1 タートルグラフィクスの基本命令

学習目標とキーワード

学 習 目 標　タートルグラフィクスの基本命令を使い、図形を描けるようにしましょう。

キーワード　・ライブラリ　・モジュール　・初期設定　・直進　・回転　・ペン　・円弧　・塗り

タートルグラフィックスとは

タートルグラフィックスは、タートル（亀のペン）を命令で移動させ、移動した軌跡を画面に描画するプログラミング教育用ツールです。Pythonではライブラリからタートルグラフィックスのモジュールを読み込むことで使用できます。

【注意】このシートでは、標準の ColabTurtle モジュールを拡張した、ColabTurtlePlus を使用しています。

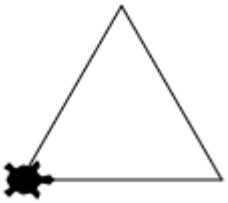
1　図形描画の基本

次のプログラムを実行して、タートルグラフィックスの基本を学びましょう。

プログラム

```
!pip install ColabTurtlePlus          #Claboratory用タートルグラフィックスモジュールの読み込み
from ColabTurtlePlus.Turtle import *  #最初の２行は１つのノートブックで最初に１回実行すればよい
clearscreen()                  #表示画面の初期化　800×600　背景白　ペン黒　位置原点　方向右
shape("turtle2")               #タートルの形状を設定する
for _ in range(3):             #以下の命令を３回繰り返す（変数は使わないので設定してない）
  forward(100)                 #100前進する
  left(120)                    #右に120度回転する
```

出力



亀は　100直進して120°右に回転します。これを３回繰り返すと正三角形の軌跡になります。亀の視点で考えてみましょう。

2　タートルグラフィックスの基本命令

タートルグラフィックスの主な基本命令を示します。最初はどのような命令があるか概要を知り、プログラミングする時に詳しく見てください。

タートルグラフィックスの基本命令　［ 命令　|　命令の省略形 ］

ライブラリ	!pip install ColabTurtlePlus	Pythonのライブラリサイトからインストールします。
モジュール	from ColabTurtlePlus.Turtle import *	モジュールをインポートします。
初期化	clearscreen( )	描画前に実行します。初期値は次のとおりです。 画面サイズ　800×600（中央座標 0,0 ）　ペン(亀）位置　中央 ペン向き　右　　速さ 6　ペンサイズ 1　ペン色　黒　背景色　白
移動	goto(x,y) 座標　x,y　に移動	setx(x)　x 座標だけ移動　　sety(y)　y座標だけ移動
向き	setheading(角度) | heading(角度) | face(角度)	初期設定の向きを０°として向きを設定
直進	forward(ピクセル) | fd(ピクセル)	ピクセル前進　backward(ピクセル) | bk(ピクセル) ピクセル後退
回転	right(角度) | rt(角度)	度右回転　　left(角度) | lt(角度)　度左回転
ペン状態	penup() | pu()　ペンを上げる	pendown() | pd()　ペンを下げる
描画速度	speed(速度)　速度は１～10 で10が最速	
アニメ省略	speed(0) にして、描画の最後に　done()　を記入するとアニメーションが省略され高速になる	
ペン表示	showturtle()　ペン（亀）を表示	hideturtle()　ペン(亀）を非表示
ペンサイズ	pensize(太さ) | width(太さ)	ペンの太さ(太さは１以上の整数)
ペン形状	shape(形状名)　ペン形状名	“classic”,”turtle”,”circle”,”arrow”,”triangle”,”square”,”ring”, “blank”,および”turtle2”
背景色	bgcolor(色)　背景色を指定	
ペン色	color(色)　ペン色を指定	塗り色　fillcolor(color)　塗り色を指定

つづく

タートルグラフィックスの基本命令　つづき

線と塗り色　color(色,色)　ペンの色、塗り色
色の指定方法　1 カラーコード　例 “#aa9933” “#fd4” “rgb(255,255,0)”
2 色名 例　“white”,“yellow”,“orange”,“red”,“green”,”blue”,”purple”,”grey”
その他色名140種類は次のサイトを参照　https://www.w3schools.com/colors/colors_names.asp

円弧　circle (半径, 範囲, ステップ) 半径が＋の場合円の中心はペンの左側にあり反時計回りに円を描きます。ーの場合円の中心はペンの右側にあり時計回りに円を描きます。範囲は弧の角度で 0 ～360の値です。省略した場合は正円を描きます。ステップは円を連続直線で描くときに使います。通常は省略します。

塗り　begin_fill()　～描画～　end_fill()
図形を塗る場合は、図形描画の前に begin_fill()　図形描画の後に end_fill()　を実行します。
複雑な図形の場合、予想している塗りの結果にならない場合があります。

参考 (2024.2確認)　https://pypi.org/project/ColabTurtlePlus/
https://larryriddle.agnesscott.org/ColabTurtlePlus/documentation2.html

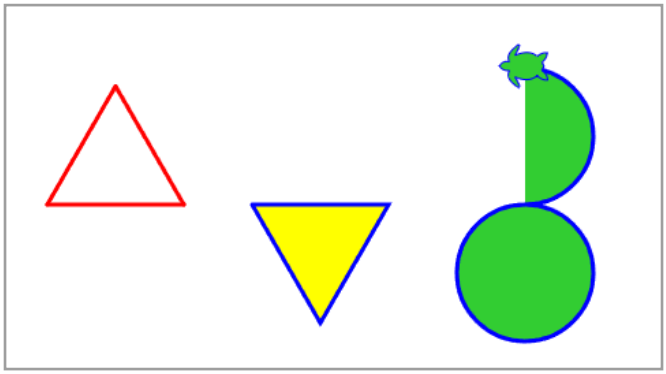
2　色々な図形の描画

次のプログラムを実行して、多角形、円弧、塗り、の基本を学びましょう。
また、数値や色を変えて、動作を確認してください。

プログラム

```
!pip install ColabTurtlePlus      #Claboratory用タートルグラフィックスモジュールの読み込み
from ColabTurtlePlus.Turtle import *  #最初の２行は１つのノートブックで最初に１回実行すればよい
clearscreen()                     #表示画面の初期化　800×600　背景白　ペン黒　位置原点　方向右
shape("turtle")                   #タートルの形状を設定する　初期値は亀
pensize(3)                        #ペンの太さを設定する
speed(7)                          #描画速度を設定する　1～10
penup();backward(200);pendown()   #ペンを上げて200後退してペンを下す（；を使って1行にしています）
color("red", "blue")              #線を赤色、塗りを青色にする
for _ in range(3):                #以下の命令を３回繰り返す（３角形を描きます）
  forward(100)                    #100前進する
  left(120)                       #左に120度回転する
penup();forward(150);pendown()    #ペンを上げて200前進してペンを下す（；を使って1行にしています）
begin_fill()                      #以下の図形を塗る
color("blue", "Yellow")           #線を青色、塗りを黄色にする
for _ in range(3):                #以下の命令を３回繰り返す（塗った３角形を描きます）
  forward(100)                    #100前進する
  right(120)                      #右に120度回転する
end_fill()                        #ここまでの図形を塗る
penup();forward(200);pendown()    #ペンを上げて200前進してペンを下す（；を使って1行にしています）
begin_fill()                      #以下の図形を塗る
color("blue", "LimeGreen")        #線を青色、塗をライムグリーンにする
circle(-50)                       #タートルの右側に半径50の円を描く（塗った円を描きます）
circle(50,180)                    #タートルの左側に半径50、180度の円弧を描く（塗った半円を描きます）
end_fill()                        #ここまでの図形を塗る
```

