プログラミング実習I クラス5 (井村担当)

知能・機械工学課程 井村 誠孝

m.imura@kwansei.ac.jp

タートルグラフィックス

- 1967年にMIT(Massachusetts Institute of Technology)で開発された教育向けプログラミング言語Logoで導入された，図形描画システム．
  - 理解が容易な命令群
  - 制御構文(if, for, while)との組み合わせにより，複雑な図形の描画が可能
- Python公式ドキュメント
  - https://docs.python.org/ja/3/library/turtle.html

実世界の亀ロボットを動かすことに
使われたので「タートルグラフィクス」

LOGO turtle

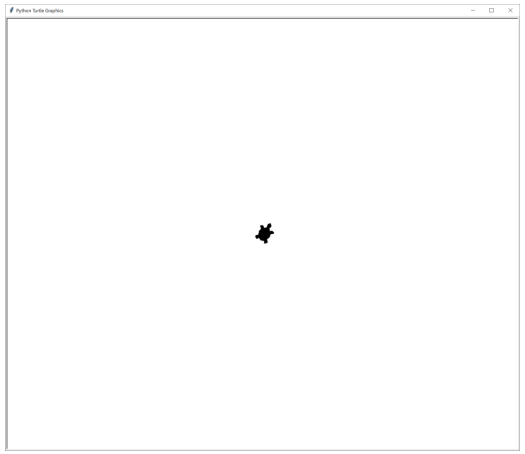
The first turtle robot used with the LOGO programming language was built at the MIT AI Lab in 1970.

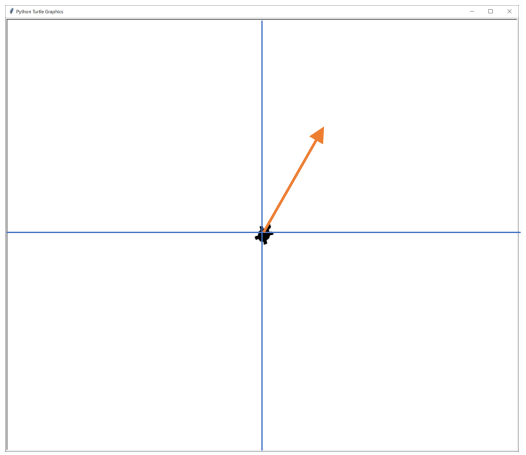
A professor at the Media Lab comments, "The computer language Logo was developed at MIT in the 70s-early 90s. It changed the way people thought about computer, children, and as its chief developer Seymour Papert would put it, "powerful ideas." The Logo floor turtle, an early Logo computer, should be part of the exhibition."

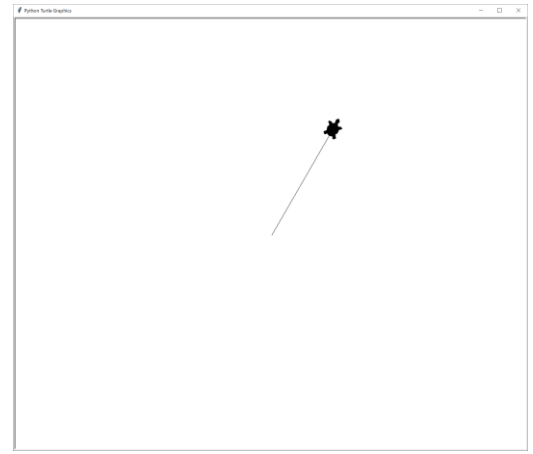
http://museum.mit.edu/nom150/entries/1158

タートルで図形を描画するとは?

- 2次元平面(キャンバス)上にタートル(亀)がいる．
- タートルの状態は位置と向きで表される．
  - 2次元なので，位置は2自由度，向きは1自由度．
- タートルを命令(メソッド)で移動させると，移動軌跡に線を描くことができる．







「前へ300進め」

タートルの初期化

- turtleモジュールをimportする．
- turtle.Turtle型のインスタンスを作成する．
  - 初期位置は(0, 0)，向きは右向き．
- 以降はこのインスタンスのメソッドを用いて操作を行う．

インスタンスとは，
あるクラス(設計図)の実体のこと．
ここでは変数(の中身)と思ってください．

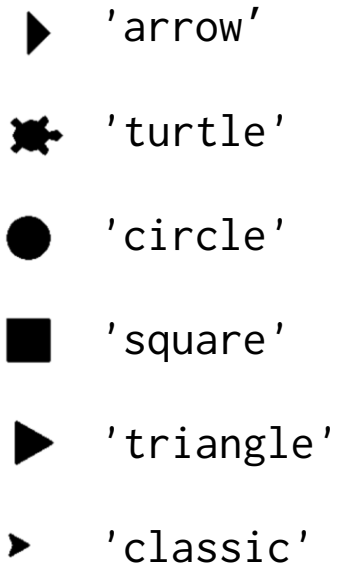
```
import turtle

kame = turtle.Turtle()
kame.shape('turtle')
kame.shapesize(2, 2, 3)
```

