

Python入門

Turtleで視覚的にわかる、構文の基礎

亀って可愛いですよね

そんな亀さんを自由自在に操れたら嬉しいはずです。

今回は大学の授業や研究でよく使われるPythonというプログラミング言語を用い、亀を動かしその軌跡で絵を描いていこうと思います。

1 Pythonの基礎

テキストの出力

`print("出力したい文字")` で出力することができる

例1-1

↓ここをクリックすることで実行できる

```
In [ ]: print("こんにちはPython")
```

こんにちはPython

解説

`print()` は、文字を出力するための関数である。

Pythonにはさまざまな関数が存在し、関数を介してさまざまな機能を用いることができる。

練習問題1-a

自分の好きな食べ物を出力してみよう

```
In [ ]: # 練習問題1-1をここに書く  
print("りんご")
```

りんご

数値の扱い

文字列を出力する際にはダブルクオーテーション(`" "`)を使用した。

ダブルクオーテーションを使用せず、数字をそのまま入れることによって数値を出力することができる。

例1-2

1行目の例では、数値として 1 を出力している。

2行目では、 `" "` をついているため、文字列として出力している。

一見すると同じように見えるが、後ほど違いを解説する。

```
In [ ]: print(1) # 数値としての出力  
print("1") # 文字列としての出力
```

```
1  
1
```

例1-3

下記の例では、四則演算を使用している。

例1-2で、数値としての出力と文字列としての出力を区別したのは、演算ができるかできないかが変わるためにある。

四則演算は、下記の演算子によって計算することができる。

記号	意味
+	加算
-	減算
*	乗算
/	除算
%	剰余
**	べき乗
()	カッコ

```
In [ ]: print(1 + 1) # 計算されて2が出力される  
print("1" + "1") # 文字列なので、単に連結されて11が出力される  
print(33 ** 2 % 7)
```

```
2  
11  
4
```

練習問題1-b

それぞれのコメントの下に、それにあった解答を出力するプログラムを書こう

ヒント

出力は `print` を使うよ

```
In [ ]: # 40 + 11  
print(40 + 11)  
# 30 - 6  
print(30 - 6)  
# 33 × 5  
print(33 * 5)  
# 20 ÷ 3  
print(20 / 3)  
# 25 の 11 乗  
print(25 ** 11)
```

```
51  
24  
165
```

```
6.6666666666666667  
2384185791015625
```

変数

一度書いた文字や、すでに計算した数値をもう一度書くのは効率が悪い。

そのため、Pythonをはじめとする多くのプログラミング言語には**変数**というものが存在する。

変数は箱のようなもので、プログラマが自由に作ることができる。

変数名(任意) = 入れたい値

ここでいう = は、数学的な等号の意味ではなく、**代入する**という意味になる。

また、変数はそのまま `print()` 関数の中に入れることで、出力できる。

例1-4

```
In [ ]: my_name = "Taiki Sugawara"  
        print(my_name)  
  
        my_age = 18 + 4  
        print(my_age)  
  
        print(my_name + "の年齢は" + str(my_age) + "です。")
```

```
Taiki Sugawara  
22  
Taiki Sugawaraの年齢は22です。
```

解説

先ほど "" で囲った場合、文字列になった。囲わない場合は変数として扱われる。

練習問題1-c

5世紀、中国の數学者祖 沖之（そ ちゅうし、429年 - 500年）は、円周率を

$$\pi = 355/113$$

と定義した。

この式を用いて円周率を定義したのちに、それぞれの問題を出力しよう。

ヒント: 円周率は一度しか求めない

```
In [ ]: # 練習問題1-c  
  
pi = 355 / 113  
# 直径2の円の面積  
print(1 * 1 * pi)  
# 直径5の球の体積  
print(4 * pi * 5 ** 3 / 3)  
# 直径6の球の表面積  
print( 4 * pi * 6 ** 2)
```

```
3.1415929203539825  
523.5988200589971  
452.3893805309735
```

発展問題1-a

ある日の気温が20°Cであるとして、それを華氏(°F)で表すプログラムを作成してみよう

ヒント1

摂氏(°C)から華氏(°F)への変換式は

$$\text{華氏温度} = \text{摂氏温度} \times 1.8 + 32$$

ヒント2

数学と同じように一つの式に記号を連ねることが可能

```
In [ ]: # 発展問題1-aの答え
```

```
templatelure_c = 5  
print(templatelure_c * 1.8 + 32)
```

