1

乱数とグラフの基本

ー　シミュレーションプログラム（モンテカルロ法の基礎）　ー

学習目標とキーワード

学 習 目 標　モンテカルロ法によるシミュレーションプログラムに必要な乱数生成関数と結果の表示に便利なグラフ描画関数を使えるようにしましょう

キーワード　シミュレーション　モンテカルロ法　確率　乱数　グラフ

1　モンテカルロ法とは

複雑な数理的な問題を、乱数を用いた試行（シミュレーション）を繰り返すことで、近似的な解を求める方法です。試行数が多いほど正確な値に近づくので、繰り返しが得意なコンピュータによる処理に適しています。名前はカジノで有名なモナコ公国の都市モンテカルロから付けられました。

2　乱数の生成

シミュレーションには乱数を使用します。生成する数は、実数（浮動小数点）と整数、生成する数の範囲が固定あるいは任意に指定できる関数があります。ここでは代表的な関数を紹介します。

プログラム

```
#ランダム関数
import random                       #ランダム関数のモジュールを読み込む
print(random.random())              # 0≦X＜1の実数（浮動小数点）を生成
print(random.uniform(10,50))        #任意の範囲（ここでは10≦X≦50)の実数（浮動小数点）を生成
print(random.randrange(10))         # 0≦＜10の整数を生成
print(random.randrange(9,18,3))     #任意の範囲のステップの整数
                                    #(ここでは9≦X＜18で３つおき9,12,15）を生成
print(random.randint(7,9))          #任意の範囲のステップの整数（ここでは7≦X≦9　7,8,9）を生成
```

出力

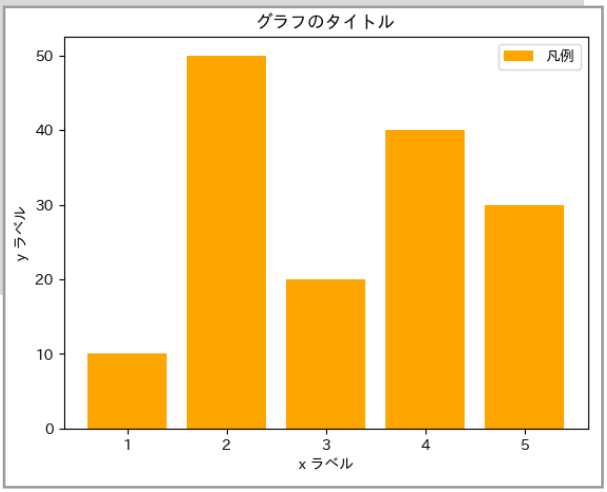
```
0.7401559475261847
13.18096667699753
5
9
7
```

乱数なので出力は毎回異なる値です

3　棒グラフの描画

シミュレーションの結果をグラフで表示することがあります。ここでは棒グラフの描画方法を説明します。

```
#棒グラフ
!pip install japanize-matplotlib  #日本語を使うモジュールのインストール（ブック内で1回実行）
import matplotlib.pyplot as plt   #グラフを使うライブラリ　名前をpltにします
import japanize_matplotlib        #日本語を使うライブラリ
x = [1,2,3,4,5]                   #X軸データのリスト
y = [10,50,20,40,30]              #Y軸データのリスト
plt.bar(x,y,color="orange",label="凡例")#棒グラフの描画
plt.xlabel("x ラベル")            #X軸のラベル
plt.ylabel("y ラベル")            #Y軸のラベル
plt.title("グラフのタイトル")     #タイトル
plt.legend()                      #凡例を表示する
plt.show()                        #グラフを表示する
```

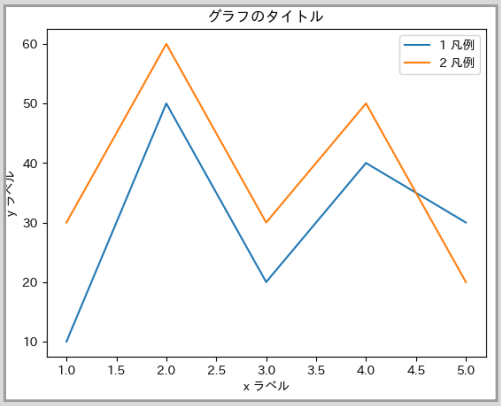


・棒グラフは、XとYのデータリストを用意して plt.bar() で描画します
・色、凡例、ラベル、タイトルは省略可能です
・日本語使う場合は、日本語モジュールのインストールとライブラリのインポートをします。インストールはブックを最初に実行する時に行います
２回目以降は行の先頭に＃を記入してコメント文にすると実行が速くなります

