

プログラムで絵を描こう

【turtle】

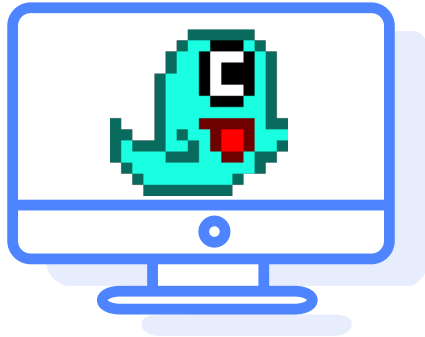
プログラミ

今日は、これまでの
プログラムの基本を使って
カメで絵を描きます。

START



解答例



1～100のいくつだと思いますか？

もっと大きいよ

もっと小さいよ

もう少し大きいけどかなり近いよ！

もうすこし小さいけど少し近いよ

当たり！

5 回で当たったよ！

50

90

70

80

72



解答例

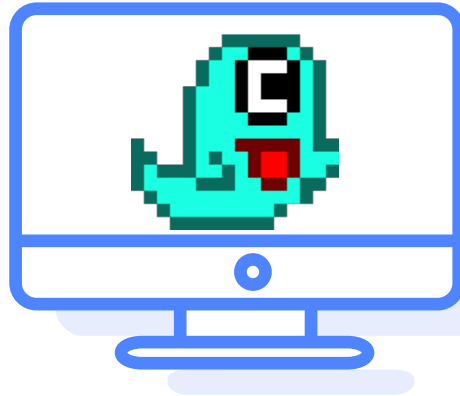
```
import random
def numCheck(indata,ans):
    if indata==ans:
        print("当たり!")
        return True
    elif indata<ans and (ans-indata) >10:
        print("もっと大きいよ")
        return False
    elif (indata>ans) and 1<=(indata-ans)<=5:
        print("もう少し小さいけどかなり近いよ!")
        return False
    elif (indata>ans) and 5<(indata-ans)<=10:
        print("もうすこし小さいけど少し近いよ")
        return False
    elif (indata<ans) and 1<=(ans-indata)<=5:
        print("もう少し大きいけどかなり近いよ!")
        return False
    elif (indata<ans) and 5<(ans-indata)<=10:
        print("もうすこし大きいけど少し近いよ ")
        return False
    else:
        print("もっと小さいよ")
        return False

ans=random.randint(1,100)
count=0
flag=True

while flag:
    count=count+1
    indata=int(input("1~100のいくつだと思いますか?"))
    if numCheck(indata,ans)==True:
        flag=False
    if count >= 10:
        print("もう一度頑張ろう!")

print(count,"回で当たったよ!")
```

解答例



曲がり角を右左どちらに曲がる？

右



お姉さんが現れた。答えにくい質問をしてきた。

ねえ、何歳に見える？

30

失礼な！そんなに老けて見える？

18

そんなに若い？嬉しいなあ。でも間違い！

28

当たり！全然当たらないじゃん！



曲がり角を右左どちらに曲がる？

左



おばさん が現れた。非常にめんどくさい質問が飛んできた。

ねえ、何歳に見える？

50

そんなに若い？嬉しいなあ。でも間違い！

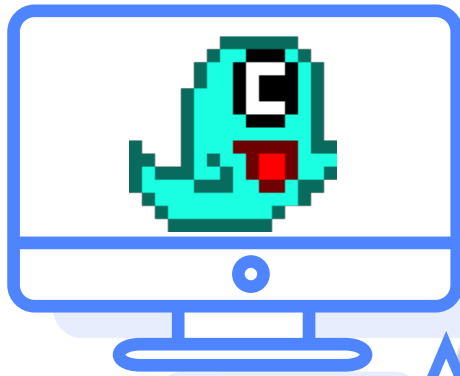
70

失礼な！そんなに老けて見える？

解答例

```
miti = int(input("あなたは今曲がり角。右に曲がるなら1、左なら2を入力し  
てください"))  
  
import random  
migi = ["お兄さん","お姉さん","会社員","OL"]  
hidari = ["（怪しい）おじさん","おばさん","おじさん","（怪しい）ばあさん",]  
a=random.choice(migi)  
b=random.choice(hidari)  
  
def numCheck(indata, ans):  
    if indata == ans:  
        print("当たり！")  
        return True  
    elif indata < ans:  
        print("そんなに若く見える？嬉しいなあ。でも間違い！")  
        return False  
    else:  
        print("失礼な！そんなに老けて見える？")  
        return False  
  
if miti==1:  
    print(a,"が現れた。答えにくい質問をしてきた。")  
    ans = random.randint(15,50)  
    count = 0  
    flag = True  
    while flag:  
        count = count + 1  
        indata = int(input("ねえ、何歳に見える？"))  
        if numCheck(indata, ans) == True:  
            flag = False  
            if count<5:  
                print("すぐ当てられちゃったなあ")  
            else:  
                print("全然当たらないじゃん！")  
        else:  
            print(b,"が現れた。非常にめんどくさい質問が飛んできた。")  
            ans = random.randint(50,100)  
            count = 0  
            flag = True  
            while flag:  
                count = count + 1  
                indata = int(input("ねえ、何歳に見える？"))  
                if numCheck(indata, ans) == True:  
                    flag = False  
                    if count<5:  
                        print("すぐ当てられちゃった,,,")  
                    else:  
                        print("全然当たんなかったなあ")
```

解答例



あなたの誕生日を当てます！開始は1、終了は2

1開始



これから言うことを「頭の中や電卓」で行ってください。

① あなたの誕生日の「月」にあたる数を2倍してください

② ①の結果に5を足してください

③ ②の結果を50倍してください

④ ③の結果に誕生日の「日」にあたる数を足してください

④の結果を入力してください

10/26

20 ?

