基礎ゼミ01Python

福田研修士2年

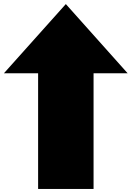
平林　新

本日の流れ

1. 目的の確認
2. **基礎分析に使える道具（コード）の紹介**
3. その他、少し便利な道具（コード）の紹介

1.目的の確認

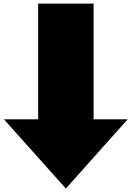
- Pythonを手足のように使って**研究を捗らせる**こと！！



- **実践的な使い方**の基盤となり、読者をアシストする教科書
- いざPythonが必要になったときに役立つマニュアル

1.目的の確認

- 実践的な使い方って？？



- データファイルの読み込み＋集計
- グラフ作成
- パラメータ推定

2.基礎分析に使える道具

① 手始めに自分の手を動かしデータを作る
② 自作データに基き仮説を立ててみる
③ 得られた結果を可視化する
④ (やや応用)実際のデータを用いて分析する

①～③で紹介する道具

- import
- numpy
- matplotlib.pyplot
- np.array
- np.array[n,m]
- plt.hist
- plt.show()
- sum
- len()
- plt.bar
- plt.pie

基礎①、の前に準備

import numpy as np

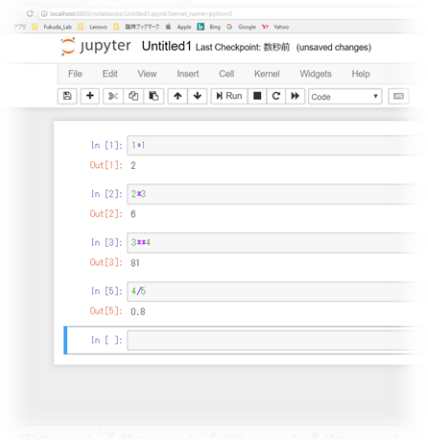
import matplotlib.pyplot as plt

importとは、自分の持っている本棚（専門用語：ライブラリ）から説明書（専門用語：パッケージ、モジュール）を持ってくる作業です

本棚を充実させたいときは、本屋さん（専門用語：コマンドプロンプト）で「pip install **」のようなコードを記述します

本屋さん→本棚、本棚→作業場所（専門用語：Jupyter Notebook）の「2つの持ってくる作業」が存在するということです

基礎①、の前に準備



Jupyter Notebook ← import ← Library ← pip ← website

package

module

module

module

基礎①、の前に準備

import numpy as np

import matplotlib.pyplot as plt

numpyとは、パッケージです

number pythonのような言葉の略だと思われます

これをimportすると、数学的な処理を行えるようになります

as npというのは、説明書（パッケージ）を短縮読みして使うことを示しています

基礎①、の前に準備©

import numpy as np

import matplotlib.pyplot as plt

こちら葛飾区亀有公園前派出所 as こちかめ

東京工業大学 as 東工大

環境・社会理工学院 土木・環境工学系 都市・環境学コース as としかん

基礎①、の前に準備

import numpy as np

import matplotlib.pyplot as plt

matplotlibはパッケージ、pyplotはモジュールです

mathematic plot liblaryの略だと思われます

matplotlib.pyplotとなっているのは、パッケージ内の特定のモジュールをimportしているということです

モジュールのかたまりがパッケージです、パッケージ＝モジュール集

基礎①、の前に準備©

import numpy as np
import matplotlib.pyplot as plt

火遁.豪火球の術
火遁.大炎弾の術
ゴムゴム.バズーカ
ゴムゴム.ガトリング

基礎①、の前に準備

【今日の研究テーマ】

「予算500円をポテトチップス、チョコレート、飲み物にどう分配して幸せを獲得するか？」

【条件】

- おやつ（飲み物含む）は500円まで
- 100円刻みでしか分配できない
- 0円（購入しないもの）があってもOK

基礎①データを自作する

arr1 = np.array([[2,1,2], [1,3,1] ,…, [2,2,1]])
print(arr1)

