

確率的モデルによる シミュレーション

23j1-215

教科書P80-81

この時間の目標

- 確率的モデルによるシミュレーション
- S モデル化とシミュレーションについてよく理解でき、活用方法を考えることができた
- A モデル化とシミュレーションについてよく理解できた
- B モデル化とシミュレーションについて理解できた
- C モデル化とシミュレーションについて理解できなかった

確率的モデルによるシミュレーション

- 条件は確率に基づいて変動する
- 何度も試して
 - 傾向を見る
 - 最良の方法を考える
- シミュレーションの結果を見て人間が判断する

問題

- 学園祭である飲み物を販売することになった。そこで、昨年までの同じ飲み物の1時間当たりの販売個数を10個単位で集計した

販売個数	10	20	30	40
度数	8	16	12	4

- 朝9時から夕方17時まで販売するとして、販売個数をシミュレーションで予測しよう

問題

- 確率で変動すると考える

販売個数	度数	累積確率	乱数の範囲
10	8		
20	16		
30	12		
40	4		

- 表を書き直した
- 1時間あたり20個・30個売れる可能性が高い

問題

- 確率で変動すると考える

販売個数	度数	累積確率	乱数の範囲
10	8	0.2	
20	16	0.6	
30	12	0.9	
40	4	1	

- 累積確率: 全体に対する割合
 - $8 / (8 + 16 + 12 + 4) = 0.2$
 - $(8 + 16) / (8 + 16 + 12 + 4) = 0.6$

問題

- 確率で変動すると考える

販売個数	度数	累積確率	乱数の範囲
10	8	0.2	0以上0.2未満
20	16	0.6	0.2以上0.6未満
30	12	0.9	0.6以上0.9未満
40	4	1	0.9以上1未満

- 乱数の範囲

- 0~1未満の乱数でどれを選ぶか

問題

- 乱数を8つ用意した

時	乱数	販売個数
9	0.859	
10	0.248	
11	0.133	
12	0.455	
13	0.339	
14	0.026	
15	0.698	
16	0.716	
合計		

販売個数	度数	累積確率	乱数の範囲
10	8	0.2	0以上0.2未満
20	16	0.6	0.2以上0.6未満
30	12	0.9	0.6以上0.9未満
40	4	1	0.9以上1未満

問題

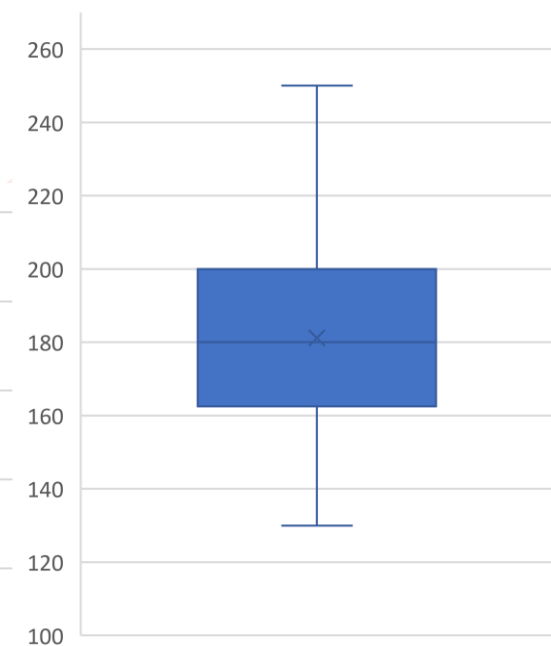
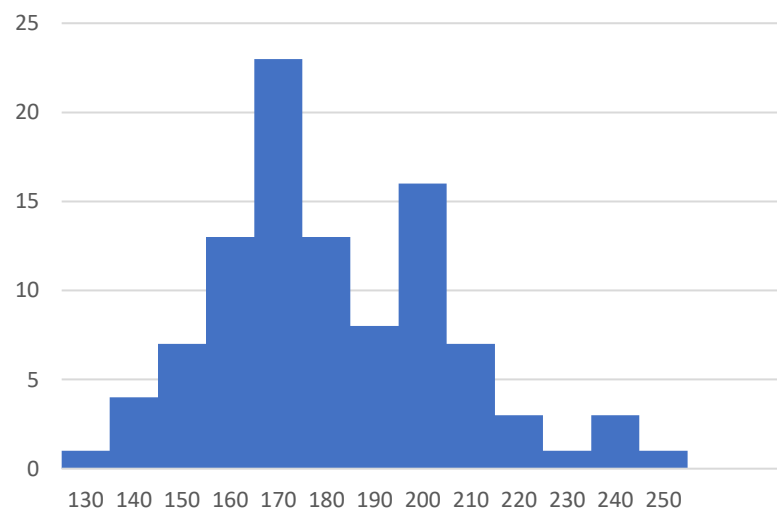
- 乱数を8つ用意した

