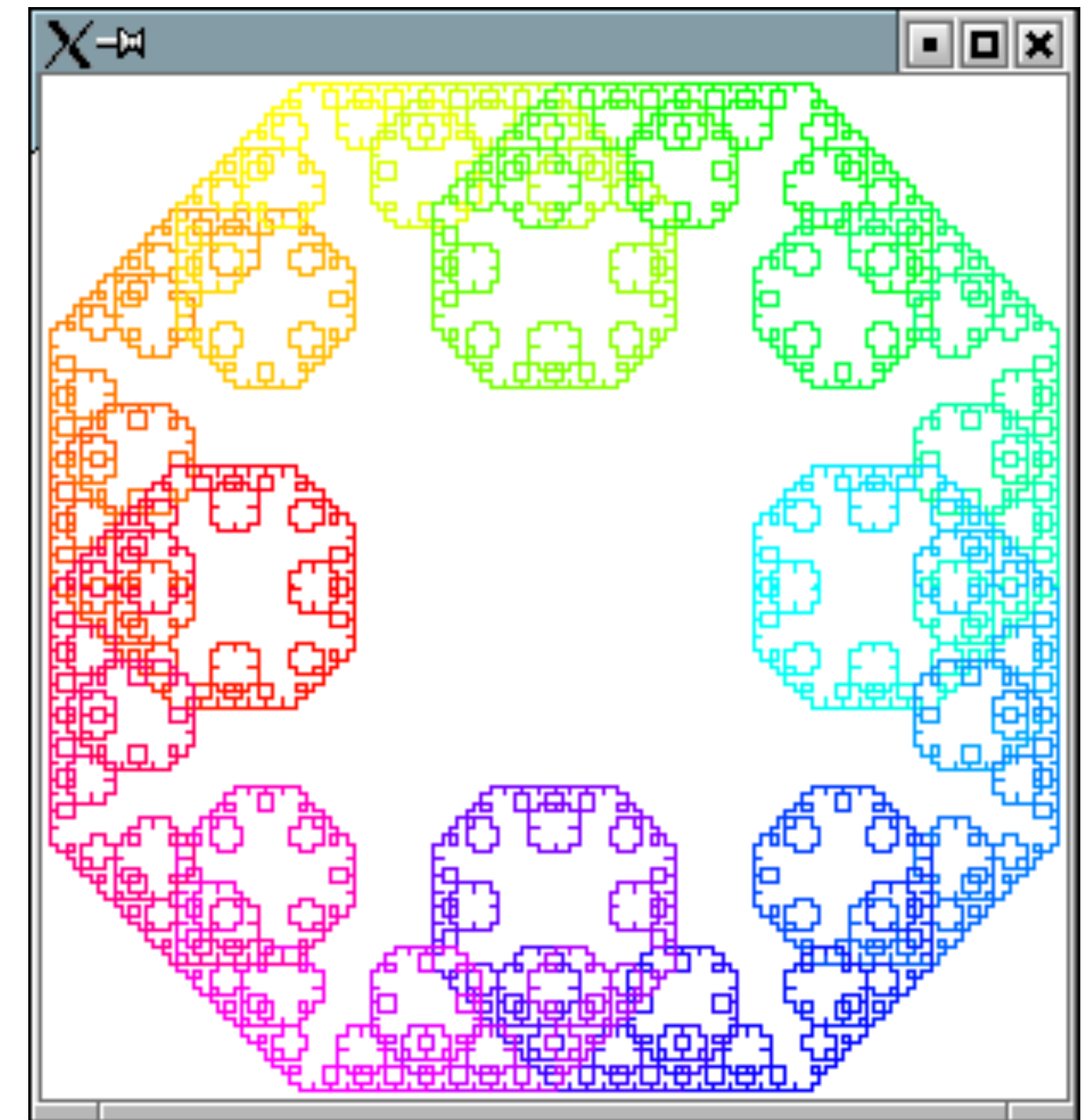


WWL 採択高校の1・2年生を対象とした  
名古屋大学附属高等学校・教養教育院共催  
「高大接続探究セミナー」

# Pythonでフラクタル を描画しよう

名古屋大学 山里敬也

[yamazato@nagoya-u.jp](mailto:yamazato@nagoya-u.jp)





ROUTE1 ← FREE HALL → ROUTE23  
JB.STUDIO K.Y

山里敬也 (YAMAZATO, Takaya) [yamazato@nagoya-u.jp](mailto:yamazato@nagoya-u.jp)  
名古屋大学 教養教育院 教授 ハイブリットラーニングセンター・センター長  
兼務：工学研究科情報・通信工学専攻,  
情報基盤センター教育情報メディア研究部門,  
附属図書館研究開発室

◆ 名大の授業：<http://ocw.nagoya-u.jp>

◆ 研究

◆ 可視光通信、ITS、確率共鳴、Open Educational Resources (OER)

◆ 趣味

◆ 自転車、ジョギング、バンド



名大の授業を体験しよう

# 「勇気ある知識人」

名古屋大学は、自由闊達な学風の下、人間性と科学の調和的发展を目的とし、創造的な研究と自発性を重視する教育を実践することによって、世界屈指の知的成果を生み出すとともに、論理的思考力と想像力に富んだ勇気ある知識人の育成を目指しています。

## OCWサポートスタッフ募集中！

「名大の授業」の制作にご協力いただける学生の皆さんを募集しています。

# 名大の授業

(名古屋大学オープンコースウェア)

→ 「名大の授業」について

このサイトの特徴と使い方、スタッフ紹介、ご利用についてのページです。

→ 用語解説

シラバス、コースウェアなど、サイト内のわかりにくい用語を解説します。

→ 名大の授業紹介&PodCast



担当教員が生の声で授業内容とその魅力を解説します。

→ スタジオチャンネル



2014-07-03UP  
名古屋大学の色々な情報を発信します。

授業一覧から探す



# 名大で実際に行っている講義内容を修正してやります

## プログラミング及び演習（3.0 単位）

ちょっとだけ

科目区分	専門基礎科目	
授業形態	講義及び演習	
対象学科	電気電子情報工学科	
開講時期 1	2 年春学期	
必修／選択	必修	
担当教員	山里 敬也 教授	米澤 拓郎 准教授

# 本講義の目的（シラバス）

**C言語**による演習を通じて、計算機を用いたより高度なプログラミング技法・問題解決技法を学ぶ。具体的には比較的大きなプログラム（500~1000行程度）を書く実力をつける。

これにより、情報リテラシーのような基礎力に加え、論理的思考力や問題解決力といった応用力も涵養する。さらには、プログラムの構成を設計（デザイン）することにより、創造力を鍛えることも狙いとする。

今回は Python を使います

達成目標：

- 目的・仕様に従いプログラムの構成要素を論理的に設計できる
- 設計に従い、効率の良いプログラム（**C言語**）の実装ができる

# 講義スケジュール

(2021/05/17現在)