arr1は器（専門用語：変数名）、ここにデータ（専門用語：配列）を代入します

np.array()はモジュール、短縮したnpという説明書集（専門用語：パッケージ）と、arrayという説明書（専門用語：モジュール）

[2,1,2]はベクトル（専門用語：1次元配列）

[[2,1,2],[1,3,1],…,[4,1,0]]は行列（専門用語：2次元配列）

行列（専門用語：2次元配列）は改行しても大丈夫らしいです

print()は()内の変数を表示するための関数、とっても便利です

```
print(arr1)
```

```
[[2 1 2]
 [1 3 1]
 [2 2 1]
 [3 2 0]
 [0 4 1]
 [2 0 3]
 [0 0 5]
 [0 5 0]
 [5 0 0]
 [1 1 3]
 [0 4 1]
 [4 1 0]
 [2 0 3]
 [0 2 3]
 [3 0 2]
 [2 2 1]
 [2 1 2]
 [0 1 4]
 [1 3 1]
 [1 1 3]
 [1 3 1]
 [2 2 1]
 [3 2 0]
 [0 4 1]
 [2 0 3]
 [0 4 1]
 [2 0 3]
 [0 0 5]
 [0 5 0]
 [5 0 0]
 [1 1 3]
 [0 4 1]
 [4 1 0]
 [2 0 3]
 [0 2 3]
 [3 0 2]
 [3 1 1]
 [4 1 0]
 [4 0 1]
 [4 1 0]
 [3 0 2]
 [3 1 1]
 [4 1 0]
 [4 0 1]
 [4 1 0]
 [3 0 2]
 [3 1 1]
 [4 1 0]
 [4 0 1]
 [4 1 0]
 [3 0 2]
 [3 1 1]
 [4 1 0]
 [4 0 1]
 [4 1 0]]
```

基礎②自作データに基き、大まかな仮説を立てる

1行目と2行目を表示する

print(arr1[0])

print(arr1[1])

pythonは0からスタートします。0, 1, 2, …

行を指定するときは[]を使います

n行は[n-1]

```
# 1行目と2行目を表示する

print(arr1[0])
print(arr1[1])
```

```
[2 1 2]
[1 3 1]
```

基礎②自作データに基き、大まかな仮説を立てる

1行1列の要素を表示する
1行2列の要素を表示する
1行3列の要素を表示する
2行3列の要素を表示する
print(arr1[0,0])
print(arr1[0,1])
print(arr1[0,2])
print(arr1[1,2])
n行m列は[n-1, m-1]

```
# 1行1列の要素を表示する
# 1行2列の要素を表示する
# 1行3列の要素を表示する
# 2行3列の要素を表示する

print(arr1[0,0])
print(arr1[0,1])
print(arr1[0,2])
print(arr1[1,2])
```

```
2
1
2
1
```

基礎②自作データに基き、大まかな仮説を立てる

1列目と2列目を表示する

print(arr1[:,0])

print(arr1[:,1])

n列目の全体を指定するときは、[:,n-1]

```
# 1列目と2列目を表示する

print(arr1[:,0])
print(arr1[:,1])
```

```
[2 1 2]
[1 3 2]
```

基礎③結果を可視化する

1列目に対してヒストグラムを作成する

ポテチに0円投じる人数、100円投じる人数…

plt.hist(arr1[:,0], bins=6, range=(0,6))

plt.show()

matplotlib.pyplot.histが正式名称です

plt.hist(配列, オプション)

