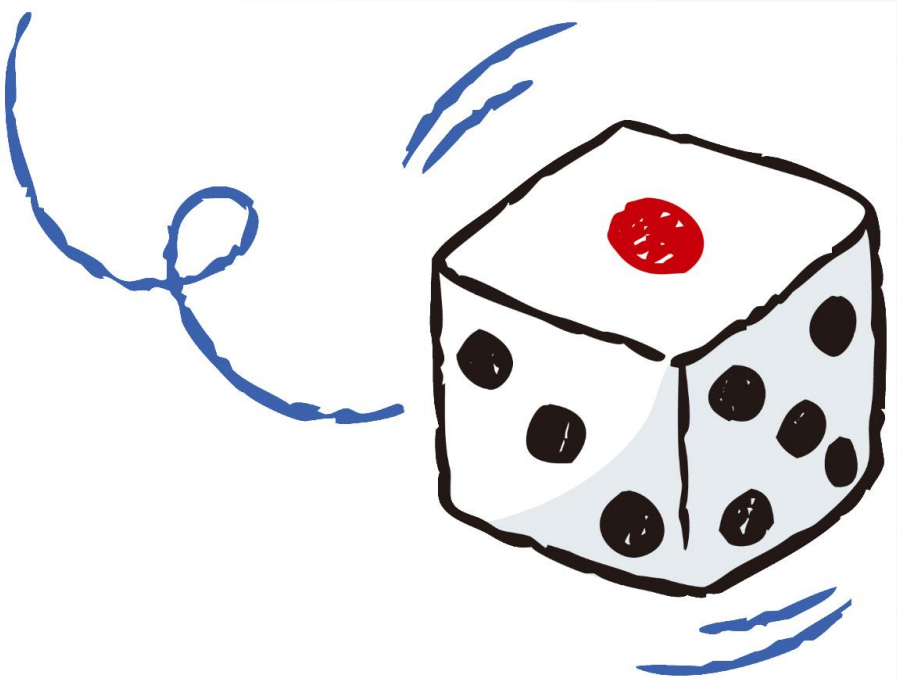
プログラミング教室のテクノロ

Pythonの道

簡単なプログラムを作ろう③

もくじ

・すごろくゲームプログラムを作る



すごろくゲームプログラムを作る

すごろくはサイコロを振ってコマを進めるので乱数を使うよ。まずは関数から学習しよう。

```
プレイヤ = 6    ・・・・変数”プレイヤ”に6を代入する
def 盤面():     ・・・・関数“ 盤面”を定義する
    print(“ □” *(プレイヤ-1) + “ P” + “ □” *(30-プレイヤ))
                ・・・・関数“ 盤面”の中身「□□□□□P □□□□・・ □□□□」
                        を表示する
盤面()          ・・・・関数“ 盤面”を呼び出す
```

※変数は通常、アルファベットで定義する。理解が追いつくまで、かな文字で変数を定義してもOK。ただし、他のプログラミング言語ではアルファベットでしか変数を定義できないことは知っておこう。

