モンティー・ホール問題から

プログラミングを学ぼう！

KUMEC
Kyoto University Medical Engineers Club

基礎編

モンティー・ホール問題とは？

直感で正しいと思える解答と、論理的に正しい解答が異なる問題

アメリカでモンティー・ホールが司会を務めるゲームショー番組「Let's make a deal」で行われたゲームのひとつ。

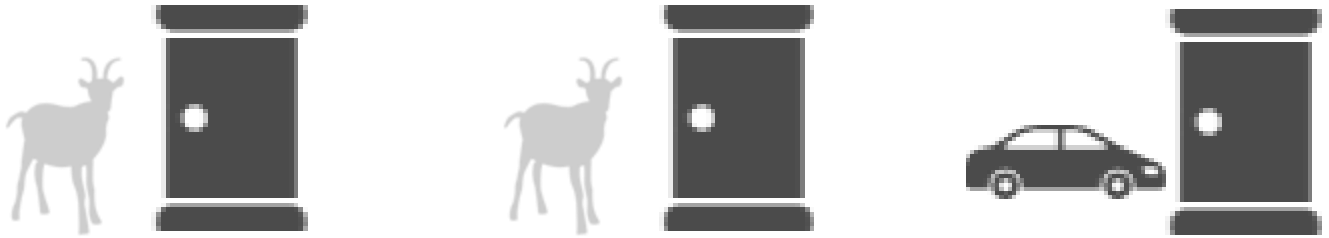
３つのドアの向こうに、当たりのスポーツカーとハズレのヤギが入っている。

回答者は、まず３つのドアの中から、当たりと思う一つのドアを選ぶ。

司会者モンティーは、選ばなかった２つのドアからハズレのドア一つを開ける。

回答者は、残ったドアからもう一度ドアを選ぶチャンスがもらえる。

このとき、回答者はドアを選び直した方がよいか否か。



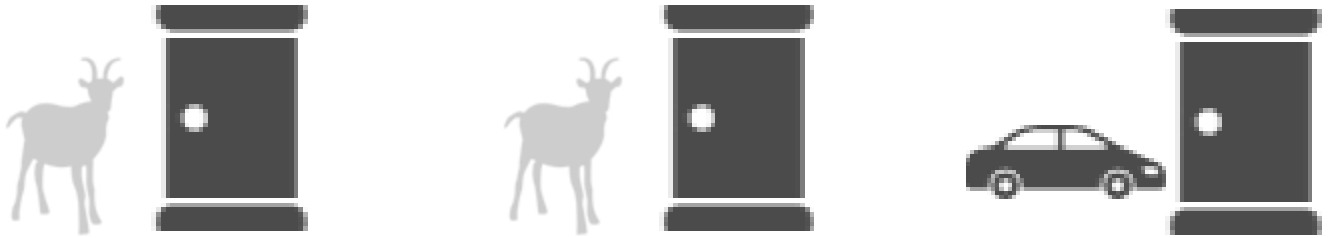
モンティー・ホール問題とは？

モンティー・ホール問題とは？

モンティー・ホール問題の証明

当時、最も高いIQとして知られていたマリリン・ヴォス・サヴァントは、「変えるべき」と言ったが、他の数学者たちはそんなわけないと反論した

- マリリンが連載していたコラム「マリリンにおまかせ」で、この問題が投稿され、「変えると当たる確率が２倍」と答えたことによって、論争が勃発
- この回答が誤っているという投書が１万通も殺到し、その中には博士もたくさんふくまれていた。