25 ?

1250 ?

1276 ?

1276

あなたの誕生日は 10月 26日ですね！！
どうでしたか？当たっていましたか？

解答例

```
def differ2(a,b):  
    c = a - b  
    return c  
  
def birthdayCheck():  
    print("これから言うことを「頭の中や電卓」で行ってください。")  
    input(" (何かキーを押してください) ")  
    print("① あなたの誕生日の「月」にあたる数を2倍してください")  
    input(" (できたらキーを押してください) ")  
    print("② ①の結果に5を足してください")  
    input(" (できたらキーを押してください) ")  
    print("③ ②の結果を50倍してください")  
    input(" (できたらキーを押してください) ")  
    print("④ ③の結果に誕生日の「日」にあたる数を足してください")  
    input(" (できたらキーを押してください) ")  
    indata = int(input("④の結果を入力してください→"))  
  
    kekka = differ2(indata,250)  
    kekkaStr = str(kekka)  
    length = len(kekkaStr)
```

1276

1276 - 250 = 1026 → 10月26日

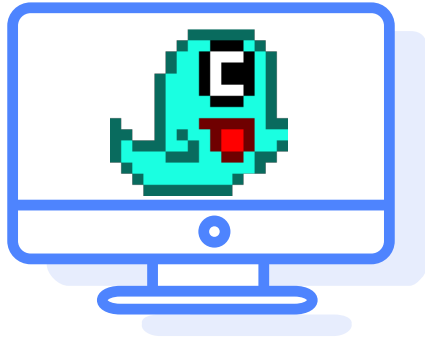
351

351 - 250 = 101 → 1月01日

```
if length == 3:  
    hund = (kekkaStr[-3])  
    ten = (kekkaStr[-2])  
    one = (kekkaStr[-1])  
    print("あなたの誕生日は",hund,"月",ten,one,"日ですね！！")  
    print("どうでしたか？当たっていましたか？")  
    print("最初に戻ります")  
    input(" (何かキーを押してください) ")  
  
elif length == 4:  
    thou = (kekkaStr[-4])  
    hund = (kekkaStr[-3])  
    ten = (kekkaStr[-2])  
    one = (kekkaStr[-1])  
    print("あなたの誕生日は",thou,hund,"月",ten,one,"日ですね！！")  
    print("どうでしたか？当たっていましたか？")  
    print("最初に戻ります")  
    input(" (何かキーを押してください) ")
```

```
flag = True  
while flag:  
    start = int(input("あなたの誕生日を当てます！開始するには1を、終了するには2を入力してください"))  
  
    if start == 1:  
        birthdayCheck()  
        flag = True  
    elif start == 2:  
        print("ありがとうございました。友達ともやってみてくださいね")  
        flag = False  
    else:  
        print("もう一度やり直してください")  
        flag = True
```

解答例



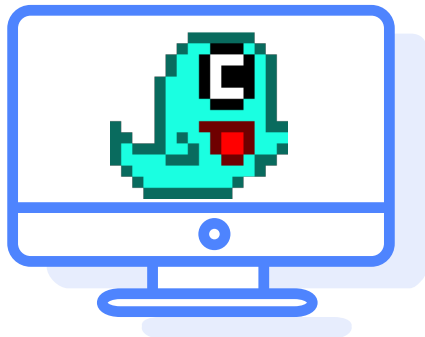
私のペットはなんでしょう？カタカナで答えてね。

イヌ

残念！ヒントは、 ひまわりの種を好みます。

ハムスター

正解です！私のペットは、 ハムスター です。



私のペットはなんでしょう？カタカナで答えてね。

ハムスター

残念！ヒントは、 カラフルな鳥です。

インコ

正解です！私のペットは、 インコ です。



解答例

```
animals=["イヌ","ネコ","ウサギ","ハムスター","インコ","カワウソ"]
tips=["ワンワンと鳴きます。","ニャーニャーと鳴きます。","耳が長いです。","ひまわりの種を好みます。",
      "カラフルな鳥です。","魚を好みます。"]
```

```
import random
def Pets(predict,answer):
    if predict==answer:
        print("正解です！私のペットは、",answer,"です。")
        return True
    else:
        print("残念！ヒントは、",hints)
        return False
```

```
answer=random.choice(animals)
```

```
if answer==animals[0]:
    hints=tips[0]
elif answer==animals[1]:
    hints=tips[1]
elif answer==animals[2]:
    hints=tips[2]
elif answer==animals[3]:
    hints=tips[3]
elif answer==animals[4]:
    hints=tips[4]
elif answer==animals[5]:
    hints=tips[5]
```

```
flag=True
while flag:
    predict=input("私のペットはなんでしょう？カタカナで答えてね。")
    if Pets(predict,answer)==True:
        flag=False
```

プログラムをもっと簡潔にする方法がありますか？




解答例

```
animals=["イヌ","ネコ","ウサギ","ハムスター","インコ","カワウソ"]
tips=["ワンワンと鳴きます。","ニャーニャーと鳴きます。","耳が長いです。","ひまわりの種を好みます。",
      "カラフルな鳥です。","魚を好みます。"]
```

```
import random
def Pets(predict,answer):
    if predict==answer:
        print("正解です！私のペットは、",answer,"です。")
        return True
    else:
        print("残念！ヒントは、",hints)
        return False
```

```
answer=random.choice(animals)
```

```
if answer==animals[0]:
    hints=tips[0]
elif answer==animals[1]:
    hints=tips[1]
elif answer==animals[2]:
    hints=tips[2]
elif answer==animals[3]:
    hints=tips[3]
elif answer==animals[4]:
    hints=tips[4]
elif answer==animals[5]:
    hints=tips[5]
```



```
for i in range(6):
    if answer==animals[i]:
        hints = tips[i]
```

```
flag=True
while flag:
    predict=input("私のペットはなんでしょう？カタカナで答えてね。")
    if Pets(predict,answer)==True:
        flag=False
```