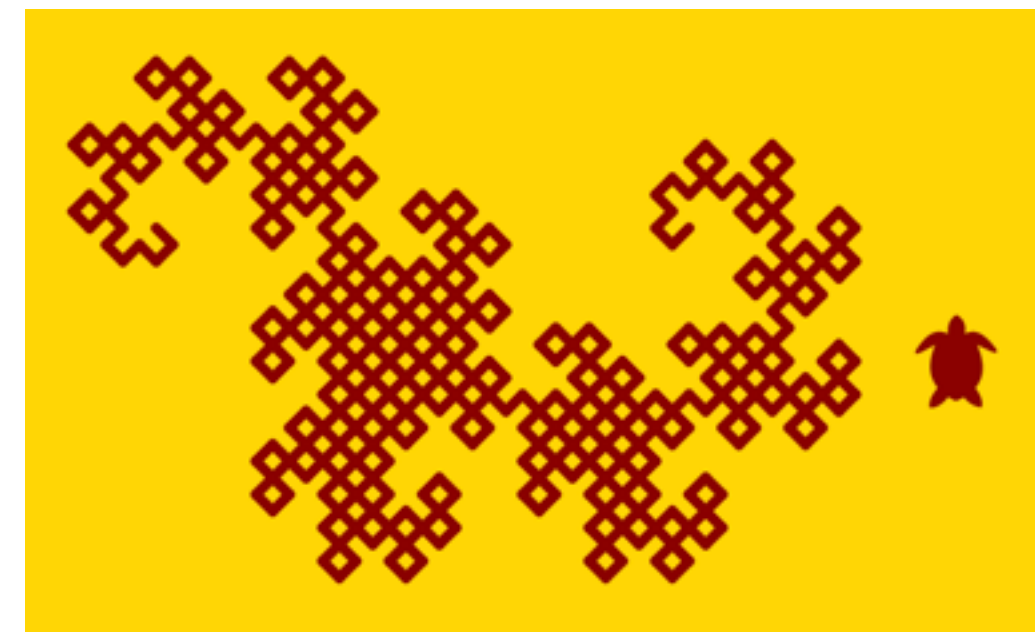
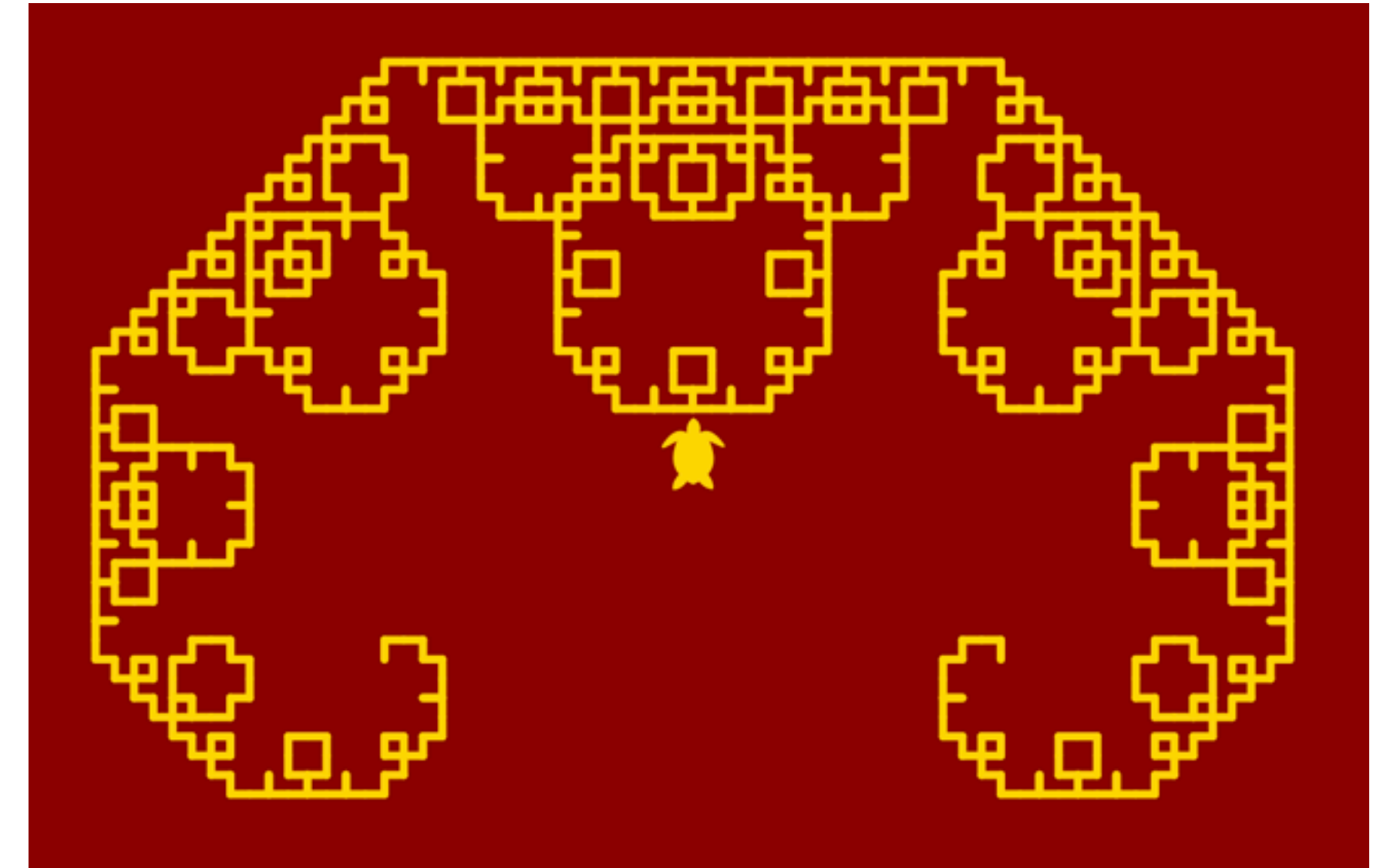
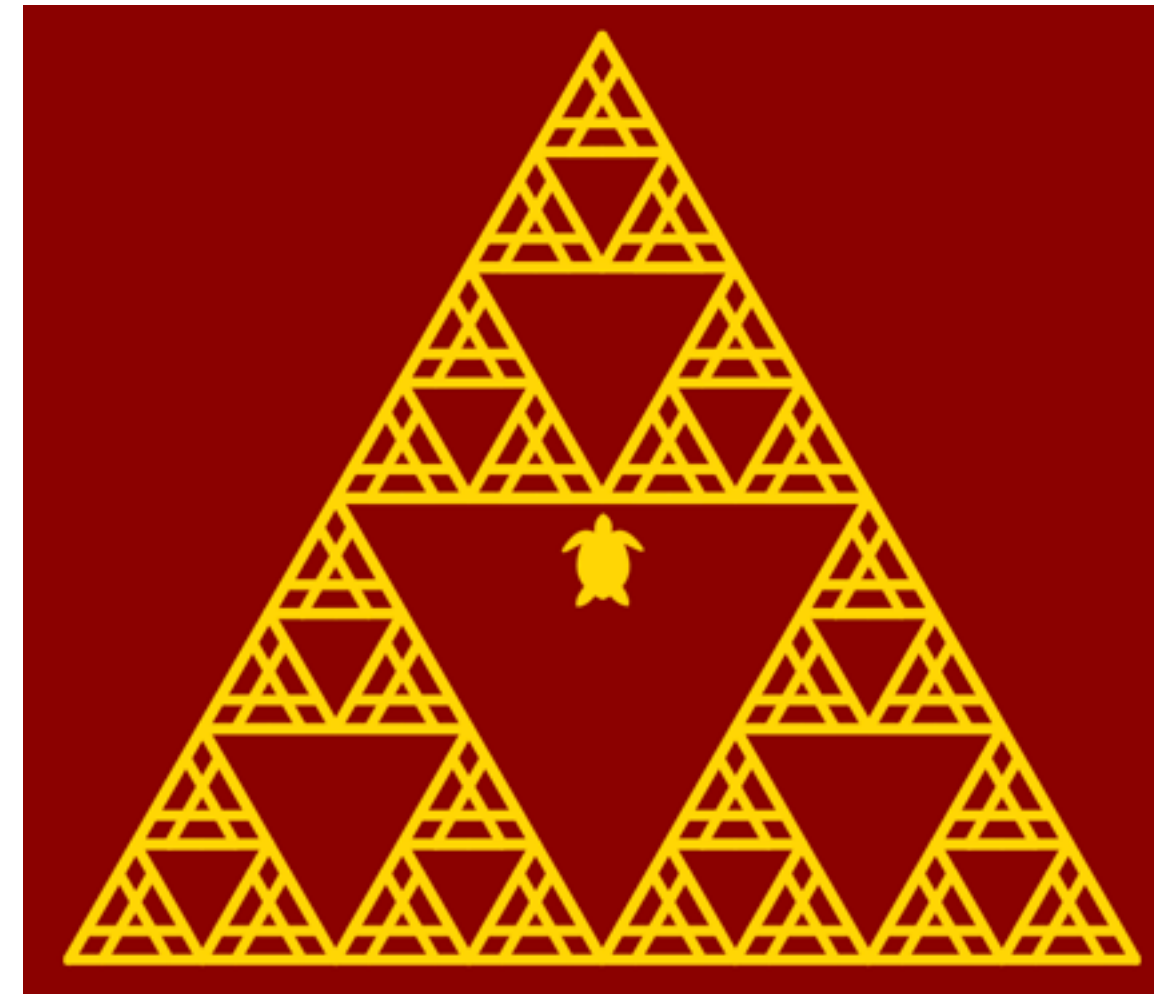
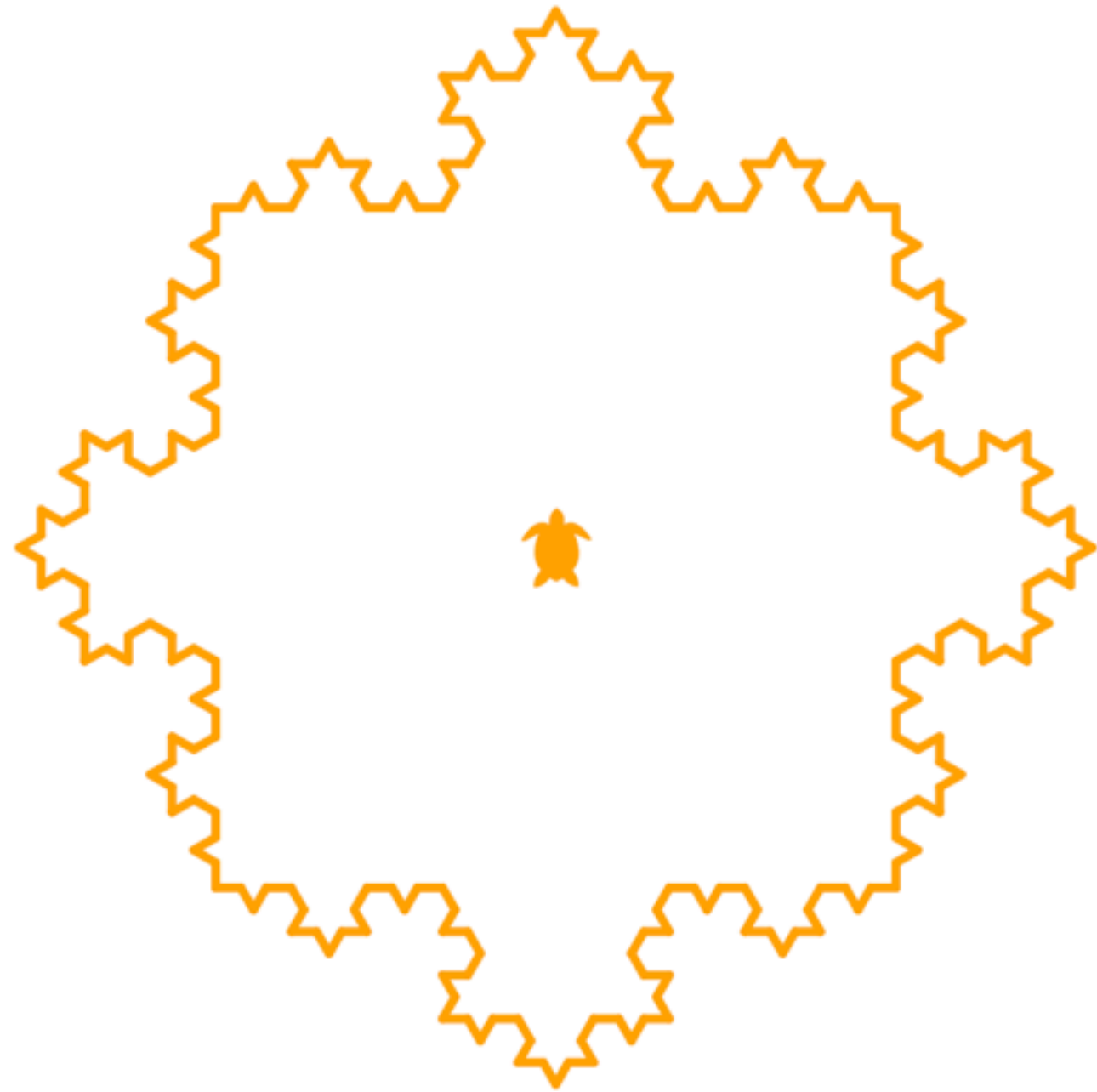
日付	内容
4/16	ガイダンス・環境構築・C言語復習（合同）
4/23	C言語復習
4/30	データ解析
5/07	データの可視化
5/14	ポインタとデータ走査
5/21	シーザー暗号
5/28	シーザー暗号解読
6/04	中間課題（合同）
6/11	ビジュアライゼーション（1）
6/18	ビジュアライゼーション（2）←再帰グラフィックス
6/25	ゲームプログラム
7/02	最終課題（合同）
7/09	課題制作時間
7/16	課題制作時間
7/30	課題発表（A, B別）
8/06	総括（合同）

取り上げる内容

# スケジュール

日時	内容
<b>8月19日（木）</b> <b>10：30～12：00</b>	Google Colabortory (Python)入門
<b>8月19日（木）</b> <b>13：00～14：30</b>	Turtle Graphics で多角形を描こう
<b>8月20日（金）</b> <b>10：30～12：00</b>	再帰関数とフラクタル （コッホ曲線, シェルピンスキーのガスケット）
<b>8月20日（金）</b> <b>13：00～14：30</b>	2分木, Levy曲線, Drangon曲線