41.0

2 Turtleの準備

1章では文字列と数値の出力のみで、味気なかった。

3章以降でTurtleと呼ばれるライブラリを用い、視覚的にプログラミングを理解する。

本章ではそのための準備を行う。理解できなくて差し支えない。

※ ライブラリとは、さまざまな機能を持つプログラムを関数などによって簡単に利用できるパックのことである。

Turtleのインストール

ライブラリは他人が書いたプログラムの集まりである。使用するにはプログラムをインストールする必要がある。

インストールには下記のコマンドを使用する。コマンドは1回実行すれば良い。

```
pip3 install インストールしたいライブラリの名前
```

下記コマンドを実行してColabTurtleをインストールしよう。

```
In [ ]: !pip3 install ColabTurtle
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/  
Collecting ColabTurtle  
  Downloading ColabTurtle-2.1.0.tar.gz (6.8 kB)  
    Preparing metadata (setup.py) ... done  
Building wheels for collected packages: ColabTurtle  
  Building wheel for ColabTurtle (setup.py) ... done  
    Created wheel for ColabTurtle: filename=ColabTurtle-2.1.0-py3-none-any.whl size=7656 sha256=05a2130b429fdd6ea0a59ece51ea0e853d3e6307cb12195da229fa65cddb863fce1e52dd24b0eb61fc  
    Stored in directory: /root/.cache/pip/wheels/a9/85/dc/29b6b43c4c6c0fe37192ccad65fe2adcef1e52dd24b0eb61fc  
Successfully built ColabTurtle  
Installing collected packages: ColabTurtle  
Successfully installed ColabTurtle-2.1.0
```

Turtleのインポート

インストールをするだけではライブラリを使用することはできない。
Pythonプログラムの冒頭で、ライブラリを使うことを明言する必要がある。

下記プログラムを実行し、ColabTurtleをインポートしよう。

```
In [ ]: from ColabTurtle.Turtle import *
```

これによって、ColabTurtleライブラリに入っている関数を使用できるようになった。

3 Turtleを使った描画の基本

いよいよTurtleを使用していく。

Turtleの関数の使い方は、ライブラリの配布場所で述べられている。

優秀な諸君なら、英語で書いてあっても問題ないと思うので、読み進めて各自やっていただいて構わない。

[ColabTurtleの関数の使い方](#)

とりあえず動かしてみる

何はともあれ下記コードを動かしてみてほしい。

亀が出現し動く。その軌跡が図形を描く。

例3-1

Turtleにある3種類の関数を使用し、「のような図形を書いている。

```
In [ ]: initializeTurtle() # カメを初期化  
forward(100) # カメを前に100px進める  
right(90) # カメを右に90°回転  
forward(100) # カメを前に100px進める
```