困ったことや質問はありませんか？

難しくなってきた不安。



だんだん複雑になって難しくなってきた。



できることが少しずつ増えて、動くとうれしい。



パソコンがフリーズして消えてしまい、やり直していて課題の提出が遅れてしまいました。



すべて正しく動作するかを確かめるのに何回もやり直したのでかなり疲れた。ゲーム制作の修正の大変さを少し垣間見た。



ゲーム音楽が好きなので、授業で使われているチャイムやアラーム音をいつも楽しく聞いています。



困ったことや質問はありませんか？

プログラマーで稼いでる人は、どれくらいスキルがあるんですか？



稼いでいる人は「製品やサービスを創り出す視点」がある。プログラムスキルだけあって「プログラムを書くロボット」みたいな人はあまり稼いでいない。



「ググるスキル」が重要。プログラムミスはエディターが教えてくれるので正確に書く必要はない。「考える力」と、あの情報はあのあたりにありそうだと検討をつけてググれる力があれば作れる。

「得意分野があること」も重要。なんでもできます、という人は、依頼するほうからすると任せにくく、これは得意です、というのがハッキリしている人に任せたくなる。仕事をするには、まず相手から仕事を受けることが必要。

「普通にコミュニケーションできる力」も重要。バグのほとんどは、コミュニケーション不足。「こんな話聞いていない」「こう言ったつもりだったのに」でトラブルが起こり、変な建て増し修正をして、さらにバグることがほとんど。意思疎通ができてるプロジェクトはどうまく動く。

「友だちを作る力」も重要。プログラミングは情報科学。毎日、新技術が生まれ、トレンドも変わっていて、すべての情報を一人でチェックするのは大変。いろんな友だちがいると、こんな技術が面白いぞ、最近これが注目されてるぞ、と教えてもらえる。最近こんなの作ったぞ、と自慢してくるのを聞くのも重要。人は人から一番影響をうけて成長できる。

まとめ：変数、順次、分岐

データを保存する

変数名 = 値

```
score = 10000
```

順番に処理する

順次

```
h = float(input("身長何cmですか？")) / 100.0
w = float(input("体重何kgですか？"))
bmi = w / (h * h)
print("あなたのBMI値は、", bmi, "です。")
```

データを調べて、場合分けする

分岐 (if)

```
if age < 20:
    print("お酒は二十歳を過ぎてから")
else :
    print("飲み過ぎにご注意ください")
```

まとめ：リスト、反復

たくさんデータを用意する

リスト名 = [値,値,値,値,値]

```
fruits = ["イチゴ", "桃", "梨", "ブドウ", "メロン"]
```

データをすべて、くり返す

反復 (for)

```
total = 0
data = [12, 34, 56, 78, 90]
for value in data:
    total = total + value
print("合計は", total)
```

ある状態の間は、くり返す

反復 (while)

```
import random
flag = True
while flag:
    dice = random.randint(1, 6)
    if dice == 6 :
        flag = False
```

まとめ：関数

ややこしい仕事をまとめる

関数 (def)

```
import random

def dice():
    kekka = random.randint(1,6)
    print(kekka)

dice()
```

データを渡すと結果を返す

関数 (def)

```
def bmi(h,w):
    ans = w / (h * h)
    return ans

h = float(input("身長何cmですか？")) / 100.0
w = float(input("体重何kgですか？"))
print("あなたのBMI値は、", bmi(h, w), "です。")
```



タートルグラフィックス



前進

```
from turtle import *  
shape("turtle")  
color("red")  
forward(100)  
done()
```

線の色（赤）

入力して試してみましょう



赤いカメを、
前へ100進めます



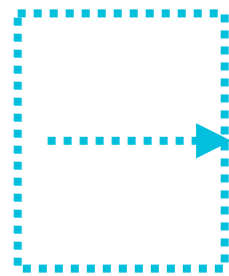
正方形

```
from turtle import *  
shape("turtle")  
color("red")
```

```
speed(10)
```

移動スピード、速く

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



```
    forward(100)
```

```
    left(90)
```

```
done()
```

入力して試してみましょう

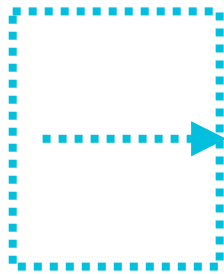
前へ100、左へ90度
を4回くり返します



六角形

```
from turtle import *  
shape("turtle")  
color("red")  
speed(10)
```

```
for i in range(6):  
    forward(100)  
    left(60)  
done()
```



六角形を描く

修正して試してみましょう

前へ100、左へ60度
を6回くり返します



六角形の関数

```
from turtle import *  
shape("turtle")  
color("red")  
speed(10)
```

```
def hexagon():
```

```
    for i in range(6):  
        forward(100)  
        left(60)
```

```
hexagon()  
done()
```

六角形を描く

修正して試してみましょう

「六角形を描く仕事」
を関数にまとめる



六角形を10回描く

```
from turtle import *  
shape("turtle")  
color("red")  
speed(10)  
def hexagon():  
    for i in range(6):  
        forward(100)  
        left(60)  
for i in range(10):  
    hexagon()  
    left(360/10)  
done()
```