# 最終目標：再帰関数をつかった絵を描こう



2日目にチャレンジします！



# Google Colabortory

## Python入門

# 準備（事前課題）

## Google Colaboratory

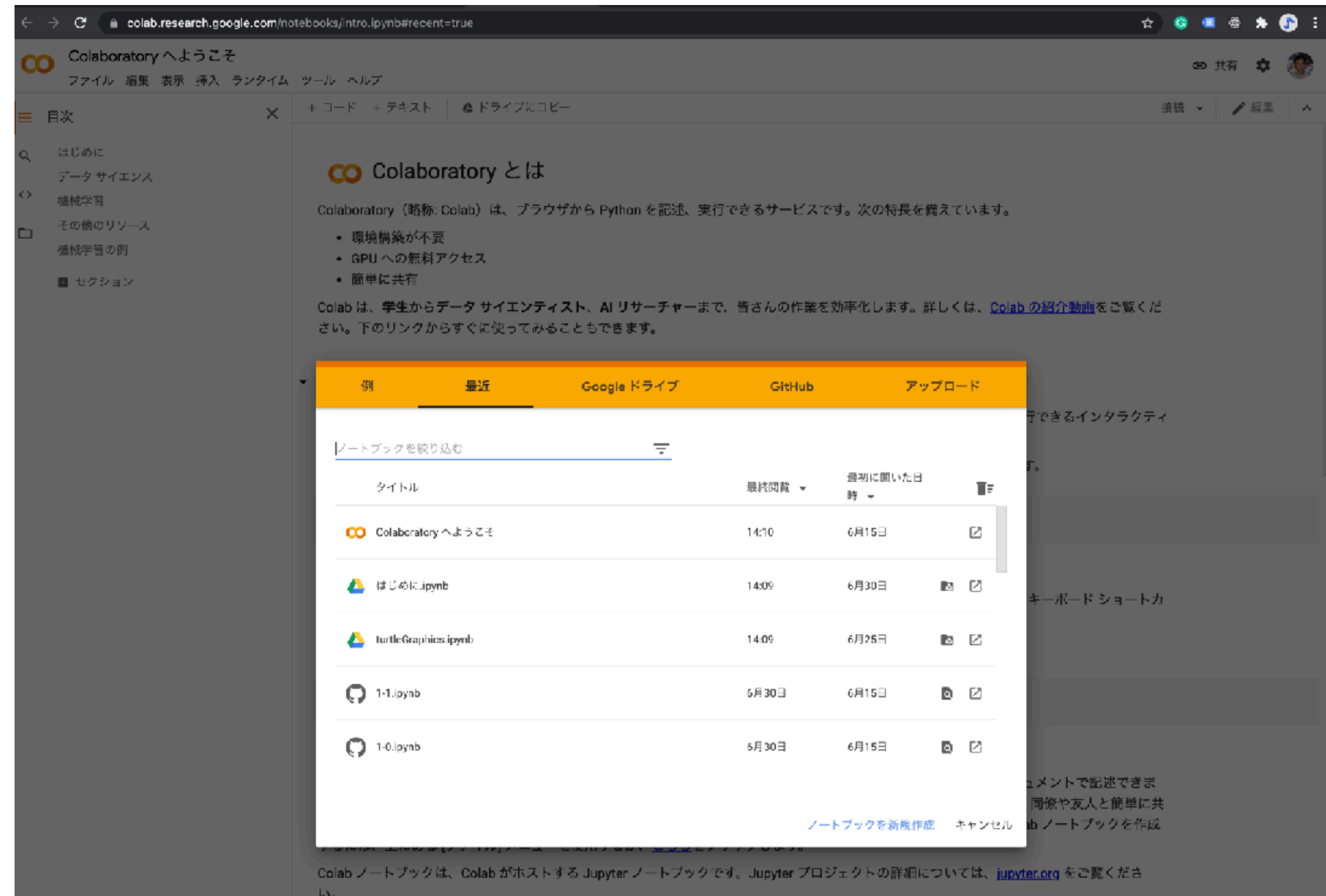
- この講義では Google Colaboratory（コラボと呼びます）を使います。
- コラボの利用には Google アカウントが必要です。事前に取得しておいてください。
- コラボはPCのブラウザから利用できます。自宅からでもOK！
  - ブラウザは何でも良いと思います。もし、うまくいかないようであれば Google Chromeをダウンロードしてご利用ください。



# コラボノートブック

## Colab Notebook

- コラボを立ち上げると右のような画面になります
- 附属学校版NUCTからファイルをダウンロードし、ご自身のコラボにアップロードしてください。
- コラボの使い方
  - <https://sites.google.com/view/ipsjmooc/How2>



# コラボの基本

## コラボの code セルを使ってみよう

これが codeセル.

```
[ ] 1+1
```

押下すると pythonプログラム が実行できます.

警告がでるけど心配いりません.



▶ 1+1

2

実行結果

確認して押下



# Codeセルにプログラムを入力してみよう

## Pythonが実行できます

- cut, copy, paste, undo ができます
  - Ctrl+c, Ctrl+x, Ctrl+v, Ctrl+z
- 四則演算もできます

演算子	説明	例	結果
**	累乗	2**3	8
*	かけ算	3*5	15
/	割り算	10/3	3.33
//	整数の割り算 (小数点以下切り捨て)	10//3	3
%	剰余 (余り)	10%3	1
+	足し算	1+2	3
-	引き算	10-3	7

# 変数も使えます

codeセルで実行してみよう

変数 a に「2」を代入

コメント文（実行されない）

```
a=2
b=3
c=a*b
# 結果をプリント
print("c=", c)
```

print()関数で c を表示

変数は短い名前が良い

例：a, b, aa, bb

- 数字から始めてはいけない
- ハイフン (-) , スペース, 特殊文字は使えない

必要に応じてコメントを入れる

# から始まる行はコメント



# 関数を定義しよう

def 関数名 (引数1, 引数2, ...):

*c = multiply(a, b)*

*multiply(a, b) = a \* b*

TABもしくはスペース4つ開ける

関数は「def」で宣言

関数が受け取る値：引数

関数の結果（返値）は「return」で指定



```
def multiply(a,b):  
    c = a*b  
    return c
```

```
# multiply(2,3) の結果をプリント  
print("c=",multiply(2,3))
```



c= 6

pythonの関数名は**スネークケース**で書くことが多い

例：turtle\_graphics

- 分かりやすい名前をつける
- 小文字で始まる
- 単語を組み合わせる場合アンダースコア (   ) で繋げる

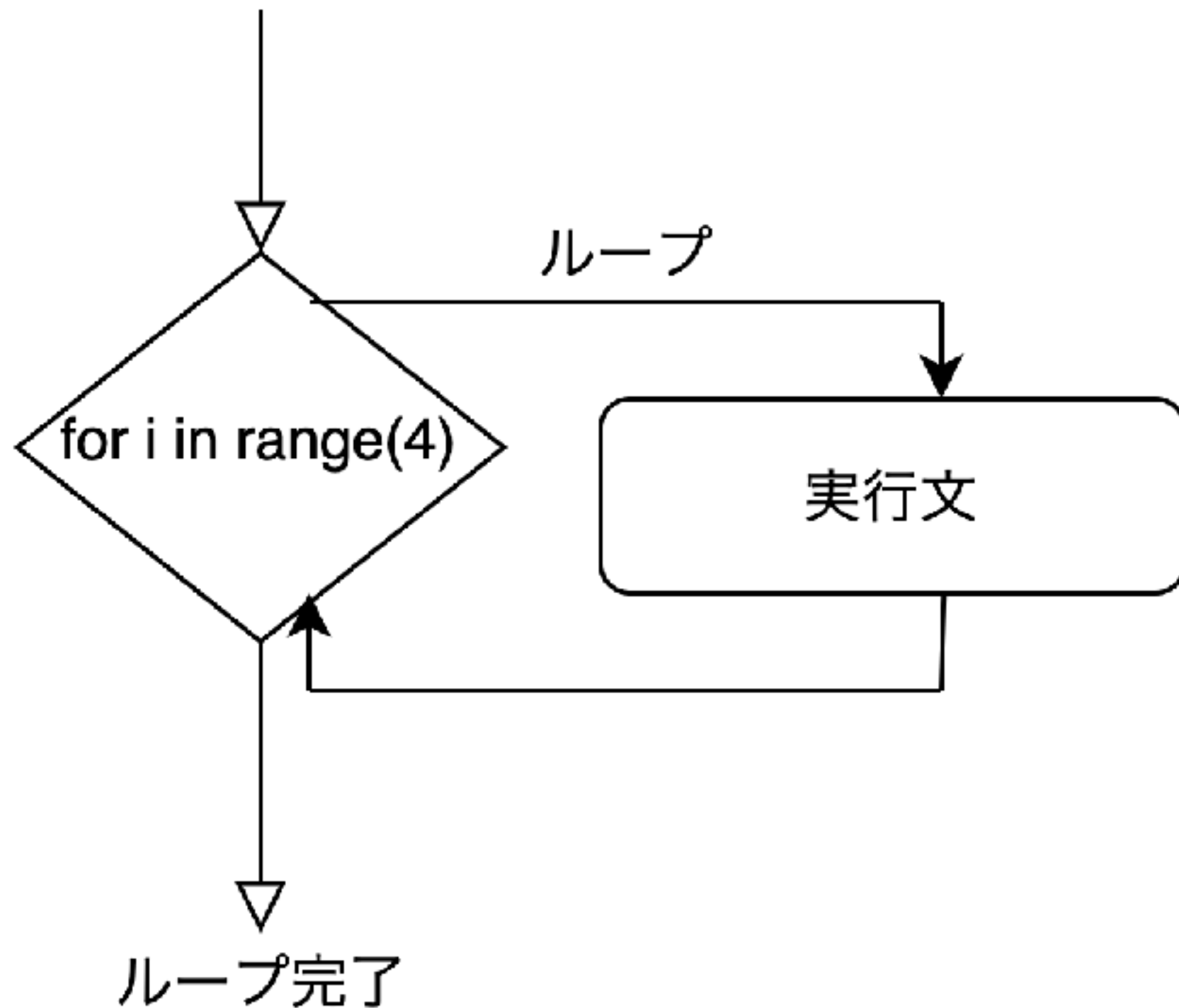
# Pythonの命名規則

ほぼスネークケース（単語と単語をアンダースコアでつなぐ）

変数	スネークケース	snake_case
定数	スネークケース（全て大文字）	SNAKE_CASE
グローバル変数	スネークケース	snake_case
関数	スネークケース	snake_case
関数の引数	スネークケース	snake_case
クラス	パスカルケース	PascalCase
インスタンス変数	スネークケース	snake_case
メソッド	スネークケース	snake_case
パッケージ	スネークケース	snake_case
モジュール	スネークケース	snake_case



# for ループと range くり返し



```
for i in range(4):  
    print(i)
```

for 変数 in オブジェクト:  
 実行文

文字列・リスト・関数など

range()  
整数列のリストを返す関数

range(start, stop[, step])

range(0,4,1)

> 0, 1, 2, 3

range(4)

> 0, 1, 2, 3

range(0,4,2)

> 0, 2

range(4,0,-1)

> 3, 2, 1, 0

# ターゲットグラフィックス



# タートルグラフィックス

LOGO: by シーモア・パパート

**pen down**



子供用のプログラミングシステム  
わかりやすくするため、簡単なグラフィックスを利用

亀が

前進 : forward(距離)

回転 : right(角度), left(角度)

ペンの上げ下げ : up(), down()

# タートルグラフィックスの主な関数

<https://github.com/tolgaatam/ColabTurtle>

関数	短縮形（別表記）	説明
<b>forward(units)</b>	fd(units)	亀を units（ピクセル）前進
<b>backward(units)</b>	bk(units)	亀を units（ピクセル）後進
<b>right(degrees)</b>	rt(degrees)	亀を degrees 度 右に向ける（回転）
<b>left(degrees)</b>	lt(degrees)	亀を degrees 度 左に向ける（回転）
<b>speed(s)</b>		亀をスピード s で動かす．1が最も遅く，13が最も早い．初期値 4
<b>penup()</b>	up()	ペンを上げる（以後，描画しない）
<b>pendown()</b>	pd()	ペンを下げる（以後，描画をはじめる）
<b>setpos(x,y)</b>	goto(x,y)	座標 (x, y) へ亀を移動
<b>bgcolor()</b>		背景の色を指定 ( <a href="https://www.w3schools.com/colors/colors_names.asp">https://www.w3schools.com/colors/colors_names.asp</a> )
<b>pencolor()</b>	color()	ペンの色を指定 ( <a href="https://www.w3schools.com/colors/colors_names.asp">https://www.w3schools.com/colors/colors_names.asp</a> )
<b>pensize(w)</b>	width(w) pensize(w)	ペンの大きさ w を指定
<b>position()</b>	pos()	亀の現在の座標 (x,y) を取得
<b>heading()</b>	getheading()	亀が向いている方向（角度）を取得

# Colabでのタートルグラフィックス

<https://github.com/tolgaatam/ColabTurtle>

- まず, 以下のコマンドでタートルグラフィックスをインストール

```
!pip3 install ColabTurtle
```

```
from ColabTurtle.Turtle import *
```

- 図を書くには初期化が必要です.

```
initializeTurtle( )
```

- 描画領域は以下のように設定されています.