4 折れ線グラフの描画

折れ線グラフの描画準備は棒グラフの描画方法と同じです。

```
#折れ線グラフ
#!pip install japanize-matplotlib #日本語を使うモジュールのインストール（初回実行は#を取る）
import matplotlib.pyplot as plt   #グラフを使うライブラリ　名前をpltにします
import japanize_matplotlib        #日本語を使うライブラリ
x = [1,2,3,4,5]                   #X軸データのリスト
y1 = [10,50,20,40,30]             #Y1軸データのリスト
y2 = [30,60,30,50,20]             #Y2軸データのリスト
plt.plot(x,y1,label="1 凡例")     #Y1グラフの描画
plt.plot(x,y2,label="2 凡例")     #Y2グラフの描画
plt.xlabel("x ラベル")            #X軸のラベル
plt.ylabel("y ラベル")            #Y軸のラベル
plt.title("グラフのタイトル")     #タイトルを表示する
plt.legend()                      #凡例を表示する
plt.show()                        #グラフを表示する
```

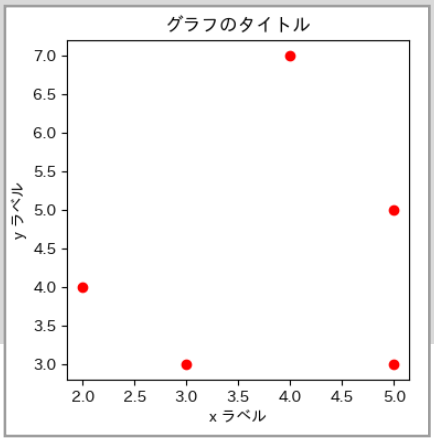


・折れ線グラフは、XとYのデータリストを用意して plt.plot() で描画します
・複数のデータを表示する時は、複数のリストを用意して、複数のplt.plot() で描画します。

5 散布図の描画

散布図で2次元データを描画しましょう。

```
#散布図
#!pip install japanize-matplotlib #日本語を使うモジュールのインストール（初回実行は#を取る）
import matplotlib.pyplot as plt   #グラフを使うライブラリ　名前をpltにします
import japanize_matplotlib        #日本語を使うライブラリ
x = [2,5,3,4,5]                   #X軸データのリスト
y = [4,5,3,7,3]                   #Y軸データのリスト
plt.figure(figsize=(4,4))         #グラフの幅と高さの指定
plt.scatter(x,y,c="red")          #散布図の描画　色は省略可
plt.xlabel("x ラベル")            #X軸のラベル
plt.ylabel("y ラベル")            #Y軸のラベル
plt.title("グラフのタイトル")     #タイトルを表示する
plt.show()                        #グラフを表示する
```

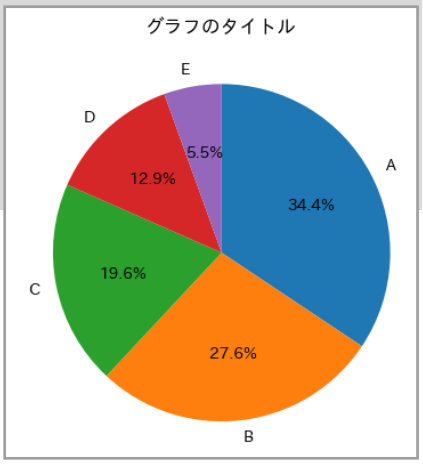


・散布図は、XとYの組のデータリストを用意して plt.scatter() で描画します

6 円グラフの描画

円グラフで割り合いのデータを描画しましょう。

```
#円グラフ
#!pip install japanize-matplotlib #日本語を使うモジュールのインストール（初回実行は#を取る）
import matplotlib.pyplot as plt   #グラフを使うライブラリ　名前をpltにします
import japanize_matplotlib        #日本語を使うライブラリ
labels = ["A","B","C","D","E"]    #ラベルのリスト
data = [56,45,32,21,9]            #データのリスト
plt.pie(data, labels=labels, autopct='%1.1f%%',¥
        counterclock=False, startangle=90)     #円グラフの描画
plt.title("グラフのタイトル")     #タイトルを表示する
```



・円グラフは plt.pie() で、データの割り合いを自動的に計算して描画します
・円グラフ plt.pie() の基本パラメータ
　data ：表示データのリスト
　labels=labels：ラベルを表示する
　autopct='%1.1f%%': 小数点1位まで表示
　counterclock=False：反時計回りに円を描く　時計回りはTrue
　startangle=90　：　データの描き始めを上方向（90°）からにする