時	乱数	販売個数
9	0.859	30
10	0.248	20
11	0.133	10
12	0.455	20
13	0.339	20
14	0.026	10
15	0.698	30
16	0.716	30
合計		170

販売個数	度数	累積確率	乱数の範囲
10	8	0.2	0以上0.2未満
20	16	0.6	0.2以上0.6未満
30	12	0.9	0.6以上0.9未満
40	4	1	0.9以上1未満

- 何回もやってみることで予測ができる

- 100回やると



今日の内容

- さいころのシミュレーション
 - 疑似さいころを100回/1000回振ってさいころとして妥当か考えてみよう
- ガチャのシミュレーション
 - ガチャ100回やったら当たるのか？

実習の目的

- 実際にシミュレーションしてみる
 - 数値を変えるとどうなるか試してみる
 - どこを変えたら良いかを理解する
- 仕組みがどうなっているか考える
 - 理解すべきスライドには ☆重要 マークが付いている
 - 式の意味や作り方をしっかり読んで考える
- できたらOKの課題ではない

今日の進め方

- さいころのシミュレーション
 - Excelでシミュレーション
 - Pythonでシミュレーション
- ガチャのシミュレーション
 - Excelでシミュレーション
 - Pythonでシミュレーション
- 情報 I 第15回[シミュレーション実習2]

Excelは
スタート→Excel
→[空白のブック]
から始める

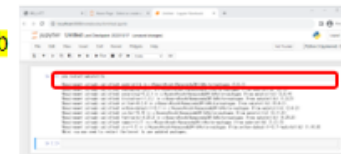
Pythonは
JupyterNotebook



少なくとも
Excelと
Pythonを
一つずつ

JupyterNotebookにおまじない

1. JupyterNotebookのセルに以下の文字列を入力
`pip install matplotlib`
2. [Shift]+[Enter]
3. 情報 I 第14回[シミュレーション実習1]に取り組む



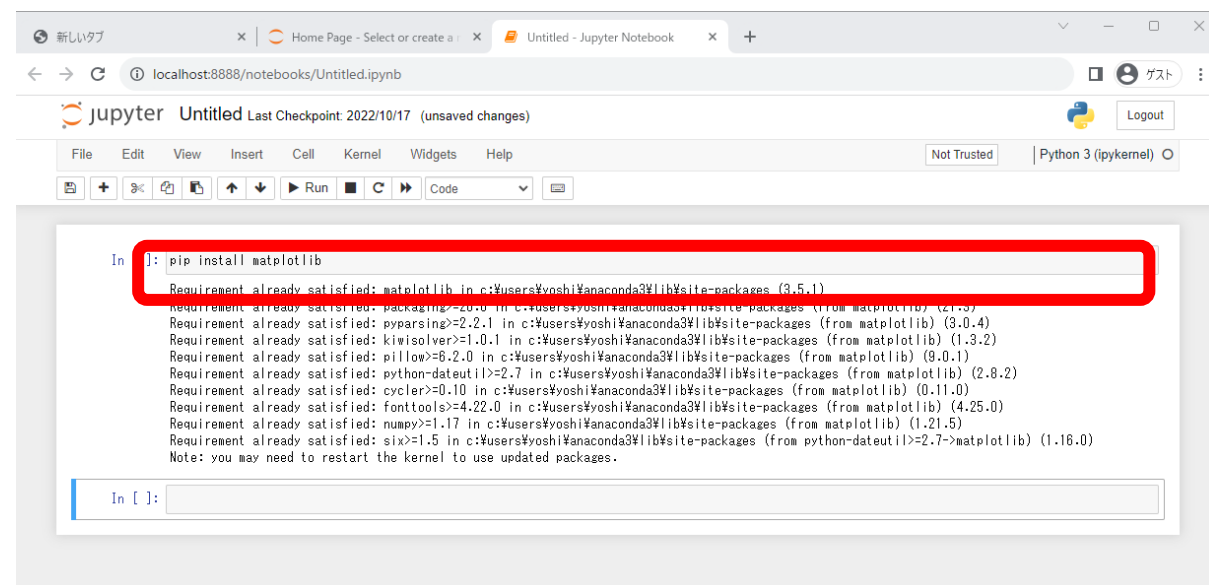
Jupyter Notebookにおまじない

1. Jupyter Notebookのセルに以下の文字列を入力

```
pip install matplotlib
```

2. [Shift]+[Enter]