- 画面サイズ : 800 x 500

- 亀の位置 : 画面の中央 (400,250)

- 亀の向き : 上

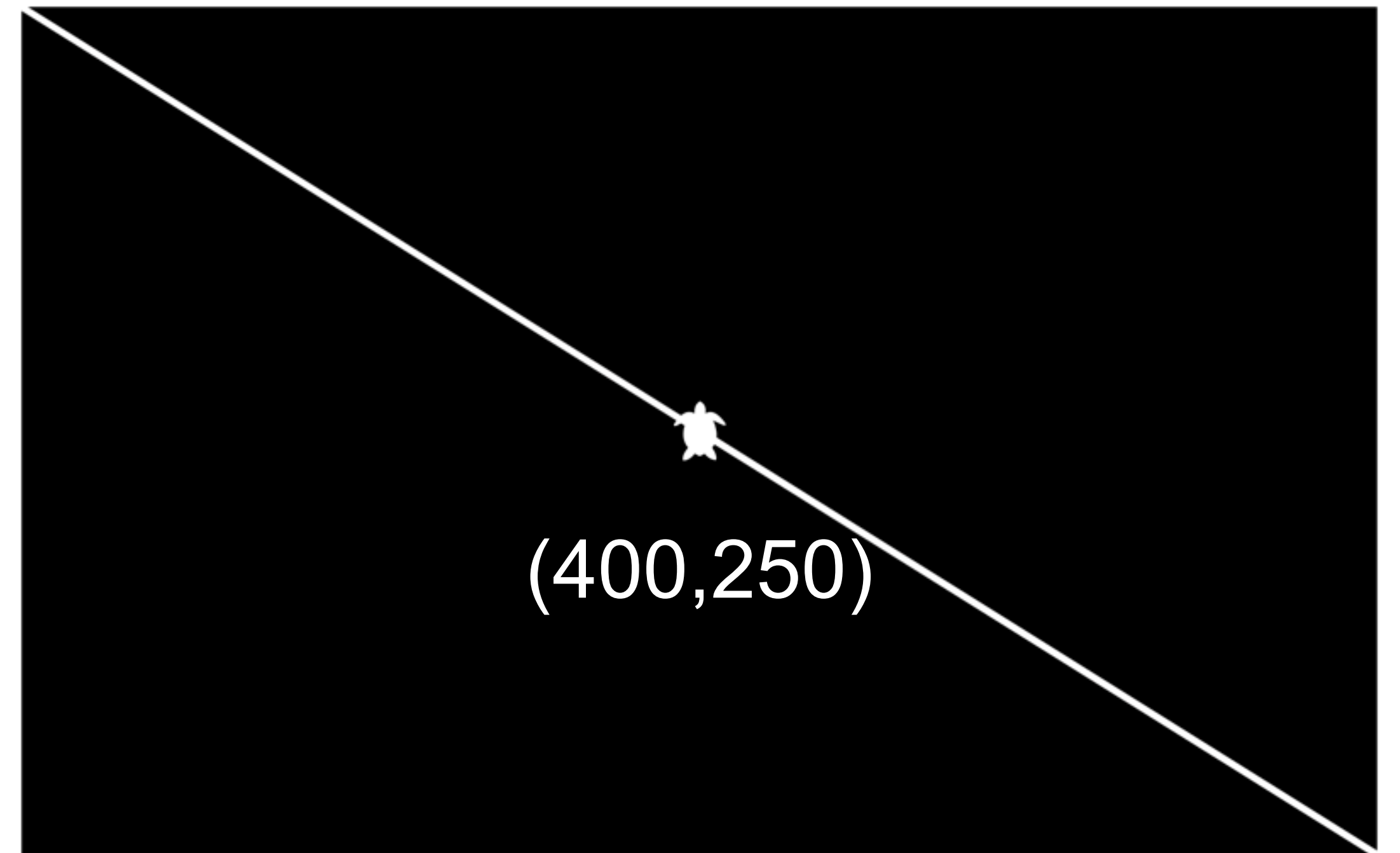
- 亀の動く速さ : speed = 4

- ペンのサイズ : width(4)

- ペンの色 : color("white")

- 背景の色 : bgcolor("black")

(0,0)



(400,250)

(800,500)

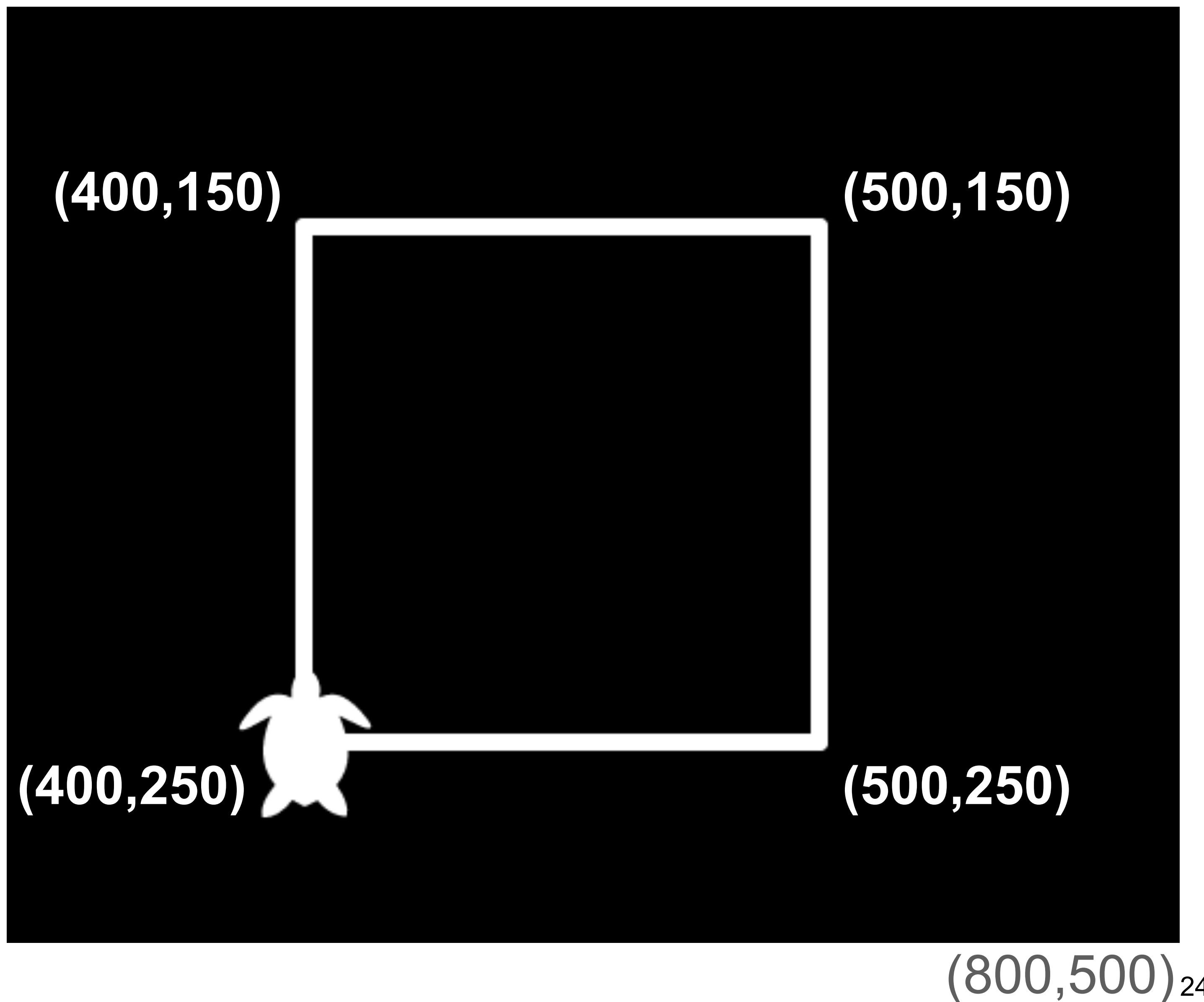


# タートルグラフィックス

まずは設計から

- 800x500の画面に一边が100の正方形を亀の軌跡で描く
- 画面は左上が  $(0,0)$  , 右下が  $(800,500)$
- 亀の最初の位置は方眼紙の中心  $(400,250)$

$(0,0)$

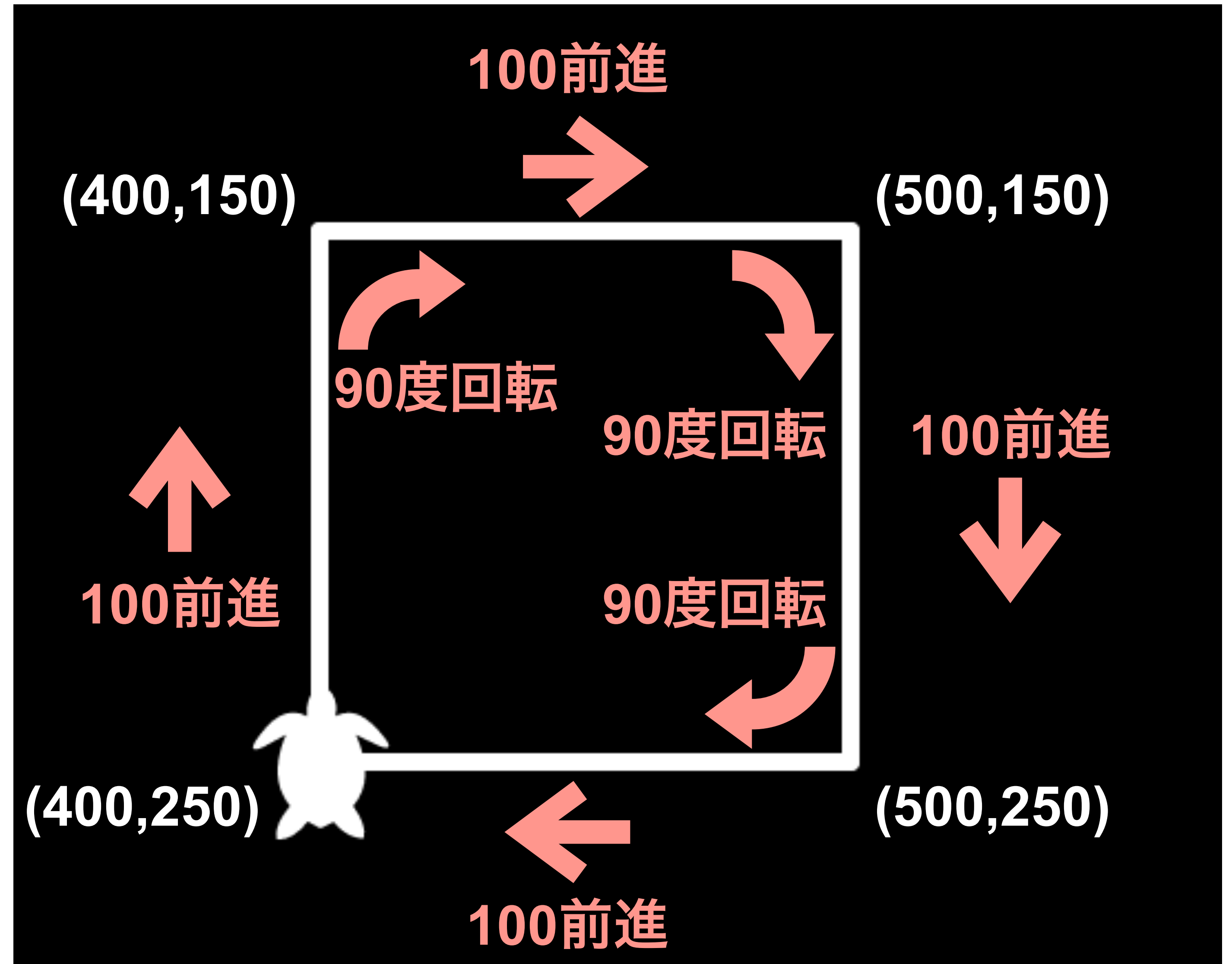


# タートルグラフィックス

亀を動かそう

- 1) 100前進
- 2) 90度回転
- 3) 100前進
- 4) 90度回転
- 5) 100前進
- 6) 90度回転
- 7) 100前進

→ 正方形が描画できる



# タートルグラフィックス

コマンドで書くと

forward(100)

right(90)

forward(100)