解説

Turtleの関数にはカメを動かしたり、色を変えたりする機能がある。

[ColabTurtleの関数の使い方](#)にそれぞれの使い方が書いてあるため、各自読んで自由に動かしてみてほしい。

`initializeTurtle()` が特殊な関数であり、これを呼び出すとカメを描画しているウインドウを初期化する。

この関数が呼ばれなければ、前の軌跡やカメの位置などが保存されている。

練習問題3-a

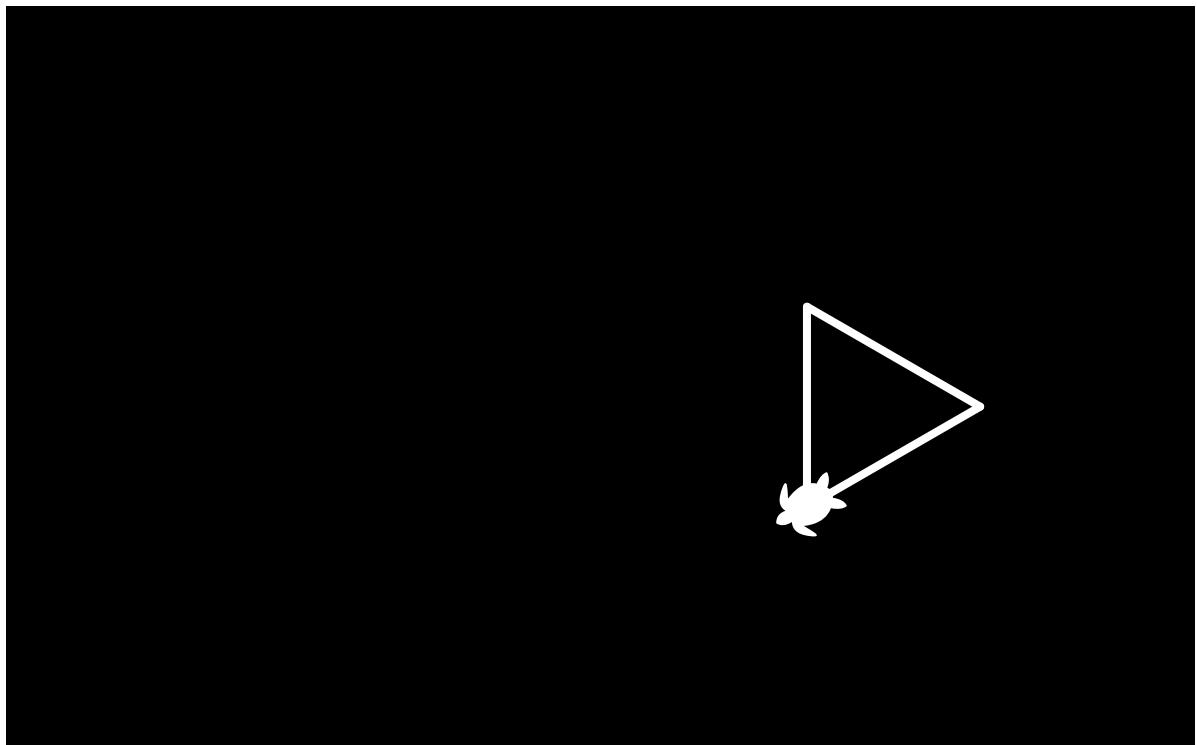
ウインドウを初期化し、軌跡で三角形を描画してみよう。

辺の長さや、どんな三角形かは問わない。

ヒント: 正三角形を書くのが簡単。正三角形の角度は?

In []:

```
# 練習問題3-a
initializeTurtle()
forward(100)
right(120)
forward(100)
right(120)
forward(100)
```



発展問題3-a

`right()`, `left()`, `forward()` 以外に5つの関数を用い、自由に描画してみよう。

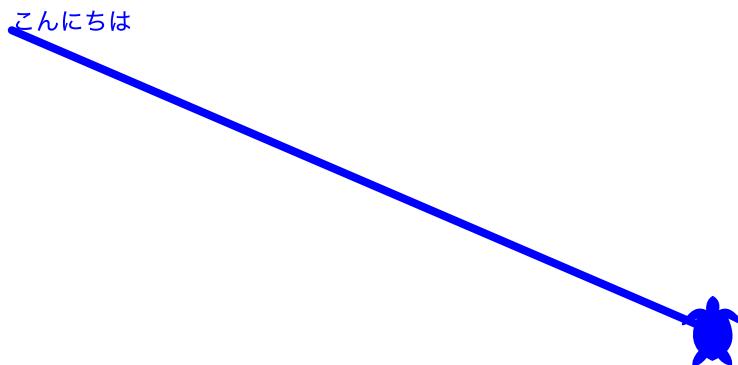
ヒント: いくつかの有用そうな関数を示す。発展問題ではこれ以外の関数も使った独自性のある図形を期待する。

`color(r,g,b)` : カメとペンの色を変える。色はRGB表記で入力する。色は[こういったサイト](#)で確かめられる

`goto(x,y)` : カメを指定したx,y座標に移動させる。軌跡は描かれる。
x,y座標は下記のようになっている。

` | x ---+-----> ||| | y √

```
In [ ]: # 発展問題3-α
initializeTurtle()
goto(10,10)
penup()
goto(50,100)
pendown()
color('blue')
speed(7)
write("こんにちは")
bgcolor('white')
home()
```