モンティー・ホール問題の証明

PCを使って数百回シミュレーションを行い、マリリンの主張が正しいと証明された

- ロバート・サッチス博士「プロの数学者として、一般大衆の数学的知識の低さを憂慮する。自らの間違いを認める事で現状が改善されます」
- スコット・スミス博士「君は明らかなヘマをした（中略）世界最高の知能指数保有者である貴女が自ら数学的無知をこれ以上世間に広める愚行を直ちに止め、恥を知るように！」
- E・レイ・ボボ博士「（前略）現在、憤懣やるかたない数学者を何人集めれば、貴女の考えを改める事が可能でしょうか？」
- プロ数学者ポール・エルデシュの弟子だったアンドリュー・ヴァージョニが本問題を自前のパーソナルコンピュータでモンテカルロ法を用いて数百回のシミュレーションを行うと、結果はサヴァントの答えと一致。
- その後、カール・セーガンら著名人らがモンティーホール問題を解説、サヴァントの答えに反論を行なっていた人々は、誤りを認める。

どうやって実装する？

このモンティーホール問題を実際にシミュレーションで証明しよう！

今回は、モンティーホール問題に即したプログラムを実際に作成し、それぞれ100回ぐらいシミュレーションを行って、勝率にどれぐらい差が出るか確認してみる。

※実際は、数学的に正しくするため、モンテカルロ法を用いるが、今回はプログラムを簡単にするため、数学的正しさは考えないこととする

モンティーホール問題

- ３つのドアに当たりが一つ入っている。
- 回答者は、まず３つのドアの中から、１つドアを選ぶ。
- 司会者モンティーは、選ばなかった２つのドアからハズレのドア一つを開ける。
  - もし回答者が当たりを選んでいたら、２つのハズレのうちランダムにドアを一つ開ける
  - もし回答者がハズレを選んでいたら、ハズレのドアを開ける
- 回答者は、モンティーが残したドアを選び直す or ランダムに残されたドアを選ぶ

どうやって実装する？

この問題をどうやってコードで書けばいい？

まずは、コードで書けるように、どんなフローチャート・タスクにわけられる？
(使用するプログラミング言語はPythonを想定しています。)

モンティーホール問題

- ３つのドアに当たりが一つ入っている。
- 回答者は、まず３つのドアの中から、１つドアを選ぶ。
- 司会者モンティーは、選ばなかった２つのドアからハズレのドア一つを開ける。
  - もし回答者が当たりを選んでいたら、２つのハズレのうちランダムにドアを一つ開ける
  - もし回答者がハズレを選んでいたら、ハズレのドアを開ける
- 回答者は、モンティーが残したドアを選び直す or ランダムに残されたドアを選ぶ