出力



2 タートルグラフィクスで正多角形を描こう

学習目標とキーワード

学 習 目 標　タートルグラフィクスで多角形を作図して、繰り返し制御のプログラムをマスターしよう。

キーワード　・for文による繰り返し　　・for文による入れ子の繰り返し（２重ループ）

1　正多角形の描画

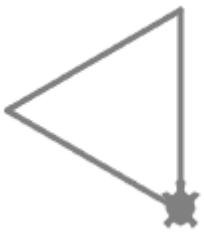
前章で説明したように、タートルグラフィックスでは、直進と回転を繰り返すことで正多角形を描くことができます。回転角度と繰り返し回数を変えて、様々な正多角形を描いてみましょう。

次のプログラムを入力して正三角形を描いてください。

プログラム

```
!pip install ColabTurtlePlus          #Claboratory用タートルグラフィックスモジュールの読み込み
from ColabTurtlePlus.Turtle import *  #最初の２行は１つのノートブックで最初に１回実行すればよい
clearscreen()                 #表示画面の初期化　800×600　背景白　ペン黒　位置原点　方向右
speed(8)                      #描画速度8
shape("turtle2")              #タートル形状亀２
face(90)                      #初期状態の向きを上方向にする
bgcolor("white")              #背景白色
color("grey")                 #描画色灰色
pensize(3)                    #ペン太さ３
for _ in range(3):            #以下の命令を３回繰り返す（変数は使わないので設定してない）
  forward(100)                #100前進する
  left(120)                   #左に120度回転する
```

出力



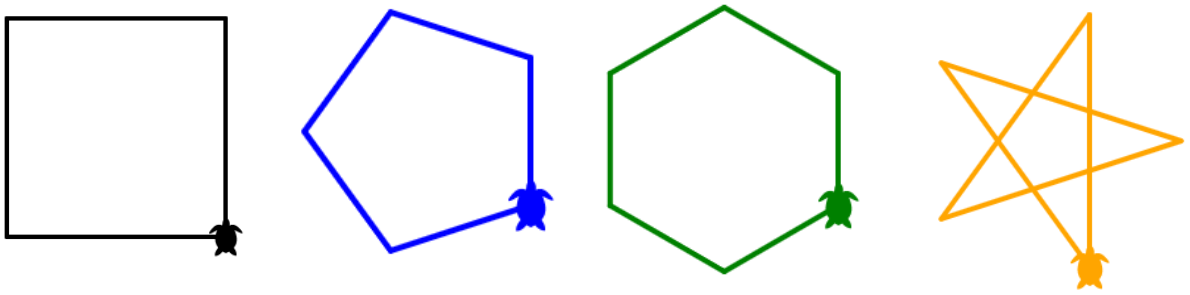
亀は初期状態で上方向を向き、100直進して120°右に回転します。これを３回繰り返すと正三角形の軌跡になります。亀の視点で考えてみましょう。

課題 2-1

・タートルグラフィックスで次の図形を描画してください。

【アドバイス　一片の長さを調節して図形全体を描画してください。色は好みの色に変えてください。】

① 正方形　② 五角形　③ 正六角形　④ 星



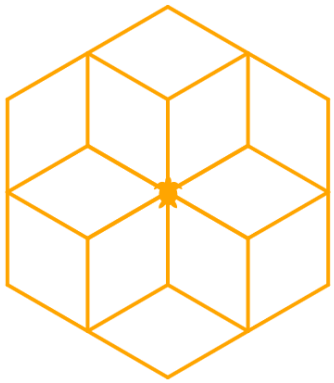
2　図形の繰り返し描画

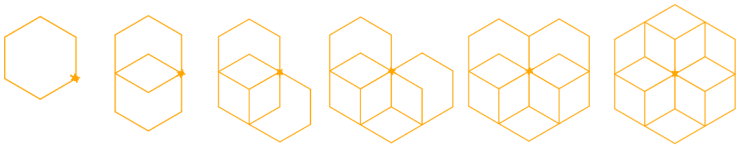
正六角形を作図するとき、開始方向を60°ずつ回転して６個描画すると、次のような図形が描けます。

プログラム

```
!pip install ColabTurtlePlus          #Claboratory用タートルグラフィックスモジュールの読み込み
from ColabTurtlePlus.Turtle import *  #最初の２行は１つのノートブックで最初に１回実行すればよい
clearscreen()          #描画エリアの初期化
speed(8)               #描画速度の指定
shape("turtle2")       #タートル形
face(90)               #初期状態の向きを上方向にする
bgcolor("white")       #背景白色
color("black")         #描画色灰色
pensize(3)             #ペン太さ３
bgcolor("white")       #背景色の指定
color("orange")        #線の色指定
for j in range(6):     #６０°ずつ回転した正六角形の描画を６回繰り返す
  for i in range(6):   #線の描画を６回繰り返して正六角形を描く
    forward(100)       #１００進む
    left(60)           # 左に６０°回転
  left(60)             #描画開始の方向を６０°回転
```

出力





1　2　3　4　5　6

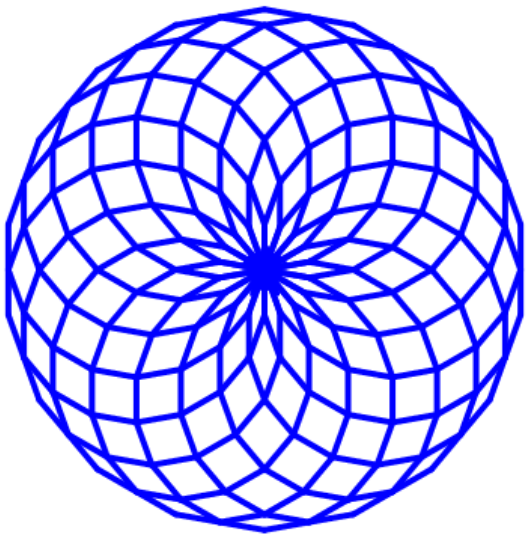
内側の For i の繰り返し（ループ）では、最初に１のような正六角形を描きます。 外側の For j の繰り返しは、正六角形を描いた後に、left(60)で亀の方向を60°左に向けます。正六角形は、先に書いた正六角形と一部が重なって描画されます。これを2,3,4,5,6のように回転しながら描画していきます。