3. 情報 I 第14回[シミュレーション実習1]に取り組む



```
In [ ]: pip install matplotlib
Requirement already satisfied: matplotlib in c:\users\yoshi\anaconda3\lib\site-packages (3.5.1)
Requirement already satisfied: packaging>=20.0 in c:\users\yoshi\anaconda3\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\yoshi\anaconda3\lib\site-packages (from matplotlib) (3.0.4)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\yoshi\anaconda3\lib\site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\yoshi\anaconda3\lib\site-packages (from matplotlib) (9.0.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\yoshi\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: cycler>=0.10 in c:\users\yoshi\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\yoshi\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: numpy>=1.17 in c:\users\yoshi\anaconda3\lib\site-packages (from matplotlib) (1.21.5)
Requirement already satisfied: six>=1.5 in c:\users\yoshi\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```



Pythonの超基本

- 変数・関数・文の見分け方
 - 関数 後ろに()が付いている
 - `print(deme)`
 - `saikoro.count(j+1)`
 - `gacha100()`
 - 変数 関数でない文字列で'で囲われていない
 - `saikoro`
 - `deme`
 - 文 行末に:(コロン)必須、次行はインデント(4字下げ)
 - `if/for/while`

さいころのシミュレーション

疑似さいころを100回/1000回振ってさいころとして妥当か考えてみよう

疑似さいころの検証(モデル)

- 乱数を使ってさいころを振る
- 出現回数を調べ妥当性を考える

例:さいころを100回振ってみると

- 乱数を使ってさいころを振る
- Excelでシミュレーション
- Pythonでシミュレーション



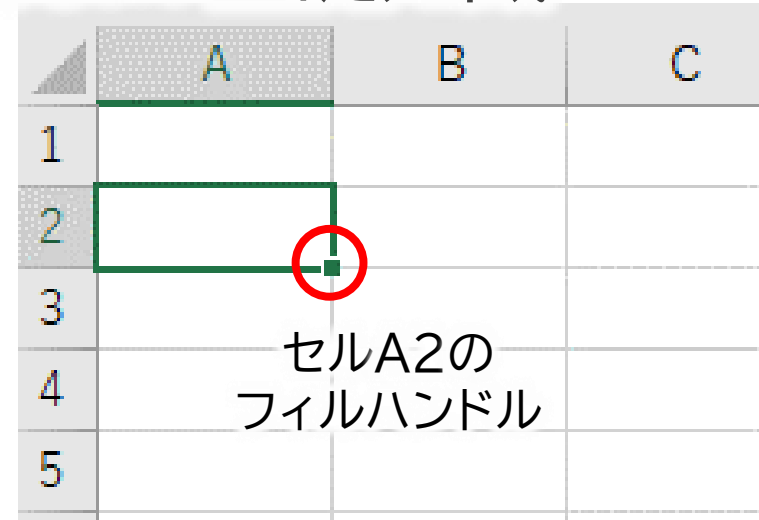
例:さいころを100回振ってみると

Excelでシミュレーション



- 手順
 - 1. 表の作成
 - 2. さいころの式を作る
 - 3. 出目のカウント
 - 4. グラフ作成
 - 5. 数値を変えてみる
-
- 手順の説明は次ページへ

Excelの基礎知識



セルA2の
フィルハンドル

おかしいときは直そうとせず
[Esc]→[Ctrl]+[Z]で
やらなかったことに

例:さいころを100回振ってみると

Excelでシミュレーション



• 1. 表の作成

1. セルB1に 出現回数 と入力
2. セルA2に 1 と入力
3. セルA3に 2 と入力
4. セルA2からセルA3をドラッグして範囲選択したら、フィルハンドルをセルA7までドラッグし、6まで作る
5. セルA10に 目 と入力
6. saikoroというファイル名で自分のドライブに保存

日本語入力は
こまめに
切っておく

	A	B	C
1		出現回数	
2	1		
3	2		
4	3		
5	4		
6	5		
7	6		
8			
9			
10	目		
11	完成イメージ		
12			



例:さいころを100回振ってみると

Excelでシミュレーション

- 2. さいころの式を作る
 1. セルA11に `=randbetween(1,6)` と入力
 2. セルA11のフィルハンドルを下にドラッグしてセルA110までドラッグする

- RANDBETWEEN(開始,終了)
 - 開始値から終了値の整数による乱数を発生する

数式は
小文字で
入力できる

例:さいころを100回振ってみると

Excelでシミュレーション



• 3. 出目のカウント

1. セルB2に `=countif(a11:a110,a2)` を入力
2. セルB2のフィルハンドルをダブルクリック

• COUNTIF(範囲,条件)

- 範囲の中から条件に合うものをカウントする

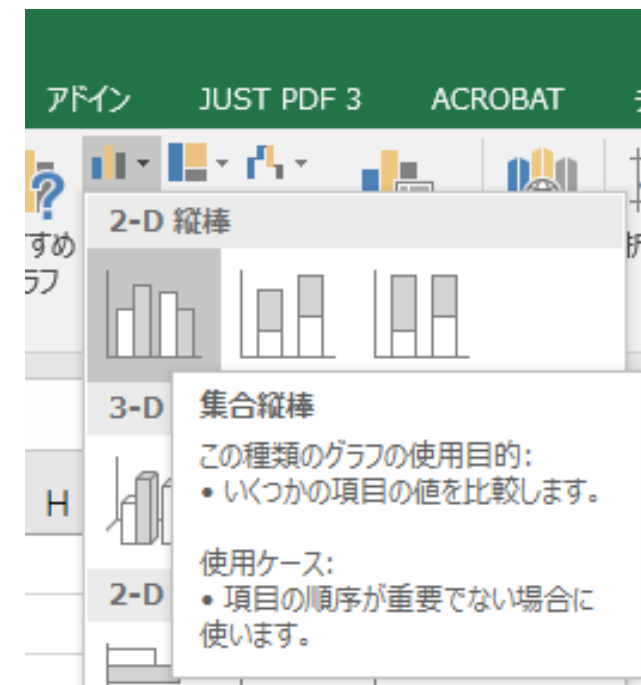
例:さいころを100回振ってみると

Excelでシミュレーション



• 4. グラフ作成

1. セルA1~B7をドラッグ
2. [挿入]→[グラフ]→[縦棒グラフ]
→[集合縦棒]



例:さいころを100回振ってみると

Excelでシミュレーション



- 4. グラフ作成-検討
- 何回か実行する
 1. セルA1をクリック
 2. [F2]キーを押して[Enter]キーを押す
- 結果のデータや度数分布を見て、さいころとして妥当か考えてみよう

例:さいころを100回振ってみると

Excelでシミュレーション



- 5. 数値を変えてみる
 - さいころを1000回振るとどうなるか
 - セルB2~B7の式は修正が必要
- 結果のデータや度数分布を見て、さいころとして妥当か考えてみよう
- 妥当かどうか判断する方法を今後数学で学ぶはず

さいころを100回振ってみると
Excel版 終了
次の課題へ>>

例:さいころを100回振ってみると

Pythonでシミュレーション



- 手順
 - 1. プログラムの作成
 - 2. 出目のカウント
 - 3. グラフ作成のために改造
 - 4. グラフ表示プログラムの追加
 - 5. 数値を変えて実行
-
- 手順の説明は次ページへ

例:さいころを100回振ってみると

Pythonでシミュレーション



• 1. プログラムの作成

1. 新しいノートブックを用意

2. 以下のプログラムを入力

3. 入力できたら実行して動作確認

• [4, 6, 2, 6, 1, 1, 3, 2, 3, 3]のように表示される

4. 100回に改造