タートルの幅の倍率

タートルの身長(?)の倍率

タートルの輪郭の太さ

6種類用意されている

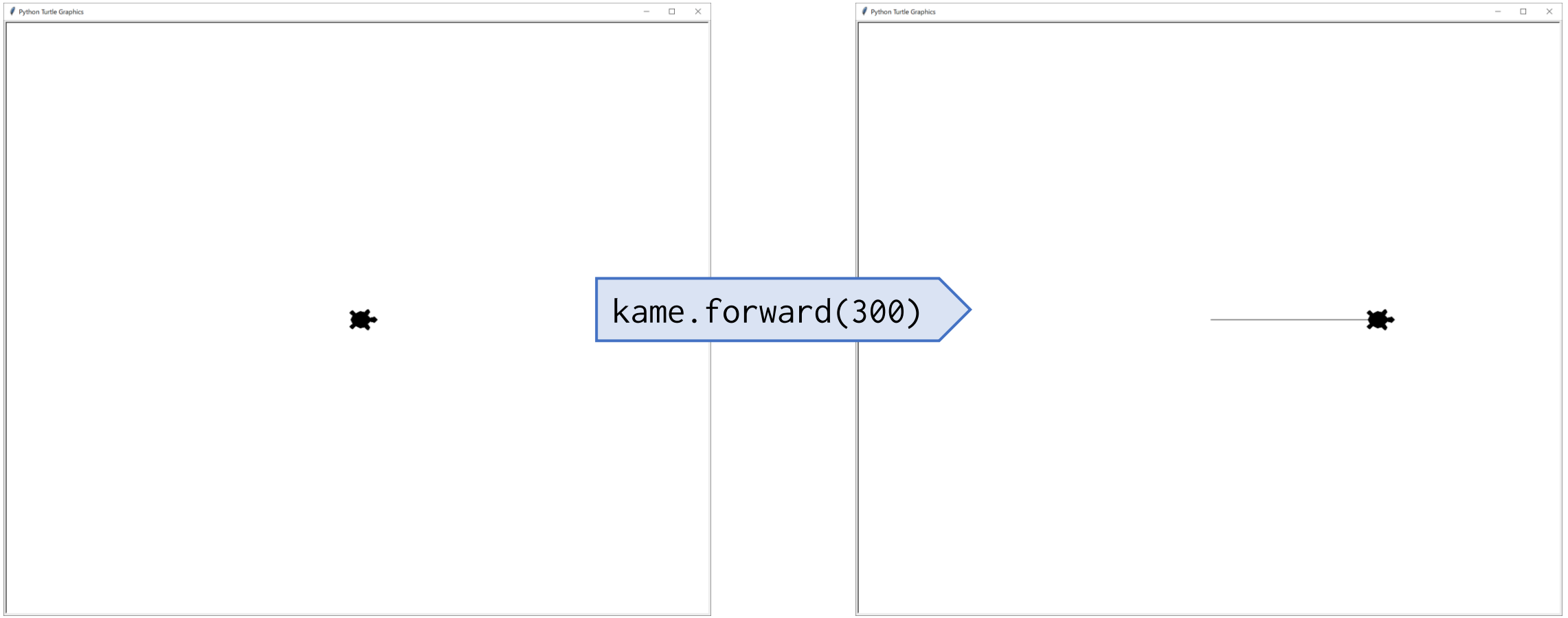


メソッドは
ある型と関連付いた関数

自分の好みに変えて
遊んでみてください．

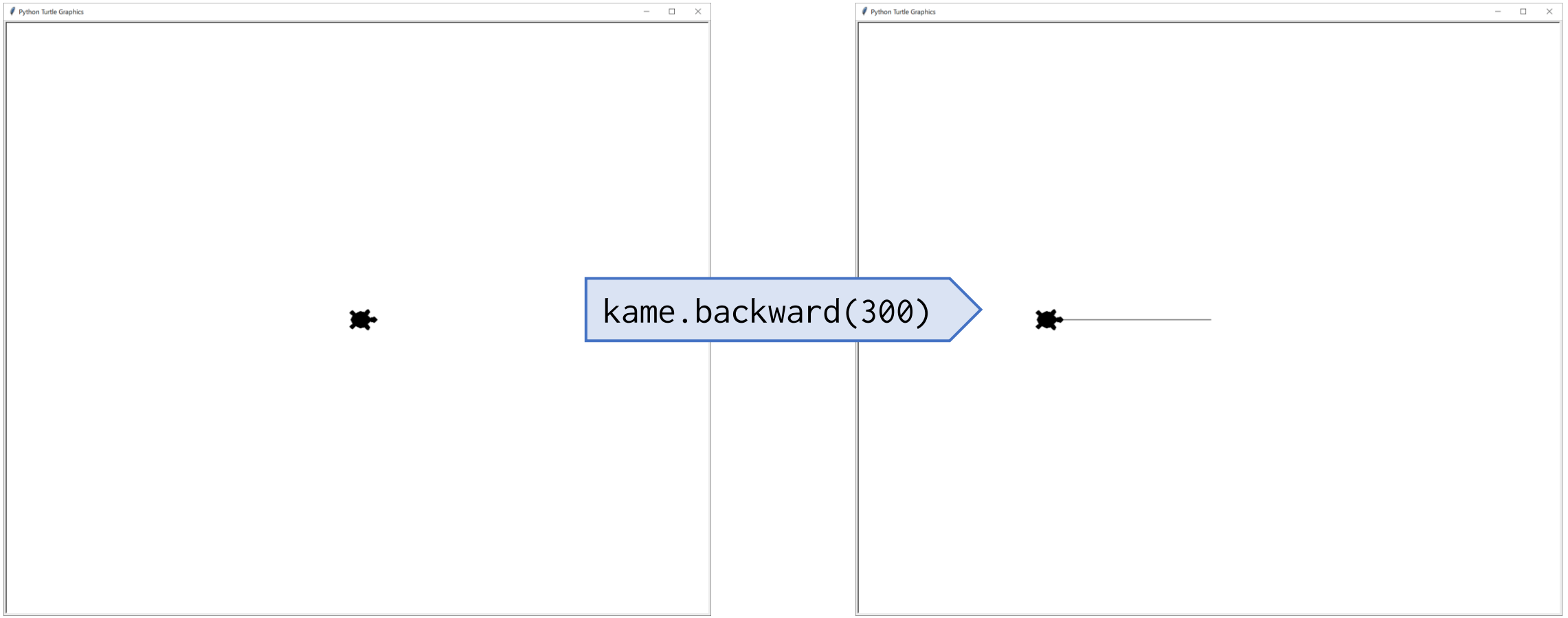
移動

- メソッド forward(): 向いている方向に進む



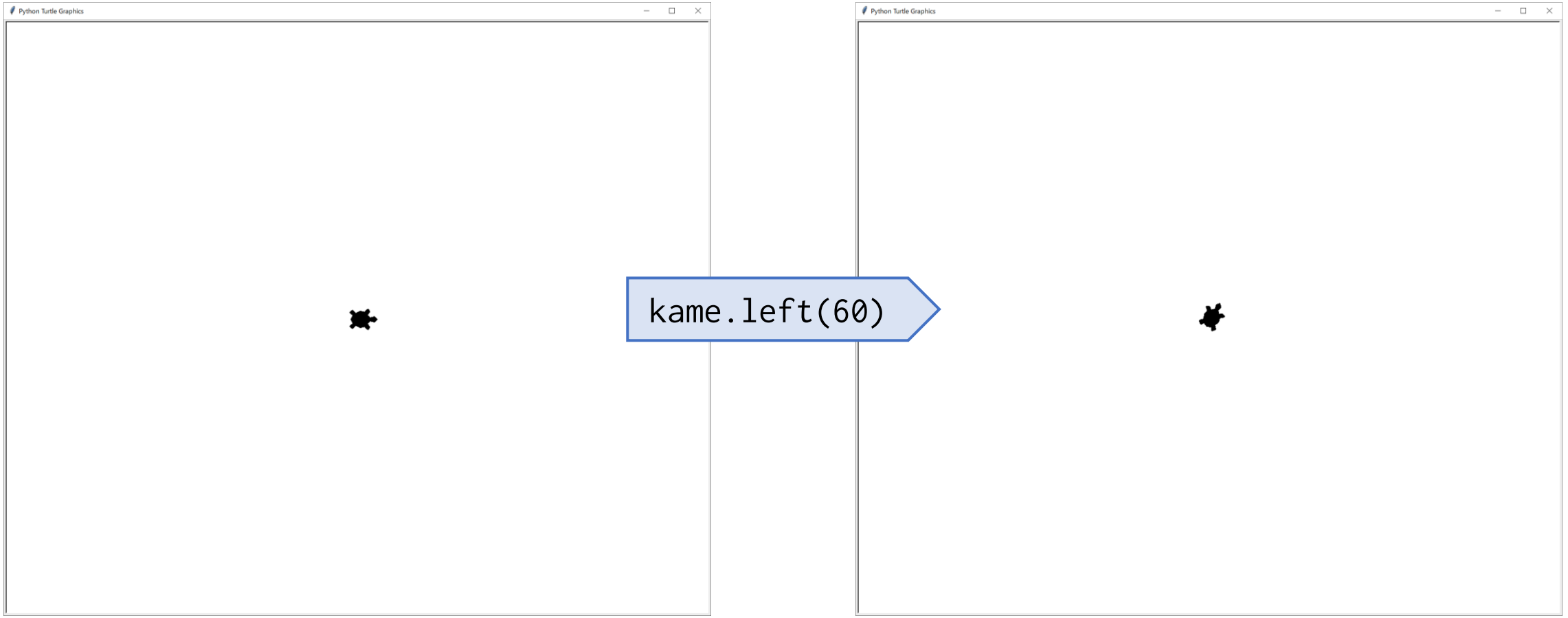
移動

- メソッド backward(): 向いている方向と逆に進む



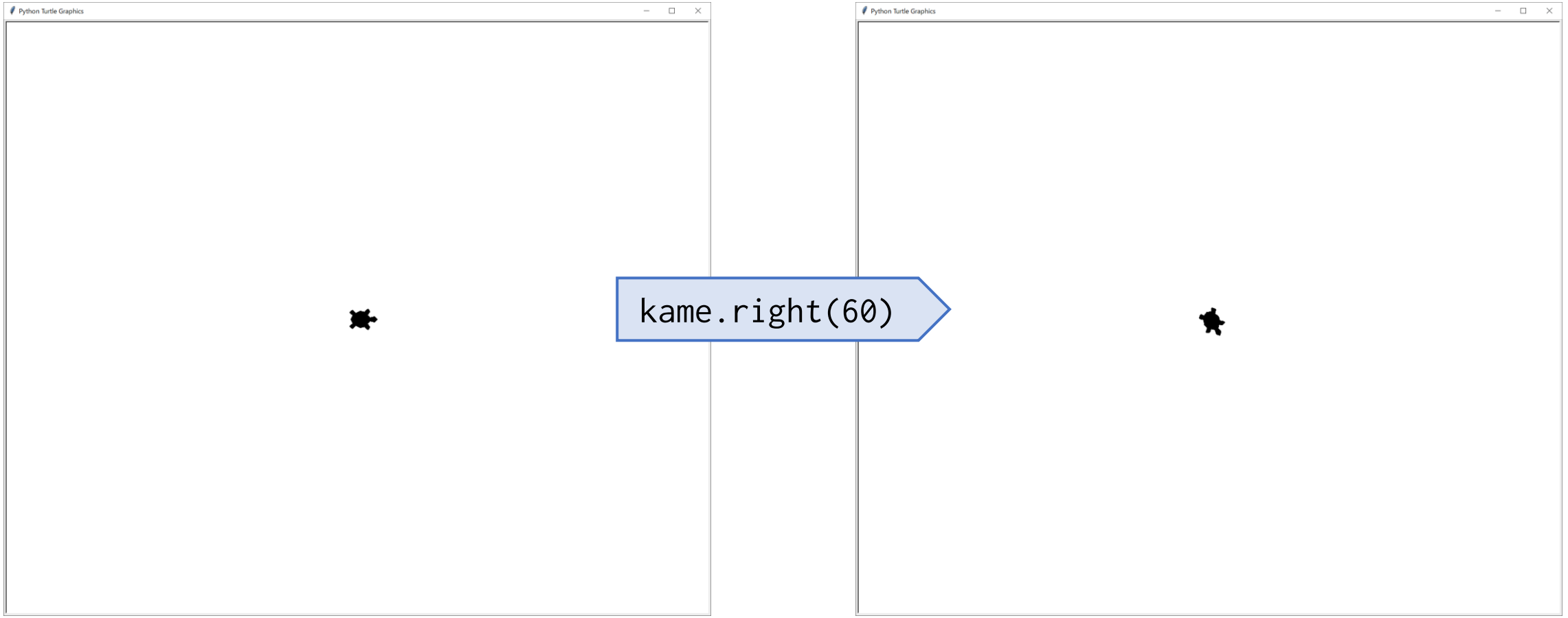
回転

- メソッド left(): 左へ回転 (角度の単位はデフォルトでは度(degree))



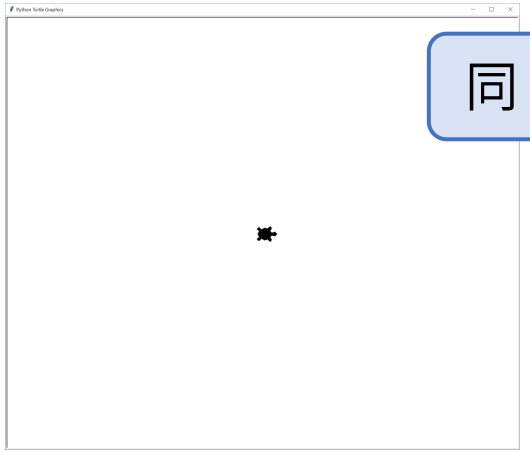
回転

- メソッド right(): 右へ回転

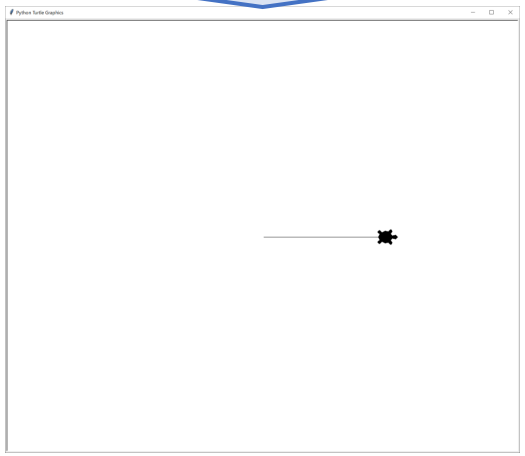


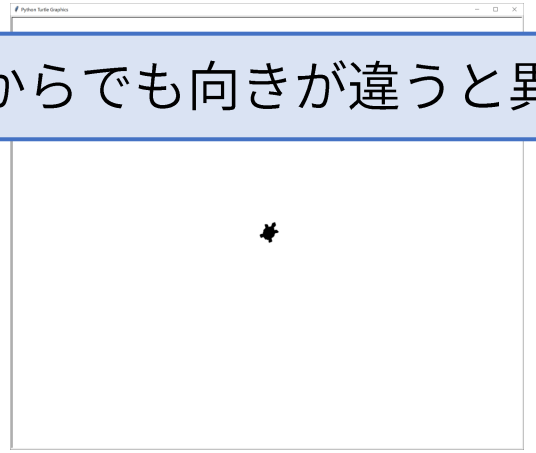
ポイント: タートルに「乗って」操作する

- 現在の位置・向きを基準とした相対的な運動を指示する



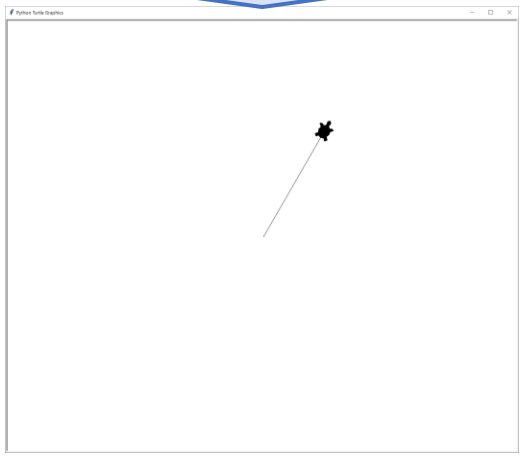
kame.forward(300)

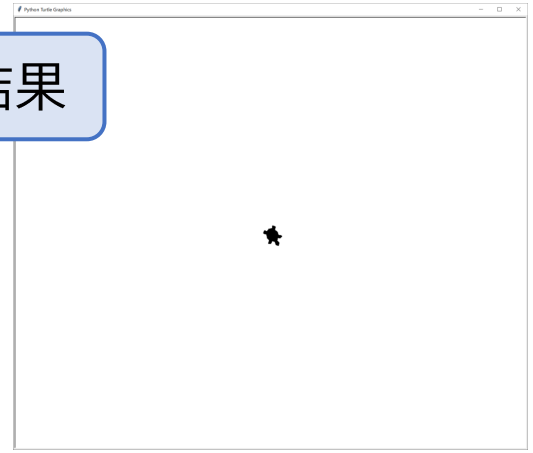




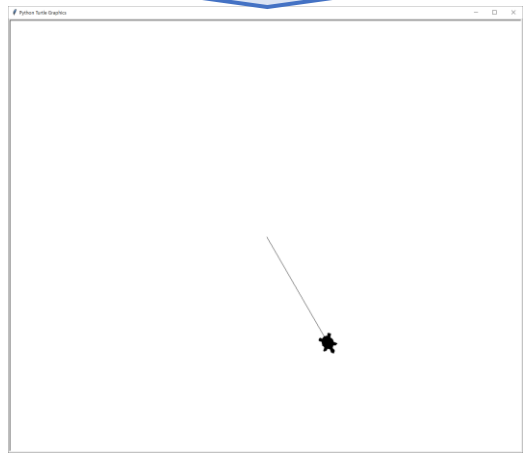
同じ位置からでも向きが違うと異なる結果

kame.forward(300)