このような繰り返し構造を「入れ子」「2重ループ」などと言います。命令は内側のループが先に実行されます。

課題 2-2

・タートルグラフィックスで次の図形を描画してください。

【アドバイス　基本の図形は何角形ですか？　一片の長さを調節して図形全体を描画してください。色は好みの色に変えてください。】



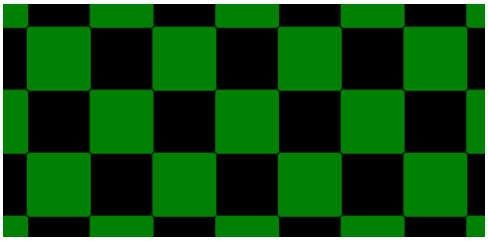
3 伝統的な連続文様を描いてみよう（市松文様）

学習目標とキーワード

学 習 目 標　市松文様の描画をとおして、システム開発のモデルを理解しましょう。

キーワード　・システム開発　・ウォーターフォールモデル　・要件定義　・基本設計　詳細設計
・テスト　・アルゴリズムの工夫　・関数

1　市松文様を描くプログラムを考えてみましょう。

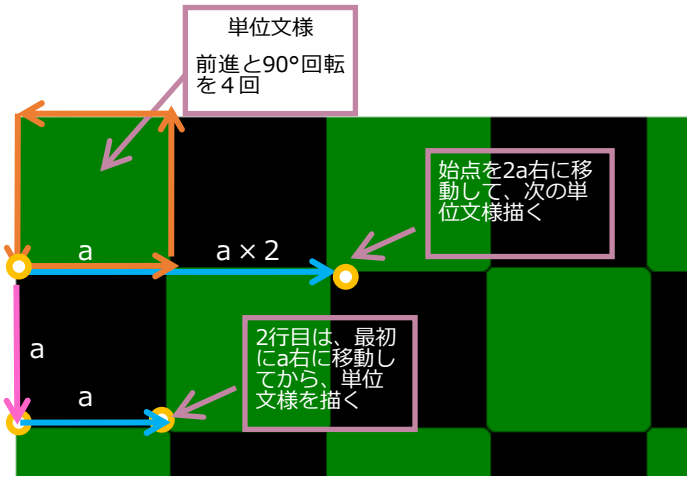


ここでは、ウォーターフォールモデルと言うシステム開発手順を参考にて、描画プログラムを作成していきます。

手順１　要件定義（どういう文様を描くか）

この市松文様は、緑の正方形と黒の正方形がたてよこに交互に並んだ文様です。

ここでは、黒い地に緑の正方形が並んでいると考えます。正方形の大きさと数、文様の大きさは指定が無いため、任意の値にします。



手順２　基本設計（どのような構成か）

この市松文様を描くプログラムは、

1 地の色やタートルの開始位置を設定する
　初期設定
2 正方形（単位文様）を描くプログラム
3 正方形を並べるプログラム
　の構成とします。

手順３　詳細設計（各構成はどのような動作か）

1 初期設定

地の色、タートルの初期位置と向き　正方形の大きさと数を設定します。

2 正方形（単位文様）の描画プログラム

一辺の距離aを直進して90°左に回転を４回繰り返して正方形を描きます。また、この図形を塗ります。

何回も使うプログラムなので関数にします。

3 正方形（単位文様）を並べるプログラム

初期位置に正方形を描き、右に辺の２倍2a移動します。その位置でまた正方形を描き横に2a移動すます。これを繰り返して１行目を描きます。

１行目の原点に戻った後に、下へa移動します。

２行目では最初に右にa移動します。そこから正方形を描き、右に2a移動する動作を繰り返します。

２行名の原点に戻った後に、下へ移動し３行目の原点とします。

以後、１行名、２行目の描画を繰り返します。

手順４　実装（プログラミング）

詳細設計を基に、プログラムを作成します。

手順５　テスト（動作を確認する）

正方形は描けるか（単体テスト）、並べることはできるか（統合テスト）テストして修正します。

手順６　運用・保守（完成と改善）

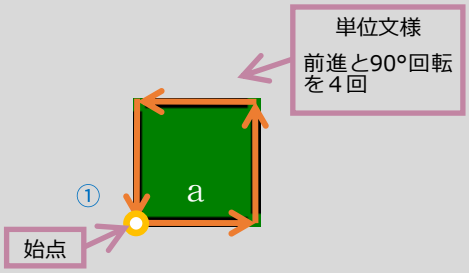
完成した描画をリリースします。また、改善する余地があれば改善します。

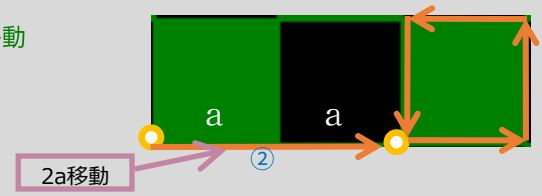
2　市松文様のプログラムを実行してみましょう。

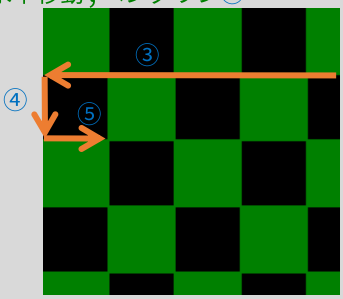
プログラム例

```
### 市松模様 ###
!pip install ColabTurtlePlus       #Claboratory用タートルグラフィックスモジュールの読み込み
from ColabTurtlePlus.Turtle import *    #最初の２行は、１つのノートブックで最初に１回実行すればよい

#単位文様（正方形）の関数
def 正方形(a):                  #関数の定義　引数aは一片の長さ①
  begin_fill()                  #塗のはじめ
  for _ in range(4):            #4回繰り返し
    forward(a)                  #距離a前進
    left(90)                    #90°左回転
  end_fill()                    #ここまでの図形を塗る
#初期設定
clearscreen()                   #ペン位置：中央　向き：左
speed(10)                       #速度10　最高速はspeed(0)　最後にdone()
shape("turtle2")                #ペン形状
bgcolor("black")                #背景黒色
color("green")                  #柄色みどり
pensize(3)                      #ペン太さ３
開始X座標=-400                  #初期ペンX座標-400
開始Y座標=200                   #初期ペンY座標　200
辺長さ=100                      #一辺の長さ　100
列数=5                          #文様列数　（緑正方形　黒正方形）×5
行数=6                          #文様行数　6
pu();goto(開始X座標,開始Y座標);pd()      #初期座標へ移動
#単位文様を繰り返し描画
for j in range(行数):           #行の繰り返し
  for i in range(列数):         #列の繰り返し
    正方形(辺長さ)              #正方形の描画
    pu();fd(辺長さ*2);pd()#ペンアップ;次の描画開始位置へ辺の2倍水平移動;ペンダウン②
   #次の行への移動
  pu()                          #ペンアップ
  setx(開始X座標)               #開始X座標へ移動③
  rt(90)                        #下へ向く
  fd(辺長さ)                    #下へa垂直移動④
  lt(90)                        #左へ向く
  fd(辺長さ*((j+1)%2))          #jが0,2,4‥偶数のときa水平移動⑤
  pd()                          #ペンダウン
#done()                         #speed(0)の時#を取ってdone()にする
```