4 制御構文

一般的にプログラミング言語には、条件分岐や繰り返しを行う制御構文と呼ばれるものが存在する。

Pythonにも制御構文は存在する。

本章では代表的な幾つかの制御構文を用い、より楽しくプログラミングをしよう。

for文

cf.) https://docs.python.org/3/reference/compound_stmts.html#the-for-statement

`for` は、繰り返し処理をするための制御構文である。

下記のように記述する。

例4-1

```
In [ ]: for i in range(5):  
    print(i)
```

```
0  
1  
2  
3  
4
```

解説

`range()` 関数は、配列と呼ばれる数値のリストを返す関数である。0から指定した回数カウントアップして出力する。今回は、`[0,1,2,3,4]` が 出力される。

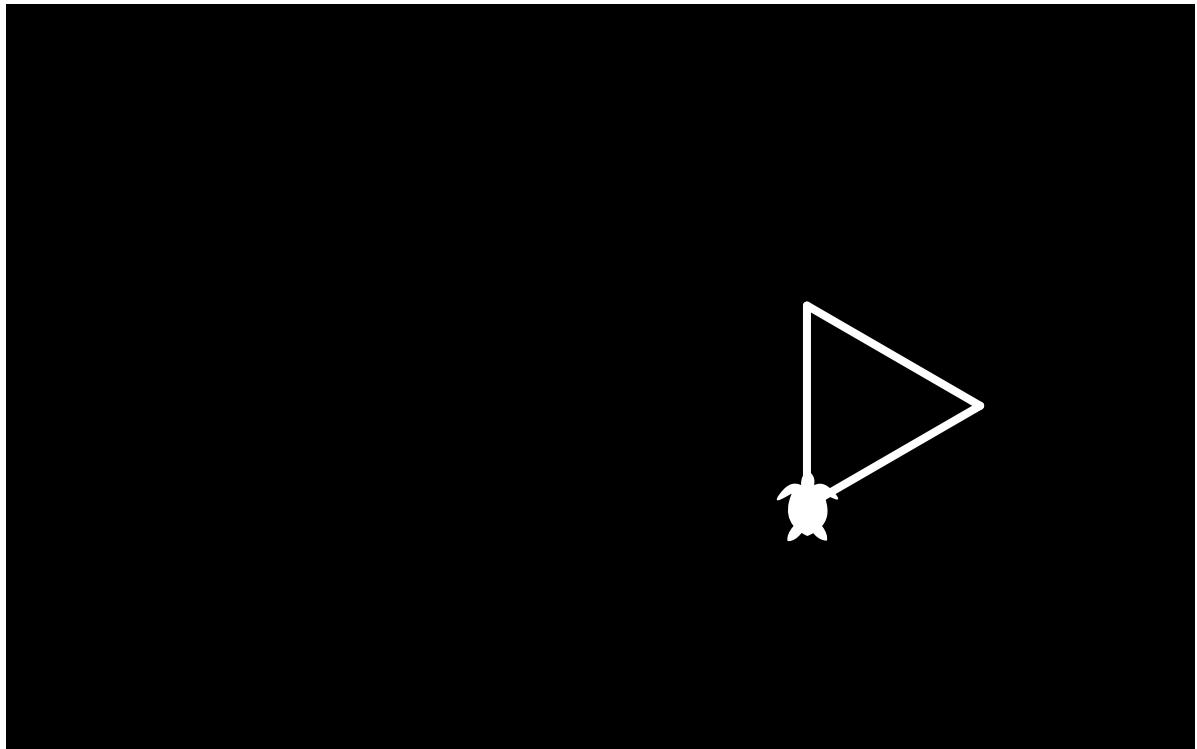
`for` の直後に書かれた変数（今回は `i`）に、リストに入っている数値が順番に入り、その後の処理が実行されていく。

繰り返し対象のプログラムはインデントをつけることに注意すること。

例4-2

`for` 文を使えば、練習問題3-aは簡単にかける

```
In [ ]: # 練習問題3-a  
initializeTurtle()  
for i in range(3):  
    forward(100)  
    right(120)
```



if文

cf.) https://docs.python.org/3/reference/compound_stmts.html#the-if-statement

`if` を使うと、条件分岐をすることができる。

先ほどの `for` と組み合わせ、下記のような処理が記述できる。

例4-3

```
In [ ]: for i in range(5):
    if i < 3:
        print(str(i) + "は3より小さい!")
    elif i == 3:
        print("3です。")
    else:
        print(str(i) + "は3より大きい!")
```

```
0は3より小さい!
1は3より小さい!
2は3より小さい!
3です。
4は3より大きい!
```