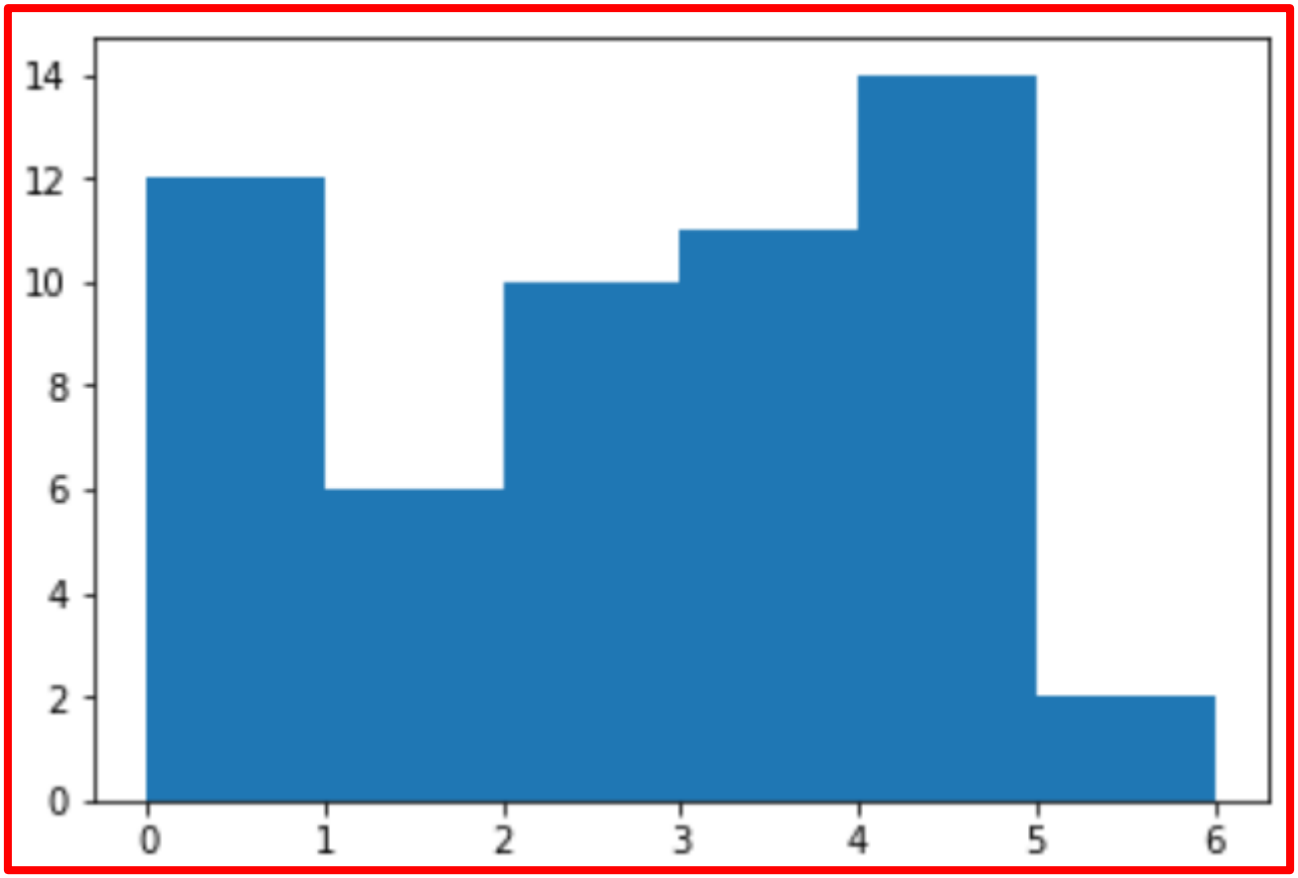
binsは何分割するか、を表します。今回は0,1,2,3,4,5なので6と設定しました。

