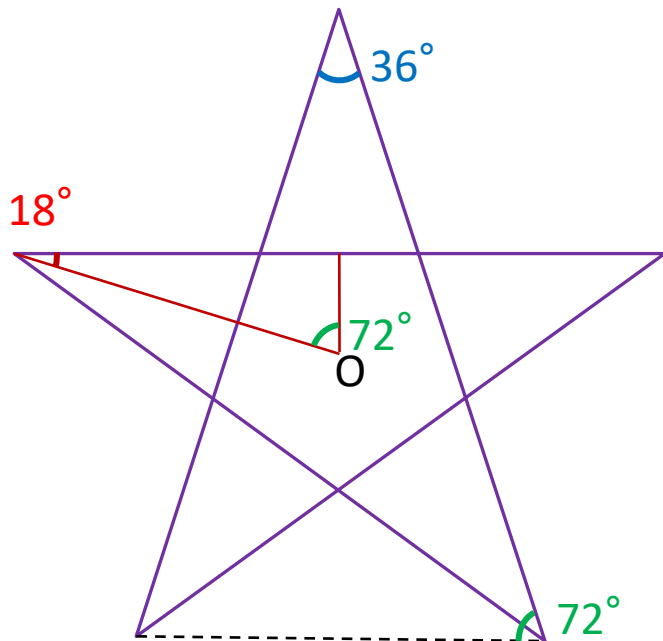
.....

"""

```
1  """
2  Turtleグラフィックスのプログラム□その3
3  正n角形を描く関数
4  """
5  from turtle import * #! この書き方は推奨されて
6
7  shape('turtle')
8
9  def n_gon(n, length):
10     """ 正n角形を描く関数
11
12     Args:
13         n: n角形
14         length: 一辺の長さ
15     """
16     angle = #? ★どう書けば良い?
17     for i in range(n):
18         forward(length)
19         right(angle)
20
21
22     # 三角形から12角形を描く
23     for n in range(3,13):
24         n_gon(n, 100)
25
26     mainloop()
```



例) ★を描く関数



```
def draw_star(length, is_fill=False):
```

```
    """★を描く
```

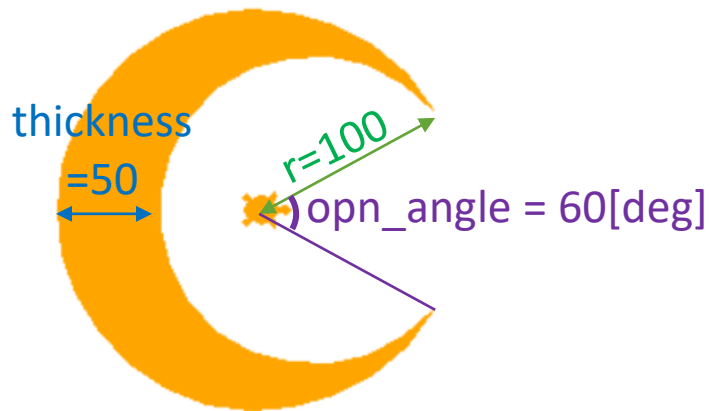
```
    Args:
```

```
        length: 1辺の長さ
```

```
        is_fill: 塗りつぶすかどうか？
```

```
    """
```

例) 月を描く関数



```
def draw_moon(r, opn_angle, thickness):
```

```
    """月を描く
```

```
    Args:
```

```
        r: 外円の半径
```

```
        opn_angle: 開口角
```

```
        thickness: 月の厚み
```

```
    """
```

```
    ...
```

```
moon(100, 60, 50)
```


from turtle import *

ワイルドカード*を使うとモジュールで公開されているすべての関数や変数などがインポートされて使えるようになる。

⇒どの名前がどの名前空間に存在しているか不明瞭であるため

Pythonコーディングガイドライン(PEP8)では**推奨されていない**。

対策1. モジュール名を省略して使う
(すべて頭に「t.」をつける) turtle_4.py

```
import turtle as t
```

```
t.shape('turtle')
```

```
t.fd(100)
```

```
...
```

```
t.mainloop()
```

対策2. Turtleクラスをインポートして使う

```
from turtle import Turtle
```

```
t = Turtle() #Turtleクラスを利用
```

```
t.shape('turtle')
```

```
t.fd(100)
```

```
...
```

```
t.screen.mainloop()
```

```
# mainloopはTurtle.screenにある
```