テクニック（交互に変わる値を作る）

　各行の正方形の書き出しは、1行目は左端、２行目は1個分右、3行名は左端、４行名は１個分右・・と、奇数行では左端、偶数行では１個分右になっています。

⑤では、行の値Ｊが0 １ ２ ３ ・・・と変化するとき(j+1)で １ ２ ３ ４・・・を作り、それを２で割った余りを計算します。余りの値は １ ０ １ ０ ・・・と交互の値になり、これに一辺の長さを掛けることにより、2行目の始点は左端から一辺の長さ、３行目は左端から距離０　・・・と、行ごとに交互に変わります。

課題 3-1

・タートルを表示して描画の手順を確認しましょう。

・speed(0) と　done()　を記入して、アニメーションを省略し、高速に描画してみましょう。

・辺の長さや文様の列、行の数を変えて描画してみましょう。

4 伝統的な連続文様を描いてみよう（鱗文様）

学習目標とキーワード

学 習 目 標　市松文様のプログラムを参考にして、鱗（うろこ）文様のプログラムを作成しましょう。

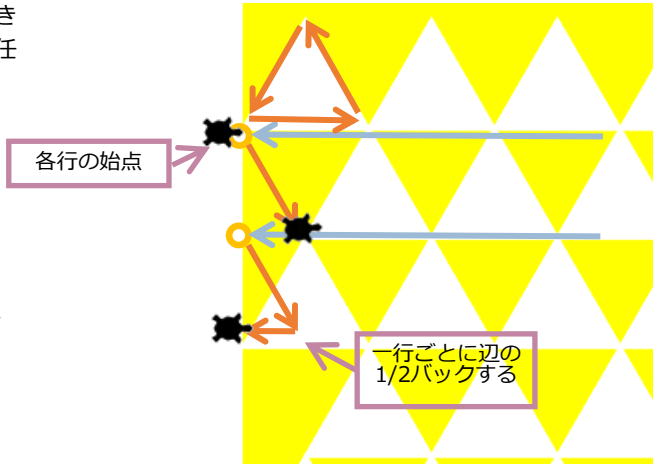
キーワード　・システム開発　・ウォーターフォールモデル　・要件定義　・基本設計　詳細設計
・テスト　・アルゴリズムの工夫　・関数

1　鱗文様を描くプログラムを考えてみましょう。

ここでは、市松文様のプログラム開発手順を参考にて、描画プログラムを作成していきます。

手順１　要件定義（どういう文様を描くか）

この鱗文様は、白と黄色の正三角形が並んだ文様です。ここでは、黄色い地に白の正三角形が並んでいると考えます。正三角形の大きさと数、文様の大きさは指定が無いため、任意の値にします。