```
import random
saikoro=[]
for i in range(10):
    saikoro.append(random.randint(1, 6))
print(saikoro)
```



例:さいころを100回振ってみると

Pythonでシミュレーション

- 1. プログラムの作成-プログラムの意味

```
import random
saikoro=[]
for i in range(10):
    saikoro.append(random.randint(1, 6))
print(saikoro)
```

- 100回に改造する

モジュールrandom をインポート
リストsaikoroを用意

10回繰り返し

1から6の整数の乱数をsaikoroに追加
saikoroを表示

例:さいころを100回振ってみると

Pythonでシミュレーション



- 2. 出目のカウント
 - 以下の部分を追加して実行し動作確認

```
import random
saikoro=[]
for i in range(100):
    saikoro.append(random.randint(1, 6))
print(saikoro)
deme=[]
for j in range(6):
    deme.append(saikoro.count(j+1))
print(deme)
```



例:さいころを100回振ってみると

Pythonでシミュレーション

• 2. 出目のカウント-プログラムの意味

```
deme=[]  
for j in range(6):  
    deme.append(saikoro.count(j+1))  
print(deme)
```

リストdemeを用意
jを0から6未満まで変えながら繰り返す
リストsaikoroにあるj+1を数えdemeに代入
demeを表示

- リスト
複数のデータを入れる変数(実はsaikoroもリスト)

例:さいころを100回振ってみると

Pythonでシミュレーション



- 3. グラフ作成のために改造
 - 以下の部分を追加して実行し動作確認

```
import random
import matplotlib.pyplot as plt
saikoro=[]
for i in range(100):
    saikoro.append(random.randint(1, 6))
print(saikoro)
deme=[]
for j in range(6):
    deme.append(saikoro.count(j+1))
print(deme)
```



例:さいころを100回振ってみると

Pythonでシミュレーション

- 3. グラフ作成のために改造-プログラムの意味

```
import matplotlib.pyplot as plt
```

モジュールmatplotlib.pyplotをpltとしてインポート

- プログラム中に何度もmatplotlib.pyplotと打つのは面倒なので、plt という略称をつけておく

例:さいころを100回振ってみると

Pythonでシミュレーション



- 4. グラフ表示プログラムの追加
 - 以下の部分を追加して実行し動作確認

```
left=[1, 2, 3, 4, 5, 6]
plt.title('Saikoro Simulation')
plt.xlabel('Me')
plt.ylabel('Kaisu')
plt.bar(left, deme, align='center')
plt.show()
```

```
import random
import matplotlib.pyplot as plt
saikoro=[]
for i in range(100):
    saikoro.append(random.randint(1, 6))
print(saikoro)
deme=[]
for j in range(6):
    deme.append(saikoro.count(j+1))
print(deme)
```