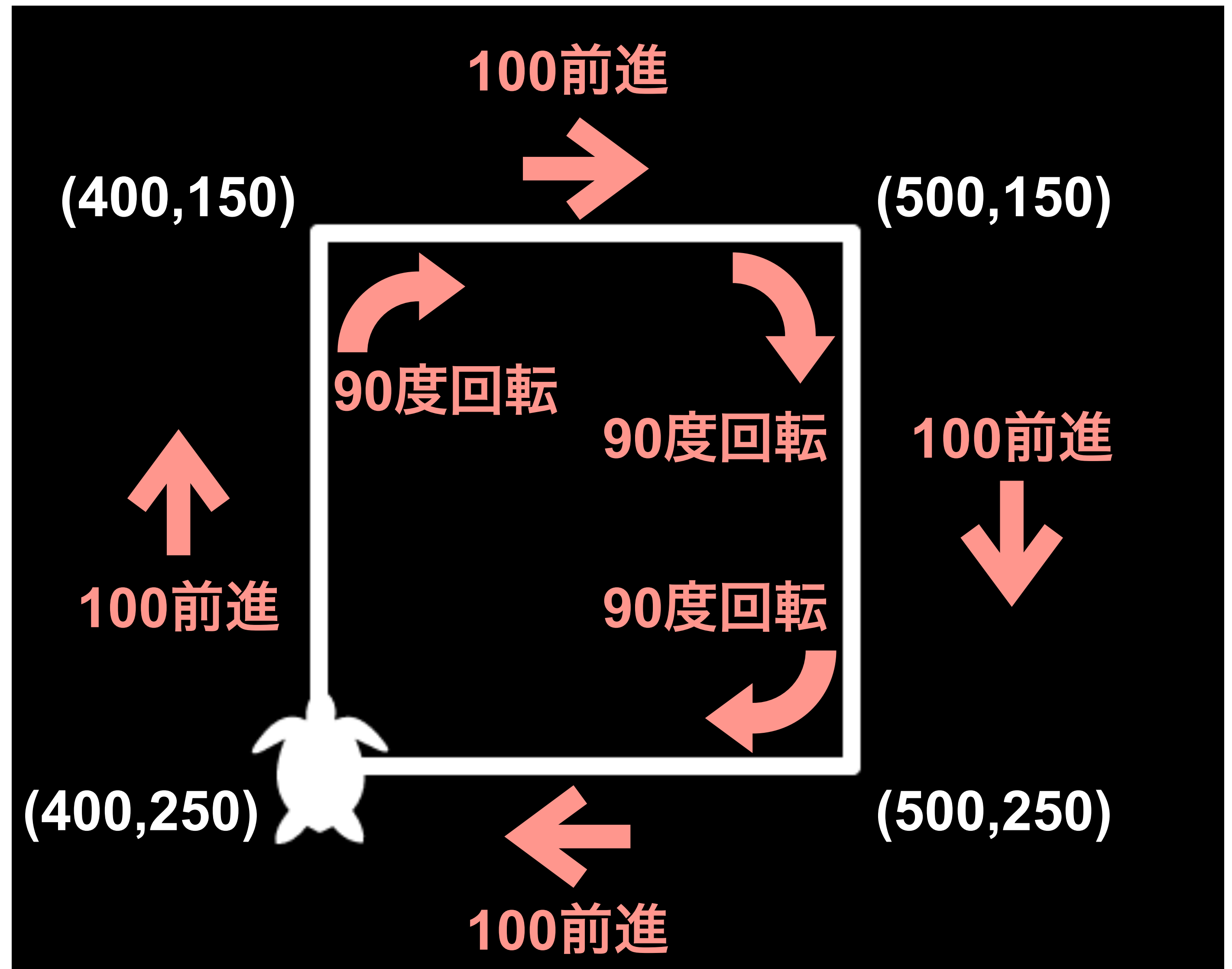
right(90)

forward(100)

right(90)

forward(100)

right(90)





# 同じ処理のくり返しはループでまわす

```
for i in range(4):  
    forward(100)  
    right(90)
```

range(start, stop[, step])

range(0,4,1)

> 0, 1, 2, 3

range(4)

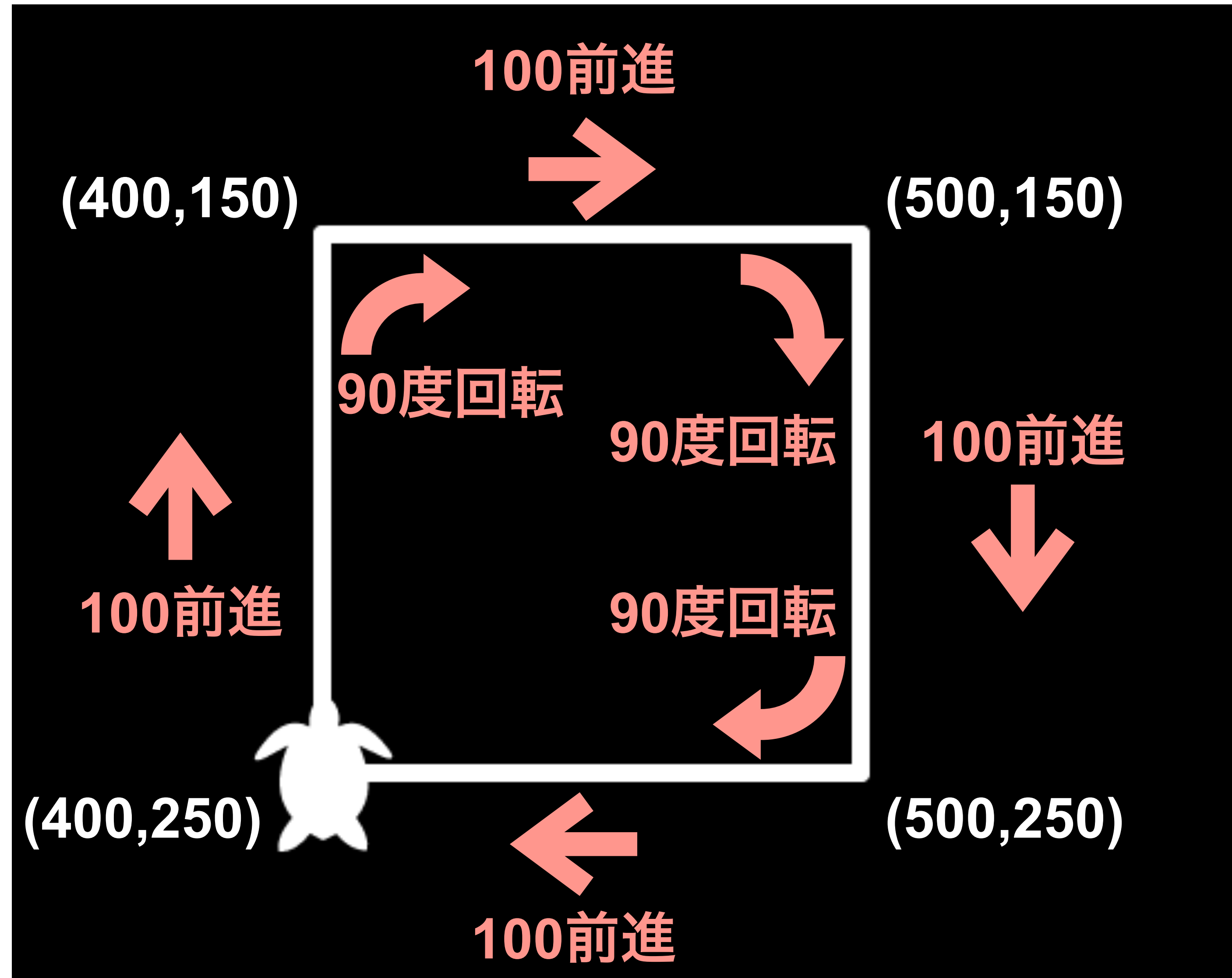
> 0, 1, 2, 3

range(0,4,2)

> 0, 2

range(4,0,-1)