※ グラフ描画の詳しい説明URL　https://qiita.com/ground0state/items/9ac0f159276d8904cb27
(2024.3 確認)　　https://pythonsoba.tech/matplotlib/

2

クラスに同じ誕生日の人がいる確率

ー シミュレーションプログラム（モンテカルロ法） ー

学習目標とキーワード

学 習 目 標　「クラスに同じ誕生日の人がいる確率」をテーマに、事象のモデル化してモンテカルロ法によるシミュレーションプログラムを作成する手法を理解しましょう。また、グラフによる視覚的表現ができるようにしましょう。

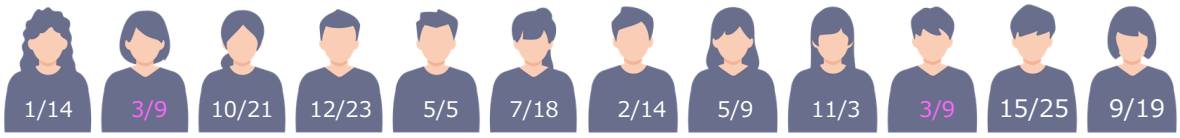
キーワード　シミュレーション　モンテカルロ法　乱数　グラフ

テーマ　クラスに同じ誕生日の人がいる確率

「40人のクラスで、同じ誕生日の人が少なくとも２人以上いる確率」はどれくらいだと思いますか？

365日（ここではうるう年は外します）からランダムに40日を選んだとき、それらの日が重複する確率は小さいと思いませんか？

モンテカルロ法のシミュレーションプログラミングで計算してみましょう。



1　シミュレーションの方針（モデル化）

（1）クラスメイトは偏りの無い確率で1年365日のいずれかの日に誕生したとします。
　　計算を簡単にするための、うるう年の2月29日は除く、産院が休診で出産が少ないとされる祝日などの偏りも考慮しない、と言うモデル化をします。

(2) 誕生日として１～365の一様乱数（偏りが無い乱数）をクラスの人数分生成します。

(3) 生成した乱数を調べ、重複があったか、重複が無かったか判断します。

(4) (2)と(4)を数多く繰り返します。例えば1000回。

(5) (4)の繰り返した数の内、重複があった回数を集計して、同じ誕生日の人がいる確率とします。

(6) クラスの人数を1～60人に変更（シミュレーション）して確率を求めます。

(7) （4）の繰り返し回数も変更（シミュレーション）して確率を求めます。

2　プログラムを実行してみましょう

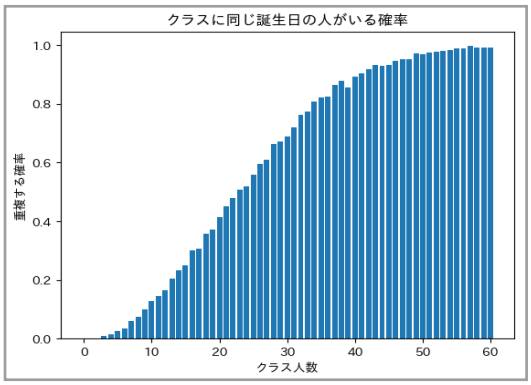
```
#クラスに同じ誕生日の人がいる確率
#!pip install japanize-matplotlib      #日本語を使うモジュールのインストール（初回実行は#を取る）
import matplotlib.pyplot as plt        #グラフのライブラリをインストール　名前をpltにします
import japanize_matplotlib             #グラフの日本語化ライブラリをインストール
import random                          #乱数ライブラリをインストール

#リストの重複を調べる関数の定義　重複があるとTuleを返す
def has_duplicates(seq):               #リストを引数として受け取る
    return len(seq) != len(set(seq))   #len()はリストの要素数を返す　set()はリストの重複要素を取り除く
                                       #重複がある場合、要素数が少なくなるのでTuleを返す

#変数・リストの初期設定
クラス定員 = 60                        #調べる最大クラス人数
試行回数 = 1000                        #シミュレーションの回数
誕生日 = []                            #誕生日のリスト定義
確率 = []                              #誕生日が重複する確率

#シミュレーション
for クラス人数 in range(クラス定員+1):  #クラス人数を0～クラス定員まで変える
  重複 = 0                             #重複回数の初期化
  for 試行 in range(試行回数):         #重複を試行回数調べる
    誕生日 = []                        #誕生日リストの初期化
    for _ in range(クラス人数):                #クラス人数分誕生日を生成する
      誕生日.append(random.randint(1, 365))    #誕生日を生成してリストにする
    if has_duplicates(誕生日) == True:         #誕生日に重複があるか調べる
      重複 = 重複 + 1                          #重複があればその回数を累計する
  print("クラス人数=",クラス人数,"重複する確率=",重複/試行回数)#クラス人数と重複する確率を表示する
  確率.append(重複/試行回数)           #確率のリストを作る
```