私が考えたフロー

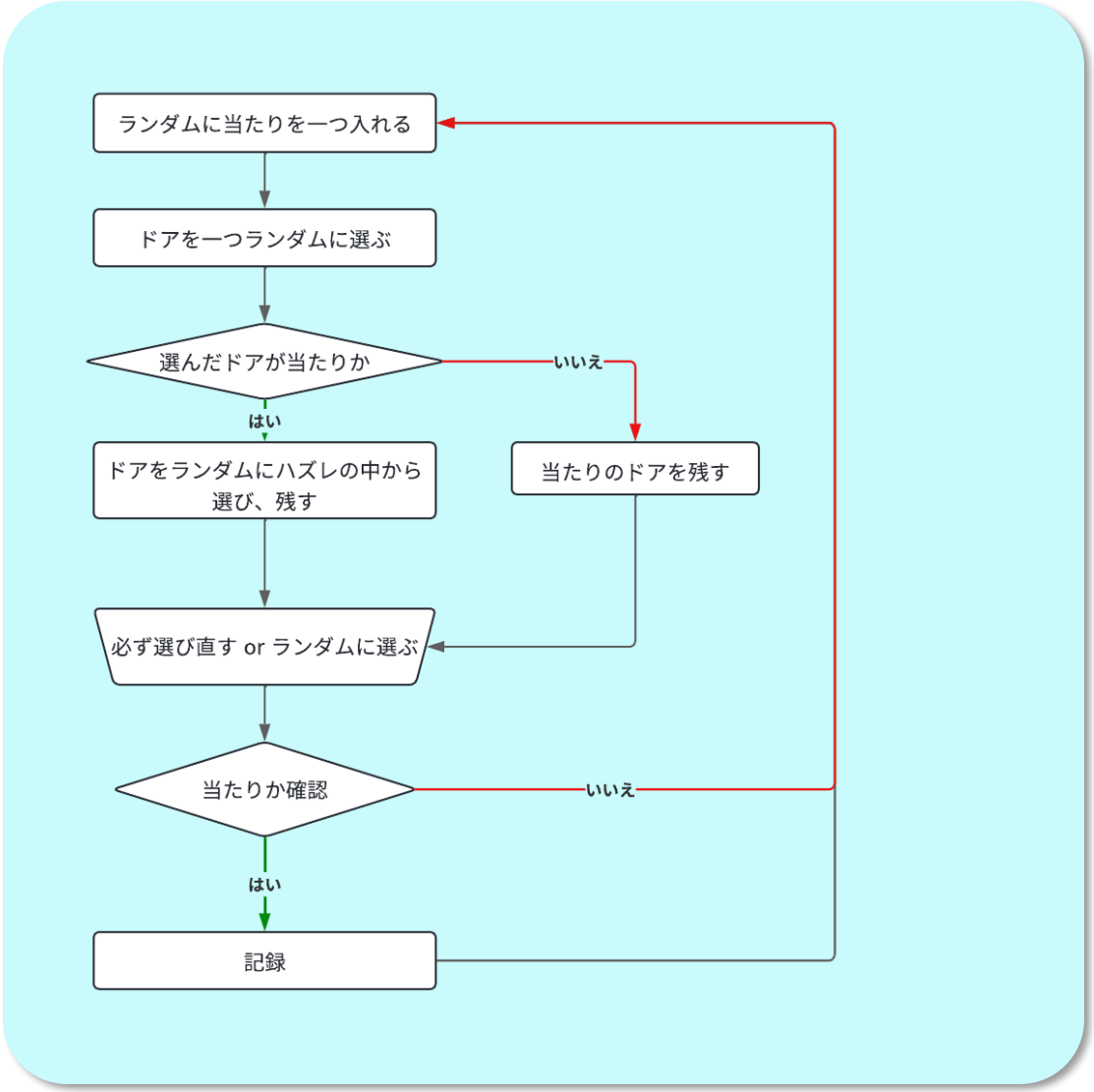
タスクをこう分けてみた！(一例)

モンティーホール問題

- ３つのドアからランダムに当たり１つを入れる
- ドアを一つ選ぶ
- モンティーが、選ばれなかったドアのうち、全てのヤギのドアを開ける(１つは残す)
  - もし回答者が当たりを選んでいたら、1つハズレのドアを残す
  - もし回答者がハズレを選んでいたら、当たりのドアを残す
- モンティーが残したドアを必ず選び直す or ランダムに残されたドアを選ぶ
- 選択したドアが当たりかどうか確認し、当たり引いたことを記録
- これを100回繰り返す

私が考えたフロー

タスクをこう分けてみた！(一例)



初心者が学びやすいプログラミング言語

1991年にグイド・ヴァン・ロッサムにより開発されたプログラミング言語
KUMECのロゴのヘビも実はpythonを意識している

特徴

- Google Colaboratoryで簡単に試せる
- AIや機械学習の分野に強い言語

KUMEC
Kyoto University Medical Engineers Club

python™

Google Colaboratory

Google Colaboratoryの使い方

KUMEC

学習教材

京大：プログラミング演習 Python 2023 ・ **東大Pythonプログラミング入門**

KUMEC
Kyoto University Medical Engineers Club

python™

Google Colaboratory

変数・関数とは？

関数(メソッド)は、数学のf(x)と似ている！

例えば：$f(x) = x^2 + 2x + 1$

- プログラミングでは、このxに当たる部分をこの関数の「引数」という
- 数学でもf(x)をg(x)やh(x)にしたように、プログラミングでもわかりやすい名前をつけていい(ex. func() etc..)
- ほかにも、print(x)など、すでに準備されているメソッドがたくさん