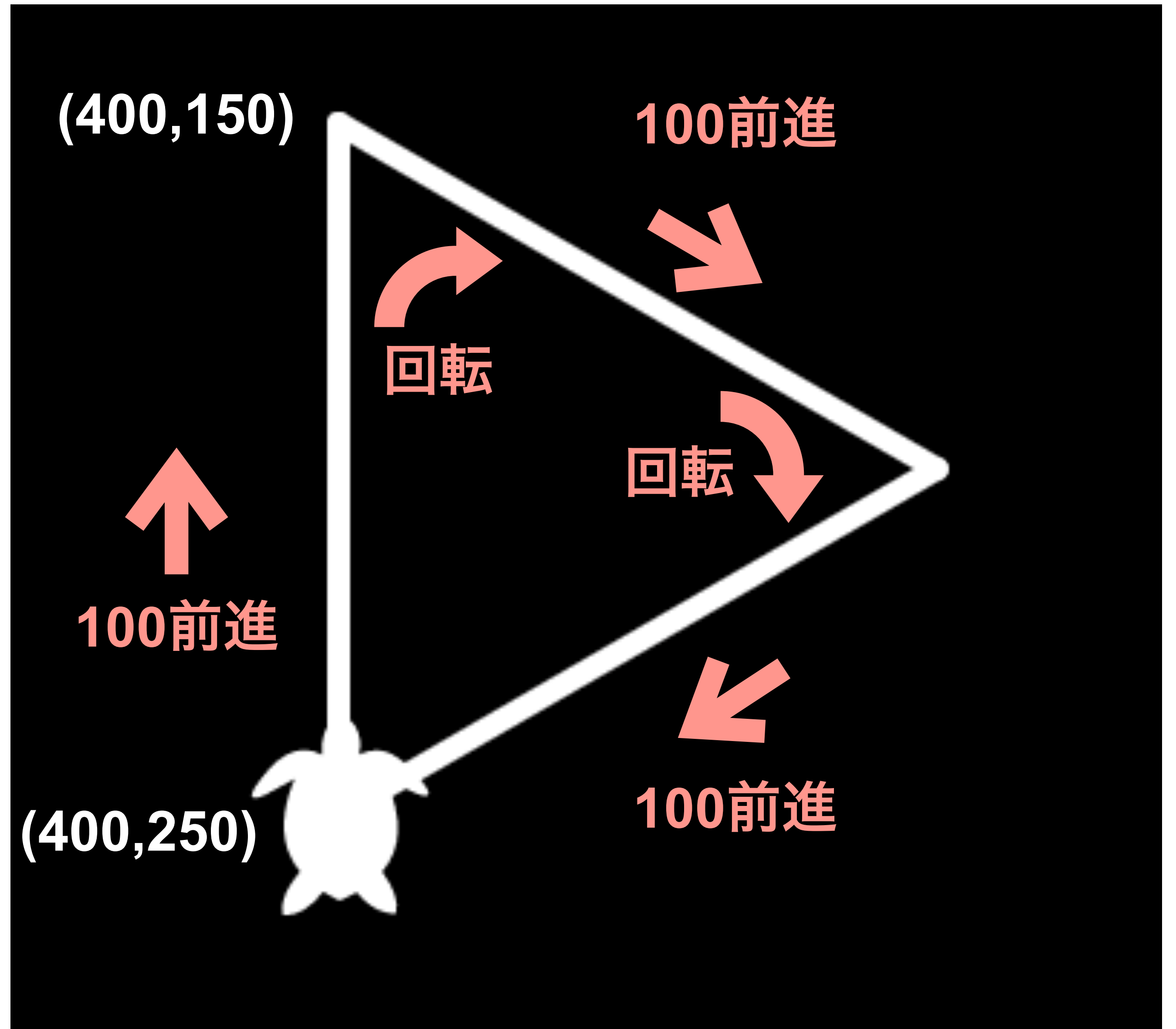
> 3, 2, 1, 0



# やってみよう：正三角形

??は自分で考えてね

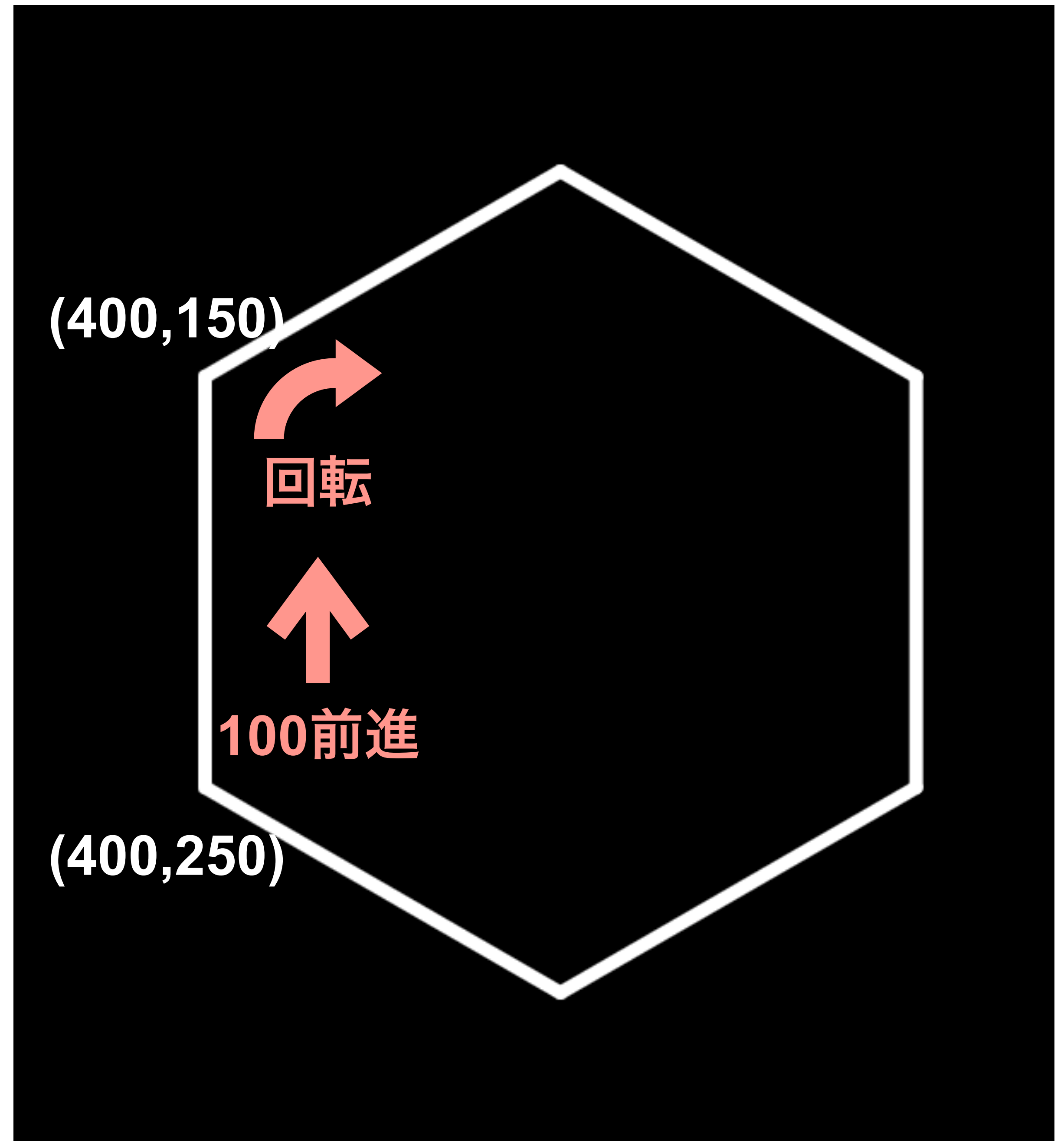
```
for i in range(?):  
    forward(100)  
    right(??)
```



# やってみよう：正六角形

??は自分で考えてね

```
for i in range(?):  
    forward(100)  
    right(??)
```





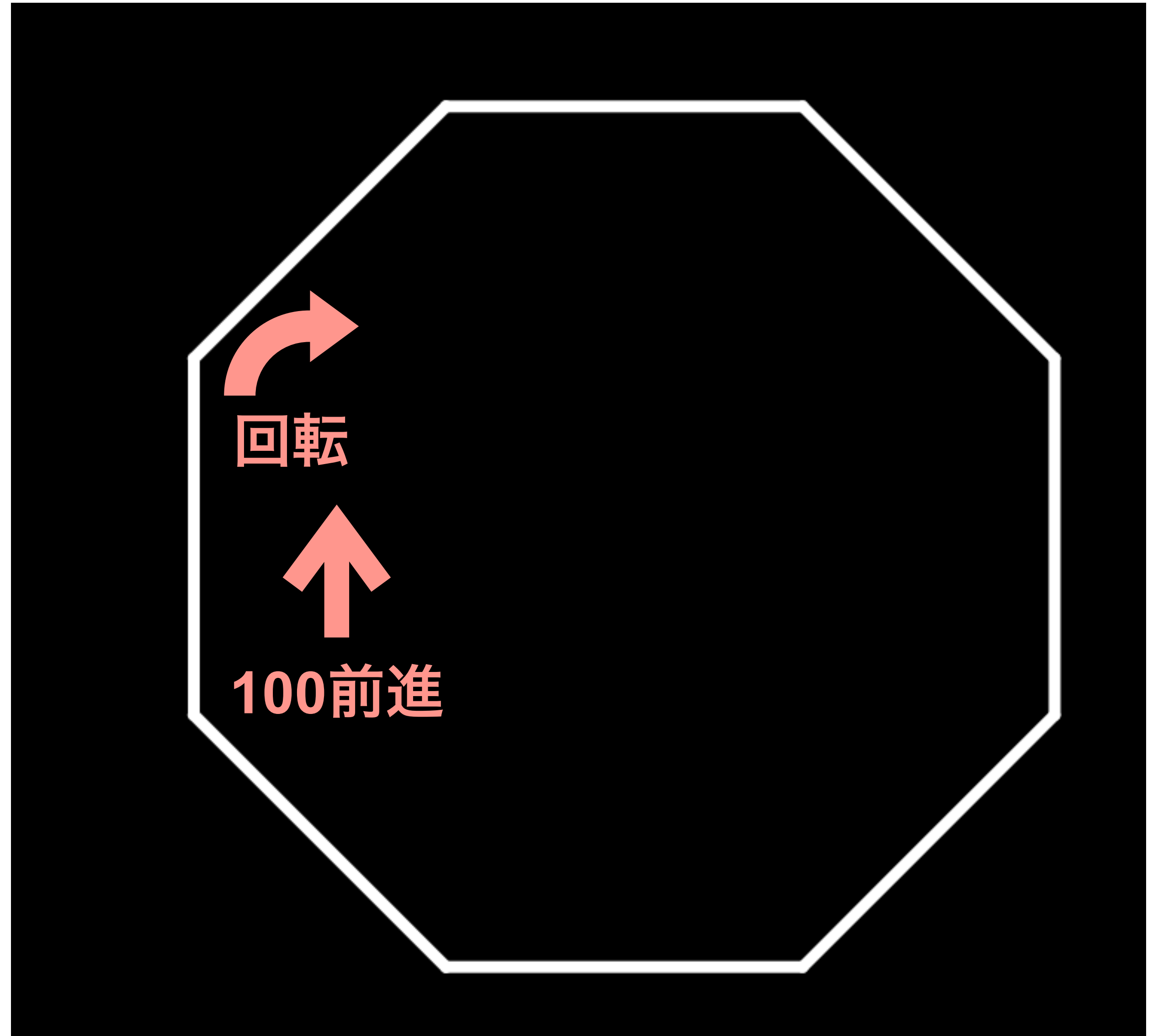
# 一般化：正n角形

??は自分で考えてね

length=100

angle = ??