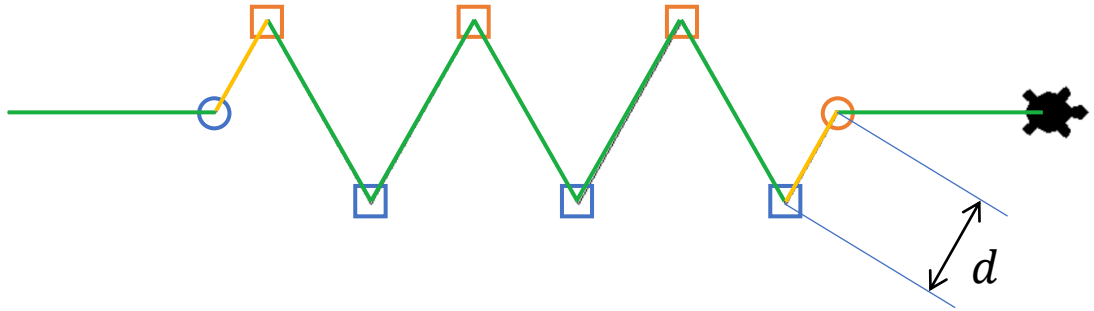
kame.forward(300)



考えてみよう

- 例題: どのような命令を実行すればよいか?

dは適当な長さ



kame.forward(d)

kame.forward(d * 2)

kame.left(60)

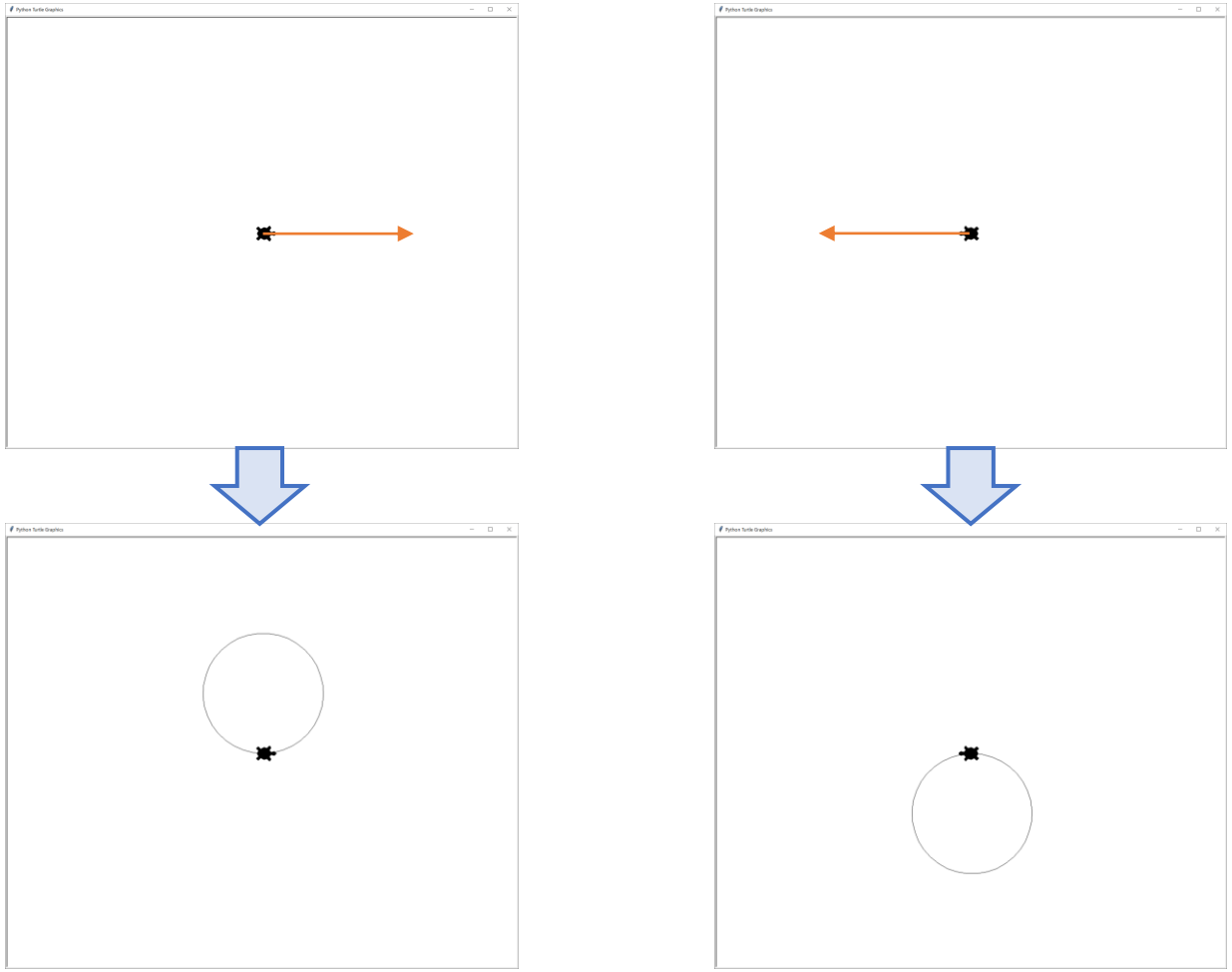
kame.left(120)

kame.right(60)

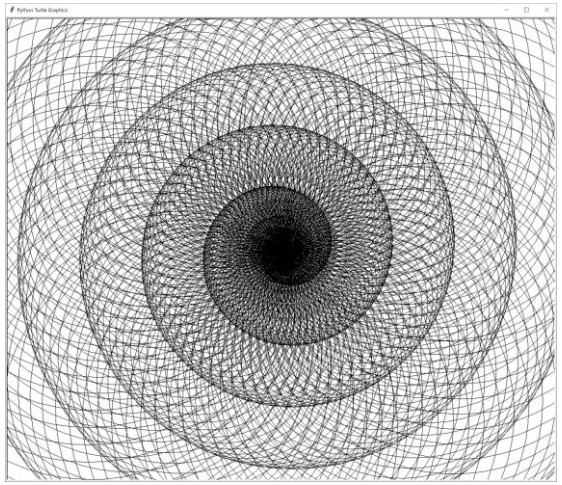
kame.right(120)

特殊な描画: 円を描く

- メソッド circle(): 進行方向左側に円を描く



繰り返しを使うと
短いコードで
複雑な図形を描ける例



うっひょ〜

制御

- メソッド undo(): 直前の操作を取り消す.
- メソッド home(): タートルの位置と向きを初期状態に戻す.
- メソッド clear(): 描かれた図形を全て消去する.

キャンバスと座標系

- キャンバスのサイズを知る
  - メソッド getscreen(): タートルが描画中の TurtleScreen オブジェクトを返す.
  - TurtleScreenのメソッド window_width(): キャンバスの横のサイズ
  - TurtleScreenのメソッド window_height(): キャンバスの縦のサイズ

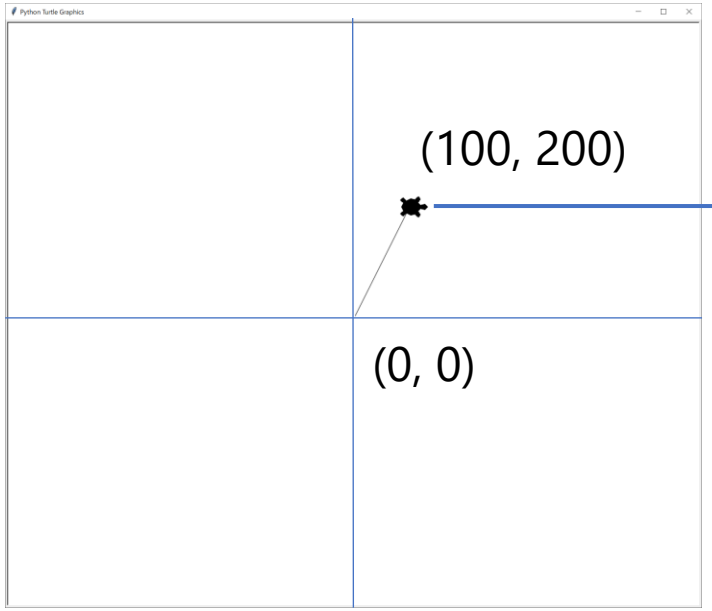
変数 w, h にそれぞれ横と縦のサイズが得られる

```
w = kame.getscreen().window_width()
h = kame.getscreen().window_height()
```

ここまででTurtleScreen型

キャンバスの座標系での操作

- メソッド position(): タートルの現在の位置を知る
- メソッド goto(): タートルを指定座標へ移動させる
  - タートルの向きは変化しない
- メソッド distance(): タートルの位置と指定された点との距離を調べる




```
kame.goto(100, 200)
```

タートルの向きが初期状態のままであり
移動方向とは異なる点に注意

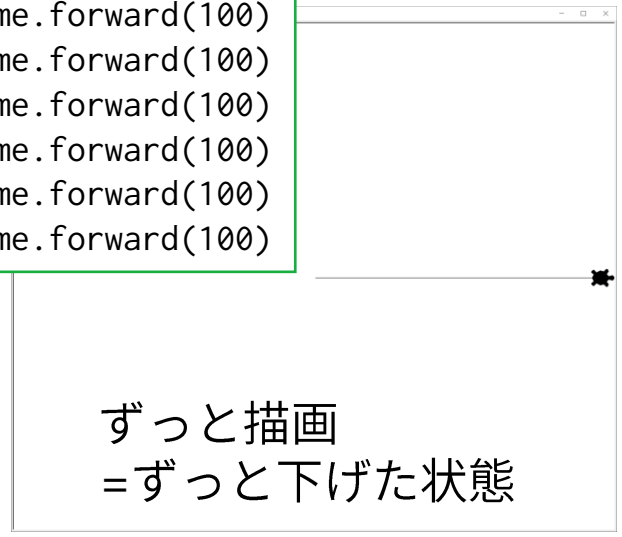
参考: タートルの向き

- メソッド heading(): タートルの現在の向きを知る
- メソッド setheading(): タートルを指定の向きにする
- メソッド towards(): タートルの位置から指定された点への向きを返す

描画の制御

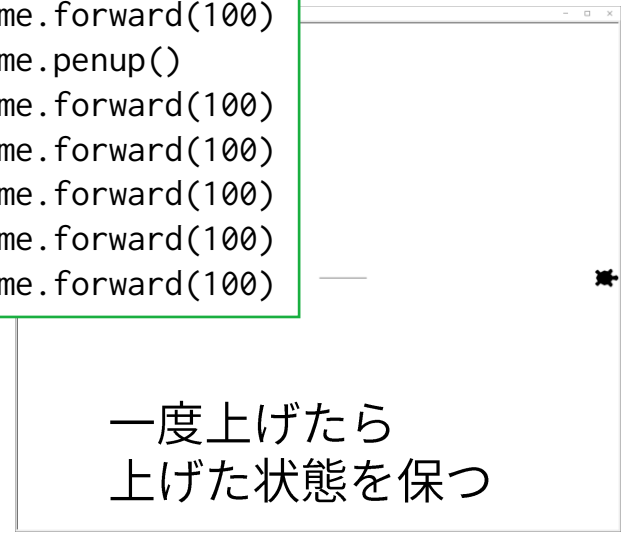
- メソッド penup(): ペンを上げる
  - その後はずっと移動しても軌跡が描かれない.
- メソッド pendown(): ペンを下げる
  - 移動すると軌跡が描かれる状態になる.
- メソッド isdown(): ペンの状態を知る

```
kame.forward(100)
kame.forward(100)
kame.forward(100)
kame.forward(100)
kame.forward(100)
kame.forward(100)
```



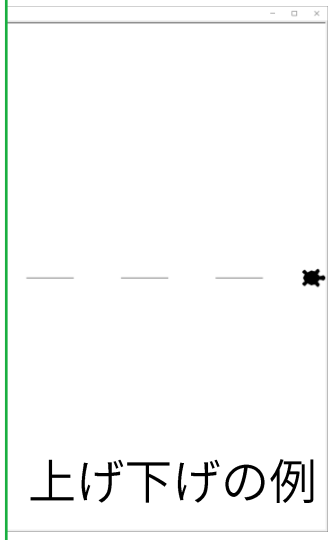
ずっと描画
=ずっと下げた状態

```
kame.forward(100)
kame.penup()
kame.forward(100)
kame.forward(100)
kame.forward(100)
kame.forward(100)
kame.forward(100)
```



一度上げたら
上げた状態を保つ

```
kame.forward(100)
kame.penup()
kame.forward(100)
kame.pendown()
kame.forward(100)
kame.penup()
kame.forward(100)
kame.pendown()
kame.forward(100)
kame.penup()
kame.forward(100)
kame.pendown()
```



上げ下げの例

参考: 描画速度が遅い...