つづく

```
#グラフ表示
x=range(len(確率))                          #確率のデータ個数だけx軸のリストを作る
plt.bar(x,確率)                             #確率の棒グラフを描く
plt.xlabel("クラス人数")                    #X軸のラベル
plt.ylabel("重複する確率")                  #Y軸のラベル
plt.title("クラスに同じ誕生日の人がいる確率") #タイトルを表示する
plt.show()                                  #グラフの表示
```

出力

```
    略
クラス人数= 37 重複する確率= 0.864
クラス人数= 38 重複する確率= 0.878
クラス人数= 39 重複する確率= 0.855
クラス人数= 40 重複する確率= 0.892
クラス人数= 41 重複する確率= 0.903
クラス人数= 42 重複する確率= 0.918
    略
```



結果　40人のクラスでは、同じ誕生日の人がいる確率は約0.89です。　グラフを見ると、23人クラスでの確率が約0.5で60人以上では1.0に近い確率になります。

課題 2-1

(1) 試行回数を1000回から10000回に変更するとグラフはどのように変化しますか。

(2) 試行回数が1000回と10000回では同じ誕生日になる確率はどのように変化しますか。

3　計算で確率を求めてみましょう

クラスに同じ誕生日の人がいるケースは、２人が同じ、３人が同じ、2人が同じが二組・・・と数多くの組み合わせがあり計算が複雑になります。そこで、同じ誕生日の人がいないケースの確率を求め、その余事象として同じ誕生日の人がいる確率を求めます。

他の人と同じ誕生日にならない確率

$$\underbrace{\frac{365}{365}}_{\text{1人目}} \times \underbrace{\frac{364}{365}}_{\text{2人目}} \times \underbrace{\frac{363}{365}}_{\text{3人目}} \times \underbrace{\frac{362}{365}}_{\text{4人目}} \times \cdots \times \underbrace{\frac{326}{365}}_{\text{40人目}} = 0.109$$

1人目	2人目	3人目	4人目	40人目	
365日の内、同じ誕生日にならない日は365日	365日の内、同じ誕生日にならない日は１人目の誕生日を除いた364日	365日の内、同じ誕生日にならない日は前2人の誕生日を除いた363日	365日の内、同じ誕生日にならない日は前3人の誕生日を除いた362日	365日の内、同じ誕生日にならない日は前39人の誕生日を除いた326日	40回確率を積算すると、同じ誕生日にならない確率になる