Google Colaboratoryでprint(“Hello world”)と入れて実行してみよう

変数・関数とは？

変数とは、(基本的に)何でも入る箱

例えば：$x^2 + 2x + 1$

- この x みたいなかんじ

```
x = 1
y = x + 10
print(y)
```

```
x = 1
y = f (x)    #f(x)はx+10を返す関数
print( y )
print( f(x) )
```

変数・関数とは？

変数とは、(基本的に)何でも入る箱

例えば：$x^2 + 2x + 1$

- このxみたいなかんじ

```
x = “World”
y = “Hello” + x  + “!”
print(y)
```

計算してみよう！

四則演算は、数学の記号と異なるものもある

x + y	x と y の和
x - y	x と y の差
x * y	x と y の積
x / y	x と y の商
x // y	x と y の商を切り下げたもの
x % y	x / y の剰余
-x	x の符号反転
+x	x そのまま
x ** y	x の y 乗

乱数を生成してみよう！

pythonでは、簡単に乱数を発生させられる

```
import random　#乱数を使うときの(とりあえず)呪文

x = random.randint(10, 20)

print(x)
```

randint(最低値、最高値)でこの範囲の整数の乱数を生成してくれる
他にもいろんなメソッドがあるから、調べてみてね

タスク：その１-1

乱数を使って、0,1,2の中から1つランダムに出力するコードを書いてみよう

タスク：その１-2

乱数を使って、0,1の中から1つランダムに出力するコードを書いてみよう

タスク1:関数・メソッドにしよう

タスク1の関数を作ってみよう

```
def 関数名(引数1, 引数2...):
  関数化したい処理

  return 返り値
```

引数がない場合は()の中身は何も書かなくて良い
関数名は、わかりやすい名前にすると、後々使いやすい

bool型とは？

プログラミング言語にはbool型：True or Falseがある

型というものがコードにはある(今回は重要でないため割愛)
bool型は文字ではなく**True**と**False**という特別な値があるということ

今回の当たり・はずれ、あるorない、のような２つしかないものを表すのに使われる。

毎回、変数に”sport car”とか”goat”を入れるのは大変

配列とは？

変数をまとめて扱うもの

たとえば、体重のデータ5人分があるとき、

weight1 = 62, weight2 = 73, weight3=34, weight4=55, weight5=80

のように変数をたくさん作るの大変

配列を使うと…

weights = [62, 73, 34, 55, 80]

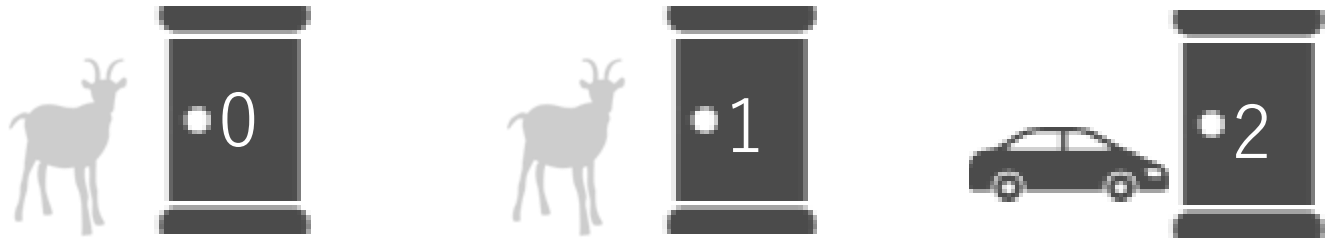
とすれば、扱いやすい！

weights[2]とすれば、変数と同じように扱える

注意：
配列の番号は 0 始まり

タスクその２

３つのドアの中に１つの当たりをランダムに入れよう



doors = [True, False, False]のような配列を作れば良さそう
ランダムに入れるときに、タスク1の関数が使えそう

できたら、タスク2も関数化しよう！

タスクその3

3つのドアの中から一つ選ぶ処理を作ろう

配列から要素を除く

N番目の配列の要素を除くときは、delを使う

```
array = [1,2,3,4,5,6,7,8]

print(array)    #[1,2,3,4,5,6,7,8]

del array[3]      #array[3] = 4

print(array)   #[1,2,3,5,6,7,8]
```