```
left=[1, 2, 3, 4, 5, 6]
plt.title('Saikoro Simulation')
plt.xlabel('Me')
plt.ylabel('Kaisu')
plt.bar(left, deme, align='center')
plt.show()
```

完成イメージ

例:さいころを100回振ってみると

Pythonでシミュレーション



- 4. グラフ表示プログラムの追加-検討
- 何回か実行する
- 結果のデータや度数分布を見て、さいころとして妥当か考えてみよう

例:さいころを100回振ってみると

Pythonでシミュレーション



- 5. 数値を変えて実行
- さいころを1000回振るとどうなるか
- 結果のデータや度数分布を見て、さいころとして妥当か考えてみよう
- 妥当かどうか判断する方法を今後数学で学ぶはず

さいころを100回振ってみると
Python版 終了
次の課題へ>>

ガチャのシミュレーション

ガチャ100回やったら当たるのか？

ガチャの検証(モデル)

- 排出確率1%のガチャ
 - 1~100までの整数の乱数を発生
 - あたりの数を1つ決めてシミュレーション
- ガチャを100回やったらどれぐらい当たるか
- 100人が100回やったらどれぐらい当たるか

例:ガチャを100回やると

- 排出確率1%のガチャ
 - 1~100までの整数の乱数を発生
 - 当たりの数を1つ決めてシミュレーション

• Excelでシミュレーション



• Pythonでシミュレーション



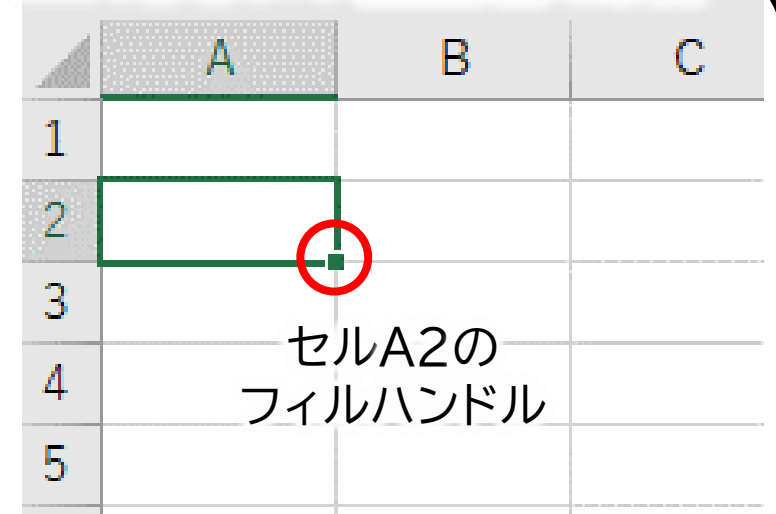
例:ガチャを100回やると

Excelでシミュレーション



- 手順
 - 1. 表の作成
 - 2. ガチャの式を作る
 - 3. 当たりの判定
 - 4. 100人分実行
 - 5. 排出回数、当選人数をカウント
 - 6. 数値を変えて実行
-
- 手順の説明は次ページへ

Excelの基礎知識



おかしいときは直そうとせず
[Esc]→[Ctrl]+[Z]で
やらなかったことに

例:ガチャを100回やると

Excelでシミュレーション



• 1. 表の作成

- セルA1に 当たり と入力
- セルA3に 排出回数 と入力
- セルA4に 当選人数 と入力
- セルB10に 1,セルC10に 2と入力
- セルB10からセルC10をドラッグして範囲選択したら、フィルハンドルをセルCW10まで右にドラッグし、100まで作る
- gacha というファイル名で自分のドライブに保存

日本語入力は
こまめに
切っておく

	A	B	C	D	E
1	当たり				
2					
3	排出回数				
4	当選人数				
5					
6					
7					
8					
9					
10		1	2	3	4
11					

完成イメージ