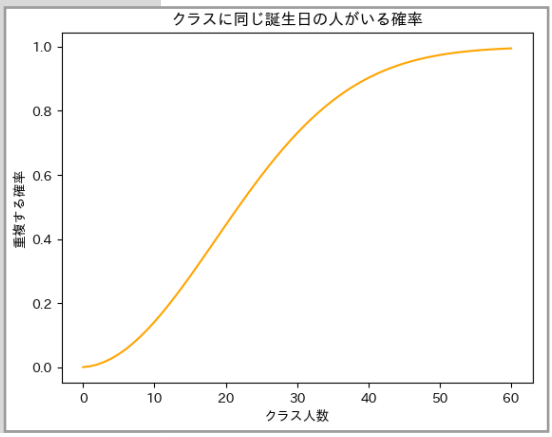
他の人と同じ誕生日になる確率

$$1 - 0.109 = 0.891 \quad 89.1\%$$

他の人と同じ誕生日になる確率は、同じ日にならない確率の余事象なので、確率１から引いた値になります。

応用　計算で確率を求めるプログラムを実行してみましょう

```
#クラスに同じ誕生日の人がいる確率
#!pip install japanize-matplotlib      #日本語を使うモジュールのインストール（初回実行は#を取る）
import matplotlib.pyplot as plt        #グラフのライブラリをインストール　名前をpltにします
import japanize_matplotlib             #グラフの日本語化ライブラリをインストール

#変数・リストの初期設定
クラス定員 = 60                        #調べる最大クラス人数
重複確率 = []                          #誕生日が重複する確率
非重複確率 = 0                         #誕生日が重複しない確率
非重複累積確率 = 1                     #誕生日が重複しない確率の累積
#シミュレーション
for クラス人数 in range(クラス定員+1):        #クラス人数を0～クラス定員まで変える
  非重複確率 = (365 - クラス人数)/365         #誕生日が重複しない確率の計算
  非重複累積確率 = 非重複累積確率 * 非重複確率 #誕生日が重複しない確率の累積
  重複確率.append( 1 - 非重複累積確率)        #重複する確率を余事象として計算して
                                            #リストに追加
  print("クラス人数=",クラス人数+1,"重複する確率=",(1 - 非重複累積確率))
                                #クラス人数と重複する確率を表示する
#グラフ表示
x=range(len(重複確率))                 #確率のデータ個数だけx軸のリストを作る
plt.plot(x,重複確率,c="orange")        #確率の線グラフを描く
plt.xlabel("クラス人数")               #X軸のラベル
plt.ylabel("重複する確率")             #Y軸のラベル
plt.title("クラスに同じ誕生日の人がいる確率") #タイトルを表示する
```



3

円周率を求めよう

ー　シミュレーションプログラム（モンテカルロ法）　ー

学習目標とキーワード

学 習 目 標　モンテカルロ法で円周率を求める理論を学び、プログラムの手法を理解しましょう。また、グラフによる視覚的表現ができるようにしましょう。

キーワード　シミュレーション　モンテカルロ法　円周率　乱数　グラフ

1　モンテカルロ法で円周率を求める理論

（1）図１のように1辺２ｒの正方形とそれに接する半径ｒの円があります。ｒの長さにより正方形と円の大きさが決まるので、正方形の面積Ｓ＝2r×2r＝4ｒ^2 と 円の面積Ａ＝πｒ^{2}は比例します。これを比で表すと

正方形面積 Ｓ：円面積 Ａ ＝ 4ｒ^2：πｒ^2 ＝ 4 ：π

になります。

この比は図２のような1/4正方形と1/4円でも成り立つので、

正方形面積 Ｓ/4：円面積Ａ/4＝4/4：π/4 ＝ 1 ：π/4

になり、変形すると

Ｓ/4 × π/4 ＝ Ａ/4 × 1 → π/4＝Ａ/Ｓ → π＝４Ａ/Ｓ

になります。

しがたって、図２の正方形の面積と四分円の面積が分かれば円周率 π の値を求めることができます。

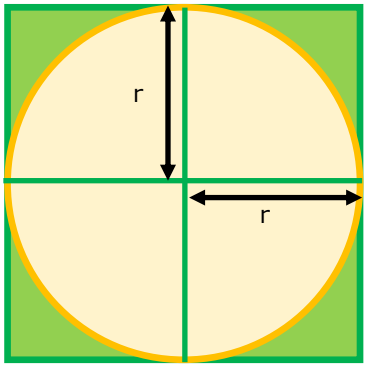


図１

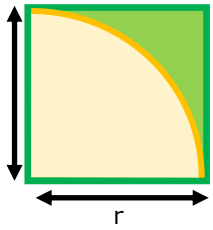


図２

（2）図３のように正方形を対角線で分割した面積が等しい三角形が２つあります。この正方形にゴマを蒔くように一様の割り合いで100の点を打つと、２つの三角形には、50：50 や 48：52のように、約1/2の点がそれぞれの領域に入ります。このシミュレーションでは「図形の面積とそこに一様に打たれた点の数は比例する」推論が成り立つことが大前提となります。

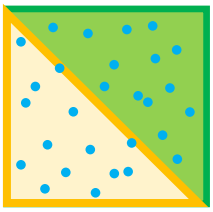


図３

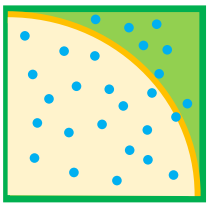


図4

（3）図４では正方形の中に四分円を描いた図形に、ランダムに点を打っています。この時、正方形に打った点の数をN、四分円の内側の点の数をMとします。点の数の比と面積の比は同じとすると（１）の式より

π＝４Ａ/Ｓ　→　π＝４Ｍ/Ｎ　→　円周率＝４×四分円内の点／正方形内の点

になり、点の数で円周率を求めることができます。

2　シミュレーションの方針（モデル化）

(1) 図５のような辺の長さ１の正方形と半径１の四分円を想定します。

(2) ランダム関数で、０以上１未満の実数ｘと実数ｙの乱数をN個生成し、点の座標（ｘ,ｙ）とします。これは、N個の点を正方形の内側に打ったことになり、正方形の面積に比例した数になります。