rangeは、ヒストグラムで表示する範囲を指定するものです。

他にも色を変える、バーの太さを変更する等できます

plt.show()を最後に記述し、ヒストグラムを表示します

基礎③結果を可視化する

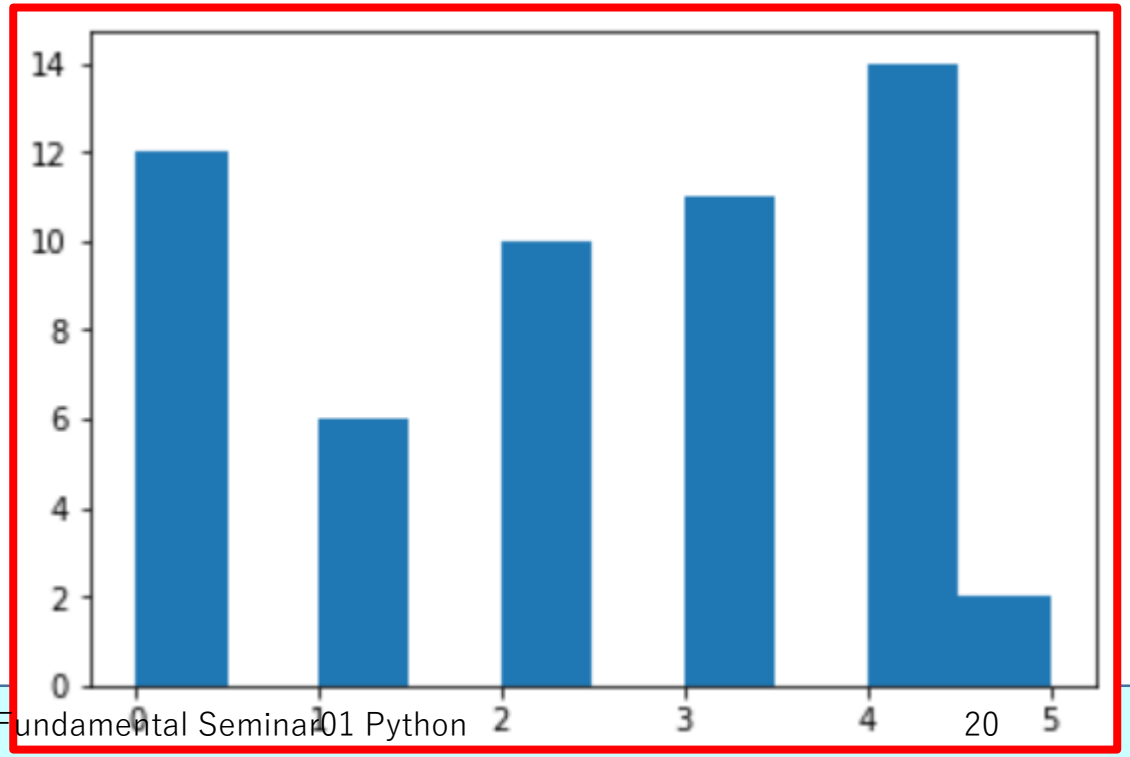


基礎③結果を可視化する

実は設定を減らしてヒストグラムを書いてみるとイマイチだった

plt.hist(arr1[:,0])

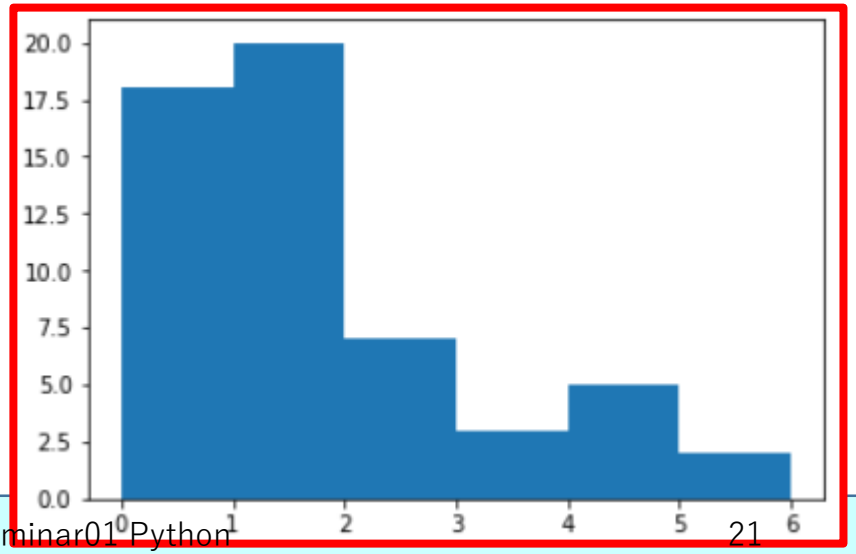
plt.show()