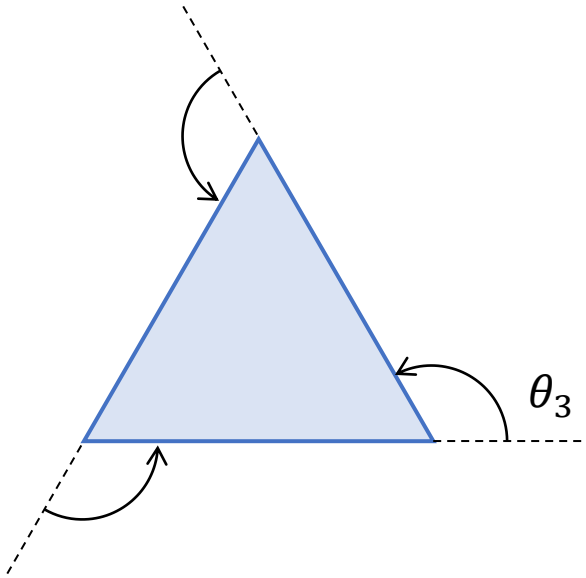
- メソッド speed(): タートルの移動速度を変える
  - 0から10までの整数を引数にとる
    - 数字が大きいほど速い
    - 0はアニメーション無し
- メソッド hideturtle(): タートルを消す

- それでも遅い場合の奥の手
  - turtle.tracer(100, 0)

いろいろ描いてみよう

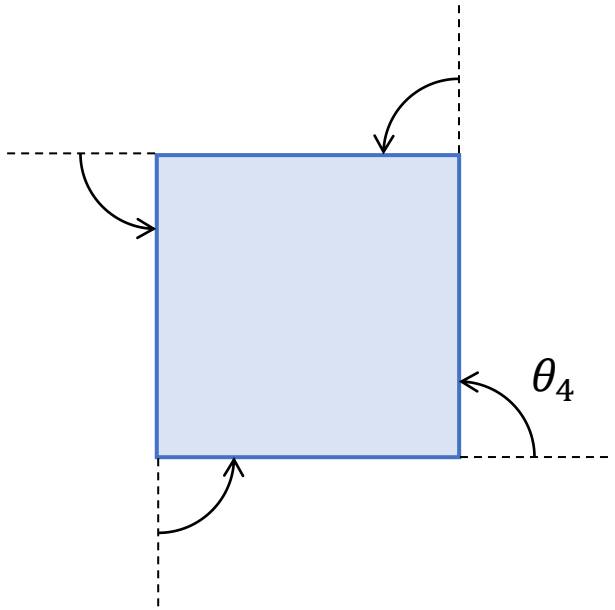
正n角形を描く

- 回転角度の法則性を考えてみよう



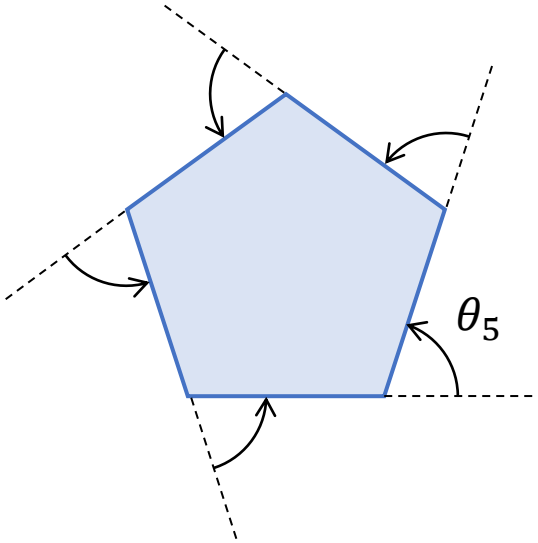
θ_3の左回転を3回行うと，
元の向きに戻る

$\therefore \theta_3 \times 3 = 360$



θ_4の左回転を4回行うと，
元の向きに戻る

$\therefore \theta_4 \times 4 = 360$

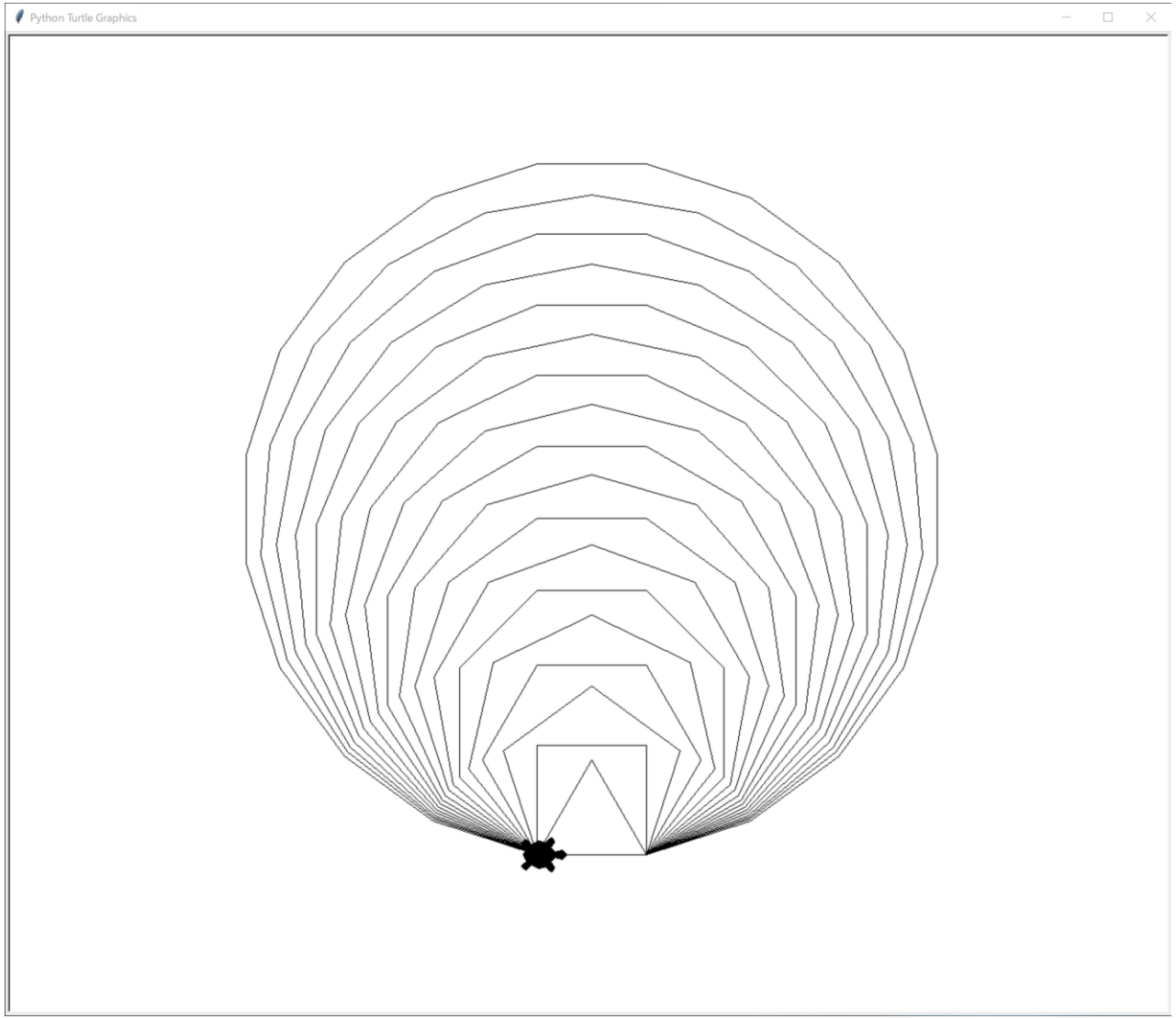


θ_5の左回転を5回行うと，
元の向きに戻る

$\therefore \theta_5 \times 5 = 360$

正n角形を描くサンプル

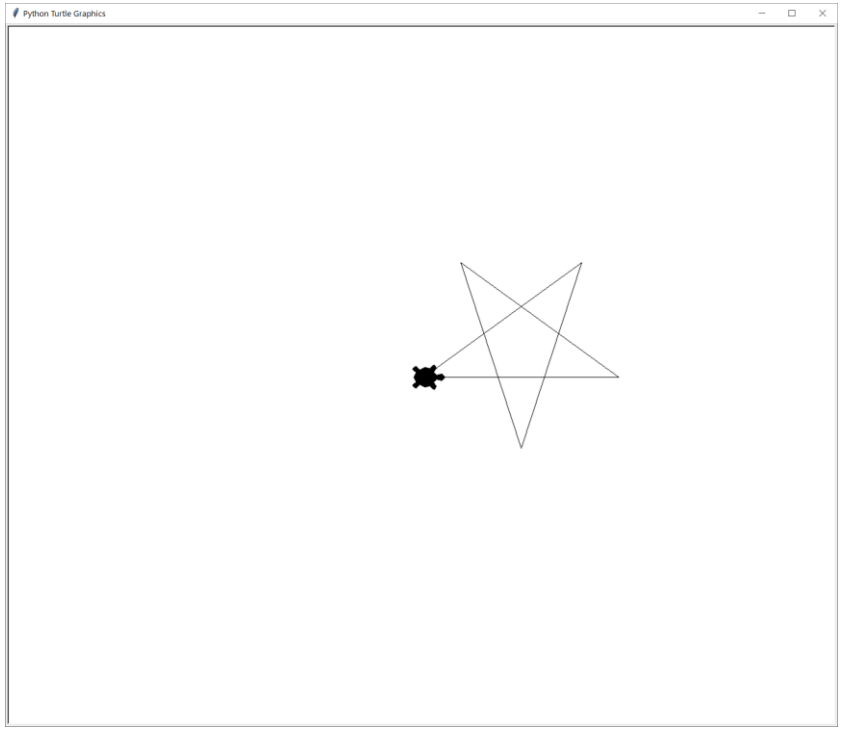
```
import turtle

kame = turtle.Turtle()
kame.shape('turtle')
kame.shapesize(2, 2, 3)

# (初期位置の移動; 略)

move_unit = 100
smallest = 3
largest = 20
for n in range(smallest, largest + 1):
    for i in range(n):
        kame.forward(move_unit)
        kame.left(360 / n)
```