If文とは？

条件分岐をしたいときに使う構文

```
x = 1

if  x == 1:
    print(x)

 # 以下説明
if 条件式:     #コロンを忘れずに！
    なんか処理      #段落(インデント)を下げるのも忘れずに！
```

条件式に使うのは
- ＝＝(等しいか？)
- ＞
- ＜

など

If文とは？

条件分岐をしたいときに使う構文

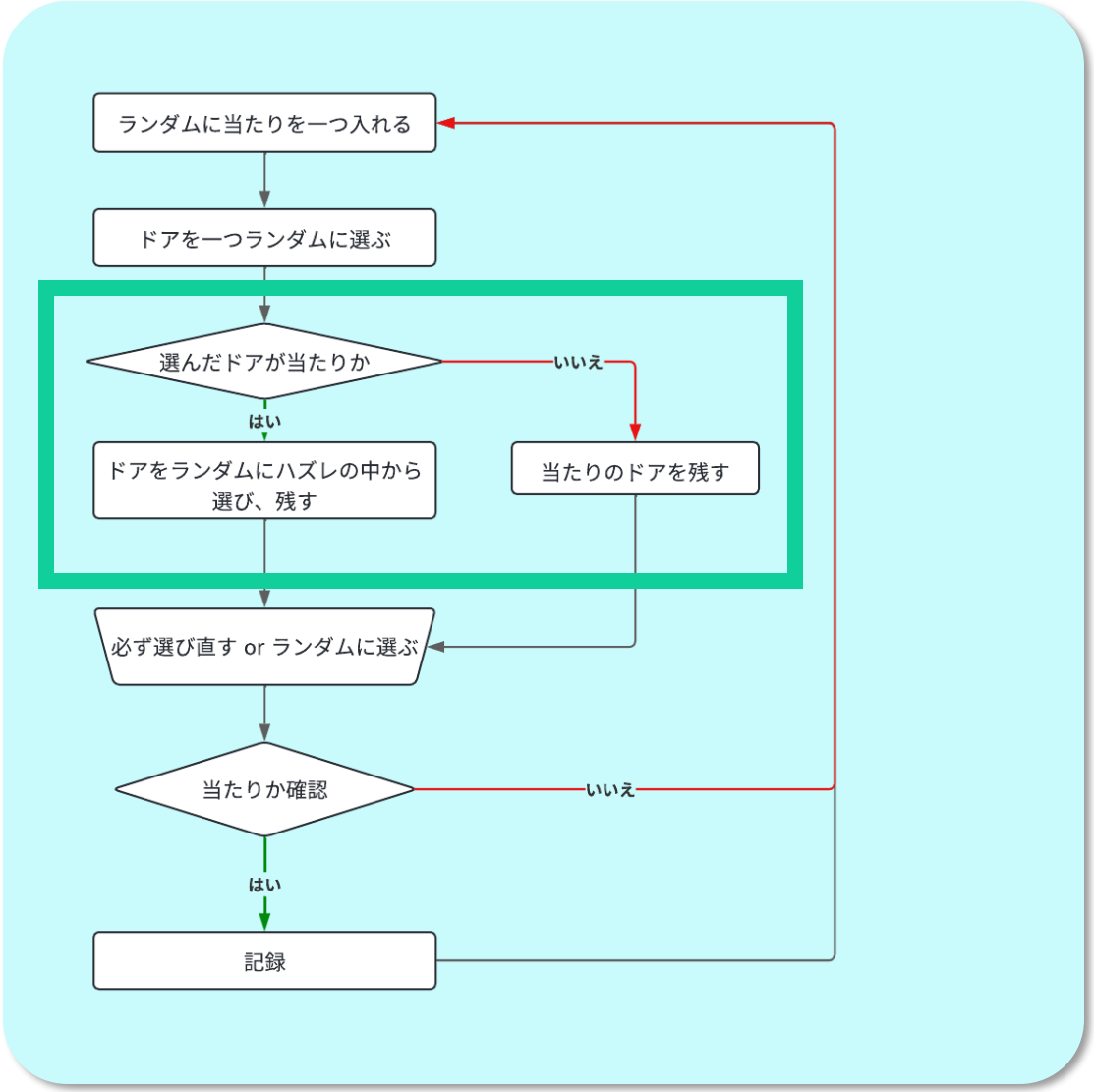
```
x = 1

if  x < 1:
    print(“1より小さい”)
elif  x < 2:
    print(“２より小さい”)
elif  x < 3:
    print(“3より小さい”)
else:
    print(“3以上”)
```

```
if 条件式:
    なんか処理
elif 条件式:
    なんか処理
else:
    なんか処理
```