(3) 四分円の内側にある点は、原点と点の距離が１以下であることで判別します。

点の座標を（ｘ,ｙ）とすると、距離dは $d=\sqrt{x^2+y^2}$ になり、

ｄが１以下の点の数Ｍが四分円の面積に比例した数になります。

(4) 点の数ＮとＭを式　π＝４Ｍ/Ｎ　で計算して円周率を求めます。

(5) １回のシミュレーションでは求めた値が真の値と離れているので、打つ点の数を増やしたり、シミュレーションを何度も行い、その平均値を求めて、真の値に近づけます。

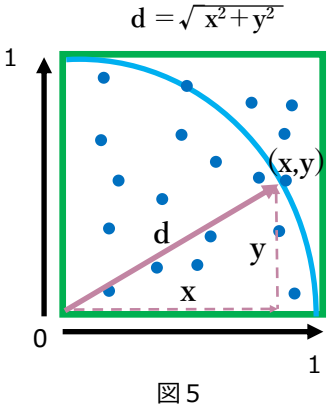
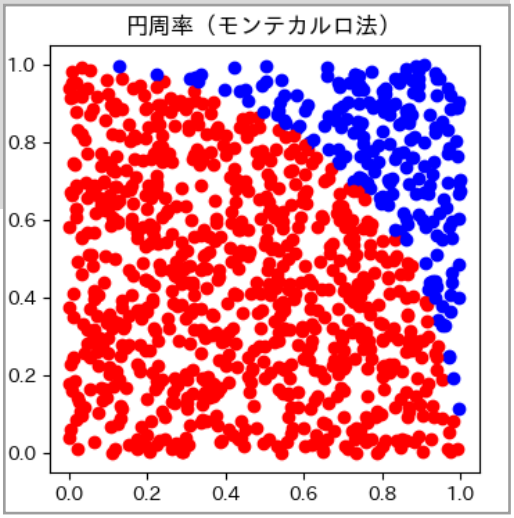


図５

3　プログラムを実行してみましょう

```
#モンテカルロ法で円周率を求める
#!pip install japanize-matplotlib #日本語を使うモジュールのインストール（初回実行は#を取る）
import matplotlib.pyplot as plt   #グラフのライブラリをインストール　名前をpltにします
import japanize_matplotlib        #グラフの日本語化ライブラリをインストール
import numpy.random as rd         # numpy(Numerical Python)数値計算ライブラリのインストール
                                  #（名前をrdにする）
plt.figure(figsize=(4,4))         # グラフの幅と高さの指定
試行回数 = 1000                   # ランダムに打つ点の総数
弧内点数 = 0                      # 四分円の内側の点の数
for i in range(試行回数):         # 点の数だけ繰り返し処理
  x = rd.random()                 # 0以上１未満の乱数
  y = rd.random()                 # 0以上１未満の乱数
  if x**2 + y**2 < 1.0:           # 円の内側の判定
      弧内点数 += 1               # 内側の時、１カウント加える
      plt.scatter(x, y, c="red")  # 赤色でプロット
  else:                           # 円の外側の時
      plt.scatter(x, y, c="blue") # 青色でプロット
print(" 円周率:", 弧内点数 * 4.0 / 試行回数)# 円周率の計算と表示
plt.title("円周率（モンテカルロ法）")# グラフのタイトル
plt.show()                        #グラフを表示
```



出力

```
円 周 率:   3.18
```

課題 3-1

（１）試行回数を増やすと結果はどうなりますか。ただし、点の数が多いと時間が掛かります。2,000回程度までにしましょう。

（２）点が四分円の内側にある判定条件が `x**2 + y**2 < 1.0` であることを説明してください。

numpy.randomモジュールによる乱数生成

乱数を発生するには 1章で紹介した、random　モジュールを使用する他に、 numpy.rangom モジュールを使用する方法があります。numpy.rangom モジュールを使うと乱数のリストを高速に生成することができます。

プログラム

```
#numpy.randomモジュールによる乱数生成
import numpy.random as rd     #numpy.randomモジュールをインポートして名前をrdとする
x = rd.random()               #０以上１未満の実数を１つ生成　rd.rand()と同じ
print(x)
x=rd.random(5)                #０以上１未満の実数を５つリストとして生成　rd.rand(n)と同じ
print(x)
x=rd.randint(3,9)             #３～９のランダムな整数を１つ生成
print(x)
x=rd.randint(3,9,5)           #３～９のランダムな整数を５つリストとして生成
print(x)
```

出力

```
0.859163068375621
[0.99534915  0.72598205 0.63904832  0.39841736  0.39089187]
4
[6 3 5 8 5]
```