手順２　基本設計（どのような構成か）

この鱗文様を描くプログラムは、

1 地の色やタートルの開始位置を設定する
　初期設定
2 正三角形（単位文様）を描くプログラム
3 正三角形を並べるプログラム
　の構成とします。

手順３　詳細設計（各構成はどのような動作か）

1 初期設定

地の色、タートルの初期位置と向き　正方形の大きさと数を設定します。

2 正三角形（単位文様）の描画プログラム
一辺の距離aを直進して120°左に回転を3回繰り返して正三角形を描きます。また、この図形を塗ります。

何回も使うプログラムなので関数にします。

3正三角形（単位文様）を並べるプログラム

初期位置に正三角形を描き、右に辺a移動します。その位置でまた正三角形を描き横にa移動すます。これを繰り返して１行目を描きます。

１行目の原点に戻った後に、60°右斜め下へa移動します。

２行目ではここを始点として三角形を並べて描きます。

３行名では原点に戻った後に、60°右斜め下へa移動し、1/2a バックします。

以後、１行名、２行目の描画を繰り返します。

手順４　実装（プログラミング）

詳細設計を基に、プログラムを作成します。

手順５　テスト（動作を確認する）

正三角形は描けるか（単体テスト）、並べることはできるか（統合テスト）テストして修正します。

手順６　運用・保守（完成と改善）

完成した描画をリリースします。また、改善する余地があれば改善します。

2　鱗文様のプログラムを実行してみましょう。

課題 4-1

・プログラム中の　ア　から　エ　に適切な字句を入れて、プログラムを完成させましょう。

・完成したら動作を確認しましょう。

・辺の長さや文様の列、行の数を変えて描画してみましょう。

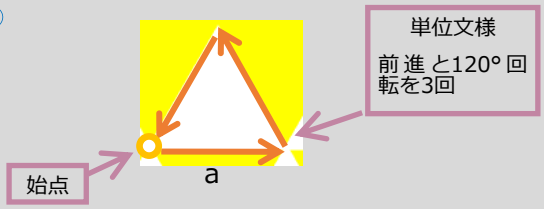
・speed(0) と　done()　を記入して、高速に描画してみましょう。

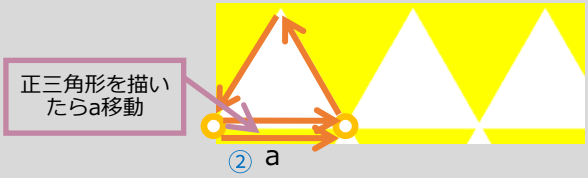
プログラム例

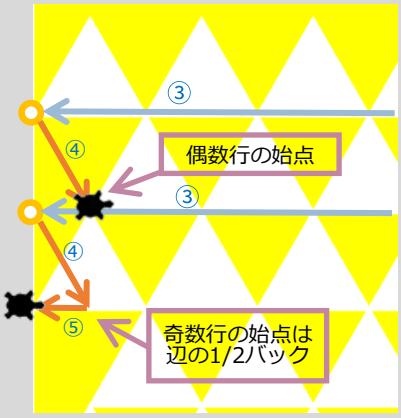
```
### 鱗模様 ###
!pip install ColabTurtlePlus       #Claboratory用タートルグラフィックスモジュールの読み込み
from ColabTurtlePlus.Turtle import *    #最初の２行は、１つのノートブックで最初に１回実行すればよい

#単位文様（正三角形）の関数
def 三角形(a):                  #関数の定義　引数aは一片の長さ①
  begin_fill()                  #塗のはじめ
  for _ in range(ア):           #ア回繰り返し
    forward(a)                  #距離a前進
    left(イ)                    #イ°左回転
  end_fill()                    #ここまでの図形を塗る
#初期設定
clearscreen()                   #ペン位置：中央　向き：左
speed(10)                       #速度10　最高速はspeed(0)　最後にdone()
shape("turtle2")                #ペン形状
bgcolor("yellow")               #背景黄色
color("white")                  #柄白色
pensize(3)                      #ペン太さ３
開始X座標=-400                  #初期ペンX座標-400
開始Y座標=200                   #初期ペンY座標　200
辺長さ=100                      #一辺の長さ　100
列数=10                         #文様列数　（正三角形)×10
行数=8                          #文様行数　6
pu();goto(開始X座標,開始Y座標);pd() #初期座標へ移動
#単位文様を繰り返し描画
for j in range(行数):           #行の繰り返し
  for i in range(列数):         #列の繰り返し
    三角形(辺長さ)              #正三角形の描画
    pu();fd(辺長さ);pd()        #ペンアップ;次の描画開始位置へ一辺長さ水平移動;ペンダウン②

  pu()                          #ペンアップ
  setx(開始X座標)               #開始X座標へ移動③
  rt(ウ)                        #ウ°右斜め下へ向く
  fd(辺長さ)                    #斜め下へa移動④
  lt(エ)                        #エ°左回転して左を向く
  bk(辺長さ/2*((j)%2))          #jが1,3,5‥奇数のときa/2水平にバック移動⑤
  pd()                          #ペンダウン
#done()                         #speed(0)の時#を取ってdone()にする
```







テクニック（図形の高さ）

　この文様の1行の高さは、正三角形の高さです。次の行へ移動するとき垂直に移動するためには、辺の長さ×√3/2あるいは、辺の長さ×cos30°　の計算が必要です。このように計算で高さを求めることもできますが、ここでは計算を避けて、斜め移動で必要な垂直の距離を求めています。

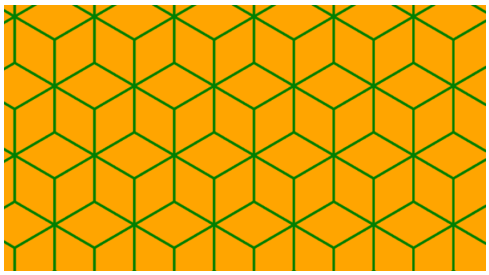
5 伝統的な連続文様を描いてみよう（毘沙門亀甲文様）

学習目標とキーワード

学 習 目 標　アルゴリズムを工夫して毘沙門亀甲文様を作成しよう。

キーワード　・システム開発　・ウォーターフォールモデル　・要件定義　・基本設計　詳細設計
・テスト　・アルゴリズムの工夫　・関数

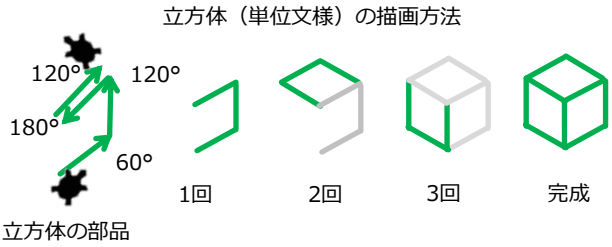
1　毘沙門亀甲文様を描くプログラムを考えてみましょう。



これまでのプログラム開発手順を参考にて、描画プログラムを作成していきます。

手順１　要件定義（どういう文様を描くか）

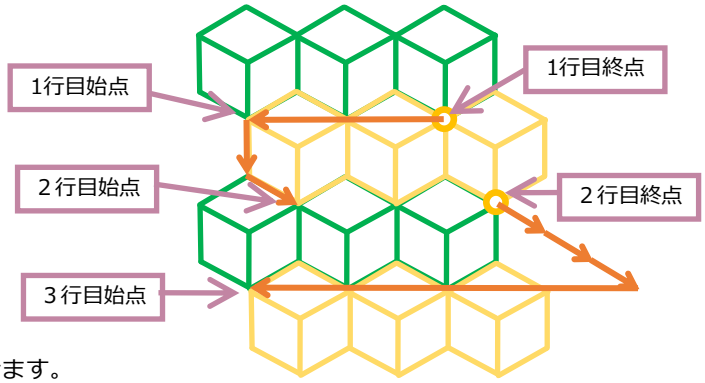
毘沙門亀甲文様は、六角形（亀甲）の組み合わせですが、見方によってはひし形の組わせや立方体の組み合わせ、さらにそれらの組み合わせに見えます。ここでは、立方体の組み合わせとして文様を描いてみます。