- ・ 六角形を描く関数を10回呼ぶ
- ・ 36度ずつ回転させる

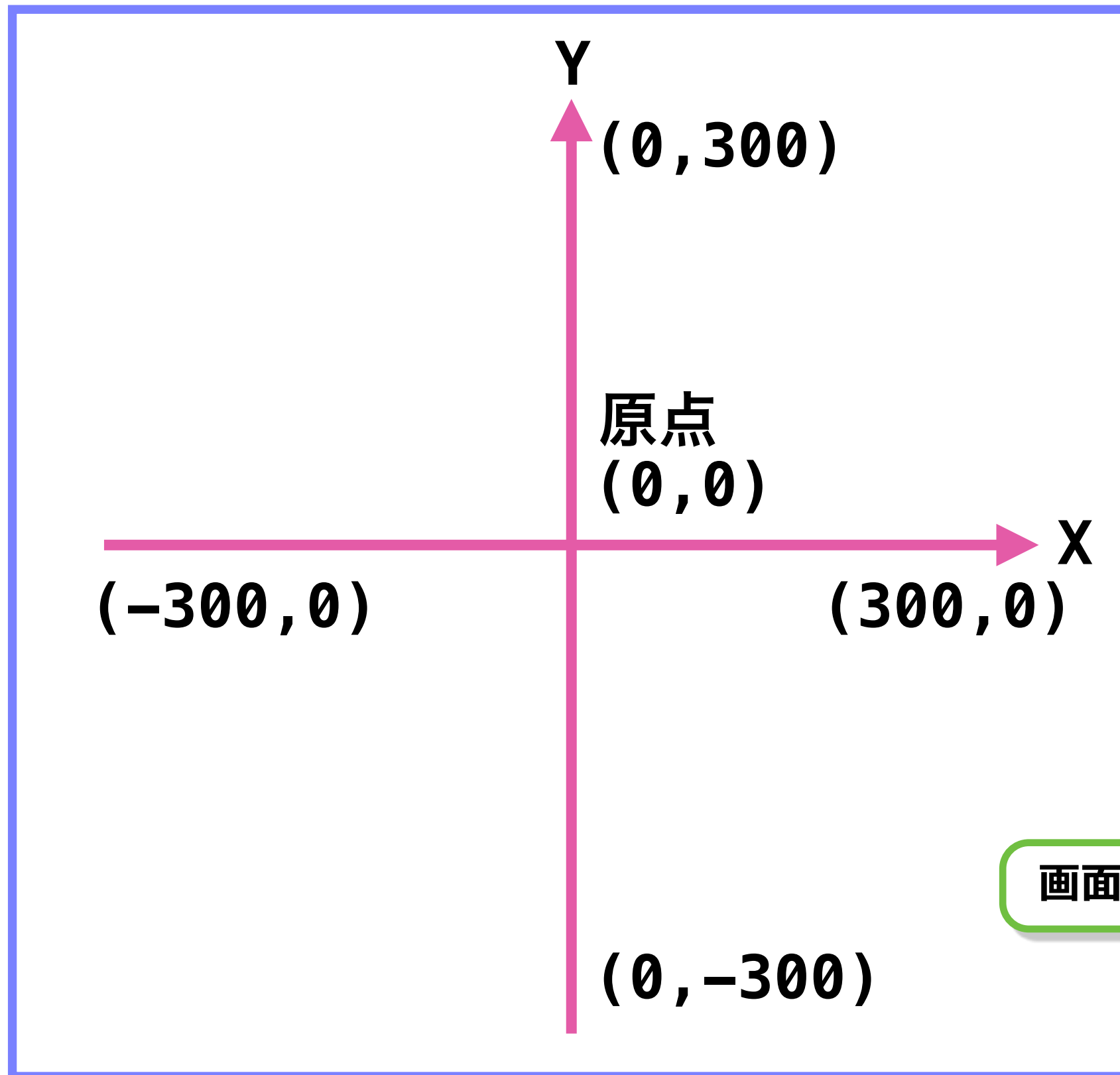
修正して試してみましょう





もっと自由に描こう
ペンの上げ下げ

タートルグラフィックスの座標



画面中央が、 $(0,0)$

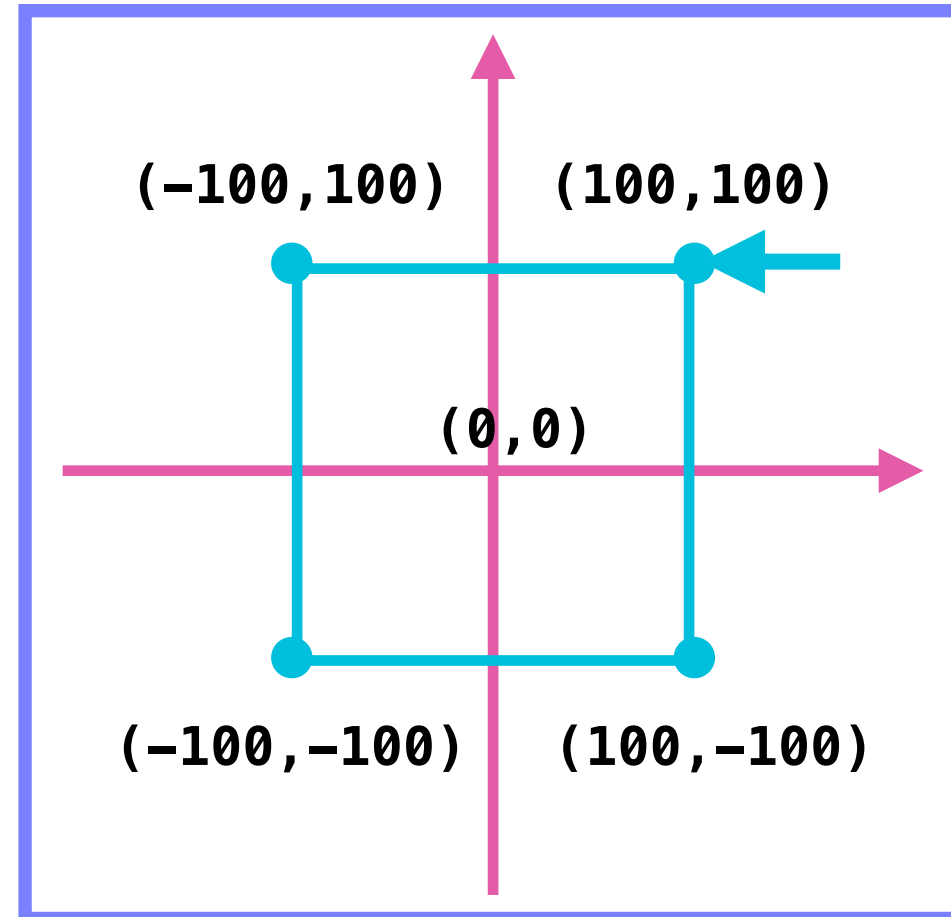


指定した座標まで移動する : goto

```
from turtle import *  
shape("turtle")  
color("orange")  
goto(100,100)  
goto(-100,100)  
goto(-100,-100)  
goto(100,-100)  
goto(100,100)
```

done()

指定した位置に移動



座標の位置で絵を描ける



ペンを上げる、下げる : penup, pendown

```
from turtle import *  
shape("turtle")
```

```
color("orange")
```

```
penup()
```

ペンを上げる

```
goto(100,100)
```

```
pendown()
```

ペンを下ろす

```
goto(-100,100)
```

```
goto(-100,-100)
```

```
goto(100,-100)
```

```
goto(100,100)
```

```
done()
```

ペンを上げている間は
線が描かれない



休憩



円、円弧

```
from turtle import *  
shape("turtle")  
color("red")  
speed(10)
```

```
circle(100)
```

```
done()
```

円

```
from turtle import *  
shape("turtle")  
color("red")  
speed(10)
```

```
circle(100, 90)
```

```
done()
```

円弧

指定した角度だけ円弧になる



赤い風船

```
from turtle import *  
shape("turtle")  
color("red")  
speed(10)
```

```
circle(100)  
right(90)  
circle(100, 90)
```

風船を描く

```
done()
```

入力して試してみましょう

円、右に90度、
ヒモを描く



風船を3つ描く

```
from turtle import *  
shape("turtle")  
color("red")  
speed(10)
```

```
def balloon(x,y):
```

指定した位置(x,y)に、風船を描く関数

```
    penup()
```

```
    goto(x,y)
```

ペンを上げて、開始位置(x,y)まで移動して、ペンを下ろす

```
    pendown()
```

```
    circle(100)
```

```
    right(90)
```

そこに風船を描く

```
    circle(100,90)
```

```
balloon(-200,200)
```

```
balloon(100,100)
```

```
balloon(0,-150)
```