タスクその４

If文を駆使して、司会者モンティーがドアを残す(ハズレのドアを開ける)処理を作ろう！



タスクその５-1

マリリンの言う通りに、
選択を必ず変える処理を作ろう！

タスクその5-2

マリリンの言うことを信じずに
ランダムに選び直す処理を作ろう！

for文とは

繰り返し同じ処理をさせたいときに使う構文

```
for x in [0,1,2,3]:
    print(x)
    #0 1 2 3

for x in [0,1,2,3,4,5]:
    print(x)

    if x> 2:
       break

#0,1,2
```

```
for 変数 in 配列:
    処理


for 変数 in 配列:
    処理

    if 条件式:
        break
```

for文とは

100回同じ処理をさせたいときも、100個の要素がある配列が必要？

```
for x in range(5)
    print(“Hello”)
    #Hello Hello Hello Hello Hello

for x in range(5):
    print(x)

#0,1,2,3,4
```

```
for 変数 in range(n):
    処理   #n回繰り返される
```

range(n)で生成される数字はn-1まで

for文とは

x = x + 1とは、$x_{n+1} = x_n + 1$

```
x = 0
for i in range(5):
    x = x + 1
    print(x)

#1,2,3,4,5
```

```
for 変数 in range(n):
    x = x + 1
# x+1したものが新たに代入される
```