例:ガチャを100回やると

- 2. ガチャの式を作る
 - セルB11に `=randbetween(1,100)` と入力
 - セルB11のフィルハンドルを右にドラッグしてセルCW11までドラッグする
- RANDBETWEEN(開始,終了)
 - 開始値から終了値の整数による乱数を発生する

数式は
小文字で
入力できる



例:ガチャを100回やると

- 3. 当たりの判定
 - セルB1に 50 を入力
 - 当選番号を50にする/別の数にしても良い
 - セルA11に `=countif(b11:cw11,b1)` を入力
- COUNTIF(範囲,条件)
 - 範囲の中から条件に合うものをカウントする

例:ガチャを100回やると

Excelでシミュレーション



- 4. 100人分実行
 - セルA11からセルCW11をドラッグして範囲選択
 - 選択範囲右下のフィルハンドルを下にドラッグし、CW110までドラッグする

- ガチャ100回100人分の表が完成

例:ガチャを100回やると

Excelでシミュレーション



- 5. 排出回数、当選人数をカウント
 - セルB3に `=sum(a11:a110)` を入力
 - セルB4に `=countif(a11:a110,">0")` を入力
- SUM(範囲)
 - 範囲にある数値の合計
- COUNTIF(範囲,条件)
 - 範囲の中から条件に合うものをカウントする

例:ガチャを100回やると

Excelでシミュレーション



- 5. 排出回数、当選人数をカウント-検討
- 何回か実行する
 - セルA1をクリック
 - [F2]キーを押して[Enter]キーを押す
- 結果のデータを見て考えよう
 - 思っていたようなデータか
 - ガチャとして妥当か

例:ガチャを100回やると

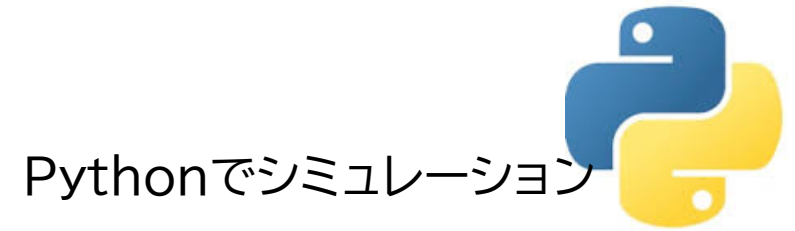
Excelでシミュレーション



- 6. 数値を変えて実行
- 100回のガチャを1000人でやるとどうなるか
 - セルB3,セルB4の式は修正が必要
- 結果のデータを見て考えよう
 - 思っていたようなデータか
 - ガチャとして妥当か

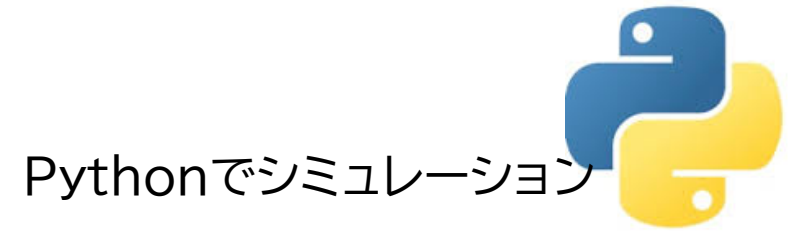
ガチャを100回やると
Excel版 終了
次の課題へ>>

例:ガチャを100回やると



- 手順
 - 1. プログラムの作成
 - 2. 当たりの判定
 - 3. 100人分のために改造
 - 4. 100人分実行
 - 5. 排出回数、当選人数をカウント
 - 6. 数値を変えて実行
-
- 手順の説明は次ページへ

例:ガチャを100回やると



- 1. プログラムの作成
 1. 新しいノートブックを用意
 2. 以下のプログラムを入力
 3. 入力できたら実行して動作確認
 4. 100回やるよう改造