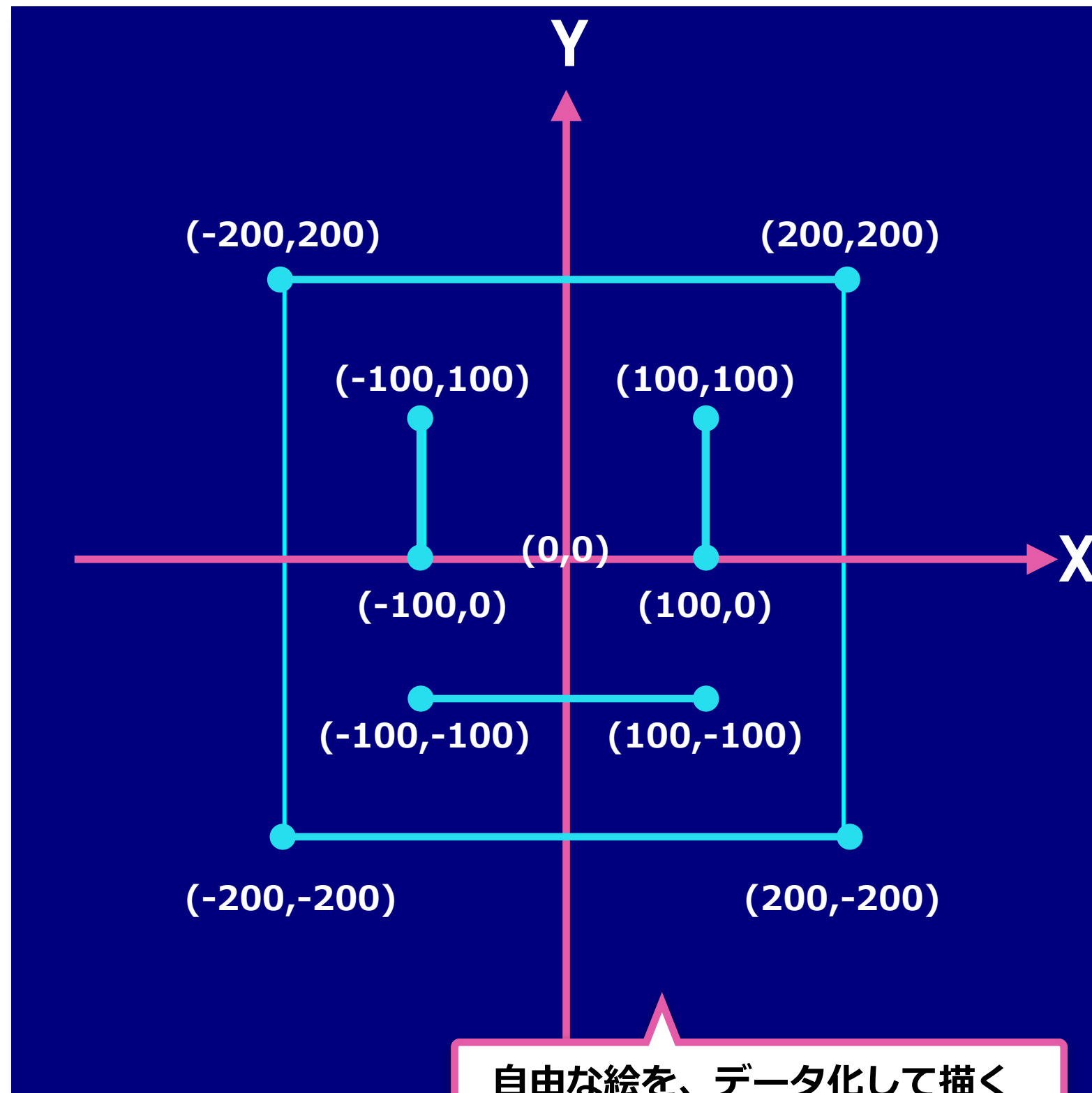
```
done()
```

風船を3つ描く

「風船を描く仕事」
を関数にまとめる
x,yで指定した位置に
風船を描く



座標を使って絵を描こう



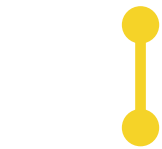
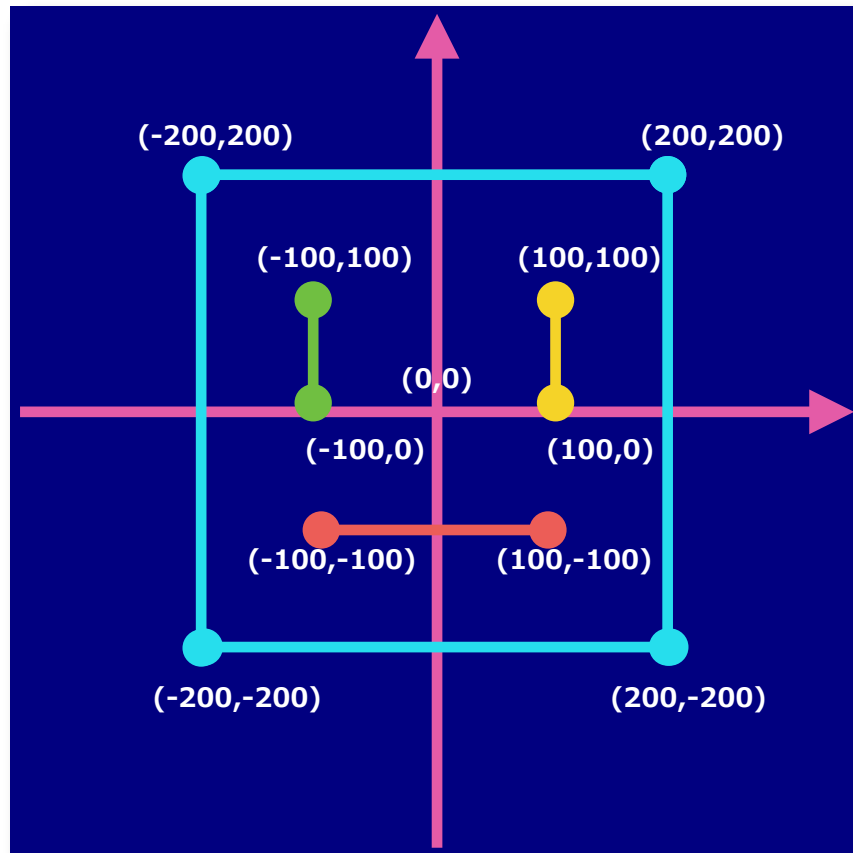
自由な絵を、データ化して描く

どうやってデータを作るか？

線をデータ化する：リストのリスト

線は、点をつないだもの

点のデータは、xyのリスト。それを複数並べたリストが線



$(100,100)-(100,0)$

→ `[[100,100],[100,0]]`



$(-100,100)-(-100,0)$

→ `[[-100,100],[-100,0]]`



$(100,-100)-(-100,-100)$

→ `[[100,-100],[-100,-100]]`



$(200,200)-(-200,200)-(-200,-200)-(200,-200)-(200,200)$

↓
`[[200,200],[-200,200],[-200,-200],[200,-200],[200,200]]`

0 1 2 3 4
 $a = [[x_0, y_0], [x_1, y_1], [x_2, y_2], [x_3, y_3], [x_4, y_4]]$

0番目の点データ

$a[0] \cdots [x_0, y_0]$

1番目の点データ

$a[1] \cdots [x_1, y_1]$

1番目の点の x の値

$a[1][0] \cdots x_1$

1番目の点の y の値

$a[1][1] \cdots y_1$

点のデータ (xyのリスト) のリスト

線のデータ

線で絵を描く1

```
from turtle import *
```

```
shape("turtle")
```

```
setup(800, 800)
```