手順２　基本設計（どのような構成か）

毘沙門亀甲文様を描くプログラムは、

1 地の色やタートルの開始位置を設定する
　初期設定
2 立方体（単位文様）を描くプログラム
3 立方体を並べるプログラム
　の構成とします。



手順３　詳細設計（各構成はどのような動作か）

1 初期設定

地の色、タートルの初期位置と向き

立方体の大きさと数を設定します。

2 立方体（単位文様）の描画プログラム

立方体の3辺を一組として描き、３つ組み合わせます。

何回も使うプログラムなので関数にします。

３立方体（単位文様）を並べるプログラム

１行目を並び終えたら、始点に戻ります。そこから、下へ一辺分、左60°方向へ一辺移動したところが２行目の始点になります。

２行目を並び終えたら、３行目の始点のY座標を求めるために、60°方向へ3辺分移動し、そこから原点のX座標に戻り３行目の始点とします。

この１行目と２行目の描画と3行目始点への移動を一組として行を繰り返します。

手順４　実装（プログラミング）

詳細設計を基に、プログラムを作成します。

手順５　テスト（動作を確認する）

立方体は描けるか（単体テスト）、並べることはできるか（統合テスト）テストして修正します。

手順６　運用・保守（完成と改善）

完成した描画をリリースします。また、改善する余地があれば改善します。

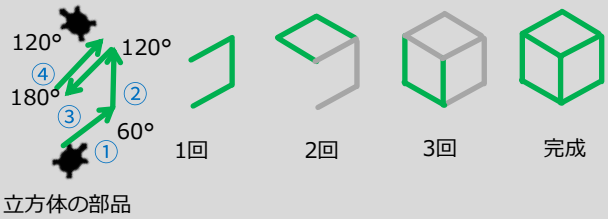
2　毘沙門亀甲文様のプログラムを実行してみましょう。

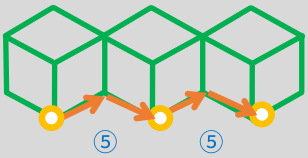
課題 5-1

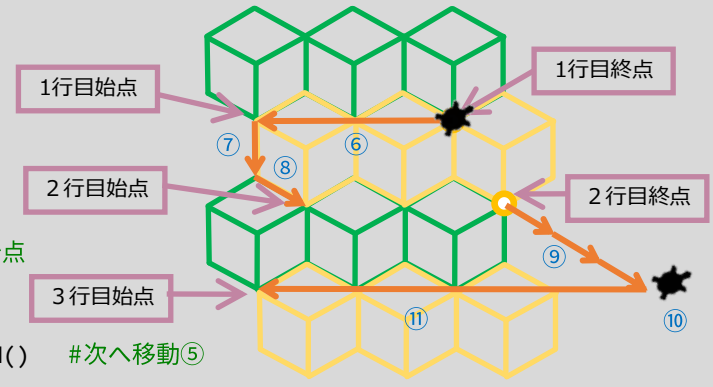
・プログラム中の　ア　から　エ　に適切な字句を入れて、プログラムを完成させましょう。
・完成したら動作を確認しましょう。
・辺の長さや文様の列、行の数を変えて描画してみましょう。
・speed(0) と　done()　を記入して、高速に描画してみましょう。

プログラム例

```
### 毘沙門亀甲文様 ###
!pip install ColabTurtlePlus       #Claboratory用タートルグラフィックスモジュールの読み込み
from ColabTurtlePlus.Turtle import *     #最初の２行は、１つのノートブックで最初に１回実行すればよい
#単位文様（立方体）の関数
def 立方体(a):                  #関数の定義　引数aは一片の長さ
  for _ in range(3):            #部品の描画3回繰り返し
    fd(a)                       #一辺直進①
    lt(60)                      #60°左回転
    fd(a)                       #一辺直進②
    lt(120)                     #120°左回転
    fd(a)                       #一辺直進③
    ア                          #180°回転
    fd(a)                       #一辺直進④
    lt(120)                     #120°左回転
#初期設定
clearscreen()                   #ペン位置：中央　向き：左
speed(10)                       #速度10　最高速はspeed(0)　最後にdone()
face(30)                        #亀の向き30°
shape("turtle2")                #ペン形状
bgcolor("orange")               #背景オレンジ色
color("green")                  #柄緑色
pensize(3)                      #ペン太さ３
開始X座標=-400                  #初期ペンX座標-400
開始Y座標=200                   #初期ペンY座標　200
辺長さ=60                       #一辺の長さ　60
列数=8                          #文様列数　8
行数=4                          #文様行数　2×4
pu();goto(開始X座標,開始Y座標);pd() #初期座標へ移動
#単位文様を繰り返し描画
for j in range(行数):           #行の繰り返し
  for i in range(列数):         #列の繰り返し（奇数行）
    立方体(辺長さ)              #立方体の描画
    pu();fd(辺長さ);rt(60);fd(辺長さ);lt(60);pd()   #次へ移動⑤
  pu()                          #ペンアップ
  setx(開始X座標)               #開始X座標へ移動⑥
  rt(イ)                        #下を向く
  fd(辺長さ)                    #一辺分直進⑦
  lt(ウ)                        #ウ°斜め方向
  fd(辺長さ)                    #一辺分直進⑧
  lt(60)                        #60°斜め上方向
  pd()                          #ペンダウン　2行目開始点
  for i in range(列数):         #列の繰り返し（偶数行）
    立方体(辺長さ)              #立方体の描画
    pu();fd(辺長さ);rt(60);fd(辺長さ);lt(60);pd()    #次へ移動⑤
  pu()                          #ペンアップ
  rt(60)                        #60°斜め下方向
  fd(辺長さ*3)                  #3つ分の辺の長さ直進⑨
  lt(エ)                        #エ°回転して左を向く⑩
  setx(開始X座標)               #開始X座標へ移動⑪
  pd()                          #ペンダウン　3行目開始点
#done()                         #speed(0)の時#を取ってdone()にする
```







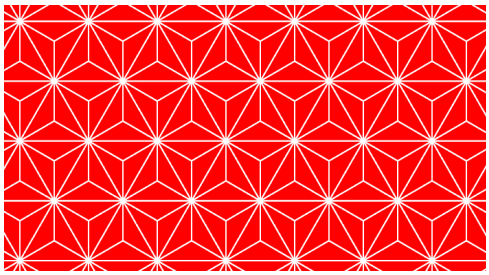
6 伝統的な連続文様を描いてみよう（麻の葉文様）

学習目標とキーワード

学 習 目 標　アルゴリズムを工夫して麻の葉文様を作成しよう。

キーワード　・システム開発　・ウォーターフォールモデル　・要件定義　・基本設計　詳細設計
・テスト　・アルゴリズムの工夫　・関数

1　麻の葉文様を描くプログラムを考えてみましょう。

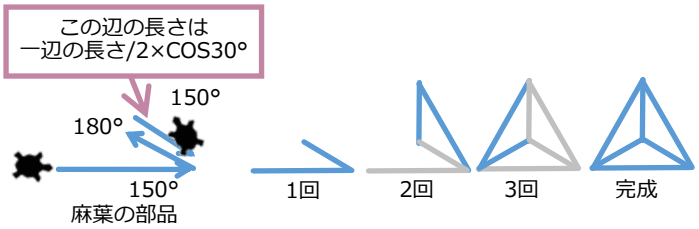


これまでのプログラム開発手順を参考にて、描画プログラムを作成していきます。

手順１　要件定義（どういう文様を描くか）

麻の葉文様は、小さな三角形の組み合わせです。文様の単位としては、最小の三角形の他に、これを３つ合わせた三角形、６つ合わせた六角形が見えます。ここでは、最小の三角形を３つ合わせた図形を単位文様とします。

麻葉（単位文様）の描画方法



手順２　基本設計（どのような構成か）

麻の葉文様を描くプログラムは、

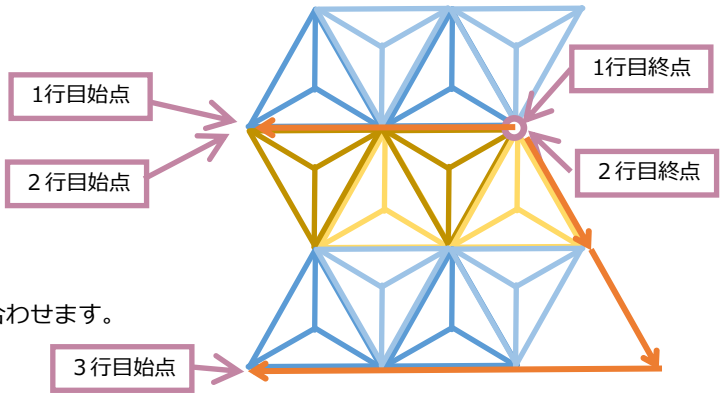
1 地の色やタートルの開始位置を設定する
　初期設定
2 麻葉（単位文様）を描くプログラム
3 麻葉を並べるプログラム