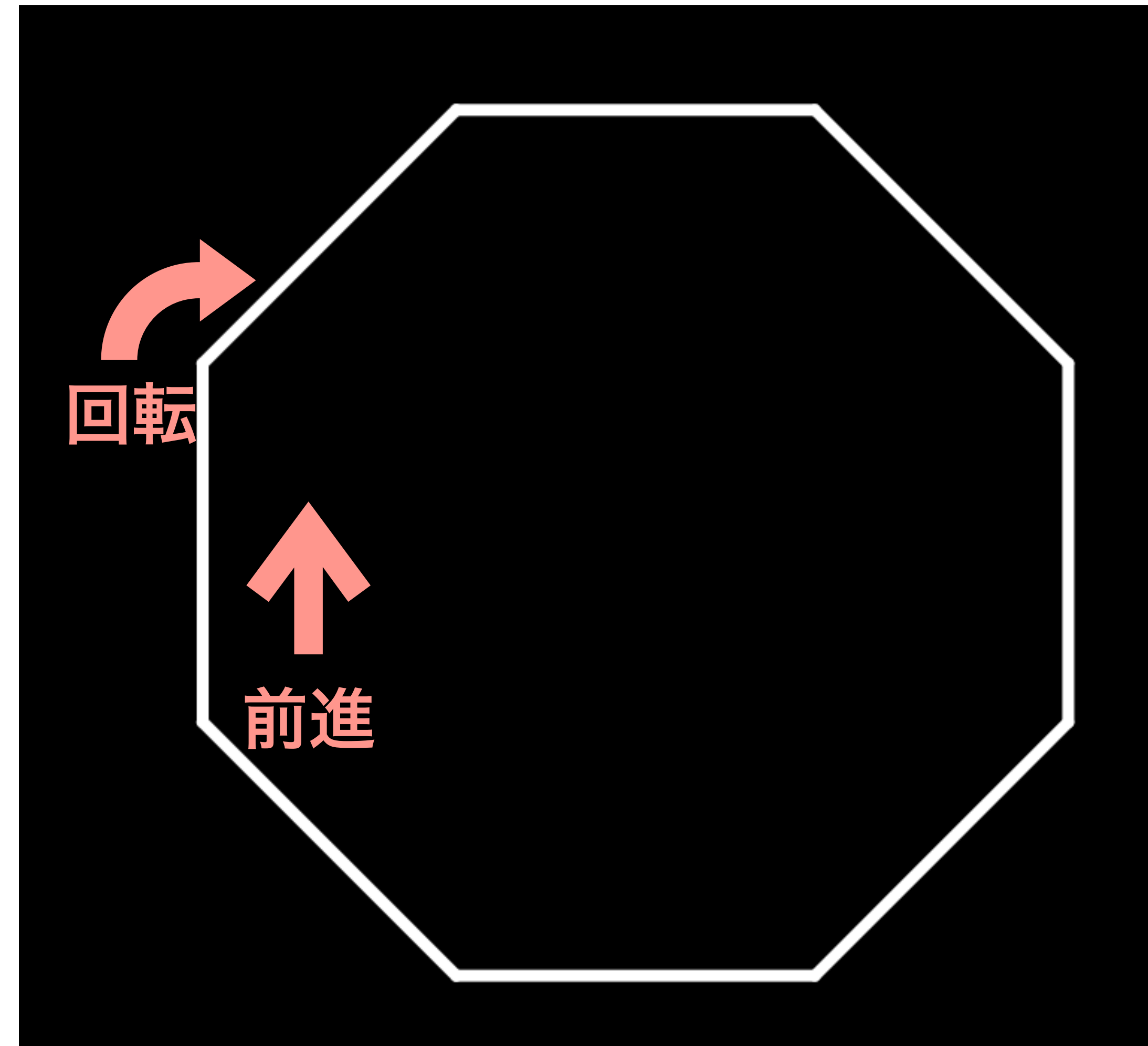
```
for i in range(?):  
    forward(length)  
    right(angle)
```



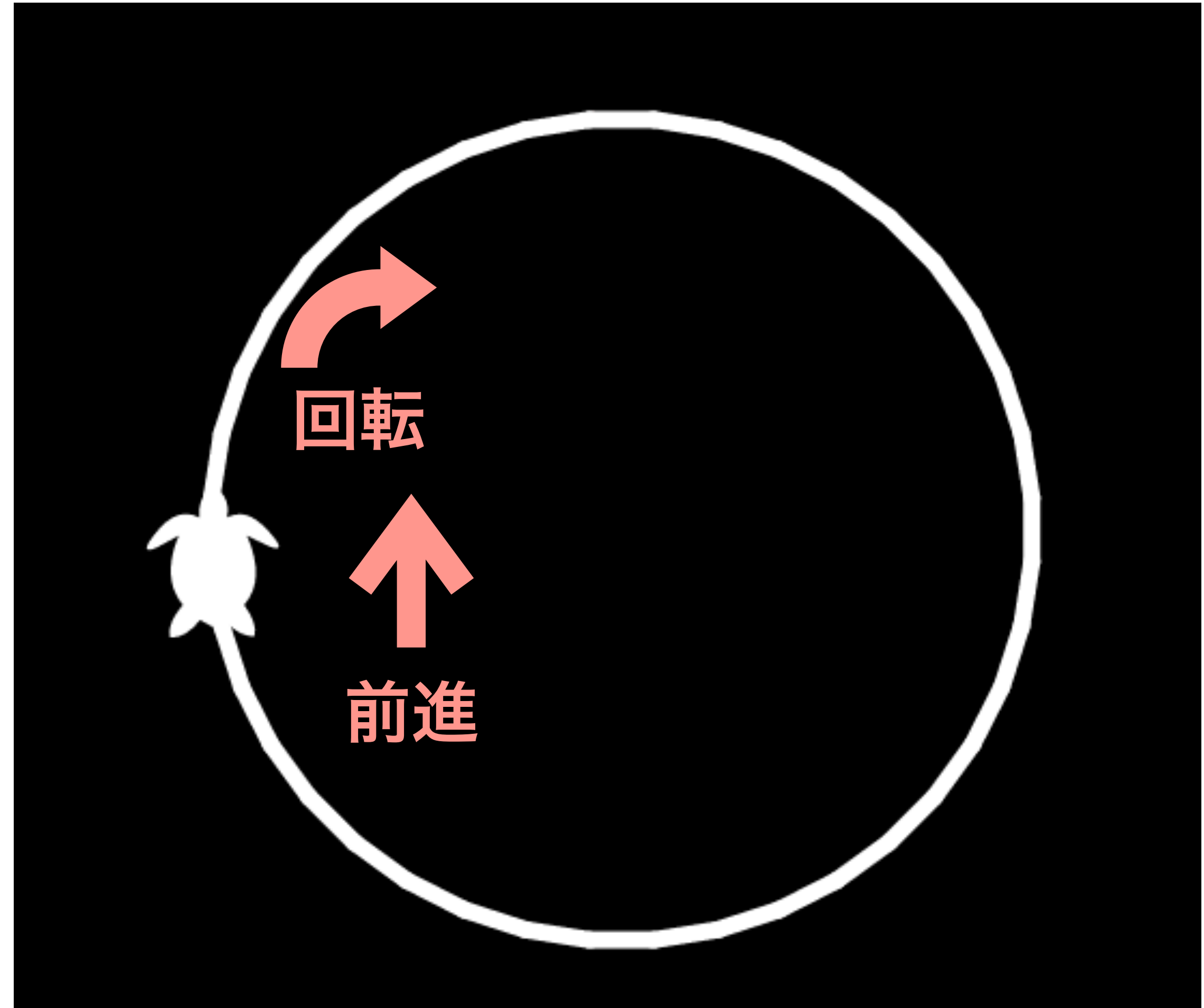
# 関数 polygon

??は自分で考えてね

```
def polygon(n, length):  
    length = ??  
    angle = ??  
    for i in range(??):  
        ??  
        ??
```