ウィンドウサイズを800x800にする

```
color("cyan")
```

線の色

```
Screen().bgcolor("navy")
```

背景の色

```
width(3)
```

```
speed(10)
```

点データのリストをつないで線を描く関数

```
def drawline(data):
```

点データの個数回くり返す

```
    for i in range(len(data)):
```

```
        if i == 0:
```

最初(0番)だけペンを上げて移動

```
            penup()
```

```
            goto(data[i][0], data[i][1])
```

```
            pendown()
```

x

y

```
drawline([[100, 100], [100, 0]])
```

```
ht()
```

```
done()
```

左目のデータ: [[開始点],[終了点]]

(100,100)



(100,0)

「線を描く仕事」

を関数にまとめる

リストで渡した点を

結んで線を引いていく

がんばれ



入力して試してみましょう

線で絵を描く2

```
:  
:  
def drawline(data):  
:
```

ここは同じ

右目と口のデータ：[[開始点],[終了点]]

```
drawline([[100,100],[100,0]])
```

```
drawline([[-100,100],[-100,0]])  
drawline([[100,-100],[-100,-100]])
```

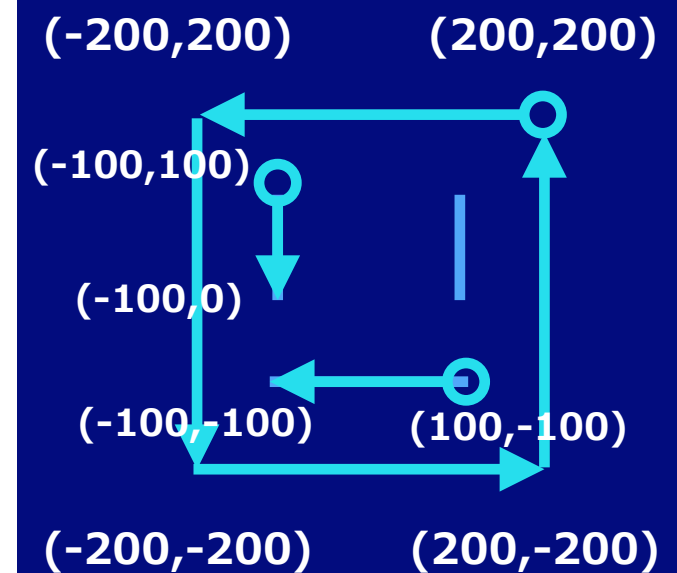
```
drawline([[200,200],[-200,200],  
[-200,-200],[200,-200],[200,200]])
```

```
ht()  
done()
```

輪郭のデータ：[[開始点],[2点目],[3点目],[4点目],[終了点]]