乱数なので出力は毎回異なる値です

4

モンティ・ホール問題

－　シミュレーションプログラム（モンテカルロ法）　－

学習目標とキーワード

学 習 目 標　モンティ・ホール問題のゲームを分析して、モンテカルロ法でシミュレーションするプログラムが理解できるようにしましょう。また、グラフによる視覚的表現ができるようにしましょう。

キーワード　シミュレーション　モンテカルロ法　モンティホール問題　乱数　グラフ

モンティホール問題

1990年ごろ、アメリカの名司会者、モンティ・ホールがテレビ番組『Let's make a deal』で、次のようなルールのゲームをしていました。

① ３つのドアがあり、その内一つのドアの先に新車が隠されています。ドアを開けて、新車があればもらえます。

② 挑戦者が一つのドアを選ぶと、司会者モンティは、残りの２つのドアの内、ハズレのドアを開けて「あなたは最初に選んだドアを変更しても良いのですが、どうしますか？」と尋ねます。

③ この問題に、世界最高のIQの持ち主である　マリリン・ボス・サバントと言う女性が、雑誌に解答を掲載しました。「挑戦者は最初に選んだドアを変更すべきである。なぜなら、ドアを変更した場合、新車を当てる確率が2倍になるから」　しかし、著名な数学者や世間の人は、この解答は間違っていると主張し、現代で言う「炎上」が起こりました。

1　シミュレーションの方針（モデル化）

ドアの選択を乱数によって決めるモンテカルロ法でシミュレーションしてみましょう

(1) ３つのドアの番号を１，２，３とし、ランダム関数で当たりを一つ決めます。

(2) ドア１，２，３からランダム関数で、挑戦者が選んだドアを一つ決めます。

(3) もし(1)と(2)で決めたドアが同じならば（最初に選んだドアが当たり）、残った２つのドアからランダム関数で一つ選択して再選択可能なドア（開けていないドア）とします。

(4) もし (1)と(2)で決めたドアが異なるならば（最初に選んだドアがはずれ）、(1)のドアを再選択可能なドアとします。これは、挑戦者が選んだドアと当たりのドアを司会者は開けることができないので、残されたはずれのドアを開くしかないためです。

(5) 再選択で選択したドアを変えるか変えないかランダム関数で選択して、当たり・はずれを判断します。

(6) (1)～(5)を繰り返し、当たりとはずれの回数を累計します。この累計を繰り返し回数で割り確率を求めます。

2　プログラムを実行してみましょう

```
#モンティ・ホール問題シミュレーション
#!pip install japanize-matplotlib          #日本語を使うモジュールのインストール（初回実行は#を取る）
import matplotlib.pyplot as plt            #グラフのライブラリをインストール　名前をpltにします
import japanize_matplotlib                 #グラフの日本語化ライブラリをインストール
import numpy.random as rd                  #numpy.randomモジュールをインポートして名前をrdとする
#初期設定
変更なし当たり確率,変更あり当たり確率=[],[]  #グラフ表示用リストの初期化
変更なし当たり,変更あり当たり=0,0           #変更なし当たり回と変更あり当たりを初期化
試行回数=1000                              #試行回数の設定
#シミュレーション
for i in range(試行回数) :                 #シミュレーションを試行回数実行
#当たりドアと選択ドアの設定
  ドアリスト=[1,2,3]                       #ドアのリストを1,2,3とする
  当たりドア=rd.choice(ドアリスト)          #ドアリストからランダムに１つ選び当たりドアとする
  選択リスト=[1,2,3]                       #プレイヤーが選択するドアリストを1,2,3とする
  選択ドア=rd.choice(選択リスト)           #選択ドアリストからランダムに１つ選び選択ドアとする
  #司会者が開かないドア（再選択が可能なドア）を決める
  if 当たりドア == 選択ドア:               #当たりドアと選択ドアが同じ場合
    if 当たりドア==1:                      #当たりドアが１の場合
      ドアリスト=[2,3]                     #司会者が開けずに残すドアのリストは 2,3
    elif 当たりドア==2:                    #当たりドアが2の場合
      ドアリスト=[1,3]                     #司会者が開けずに残すドアのリストは 1,3
    elif 当たりドア==3:                    #当たりドアが3の場合
      ドアリスト=[1,2]                     #司会者が開けずに残すドアのリストは 1,2
    再選択ドア=rd.choice(ドアリスト)       #残すドアリストからランダムに１つ選び、再選択できるドアとする
  else:                                    #当たりドアと選択ドアが異なる場合
    #当たりドアは開けられないので、再選択できるドアは当たりドアだけになる
    再選択ドア=当たりドア
```