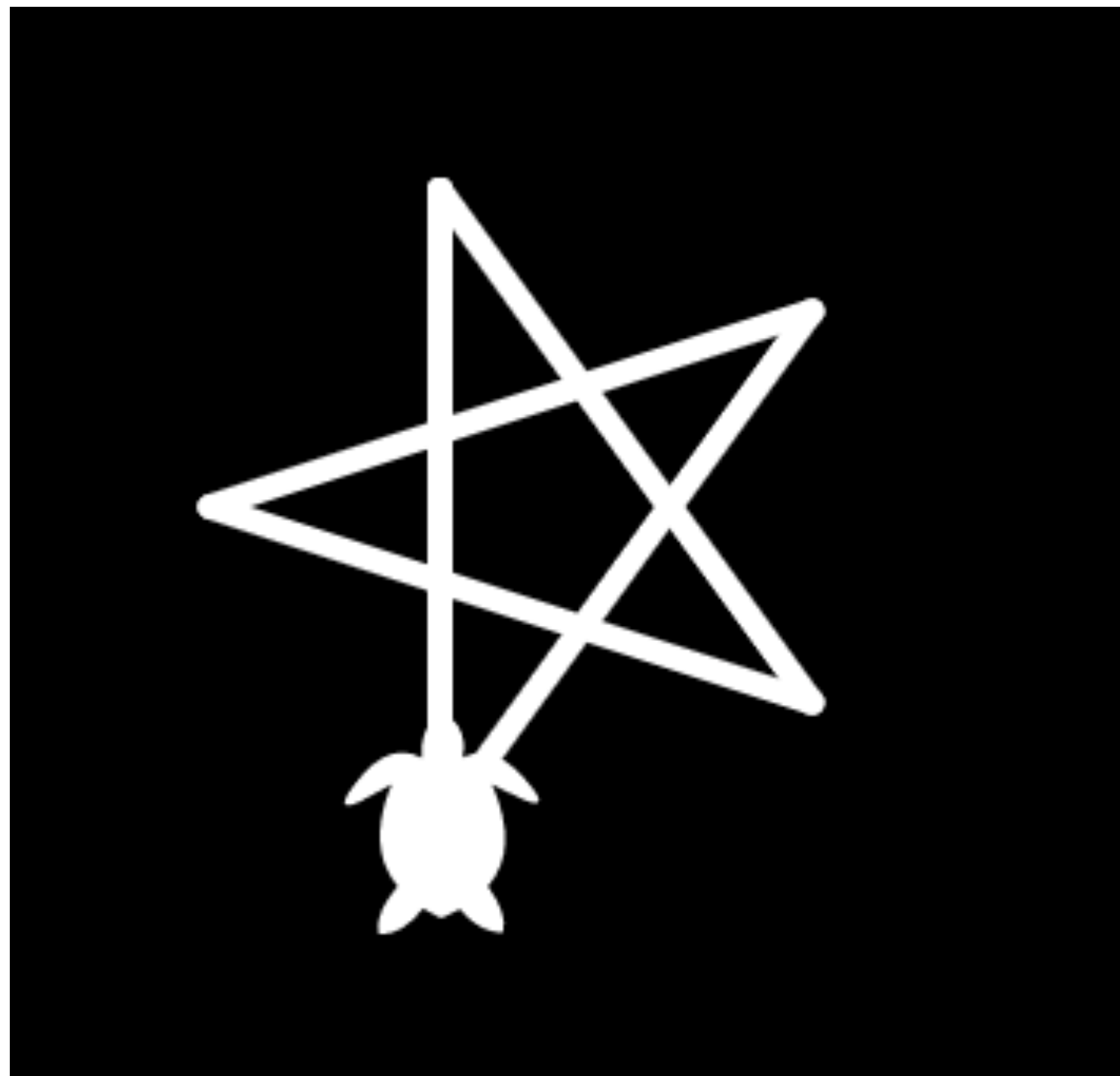
# 円はどう描く？





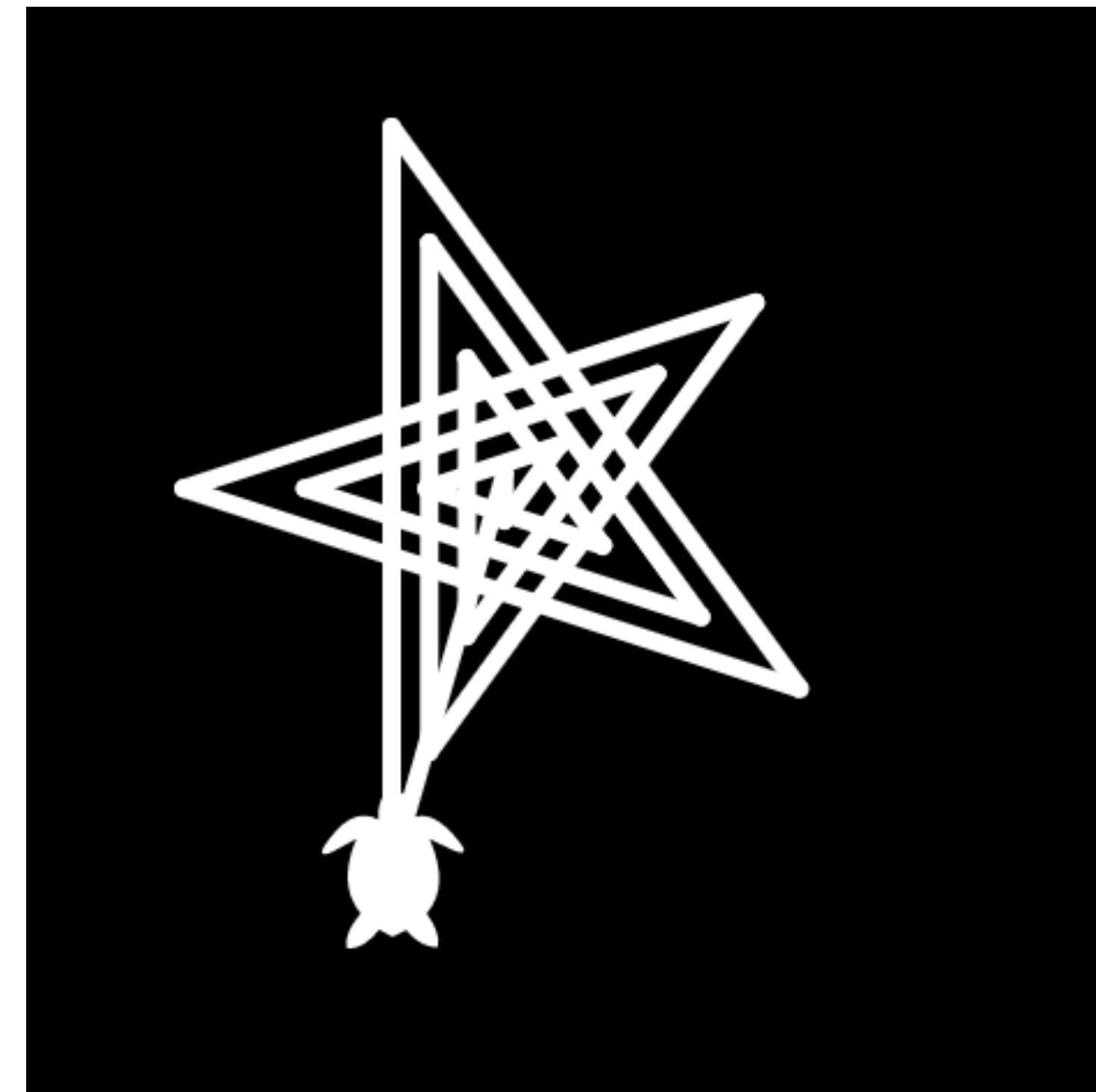
# 少し複雑な図形に挑戦

```
## Star
angle = ??
for i in range(?):
    forward(200)
    right(angle)
```



だんだん小さくなる

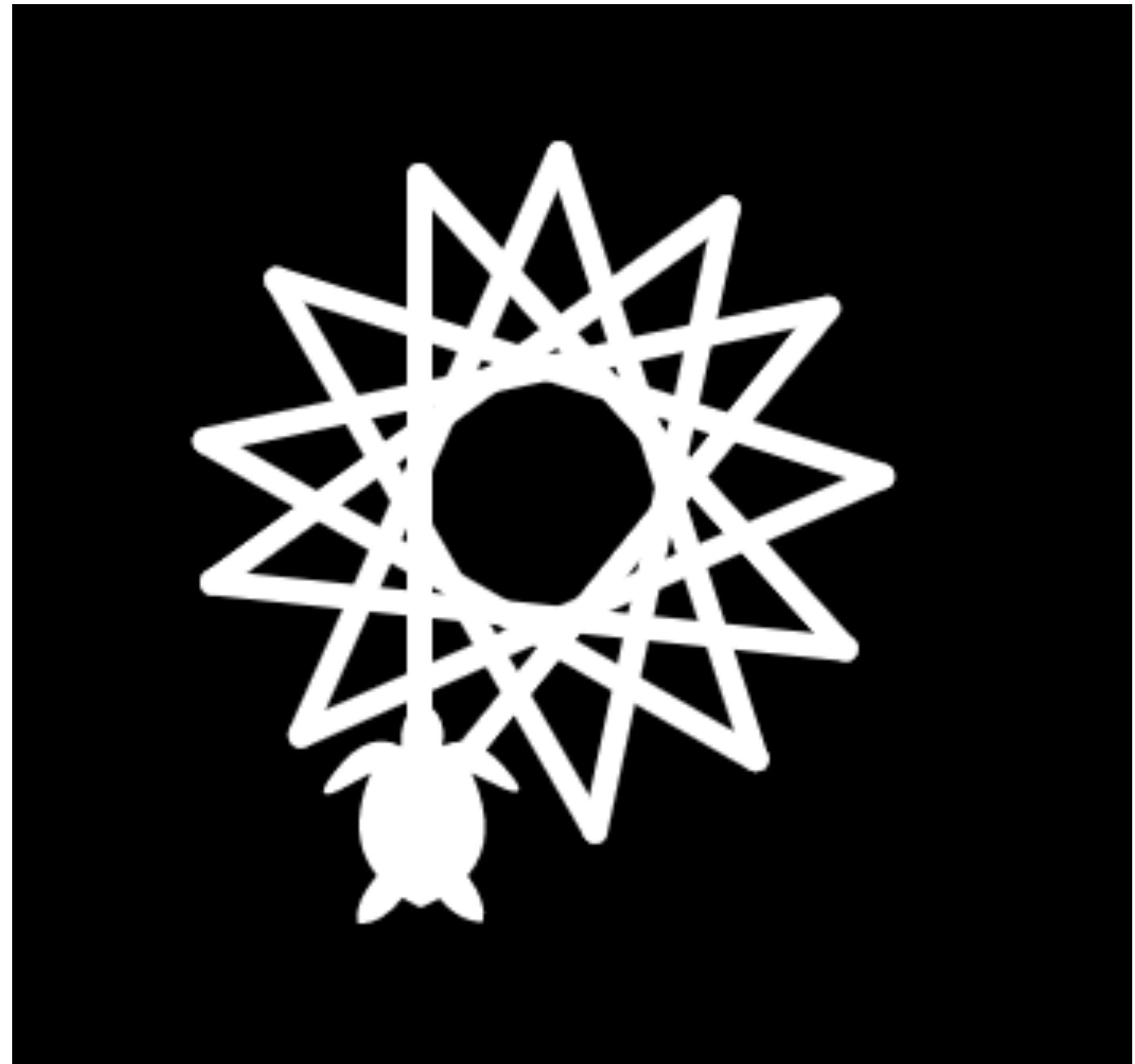
```
## Star2
angle = ??
for i in range(?):
    forward(200-i*10)
    right(angle)
```



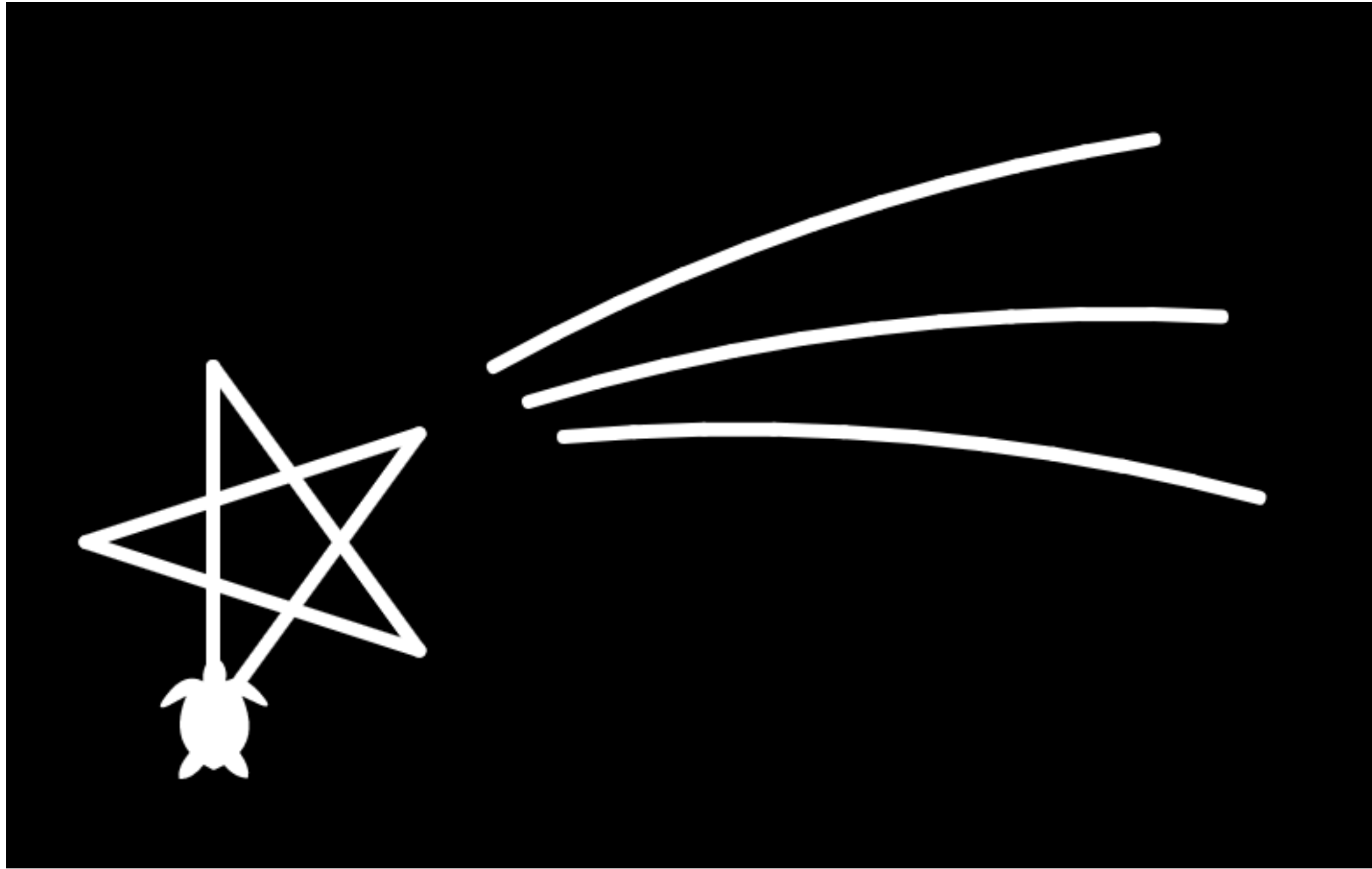
# 回りながら三角形をくり返し描く

```
## Triangle  
angle = ??  
for i in range(?):  
    forward(200)  
    right(angle + ??)
```

回転



# 流れ星は？



ペンアップ：描くのをやめる

亀の移動

ペンドاون：描きはじめる

```
## Star
angle = ??
for i in range(?):
    forward(100)
    right(angle)

right(70)
for i in range(3):
    penup()
    goto(480+i*10, 150+i*10)
    pendown()
    left(8)
    for i in range(10):
        forward(20)
        right(2)
    penup()
```



# 自分のイニシャルを描こう