修正して試してみましょう

顔が描けたよ



「タートルで自分の名前を描くプログラム」
を作ってください。



「①ゴールを決めて②計画して③作る」
のが目的です。

漢字が難しければ、カタカナでもいいよ。
それでも難しかったら、似顔絵でもいいよ。

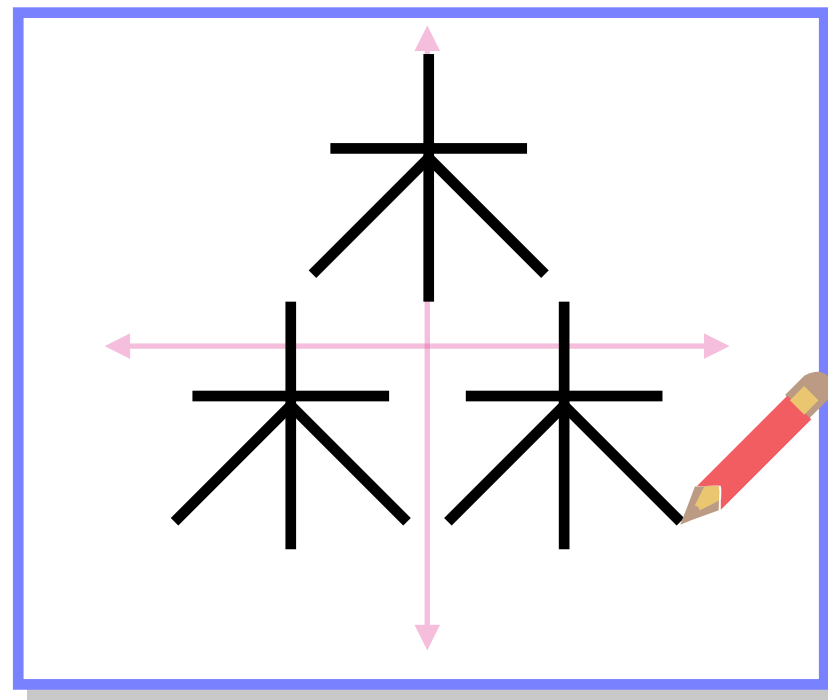


中間課題

① 何を描くか、ゴールを決める

まず、何を描くかを決めて、紙に描きましょう。

- ・ 自分の名前を、漢字、ひらがな、カタカナで描く
 - ・ または、似顔絵を描く
- の中から選んでください。



タートルで描くことを
イメージして描こう

姓（上の名前）だけでも、名（下の名前）だけでもいいし、
できる人は、姓名（上と下の名前）を描いてもいいよ。



② どのように書くか、計画する

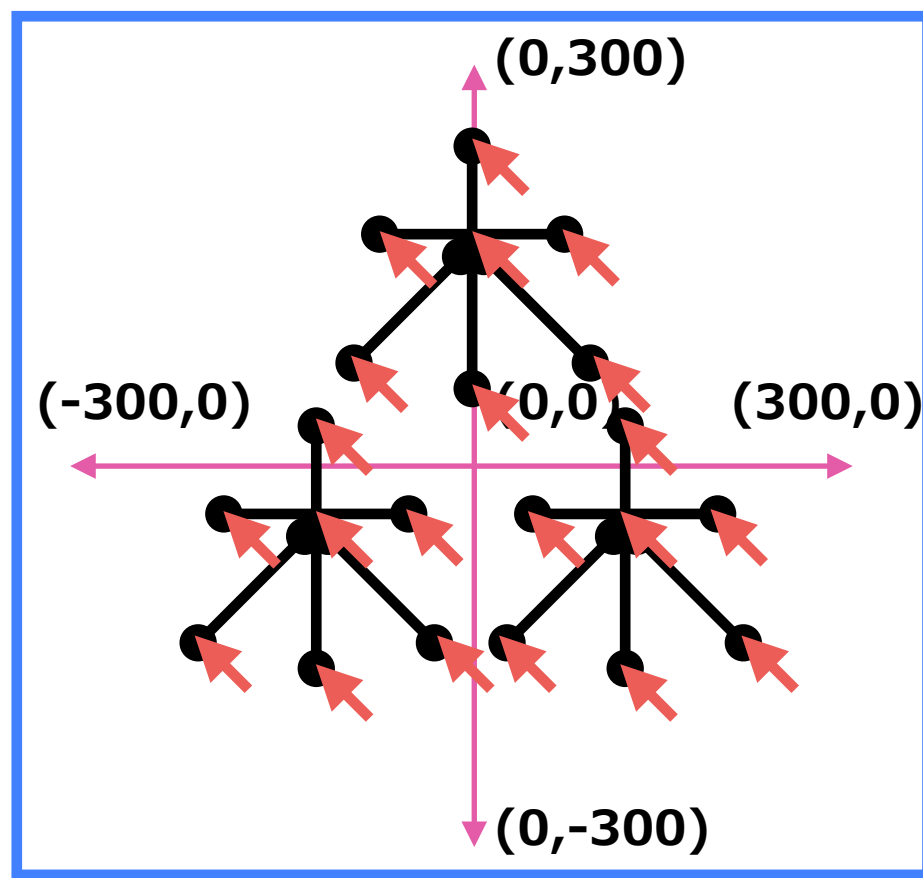
次に、どのように書くかの、計画を立てます。

すべての線（開始点、終了点）のXY座標

を調べましょう。

（方眼紙があれば便利です。）

すべての点のXY座標を調べる



[[0, 200], [0, 50]]
[[-70, 140], [70, 140]]
[[0, 140], [-70, 50]]
[[0, 140], [70, 50]]
[[-80, 30], [-80, -120]]
[[-150, -30], [-10, -30]]
[[-80, -30], [-150, -120]]
[[-80, -30], [-10, -120]]
[[80, 30], [80, -120]]
[[10, -30], [150, -30]]
[[80, -30], [10, -120]]
[[80, -30], [150, -120]]

ひと筆ずつの、データを作ろう



③ データや、組み立て方から、プログラムを作る

作ったデータから、プログラムを作ります。

```
def drawline(data):  
    print(data)  
    for i in range(len(data)):  
        if i == 0:  
            penup()  
        goto(data[i][0], data[i][1])  
        pendown()
```

```
drawline([[0, 200], [0, 50]])  
drawline([[-70, 140], [70, 140]])  
drawline([[0, 140], [-70, 50]])  
drawline([[0, 140], [70, 50]])
```

```
drawline([[-80, 30], [-80, -120]])  
drawline([[-150, -30], [-10, -30]])  
drawline([[-80, -30], [-150, -120]])  
drawline([[-80, -30], [-10, -120]])
```

```
drawline([[80, 30], [80, -120]])  
drawline([[10, -30], [150, -30]])  
drawline([[80, -30], [10, -120]])  
drawline([[80, -30], [150, -120]])
```

```
[[0, 200], [0, 50]]  
[[-70, 140], [70, 140]]  
[[0, 140], [-70, 50]]  
[[0, 140], [70, 50]]  
[[-80, 30], [-80, -120]]  
[[-150, -30], [-10, -30]]  
[[-80, -30], [-150, -120]]  
[[-80, -30], [-10, -120]]  
[[80, 30], [80, -120]]  
[[10, -30], [150, -30]]  
[[80, -30], [10, -120]]  
[[80, -30], [150, -120]]
```

データから、プログラムを作る

drawlineをすべて書いてから試すより
途中で試しながら作ると、作りやすいぞ



turtle 命令一覧

・ 動作

命令	動作
foward(歩数)	前進する
back(歩数)	後退する
left(角度)	左へ回転
right(角度)	右へ回転
goto(x, y)	指定位置へ移動
penup()	ペンを上げる
pendown()	ペンを下げる
ht()	カメを消す

・ 色の名前

色名	色名	色名
black	white	gray
red	green	blue
cyan	yellow	magenta
tomato	limegreen	navy
orange	aquamarine	steelblue
chocolate	gold	royalblue
coral	khaki	lightblue
salmon	greenyellow	cadetblue
brown	forestgreen	blueviolet

・ ペンの変更

命令	動作	値
shape(マーク)	カメの形を変える	turtle, arrow, circle, square, triangle
color("色")	色を変える	色の名前
Screen().bgcolor("色")	背景色を変える	色の名前
width(太さ)	線の太さ	
speed(速度)	移動速度	1:遅い~10:速い、0アニメーションなし
setup(幅、高さ)	ウィンドウサイズ	