単語帳　～英語も一緒に覚えてしまおう！～

■define:定義する

■print:印刷する

すごろくゲームプログラムを作る

「Run Module」またはF5キーを押してプログラムを実行する。

□□□□□P□□□□□□□□□□□□□□□□□□□□□□□□
>>>

print(“ □” *(プレイヤ-1) + “ P” + “ □” *(30-プレイヤ))の解説

print文の中身は以下の３つをつなげたもの。文章をつなげるときに「＋」を使うことを知っておこう。変数プレイヤには６が入っている。

“ □” *(プレイヤ-1) ⇒ □ × (6-1)＝□□□□□

“ P” ⇒ P

24個

“ □” *(30-プレイヤ) ⇒□ × (30-6) ＝ □□□□□□・・・□□

def関数を理解する

関数を作るにはdef():と記載する。

scratchの　「ブロック定義」
ブロックと同じ

def　関数名():　・・・・ difの後に関数名を書く。この名前で呼び出す
　　処理　　・・・・どのような処理を実行するか記載する

※「：」(コロン)の後に改行してインデントを付ける。
インデントの範囲内でしか、条件式は有効とならないことを知っておこう。

関数を呼び出すときは関数名()と記載する。

すごろくゲームプログラムを作る

CPUのコマも表示できるようにする。

```
プレイヤ = 6   ・・・・変数”プレイヤ”に6を代入する
CPU = 3        ・・・・変数”CPU”に3を代入する
def 盤面():    ・・・・関数“ 盤面”を定義する
    print(“ □”*(プレイヤ-1) + “ P” + “ □”*(30-プレイヤ))
・・・・関数“ 盤面”の中身「□□□□□P □□□□・・ □□□□」を表示する
    print(“ □”*(CPU-1) + “ C” + “ □”*(30-CPU))
・・・・関数“ 盤面”の中身「□□C□□□□・・ □□□□□□□□□」を表示する
盤面()  ・・・・関数“ 盤面”を呼び出す
```

単語帳　～英語も一緒に覚えてしまおう！～

■define:定義する

■print:印刷する

すごろくゲームプログラムを作る

「Run Module」またはF5キーを押してプログラムを実行する。

□□□□□P□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□C□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
>>>

すごろくゲームプログラムを作る

プレイヤとCPUのコマを進めていく処理を追加しよう。

```
プレイヤ = 1   ・・・・変数”プレイヤ”に1を代入する
CPU = 1        ・・・・変数”CPU”に1を代入する
def 盤面():    ・・・・関数“盤面”を定義する
    print(“□”*(プレイヤ-1) + “P” + “□”*(30-プレイヤ))
      ・・・・関数“盤面”の中身「P□□□□□□□□□・・□□□□」を表示する
    print(“□”*(CPU-1) + “C” + “□”*(30-CPU))
      ・・・・関数“盤面”の中身「C□□□□□□□□□・・□□□□」を表示する
while True:    ・・・・条件が満たされるまで以下の処理を無限に繰り返す
    盤面()     ・・・・関数“盤面”を呼び出す
    input(“Enterを押すとコマが進みます”)   ・・・入力を受け付ける
    プレイヤ = プレイヤ + 1    ・・・・変数プレイヤに1+1を代入する
    CPU = CPU + 2              ・・・・変数CPUに1+2を代入する
```

すごろくゲームプログラムを作る

単語帳　～英語も一緒に覚えてしまおう！～

■define:定義する

■print:印刷する

■while:～する間

■True：正しい

■input：入力

「Run Module」またはF5キーを押してプログラムを実行する。

□□□□□P□□□□□□□□□□□□□□□□□□□□□□□□
□□C□□□□□□□□□□□□□□□□□□□□□□□□□□□
Enterを押すとコマが進みます
□□□□□□P□□□□□□□□□□□□□□□□□□□□□□□
□□□□C□□□□□□□□□□□□□□□□□□□□□□□□□
Enterを押すとコマが進みます
>>>

while文を理解する

while文を使うことで繰り返し処理を作ることができる。

scratchの「～まで繰り返す」ブロックと同じ

while文は条件を満たしている間、処理を繰り返す。

```
while 条件:
    条件が満たされている間に実行する処理
```

```
数 = 0          ・・・・数に0を代入する
while 数 < 3:   ・・・・数が3より小さい間、while文の処理を繰り返す
    数 = 数 + 1 ・・・・数に数＋１を代入する
    print(数)   ・・・・数を表示する
```