解説

`if` の後に来る文は条件式と呼ばれる。関数が入ることもあるが、今回の例のように比較演算子が入ることもある。

記号	意味
<code>==</code>	等しい
<code>!=</code>	等しくない
<code><</code>	より小さい
<code>></code>	より大きい
<code><=</code>	以下
<code>>=</code>	以上
<code>and</code>	かつ
<code>or</code>	または
<code>not</code>	否定

`elif` は、直前の `if` の結果が偽であった際に続けて判定される。

`else` は、`if` や `elif` が偽だった時に最終的に実行される。偽だった際に無条件で実行されるため、条件式は書かない。

5 Turtleを使った大規模な描画

今まで学んだことを活かし、より複雑な図形を作ってみよう。

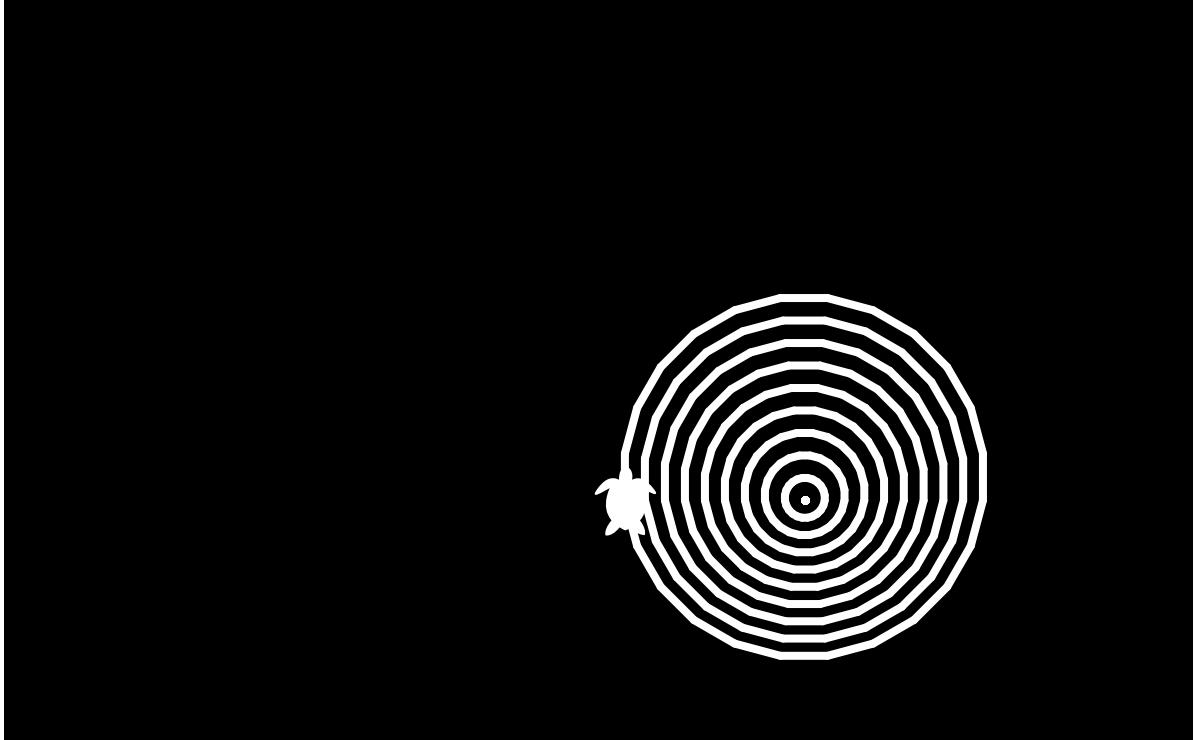
とにかくコードを書いてみよう

下記の例は今までの技術を用いたものである。

例5-1

```
In [ ]: initializeTurtle()
pi = 3.141592653589793
speed(10)
for size in range(10):
    radius = size * 10
    penup()
    w_width = window_width()
```