手順３　詳細設計（各構成はどのような動作か）

1 初期設定

地の色、タートルの初期位置と向き

麻葉の大きさと数を設定します。



2 麻葉（単位文様）の描画プログラム

麻葉内の小さな三角形を一組として３つ組み合わせます。

何回も使うプログラムなので関数にします。

3麻葉（単位文様）を並べるプログラム

１行目は麻葉（単位文様）が上向きと下向きを交互に繰り返すように、始点の移動とペンの回転をします。

１行目の描画が終了したら、ペンを開始X座標に戻し、２行目始点とします。

２行目は麻葉（単位文様）が下向きと上向きを交互に繰り返すように、始点の移動とペンの回転をします。

２行目の描画が終了したら、3行目開始Y座標まで60°右下へ一辺分移動し、そこからペンを開始X座標に戻して、３行目始点に移動します。

手順４　実装（プログラミング）

詳細設計を基に、プログラムを作成します。

手順５　テスト（動作を確認する）

立方体は描けるか（単体テスト）、並べることはできるか（統合テスト）テストして修正します。

手順６　運用・保守（完成と改善）

完成した描画をリリースします。また、改善する余地があれば改善します。

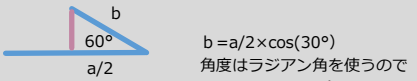
2　麻の葉文様のプログラムを実行してみましょう。

課題 6-1

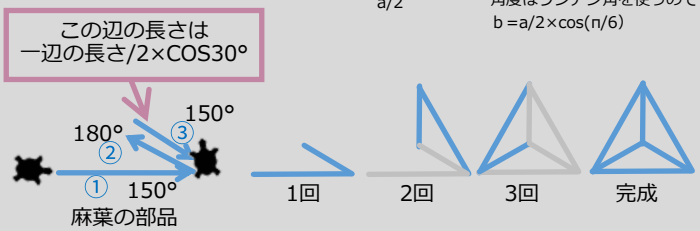
・プログラム中の　ア　から　ウ　に適切な字句を入れて、プログラムを完成させましょう。
・完成したら動作を確認しましょう。
・辺の長さや文様の列、行の数を変えて描画してみましょう。
・speed(0) と　done()　を記入して、高速に描画してみましょう。

プログラム例

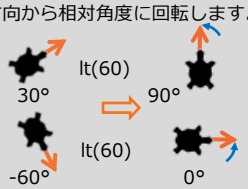
```
### 麻の葉文様 ###
!pip install ColabTurtlePlus      #Claboratory用タートルグラフィックスモジュールの読み込み
from ColabTurtlePlus.Turtle import *   #最初の２行は、１つのノートブックで最初に１回実行すればよい
import math                       #数学ライブラリのインポート（三角関数を使うため）
pi = math.pi                      #πを変数piに入れる
#単位文様（麻葉）の関数
def 麻葉(a):                      #関数の定義　引数aは一片の長さ
  b=(a/2)/math.cos(pi/6)          #小さい三角形の辺の長さb=一片の長さ/2×COS(π/6)
  for _ in range(3):              #部品の描画3回繰り返し
    fd(a)                         #一辺直進①
    ア                            #150°左回転
    fd(b)                         #短辺直進②
    イ                            #180°左回転
    fd(b)                         #短辺直進③
    lt(150)                       #150°左回転
#初期設定
clearscreen()                     #ペン位置：中央　向き：左
speed(10)                         #速度10　最高速はspeed(0)　最後にdone()
shape("turtle2")                  #ペン形状
bgcolor("red")                    #背景赤色
color("white")                    #柄白色
pensize(2)                        #ペン太さ２
開始X座標=-500                    #初期ペンX座標-500
開始Y座標=300                     #初期ペンY座標　200
辺長さ=100                        #一辺の長さ　100
列数=10                           #文様列数　10
行数=10                           #文様行数　2×10
pu();goto(開始X座標,開始Y座標);pd()#初期座標へ移動
#単位文様を繰り返し描画
for j in range(行数):             #行の繰り返し
  for i in range(列数):           #列の繰り返し
    ウ                            #ペンの方向0°（右向き）④
    麻葉(辺長さ)                  #上向き麻葉の描画
    pu();fd(辺長さ);pd()          #次の始点へ移動⑤
    face(60)                      #ペン方向60°（1時方向）⑥
    麻葉(辺長さ)                  #下向き麻葉の描画
    face(0)                       #ペン方向0°（右向き）⑦
  pu()                            #ペンアップ
  setx(開始X座標)                 #開始X座標へ移動⑧
  pd()                            #ペンダウン　2行目開始点
  for i in range(列数):           #列の繰り返し
    face(-60)                     #ペン方向-60°（5時方向）⑨
    麻葉(辺長さ)                  #下向き麻葉の描画
    face(0)                       #ペンの方向0°（右向き）⑩
    pu();fd(辺長さ);pd()          #次の始点へ移動⑪
    face(-120)                    #ペン方向-120°（7時方向）⑫
    麻葉(辺長さ)                  #上向き麻葉の描画
    face(0)                       #ペンの方向0°（右向き）⑬
  pu()                            #ペンアップ
  face(-60)                       #ペン方向-60°（5時方向）⑭
  fd(辺長さ*2)                    #辺の長さ×２直進⑮
  face(0)                         #ペンの方向0°（右向き）⑯
  setx(開始X座標)                 #開始X座標へ移動⑰
  pd()                            #ペンダウン　3行目開始点
#done()                           #speed(0)の時#を取ってdone()にする
```



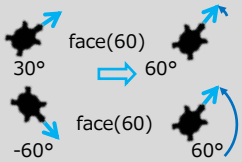
b =a/2×cos(30°)
角度はラジアン角を使うので
b =a/2×cos(π/6)

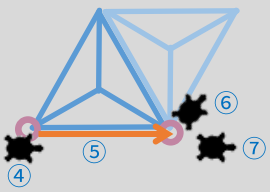


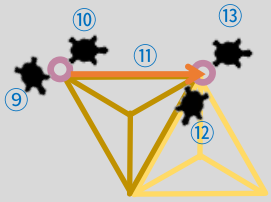
回転　left() :lt()　right():rt()は
現在の方向から相対角度に回転します。