基礎③結果を可視化する

2列目に対してヒストグラムを作成する
チョコに0円投じる人数、100円投じる人数...

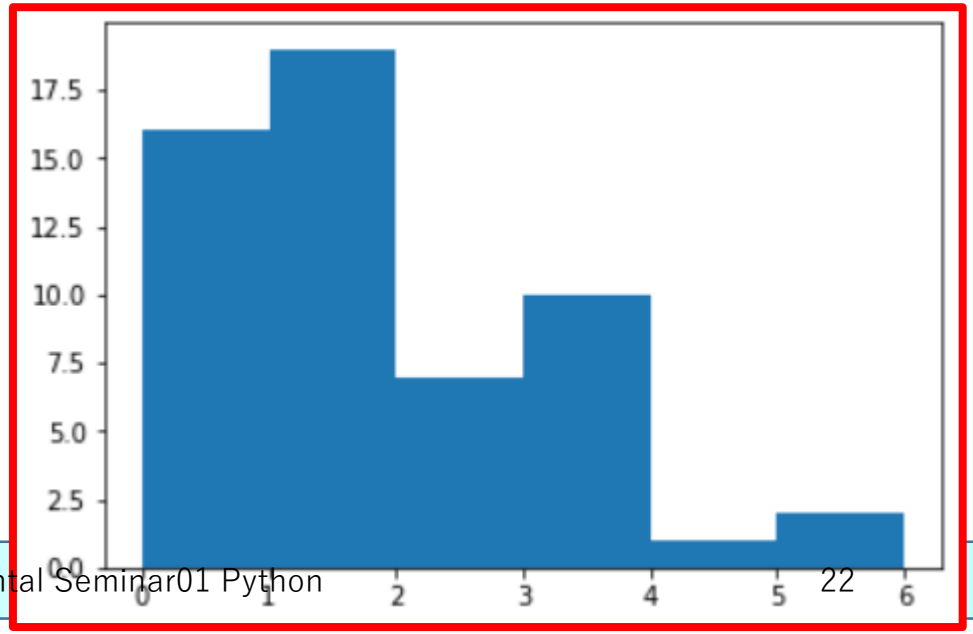
plt.hist(arr1[:,1], bins=6, range=(0,6))
plt.show()



基礎③結果を可視化する

3列目に対してヒストグラムを作成する
飲み物に0円投じる人数、100円投じる人数...

plt.hist(arr1[:,2], bins=6, range=(0,6))
plt.show()



基礎③結果を可視化する

各ヒストグラムを合わせたものを作成する

plt.hist([arr1[:,0],arr1[:,1],arr1[:,2]],stacked=False,bins=6,range=(0,6),label=["chips","chocolate","beverage"])