星形を描く

- 一筆描きの要領で描く．
  - 正五角形の辺を延長して得られる星形(星形正五角形)では，星の頂点の角度は36度 → 180-36=144度回転すればよい．

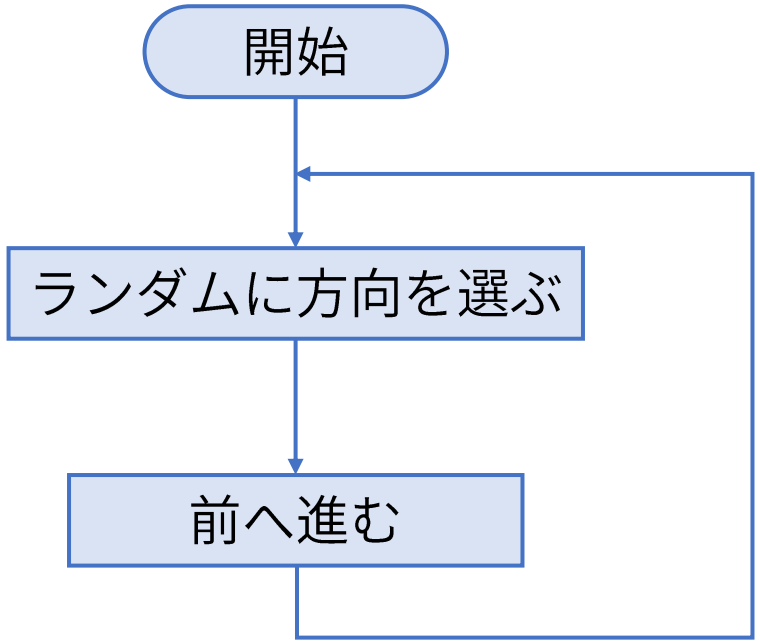
```
import turtle

kame = turtle.Turtle()
kame.shape('turtle')
kame.shapesize(2, 2, 3)

edge_length = 300
rotate_angle = 144

for i in range(5):
    kame.forward(edge_length)
    kame.left(rotate_angle)
```



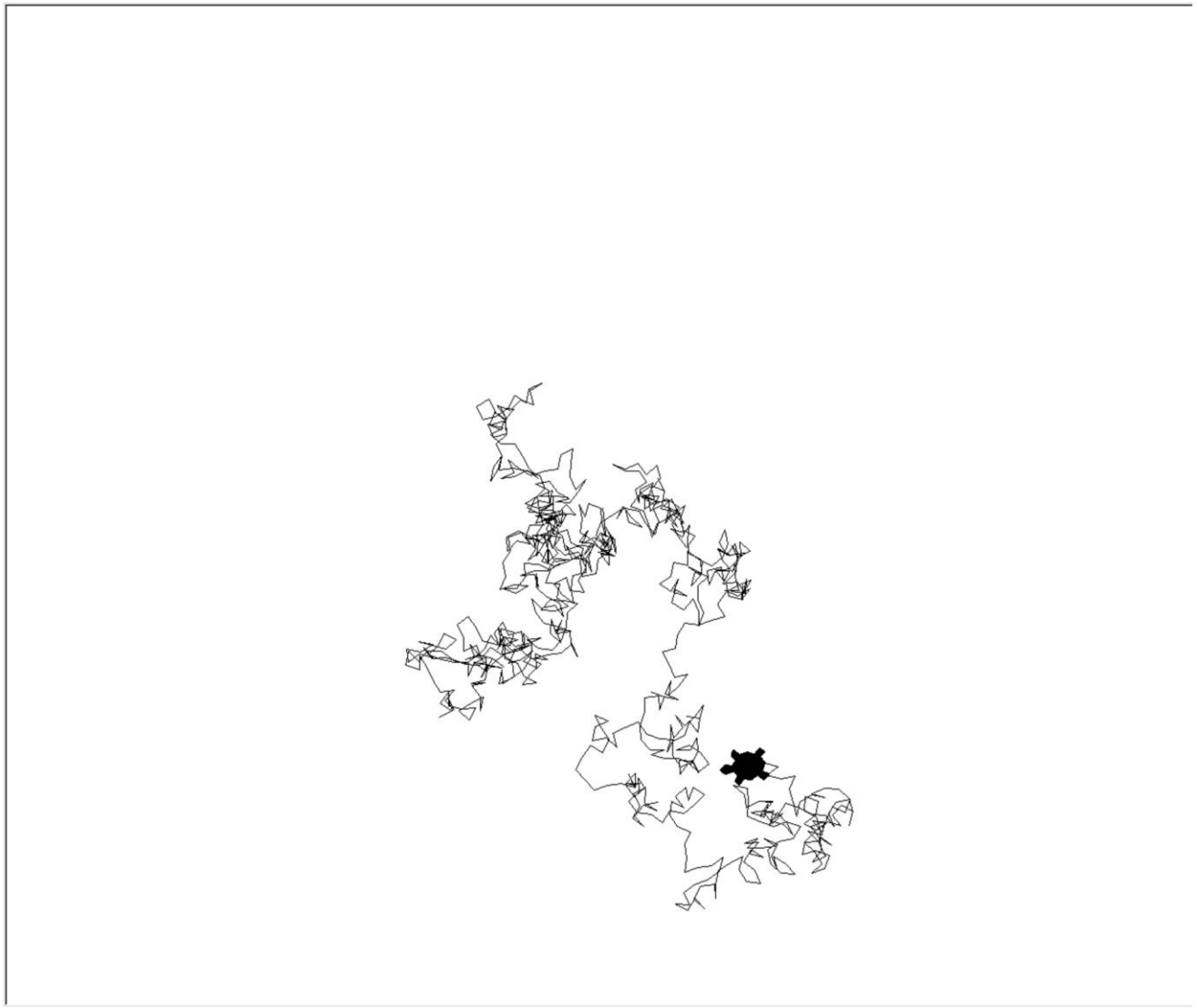
ランダムウォーク

- ランダムな方向に向きを変えては前進する．

```
import turtle
import random

kame = turtle.Turtle()
kame.shape('turtle')
kame.shapesize(2, 2, 3)

distance = 15 # 1回の移動距離

while True:
    angle = random.randrange(360)
    kame.left(angle)
    kame.forward(distance)
```



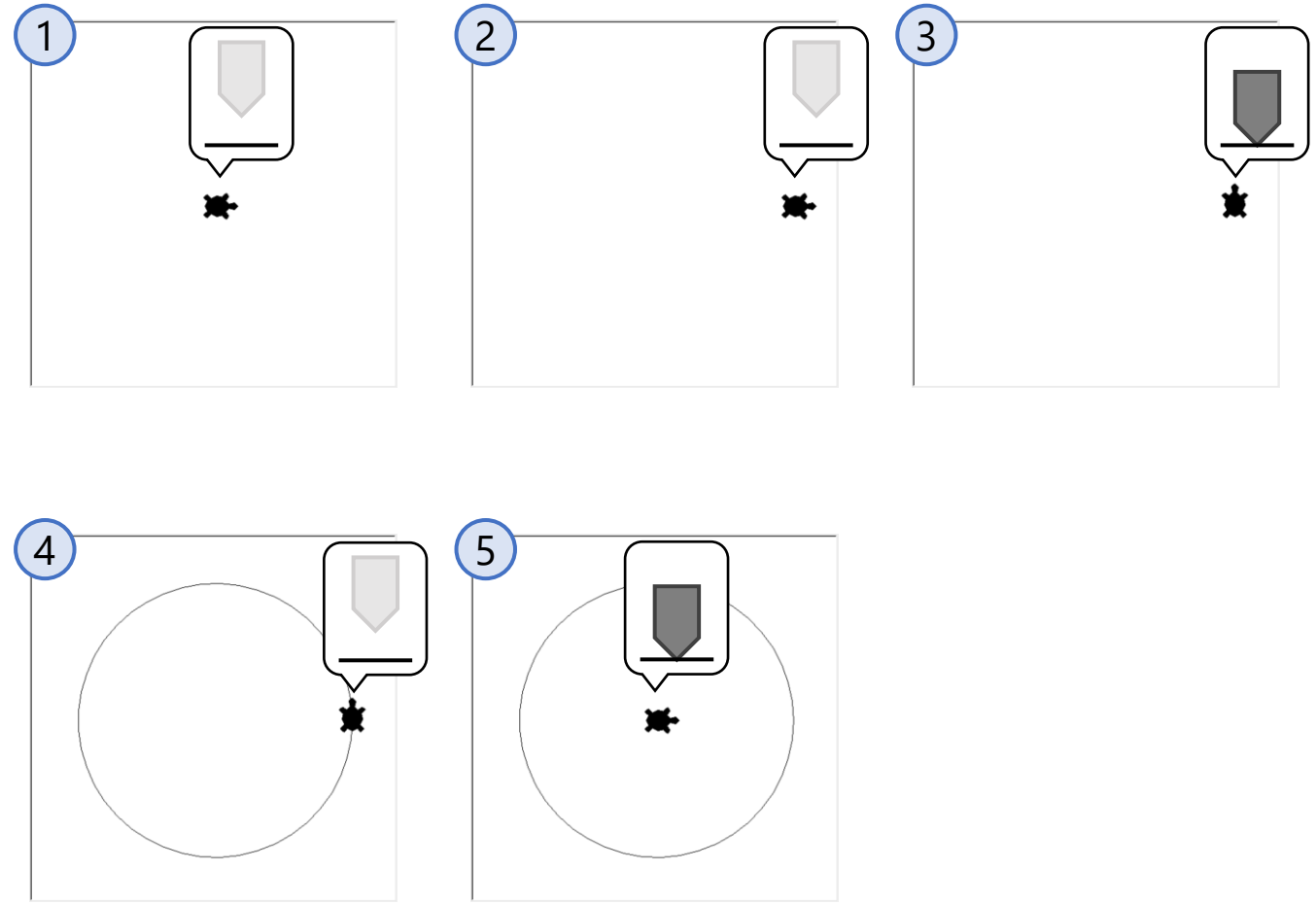
ランダムウォーク



ランダムウォーク: 円の内側だけ動く

- まず原点を中心とした円を描く

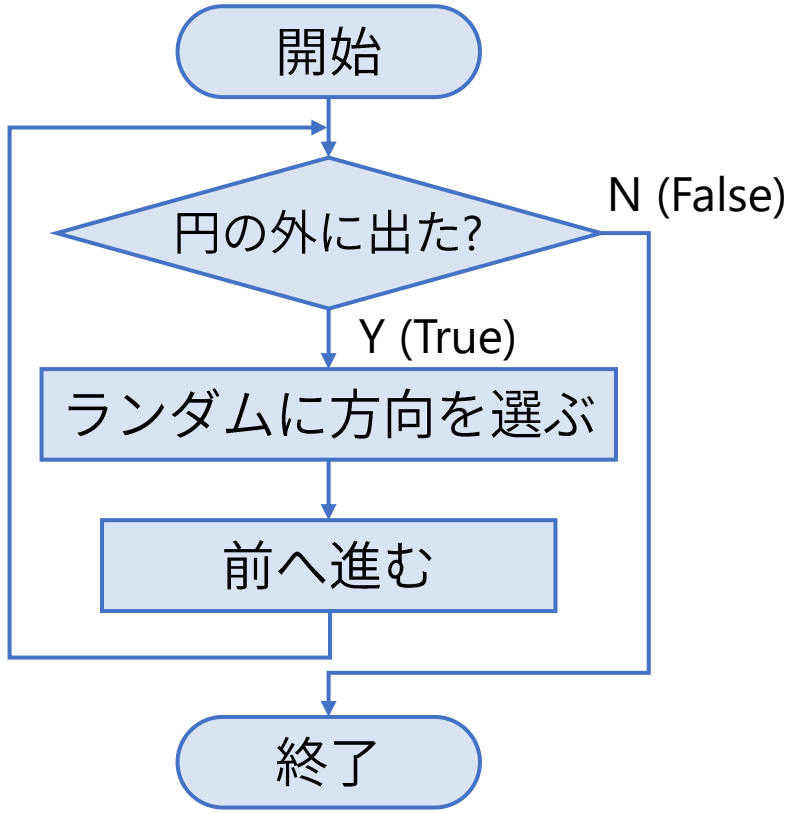
```
limit_radius = 200
kame.home()
kame.penup()
kame.forward(limit_radius)
kame.left(90)
kame.pendown()
kame.circle(limit_radius)
kame.penup()
kame.home()
kame.pendown()
```