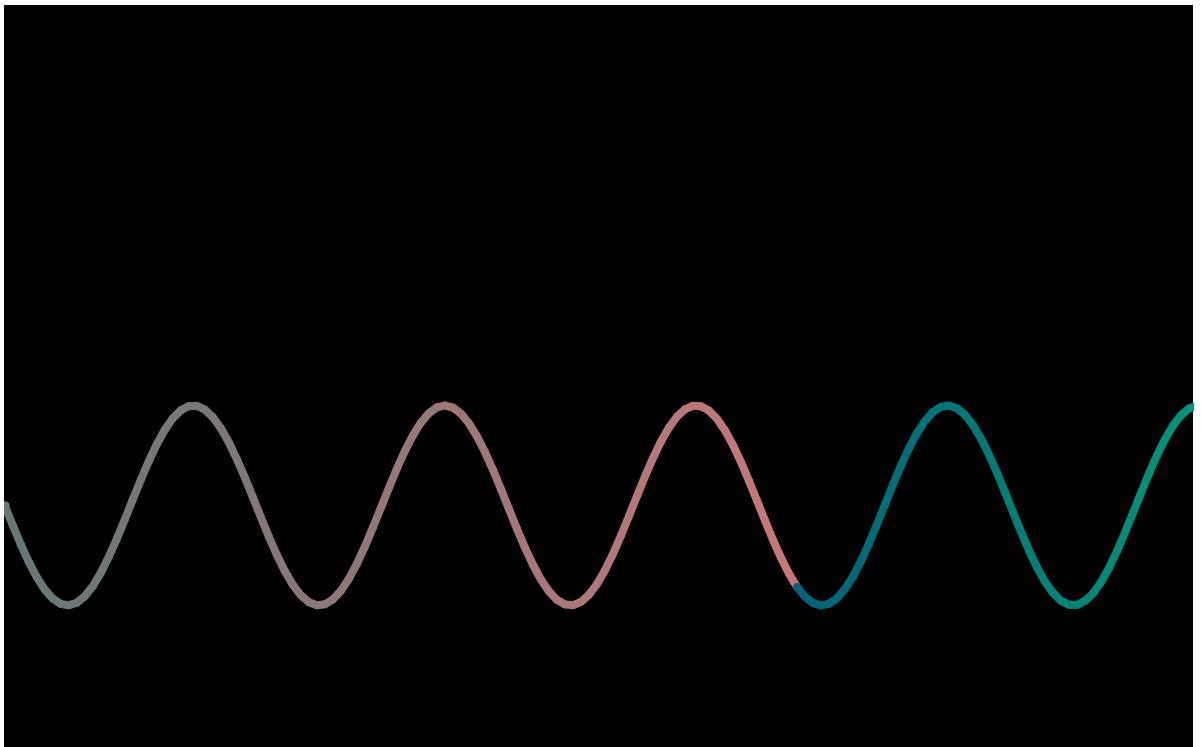
```
center_position = w_width/2
setx(center_position - radius)
pendown()
for i in range(24):
    forward(radius * 2 * pi / 24)
    right(15)
```



例5-2

```
In [ ]: import math
initializeTurtle()
speed(13)
w_width = window_width()
w_height = window_height()
step_unit = 200
x_unit = w_width / step_unit
color_unit = int(255 / step_unit)

penup()
setx(0)
pendown()
for step in range(step_unit):
    color_num = color_unit * step
    if step < step_unit / 2:
        color(color_num + 100, 120, 120)
    else:
        color(0 , color_num, 120)
    x = x_unit * step
    y=math.sin(x / 20) * 50 + w_height / 2
    goto(x,y)
```



練習問題5-1

今まで使った知識をもとに、独自に絵を描いてみよう

```
In [ ]: # 練習問題5-1
```

```
In [ ]: # 練習問題5-1(いくつも書いてみよう)
```

授業情報

概要

作者 :

菅原 太樹 (Github : [@taikis](#))

本Jupyterファイルを含む講義資料は、[GitHub](#)で公開している。

使用ライブラリ

本教材には下記のライブラリを使用している。

[ColabTurtle](#) ©2018 Tolga Atam (MIT License)