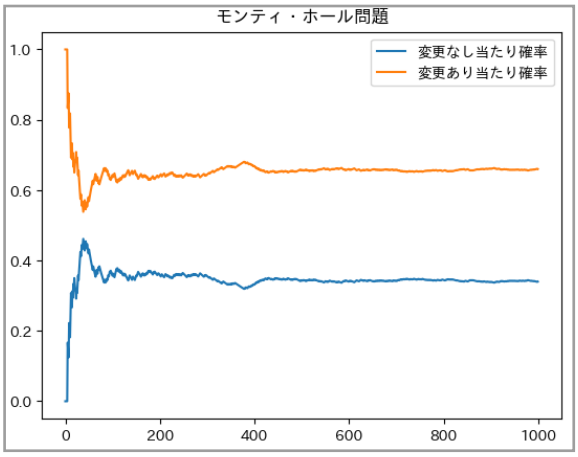
つづく

```
  #あたり判定と集計
  #変更なしで当たる場合
  if  選択ドア == 当たりドア:                    #最初に選択したドアが当たりの判定
      変更なし当たり=変更なし当たり+1             #変更なしで当たった回数を集計する
  #変更ありで当たる場合
  if  再選択ドア == 当たりドア:                  #変更したドアが当たりの判定
      変更あり当たり=変更あり当たり+1             #変更ありで当たった回数を集計する
  変更なし当たり確率.append(変更なし当たり/(変更なし当たり+変更あり当たり))#変更なしで当たる確率を計算してリストにする
  変更あり当たり確率.append(変更あり当たり/(変更なし当たり+変更あり当たり)) #変更ありで当たる確率を計算してリストにする
#結果
print("試行回数",試行回数)
print("変更なし当たり",変更なし当たり,"確率",変更なし当たり/(変更なし当たり+変更あり当たり))
print("変更あり当たり",変更あり当たり,"確率",変更あり当たり/(変更なし当たり+変更あり当たり))
# 確率グラフ表示
plt.title("モンティ・ホール問題")                  #グラフのタイトル
plt.plot(変更なし当たり確率,label='変更なし当たり確率')  #折れ線グラフと凡例の表示
plt.plot(変更あり当たり確率,label='変更あり当たり確率')  #折れ線グラフと凡例の表示
plt.legend()                                      #凡例を表示
plt.show()                                        #グラフを表示
```

出力

```
試行回数 1000
変更なし当たり 306 確率 0.306
変更あり当たり 694 確率 0.694
```

乱数なので出力は毎回異なる値です



課題 4-1

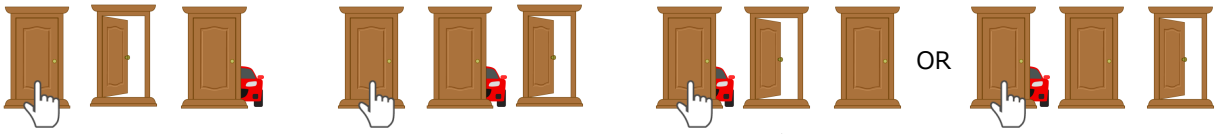
（1）シミュレーションの結果とグラフを見ると、マリリンの解答は正しいと言えますか。

（2）グラフを見るとシミュレーション開始直後は値が大きく変動しています。どうしてですか。

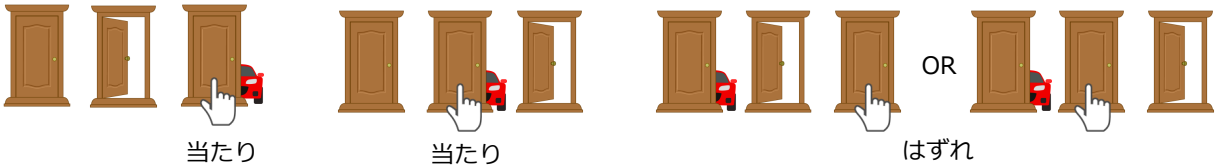
3　モンティ・ホール問題を図で考えてみましょう

（1）最初の選択で当たる確率は1/3で、選択を変更しない場合は当たる確率はそのまま1/3です。

（2）司会者がはずれのドアを開きます。選択したドアと当たりのドアは開くことができません。



（3）最初に選択したドアから変更して別のドアを開きます。当たる確率は2/3になります。



（4）この図では、一番左のドアを選択していますが、他のドアを選択しても同じ結果になります。
最初の選択を変更すると、最初にはずれを選んだ２パターンが当たりになります。また、最初にあたりを選んだ１パターンがはずれになります。つまり選択を変更したことで当たる確率は 1/3 から 2/3 の２倍になります。