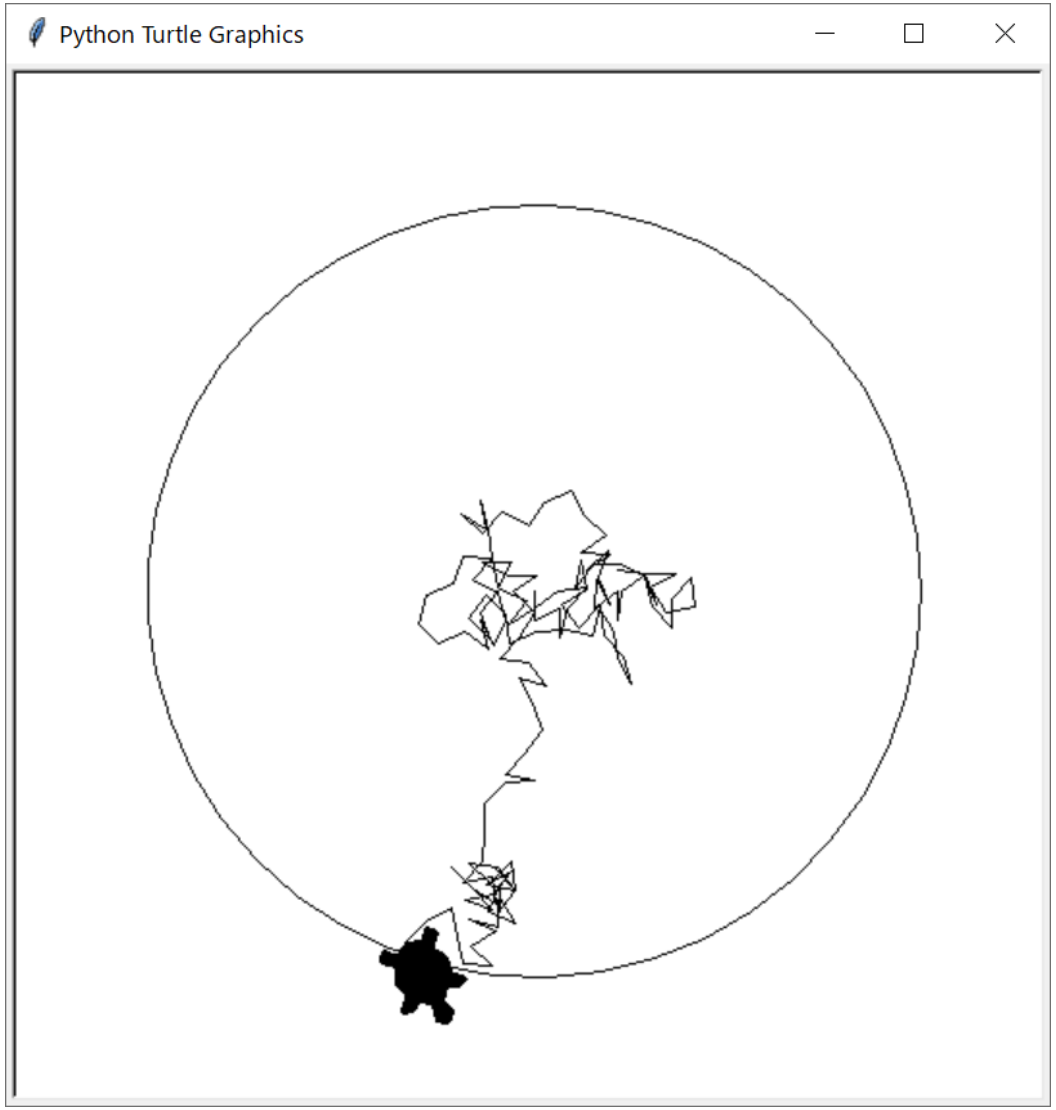
ランダムウォーク: 円の内側だけ動く

- 原点からの距離(メソッド distance()で求められる)に基づいて停止

```
import turtle
import random

kame = turtle.Turtle()
kame.shape('turtle')
kame.shapesize(2, 2, 3)

distance = 15
limit_radius = 200

# (円を描く; 略)

while kame.distance(0, 0) < limit_radius:
    angle = random.randrange(360)
    kame.left(angle)
    kame.forward(distance)
```



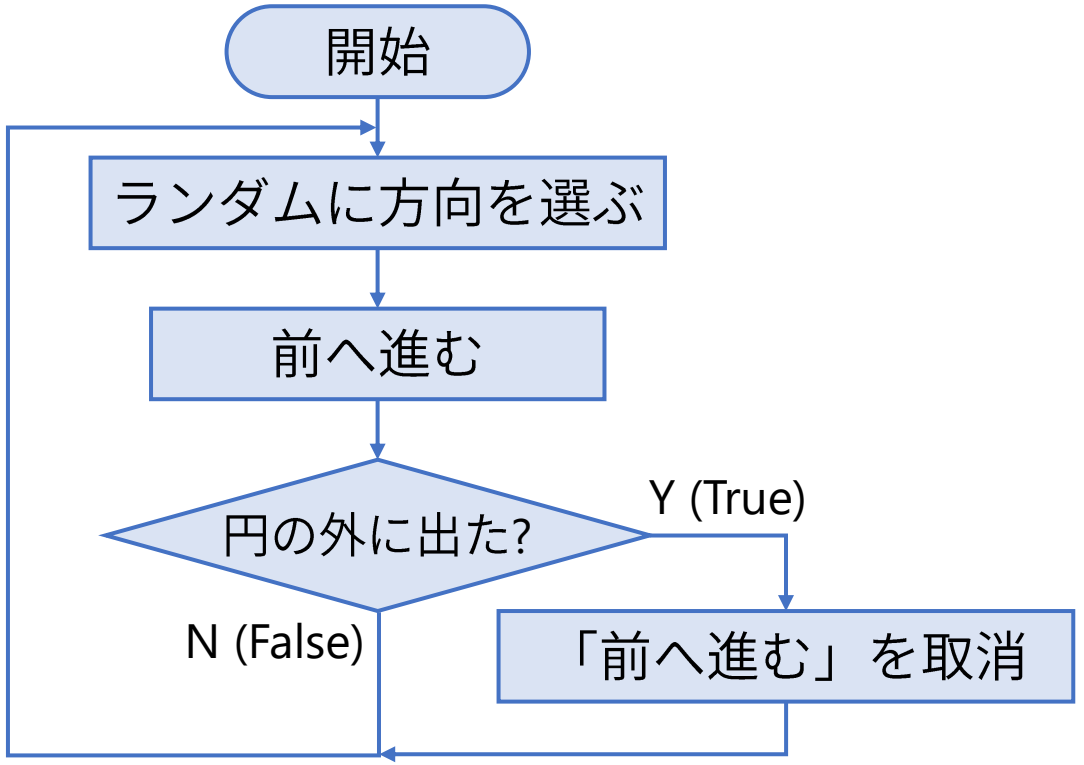
ランダムウォーク: 円の内側だけ動く



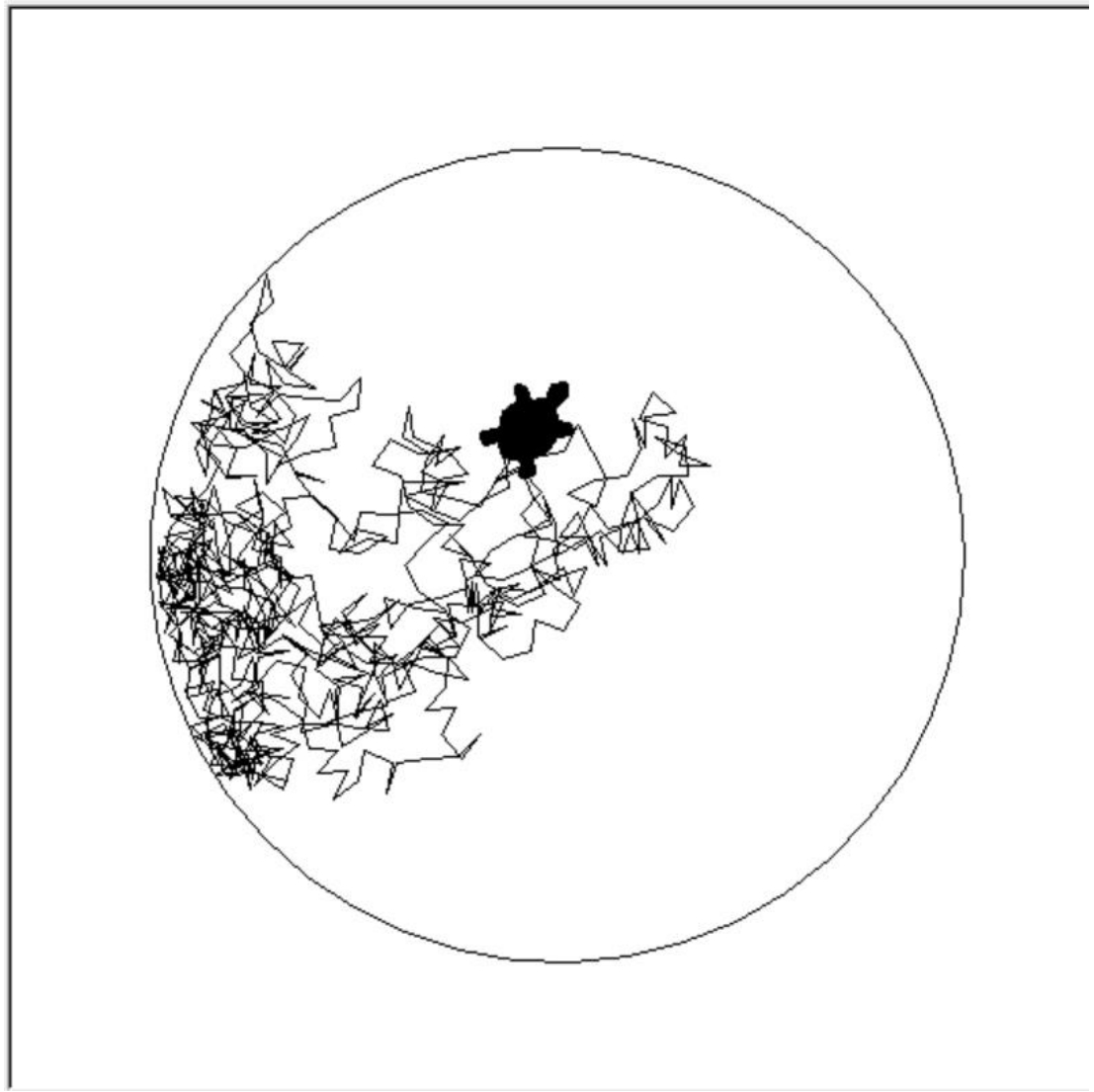
ランダムウォーク: 円の内側を永遠に動く

- 原点からの距離が一定値以上であれば直前の動作をキャンセル

```
import turtle
import random

kame = turtle.Turtle()
kame.shape('turtle')
kame.shapesize(2, 2, 3)

distance = 15
limit_radius = 200

# (円を描く; 略)

while True:
    angle = random.randrange(360)
    kame.left(angle)
    kame.forward(distance)
    if kame.distance(0, 0) >= limit_radius:
        kame.undo()
```