```
import random
gacha=[]
for i in range(10):
    gacha.append(random.randint(1, 100))
print(gacha)
```



例:ガチャを100回やると

- 1. プログラムの作成-プログラムの意味

```
import random
gacha=[]
for i in range(10):
    gacha.append(random.randint(1, 100))
print(gacha)
```

- 100回に改造する

モジュールrandomをインポート

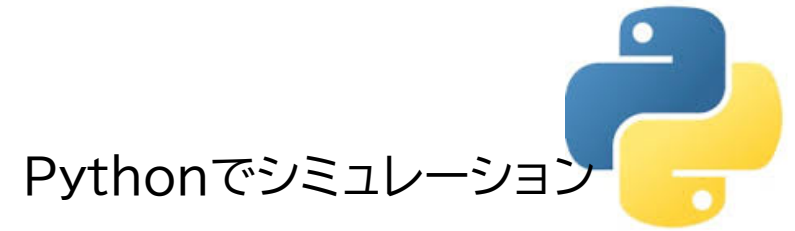
リストgachaを用意

10回繰り返し

1~100までの乱数をgachaに追加

gachaを表示

例:ガチャを100回やると



- 2. 当たりの判定
 - 以下の部分を追加して実行し動作確認

```
import random
atar i=50
gacha=[]
for i in range(100):
    gacha.append(random.randint(1, 100))
print(gacha)
tousen=gacha.count(atar i)
print(tousen)
```



例:ガチャを100回やると

- 2. 当たりの判定-プログラムの意味
 - 当選番号は1~100の間で変えてもよい

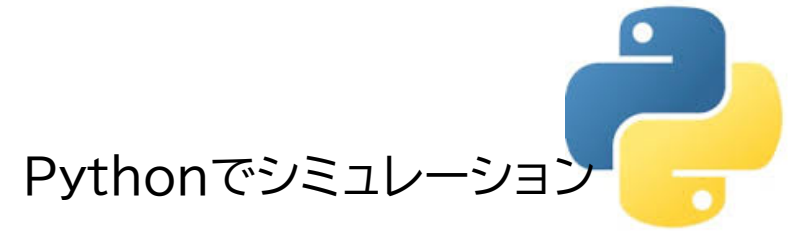
当選番号を
50にする

```
import random
atar i=50
gacha=[]
for i in range(100):
    gacha.append(random.randint(1, 100))
print(gacha)
tousen=gacha.count(atar i)
print(tousen)
```

tousenを
表示する

tousen にgachaにあるatari
の個数(排出回数)を代入

例:ガチャを100回やると



- 3. 100人分のために改造
 - 以下の部分を修正して実行し動作確認

```
import random
atar i=50
def gacha100():
    gacha=[]
    for i in range(100):
        gacha.append(random.randint(1, 100))
    touselen=gacha.count(atar i)
    return touselen

print(gacha100())
```



例:ガチャを100回やると

- 3. 100人分のために改造-プログラムの意味
 - gacha100()関数を定義/排出回数を返す

関数
gacha100()
を定義

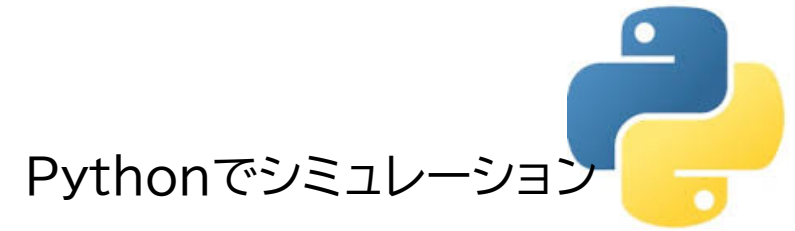
```
import random
atar i=50
def gacha100():
    gacha=[]
    for i in range(100):
        gacha.append(random.randint(1, 100))
    tousel=gacha.count(atar i)
    return tousel

print(gacha100())
```

関数
gachaの
戻り値として
当たりの回数を
返す

関数
gacha100()
を呼び出し
結果を表示

例:ガチャを100回やると



- 4. 100人分実行
 - `print(gacha100())`を消して
 - 以下の部分を追加して実行し動作確認