range(n)で生成される数字はn-1まで

タスクその6

今までのメソッドを全て合わせて一通りの処理を作ってみよう！

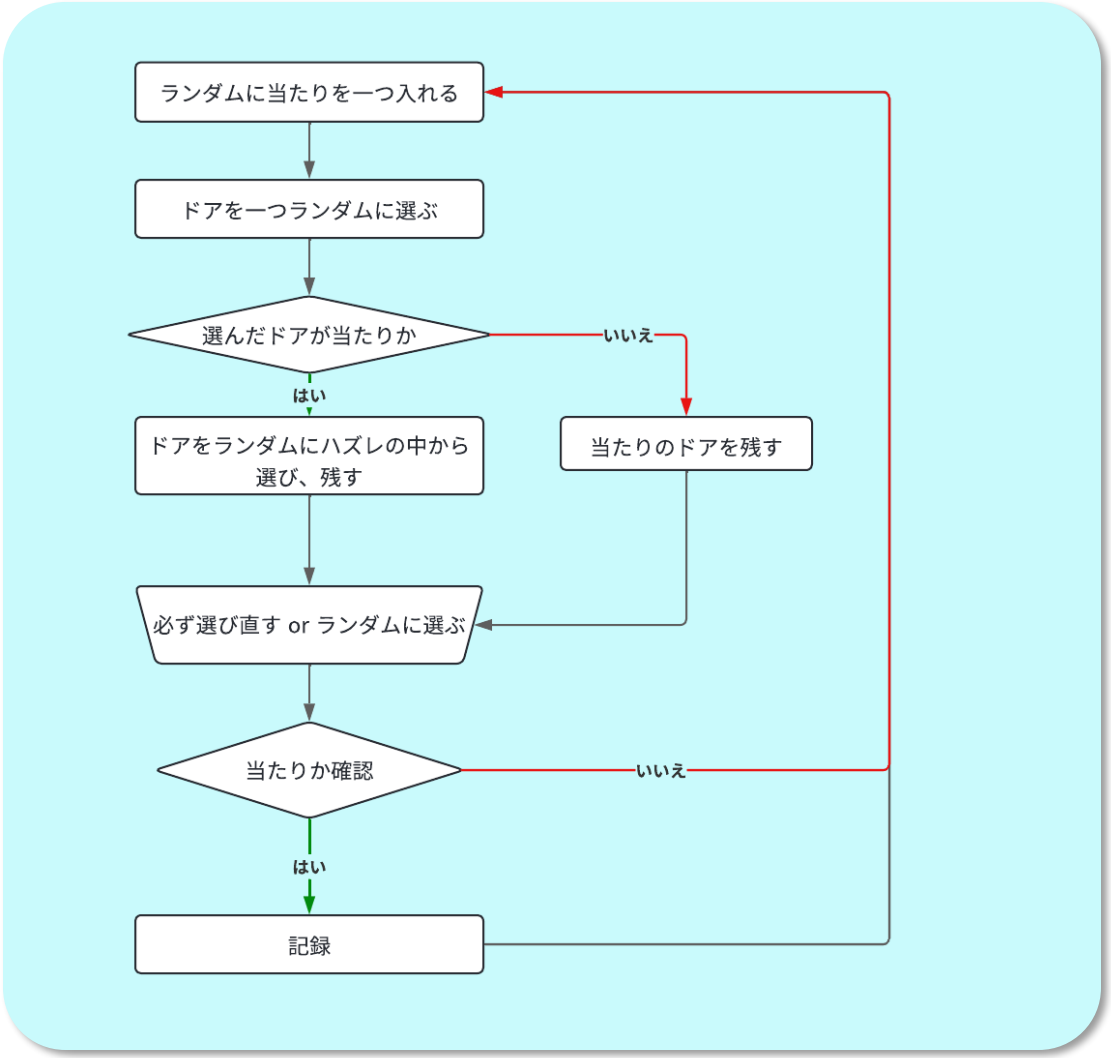
全体の処理を100回繰り返して、当たった回数を出力しよう

タスクその7

ドアの数を増やせるように、今までのメソッドを作り変えよう

モンティーが回答者のために残すドアは一つだけ

100個ドアがあれば、回答者が選ばなかった99個のドアのうちヤギが入っている98個のドアを開ける



クラスとは

変数やメソッドをまとめて使いやすくしたもの

```
class Car:
    def __init__(self, color, size):
        self.color = color
        self.size = size
        self.tire_num = 4

    def set_color(self,color):
        self.color = color
        print(color)

    def clean(self):
        self.set_color(“white”)
        print(“きれいになった”)
```

```
Prius = Car(“Blue”, 1.78)
Prius.clean()
```

```
class クラス名:
    def __init__(self,引数1):    #呪文
        self.変数1 = 引数1
        self.変数2 = 0

    def 関数名(self, 引数):
        処理
        変数 = self.変数1

#実際に使うとき
変数 = クラス名(引数1)
クラス名.関数名(引数)
```

タスクその8

マリリン信者クラスとマリリンアンチクラスを作ってみよう！

マリリン、アンドリュー、カールを
マリリン信者クラスのインスタンスとして作ってみよう

ロバート、スコット、レイを
マリリンアンチクラスのインスタンスとして作ってみよう

タスクその9

親クラスを作ろう！

マリリン信者クラスとマリリンアンチクラスでは、ほとんどのメソッドが同じ

親クラスを作って、これらのメソッドをまとめよう！

そして、信者とアンチに親クラスを継承させよう

タスク番外編

Matplotlibを使って、それぞれのインスタンスの勝率を棒グラフで作ってみよう