実行結果

```
1
2
3
>>>
```

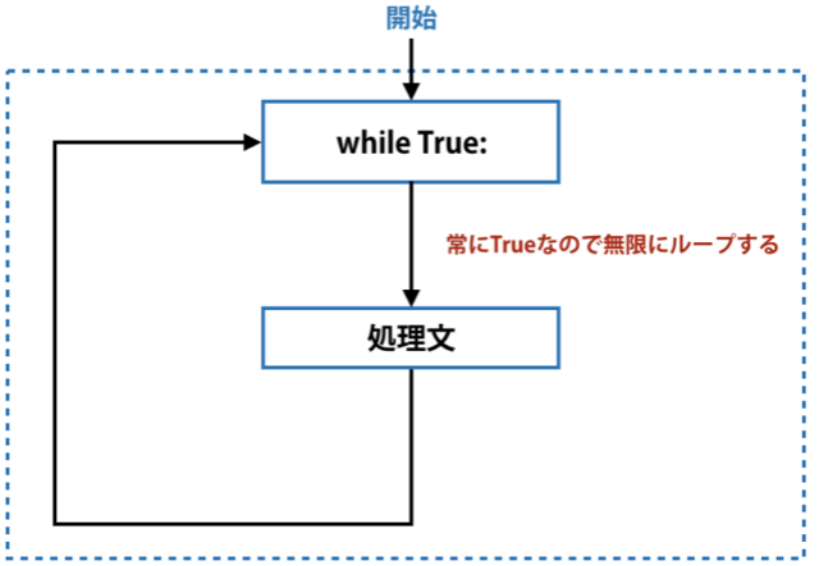
while文を理解する

while True:で無限ループを作ることができる。

while文は「ある条件を満たす間(Trueの間）、指定の処理を繰り返す」というもの。
つまり条件が常にTrue（=真）であれば、指定の処理を永遠に繰り返す無限ループになることを知っておこう。

```
while True:　・・・・TrueのTは必ず大文字Tをつかう
    処理
```

これで常に条件がTrue（=真）のため無限ループになるよ。



無限ループから抜けるにはbreakと記載することを覚えておこう

すごろくゲームプログラムを作る

乱数を使ってコマが進む数をランダムにしよう。

```
import random ・・・・ランダム関数を呼び出す
プレイヤ = 1  ・・・・変数”プレイヤ”に1を代入する
CPU = 1       ・・・・変数”CPU”に1を代入する
def 盤面():  ・・・・関数“盤面”を定義する
    print(“□”*(プレイヤ-1) + “P” + “□”*(30-プレイヤ))
                ・・・・関数“盤面”の中身「P□□□□・・□□□□」を表示する
    print(“□”*(CPU-1) + “C” + “□”*(30-CPU))
                ・・・・関数“盤面”の中身「C□□□□・・□□□□」を表示する
while True:   ・・・・条件が満たされるまで以下の処理を無限に繰り返す
    盤面()      ・・・・関数“盤面”を呼び出す
    input(“Enterを押すとコマが進みます”)・・・入力を受け付ける
    プレイヤ = プレイヤ + random.randint(1,6)
                ・・・変数プレイヤに1+(1～6)までの乱数を代入する
    CPU = CPU + random.randint(1,6)
                ・・・変数CPUに1+(1～6)までの乱数を代入する
```

すごろくゲームプログラムを作る

単語帳　～英語も一緒に覚えてしまおう！～

■import:取込む、■random:でたらめの、■define:定義する

■print:印刷する、■while:～する間、■True：正しい

■input：入力

「Run Module」またはF5キーを押してプログラムを実行する。

Enterを押すとコマが進みます

□□□□□P□□□□□□□□□□□□□□□□□□□□□□□□□

□□C□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Enterを押すとコマが進みます

□□□□□□P□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□C□□□□□□□□□□□□□□□□□□□□□□□□

Enterを押すとコマが進みます

□□□□□□□□□P□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□C□□□□□□□□□□□□□□□□□□□□□

Enterを押すとコマが進みます

>>>

random関数を理解する

random関数を使えば、あらかじめ決めておいた範囲で
無作為に数字や文字を出したりできる。

scratchの「〇から〇までの乱数」
ブロックと同じ

random関数を使用するには、randomモジュールを呼び出す必要がある。
random関数を使う前に「import random」と書くことでrandom関数を呼び出す。
つまり、「import random」はrandom関数を召喚する魔法の呪文だ！