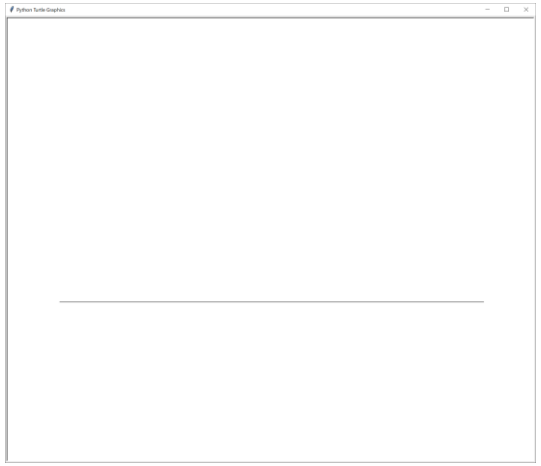


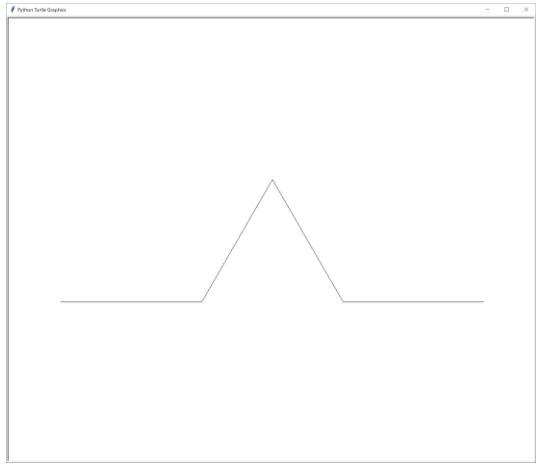
ランダムウォーク: 円の内側を永遠に動く

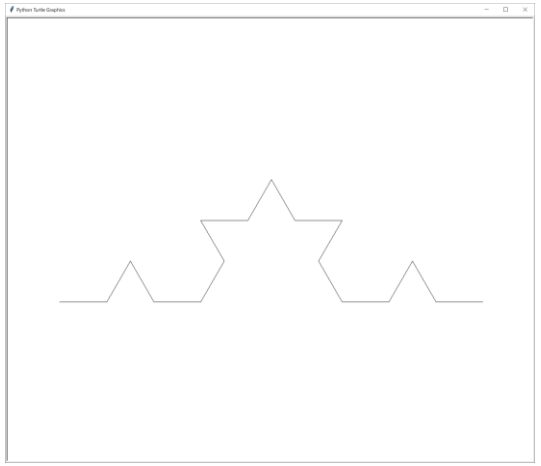


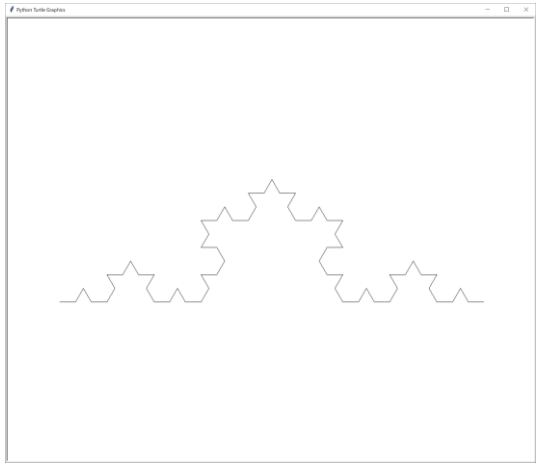
おまけ: いろいろ描いてみよう

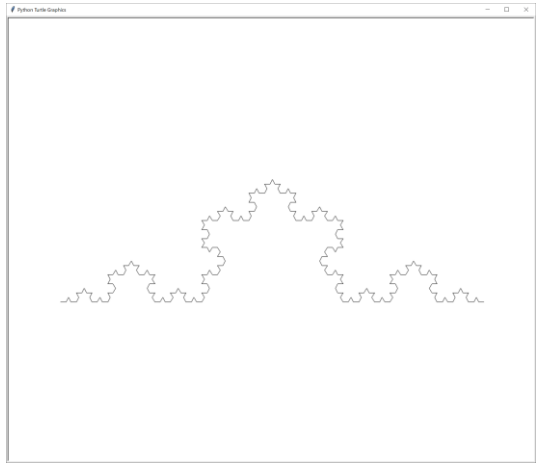
コッホ曲線

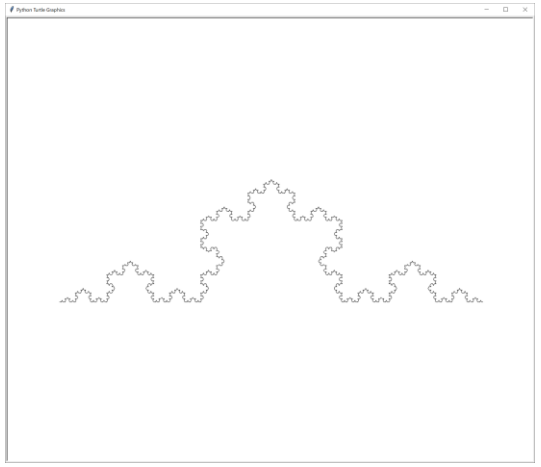


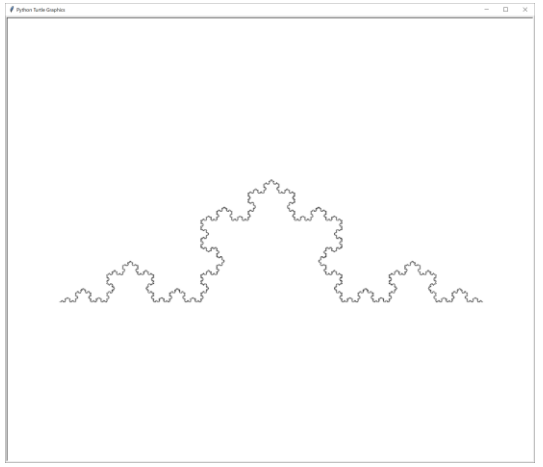




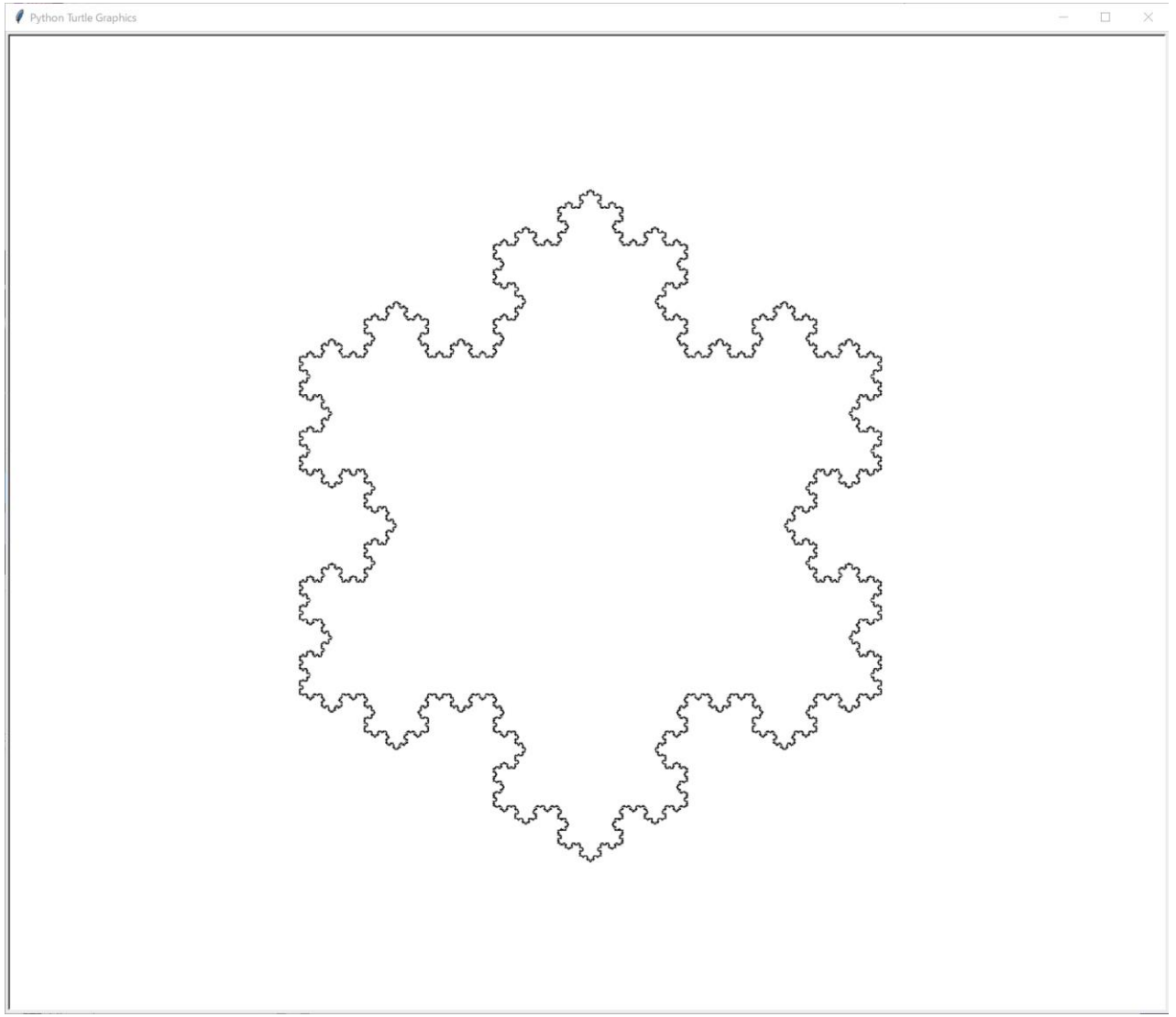