```
kekka=[]  
for i in range(100):  
    kekka.append(gacha100())  
  
print(kekka)
```

```
1 import random  
2 atari=50  
3 def gacha100():  
4     gacha=[]  
5     for i in range(100):  
6         gacha.append(random.randint(1,100))  
7     tousen=gacha.count(atari)  
8     return tousen  
9  
10 kekka=[]  
11 for i in range(100):  
12     kekka.append(gacha100())  
13 print(kekka)
```

完成イメージ



例:ガチャを100回やると

- 4. 100人分実行-プログラムの意味
 - gacha100()を100回実行し排出回数をリストkekkaに代入する

空のリスト
kekka
を用意

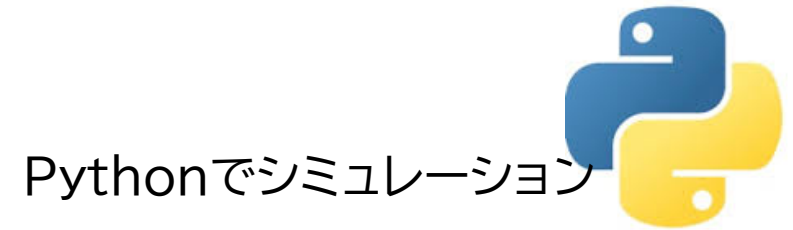
```
kekka=[]  
for i in range(100):  
    kekka.append(gacha100())  
  
print(kekka)
```

100回繰り返す

gacha100()の
結果を
リストkekkaに
追加

- リスト
複数のデータを入れる変数(実はgachaもリスト)

例:ガチャを100回やると



- 5. 排出回数、当選人数をカウント
 - kekka=[]以降を修正して実行し動作確認

```
kekka=[]
hit=0;luck=0
for i in range(100):
    kekka.append(gacha100())
    hit=hit+kekka[i]
    if kekka[i]>0:
        luck=luck+1
print(kekka)
print('hit',hit,'luck',luck)
```

```
1 import random
2 atari=50
3 def gacha100():
4     gacha=[]
5     for i in range(100):
6         gacha.append(random.randint(1,100))
7     touden=gacha.count(atari)
8     return touden
9
10 kekka=[]
11 hit=0;luck=0
12 for i in range(100):
13     kekka.append(gacha100())
14     hit=hit+kekka[i]
15     if kekka[i]>0:
16         luck=luck+1
17 print(kekka)
18 print('hit',hit,'luck',luck)
```

完成イメージ



例:ガチャを100回やると

- 5. 排出回数、当選人数をカウント-プログラムの意味
 - kekk=[]以降を修正して実行し動作確認

変数
hit(排出回数)
luck(当選人数)
を0にする

```
kekka=[]  
hit=0; luck=0  
for i in range(100):  
    kekka.append(gacha100())  
    hit=hit+kekka[i]  
    if kekka[i]>0:  
        luck=luck+1  
print(kekka)  
print('hit', hit, 'luck', luck)
```

変数hitに
排出回数を加える

変数
hit(排出回数)
luck(当選人数)
を表示する

排出回数>0なら
(当選したなら)
変数luckに1を加える

例:ガチャを100回やると

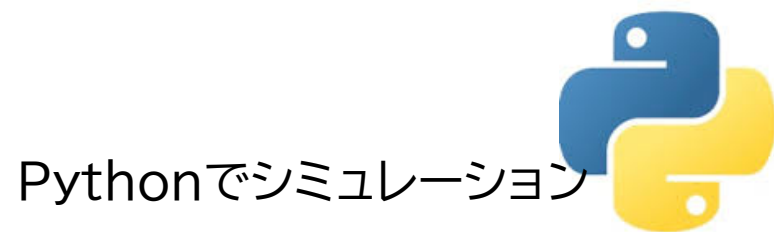
Pythonでシミュレーション



- 5. 排出回数、当選人数をカウント-検討
- 何回か実行する

- 結果のデータを見て考えよう
 - 思っていたようなデータか
 - ガチャとして妥当か

例:ガチャを100回やると



- 6. 数値を変えて実行
- 100回のガチャを1000人でやるとどうなるか
- 結果のデータを見て考えよう
 - 思っていたようなデータか
 - ガチャとして妥当か

ガチャを100回やると
Python版 終了
次の課題へ>>

時間があれば取り組もう

- それぞれの課題でまだやっていないものに取り組もう
- さいころのシミュレーション
 - [Excelでシミュレーション](#)
 - [Pythonでシミュレーション](#)
- ガチャのシミュレーション
 - [Excelでシミュレーション](#)
 - [Pythonでシミュレーション](#)