ランダム関数を呼びだしたら、random.randint(〇.△)と書くことで〇~△の範囲にある整数をランダムで出すことができるよ。

すごろくゲームプログラムを作る

if文を使ってゴールに到達したことを判定しよう。

```
import random          ・・・・ランダム関数を呼び出す
プレイヤ = 1           ・・・・変数”プレイヤ”に1を代入する
CPU = 1                ・・・・変数”CPU”に1を代入する
def 盤面():            ・・・・関数“盤面”を定義する
    print(“□”*(プレイヤ-1) + “P” + “□”*(30-プレイヤ) +”ゴール”)
                       ・・・・関数“盤面”の中身「P□□□□□□□□□・・□□□□ゴール」を表示する
    print(“□”*(CPU-1) + “C” + “□”*(30-CPU) +”ゴール”)
                       ・・・・関数“盤面”の中身「C□□□□□□□□□・・□□□□ゴール」を表示する
盤面()                 ・・・・関数“盤面”を呼び出す
print(“すごろく、スタート！”)  ・・・・「すごろく、スタート」を表示する
while True:                    ・・・・条件が満たされるまで以下の処理を無限に繰り返す
    input(“Enterを押すとコマが進みます”)        ・・・入力を受け付ける
    プレイヤ = プレイヤ + random.randint(1,6)
                       ・・・変数プレイヤに1+(1～6)までの乱数を代入する
    if プレイヤ > 30:       ・・・もし、プレイヤが30より大きくなったとき、プレイヤは30
        プレイヤ = 30           のままにしておく
    盤面()              ・・・・関数“盤面”を呼び出す
    if プレイヤ ==30:       ・・・・もしプレイヤが30のとき「あなたの勝ちです！」と
        print(“あなたの勝ちです！”)     表示する。
        break           ・・・・処理を中断する。
```

次のページに続く

すごろくゲームプログラムを作る

```
    input(" Enterを押すとコマが進みます" ) ・・・入力を受け付ける
    CUP = CPU + random.randint(1,6) ・・変数プレイヤに1+(1～6)までの乱数を代入する
    if CPU > 30: ・・・もし、CPUが30より大きくなったとき、CPUは30
        CPU = 30          のままにしておく
    盤面() ・・・・関数 " 盤面" を呼び出す
    if CPU == 30:                    ・・・・もしCPUが30のとき
        print(" コンピュータの勝ちです!" ) 「コンピュータの勝ちです！」と表示する。
        break ・・・・処理を中断する。
```

単語帳　～英語も一緒に覚えてしまおう！～

■import:取込む、■random:でたらめの、■define:定義する

■print:印刷する、■while:～する間、■True：正しい

■input：入力、■if:もし~なら、■break:中断する

すごろくゲームプログラムを作る

「Run Module」またはF5キーを押してプログラムを実行する。

P□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ゴール
C□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ゴール
すごろく、スタート
Enterを押すとコマが進みます
□□□□□□P□□□□□□□□□□□□□□□□□□□□□□□ゴール
C□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ゴール
Enterを押すとコマが進みます
□□□□□□P□□□□□□□□□□□□□□□□□□□□□□□ゴール
□□□□C□□□□□□□□□□□□□□□□□□□□□□□□□ゴール
・
・
・
Enterを押すとコマが進みます
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Pゴール
□□□□□□□□□□□□□□□□□□□□□□□□□□□C□□ゴール
あなたの勝ちです！
>>>

if文を理解する

if文を使えば、条件分岐のプログラムを作ることができる。

scratchの
「もし〇〇ならXX」ブロックと同じ

```
if 条件式(〇〇):      ・・・・もし〇〇なら　（条件分岐）
    print(" XX" )     ・・・・条件を満たす場合、XXと表示する
```

※「：」(コロン)の後に改行してインデント（半角スペース）を付ける。
インデントの範囲内でしか、条件式は有効とならないことを知っておこう。

復習＆チャレンジ

ここまで習ったことをScratchでもできるかチャレンジしてみよう。
その過程でScratchでできること、Pythonでないとできないことを整理してみよう。
例えば、データの型という概念はscratchにはあっただろうか？

メモ

プログラミング教室のテクノロ

なまえ：