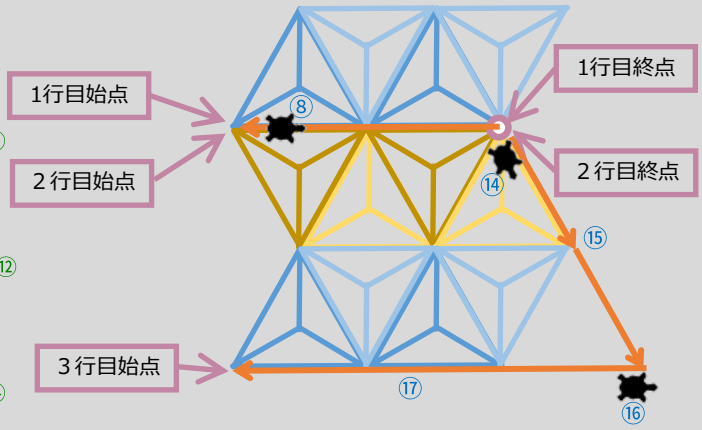


方向　face() は右向きを０°とした
絶対角度に回転します。









7 伝統的な連続文様を描いてみよう（青海波文様）

学習目標とキーワード

学 習 目 標　アルゴリズムを工夫して青海波文様を作成しよう。

キーワード　・システム開発　・ウォーターフォールモデル　・要件定義　・基本設計　詳細設計
・テスト　・アルゴリズムの工夫　・関数

1　青海波文様を描くプログラムを考えてみましょう。

これまでのプログラム開発手順を参考にて、描画プログラムを作成していきます。

波の（単位文様）の描画方法

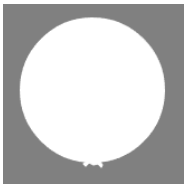
手順１　要件定義（どういう文様を描くか）

青海波文様は円弧が重なって並んでいる文様です。単位文様の弧の角度を計算するのは複雑です。そこで、同心円文様の円を重ねて描画することで表現します。

手順２　基本設計（どのような構成か）

青海波文様を描くプログラムは、

1 地の色やタートルの開始位置を設定する
初期設定
2 波（単位文様）を描くプログラム
3 波を並べるプログラム
の構成とします。



1 白く塗った円を描きます

2 青い円を描きます

3 ペンを横向きにして円の始点を設定します

4 二つ目の円を描きます

5 ペンを横向きにして円の始点を設定します

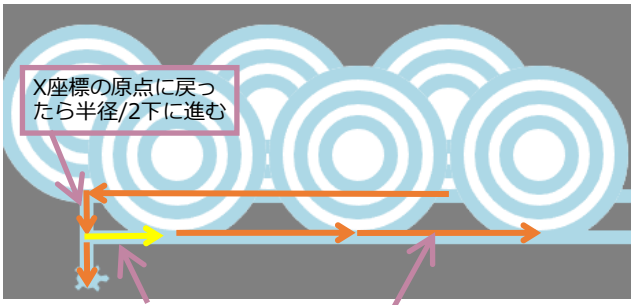
6 内側の同心円を描画して完成です

手順３　詳細設計（各構成はどのような動作か）

1 初期設定
地の色、タートルの初期位置と向き、波の大きさと数を設定します。

2 波（単位文様）の描画プログラム
白く塗った円を下地にして、その上に３つの青い同心円を描きます。
円の中心位置は、ペンの横方向に半径を指定して決めるので、同心円を描くときは、半径に合わせてペンの位置を変えます。同心円の半径や線の太さは試行錯誤で決めていきます。

3 波を並べるプログラム
波の並びは、偶数行と奇数行で開始点水平方向にずらします。



手順４　実装（プログラミング）

詳細設計を基に、プログラムを作成します。

手順５　テスト（動作を確認する）

立方体は描けるか（単体テスト）、並べることはできるか（統合テスト）テストして修正します。

手順６　運用・保守（完成と改善）

完成した描画をリリースします。また、改善する余地があれば改善します。

2　青海波文様のプログラムを実行してみましょう。

課題 7-1

・プログラム中の　ア　から　ウ　に適切な字句を入れて、プログラムを完成させましょう。
・完成したら動作を確認しましょう。
・辺の長さや文様の列、行の数を変えて描画してみましょう。
・speed(0) と　done()　を記入して、高速に描画してみましょう。

プログラム例

```
### 青海波文様 ###
!pip install ColabTurtlePlus          #Claboratory用タートルグラフィックスモジュールの読み込み
from ColabTurtlePlus.Turtle import *  #最初の２行は、１つのノートブックで最初に１回実行すればよい
#単位文様（波）の関数
def 波(a):                  #関数の定義　引数aは波の半径
  begin_fill()              #塗りの開始
  color("white","white")#線と塗りの色は白色
  circle(a)                 #半径aの円を描く
  end_fill()                #塗終わり　　白い塗円が描ける①
  color("LightBlue")        #線の色をライトブルーにする
  pensize(int(a/6))         #ペンサイズを半径の6分の１にする
  circle(a)                 #ライトブルーの円を描く②
  ア                        #上を向く
  pu();fd(0.3*a);pd()       #半径×0.3 円の中心に進む
  イ                        #右を向く③
  circle(a*0.7)             #半径×0.7 で２番目の円を描く④
  ア                        #上を向く
  pu();fd(0.3*a);pd()       #半径×0.3 円の中心に進む
  イ                        #右を向く⑤
  circle(a*0.4)             #半径×0.4 で３番目の円を描く⑥
  ウ                        #下を向く
  pu();fd(a*0.6);pd()       #半径×0.6 円の外側に進む
  イ                        #右を向く
#初期設定
clearscreen()#ペン位置：中央　向き：左
speed(10)                   #速度10　最高速はspeed(0)　最後にdone()
shape("turtle2")            #ペン形状
bgcolor("gray")             #背景ア灰色
color("white")              #柄白色
pensize(10)                 #ペン太さ10
開始X座標=-400              #初期ペンX座標-400
開始Y座標=250               #初期ペンY座標　250
半径=60                     #半径60
列数=7                      #文様列数　7
行数=22                     #文様行数　22
pu();goto(開始X座標,開始Y座標);pd()      #初期座標へ移動
#単位文様を繰り返し描画
for j in range(行数):       #行の繰り返し
  for i in range(列数):     #列の繰り返し
    face(0)                 #右向を向く
    波(半径)                #波の描画
    pu();fd(半径*2.2);pd()#半径×2.2右に進む⑦
  setx(開始X座標)           #開始X座標へ戻る⑧
  face(-90)                 #下を向く⑨
  fd(半径/2)                #半径/2　進む⑩
  face(0)                   #右を向く⑪
  fd(半径*1.1*((j+1)%2))#Jが0,2,4・・・の時(j+1)%2の値が1になり、半径×1.1右に進む⑫
  pd()                      #ペンダウン
#done()                     #speed(0)の時#を取ってdone()にする
```

① 白く塗った円を描きます　② 青い円を描きます　③ ペンを横向きにして円の始点を設定します

④ 二つ目の円を描きます　⑤ ペンを横向きにして円の始点を設定します　⑥ 内側の同心円を描画して完成です