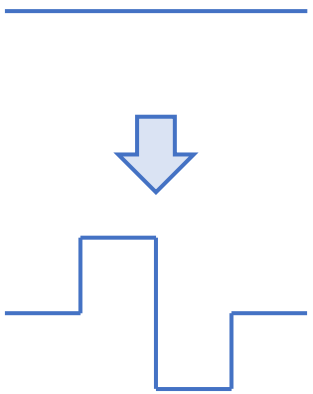


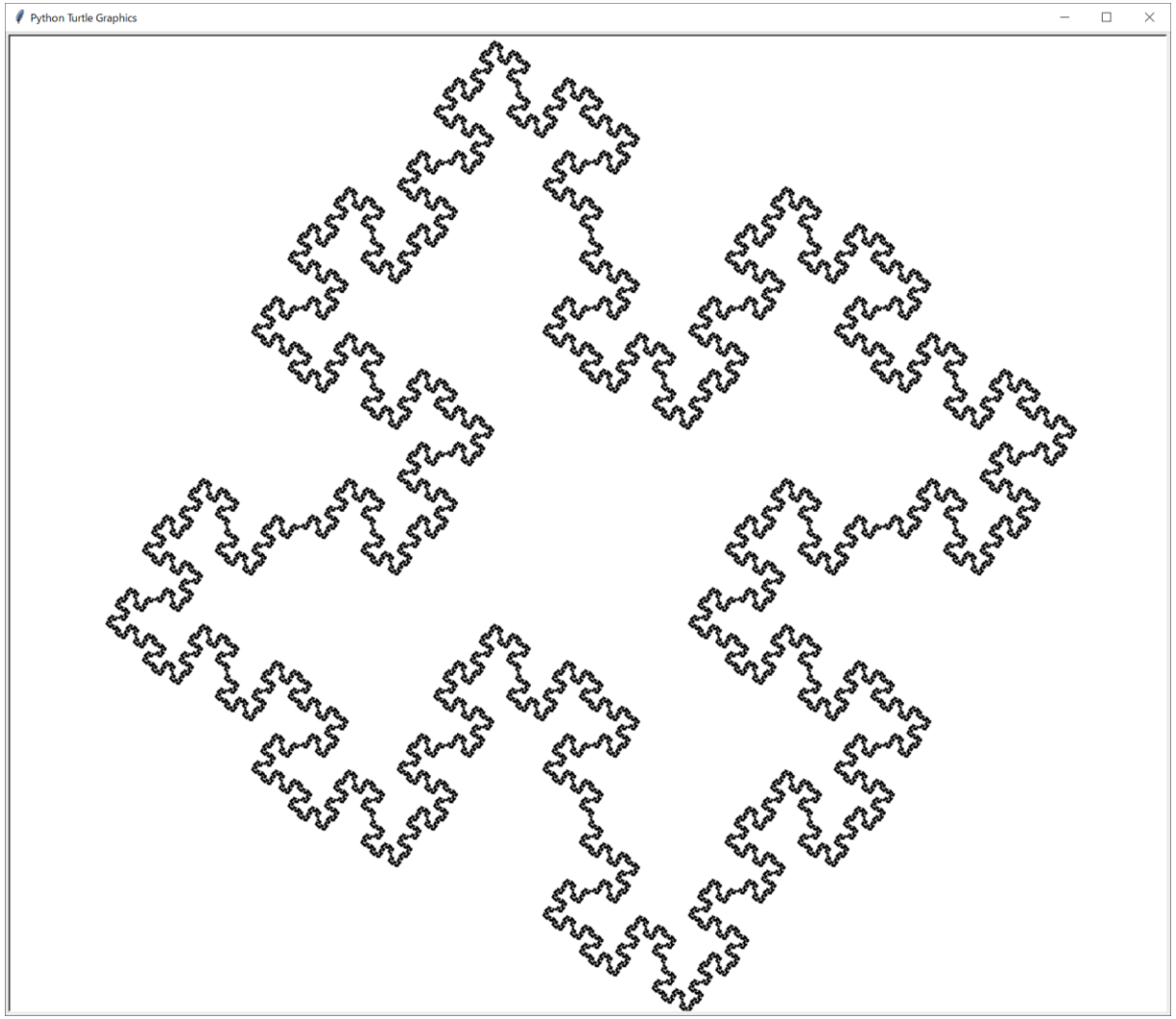


コッホ雪片

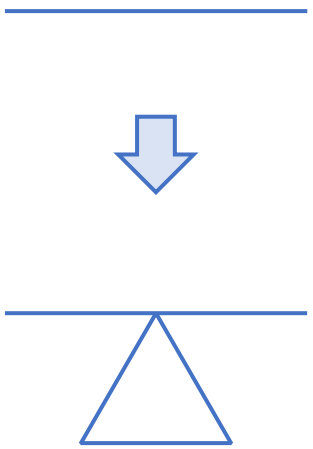


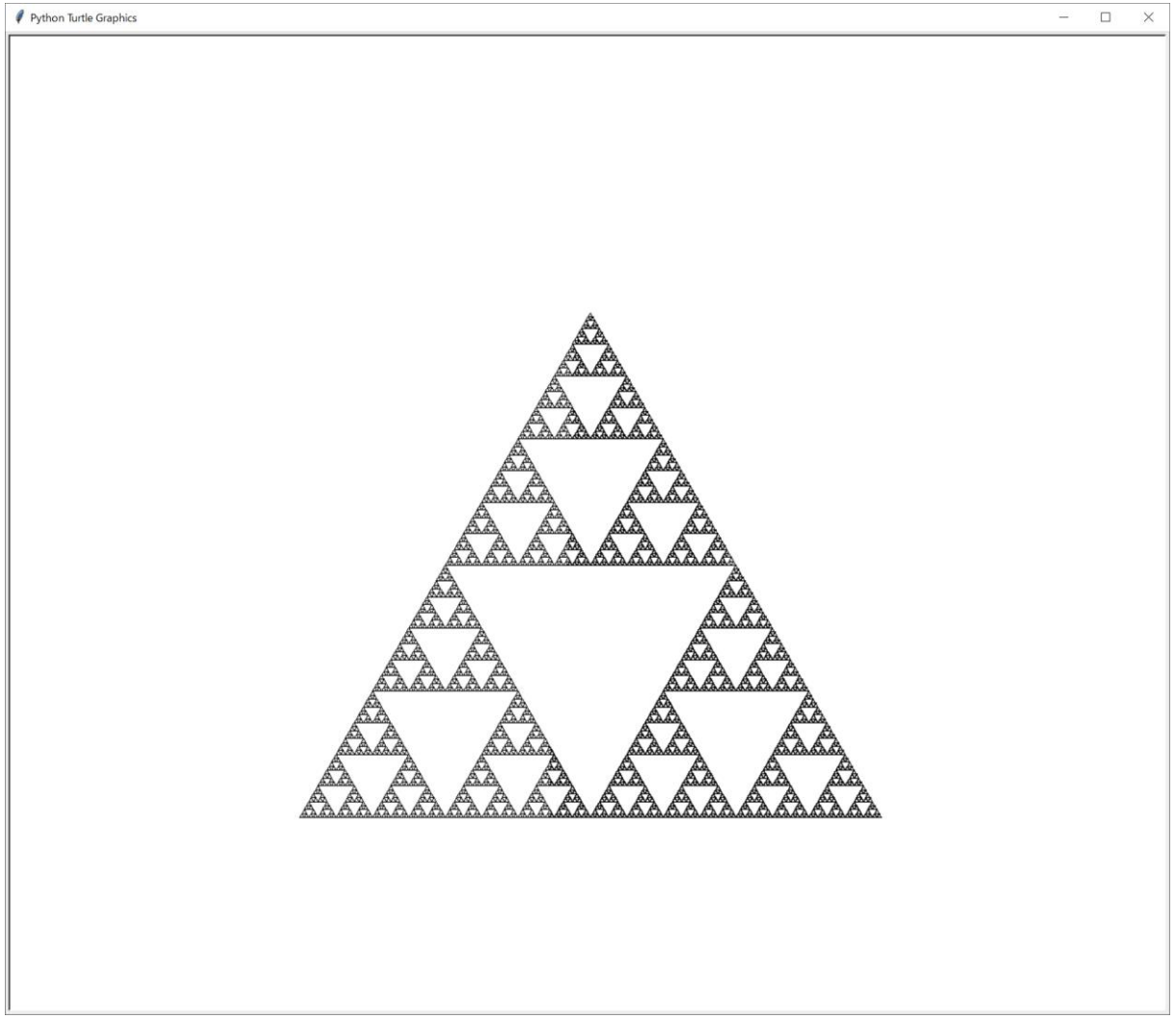
バリエーション



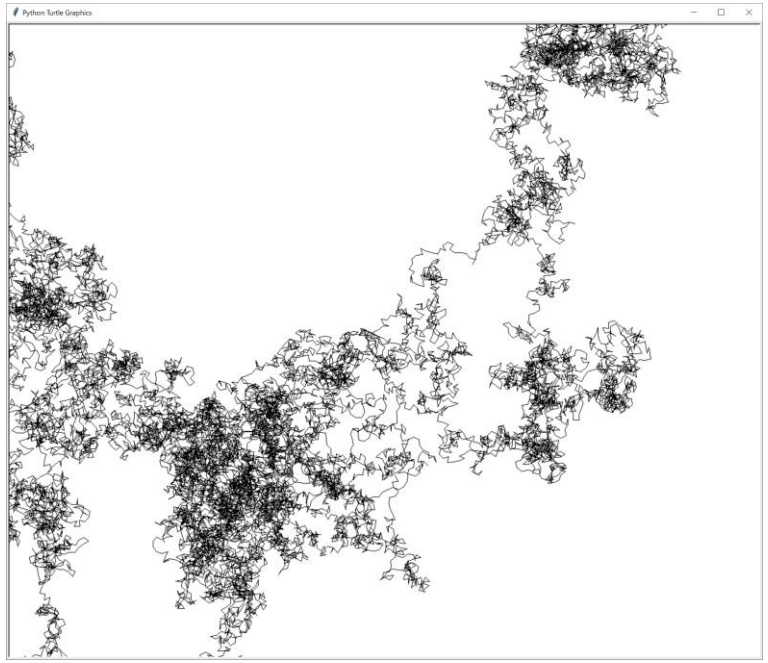


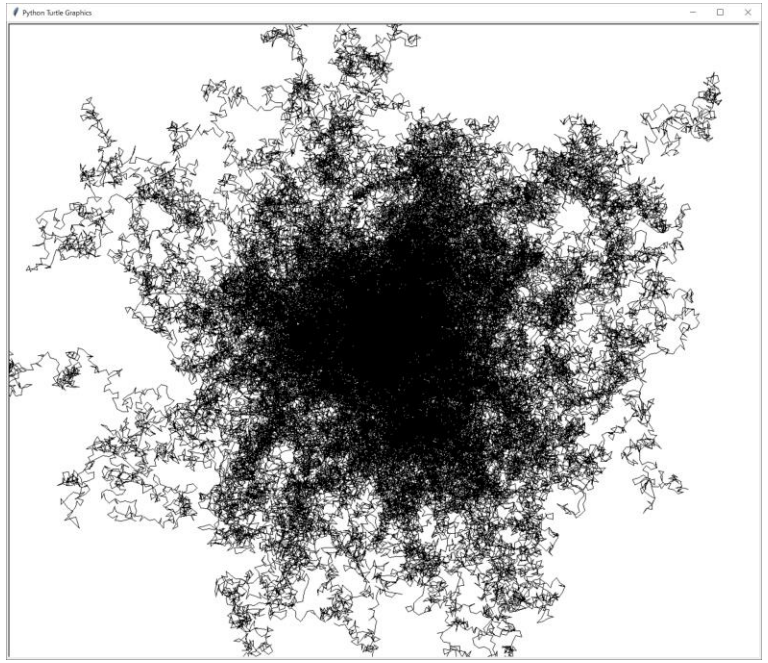
シェルピンスキーのガスケットもどき

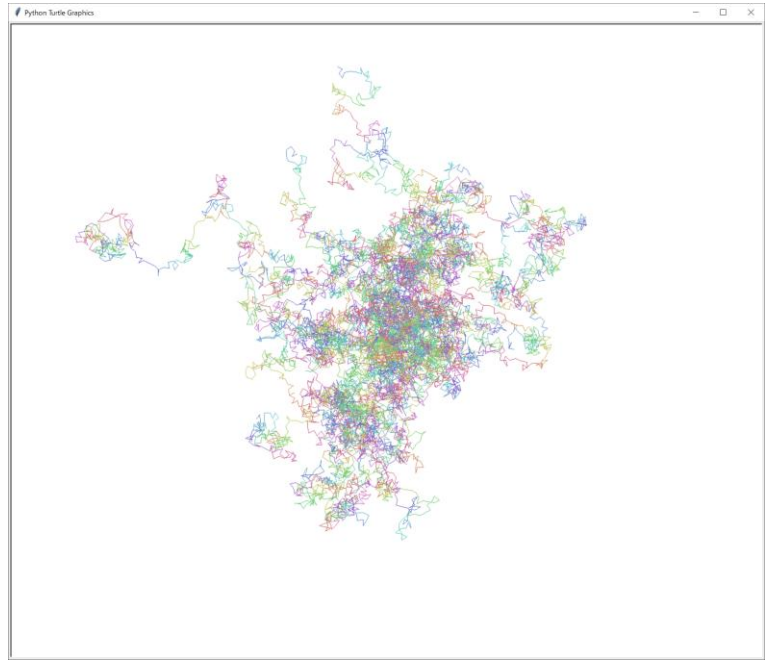




ランダムウォークいろいろ







ランダムウォークいろいろ