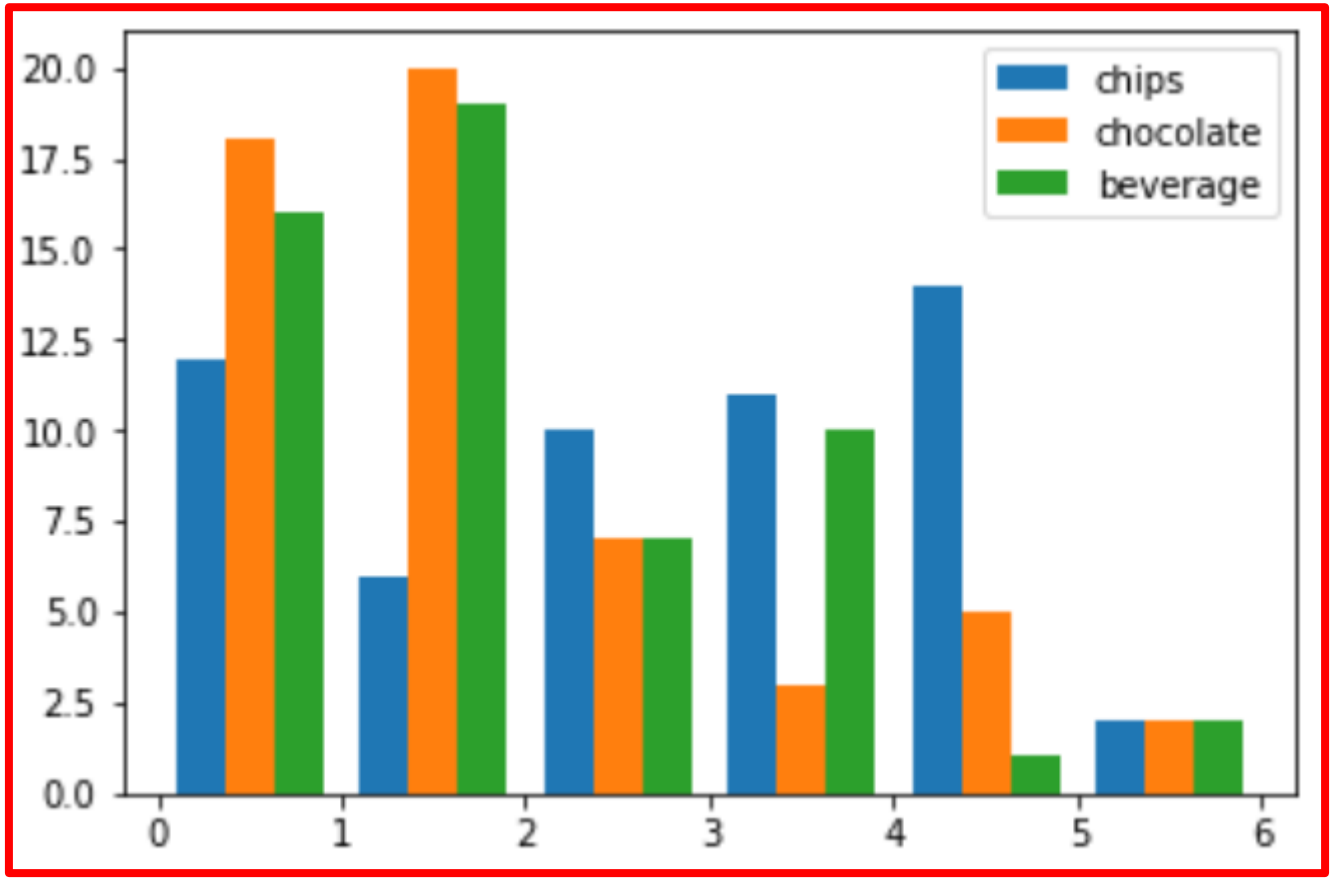
plt.legend()

plt.show()

stacked=Falseは、各要素（chipsなど）を合計して重ねるか、という設定をオフにしていることを意味します。

labelは凡例を載せる機能を持っています。ポテチ、チョコ、飲み物と今回は表しています。

基礎③結果を可視化する



基礎③結果を可視化する

各列(ポテチ、チョコ、飲み物)の平均を求める

ave1 = sum(arr1[:,0])/len(arr1)

ave2 = sum(arr1[:,1])/len(arr1)

ave3 = sum(arr1[:,2])/len(arr1)

print(ave1)

print(ave2)

print(ave3)

```
# 各列（ポテチ、チョコ、飲み物）の平均を求める

ave1 = sum(arr1[:,0])/len(arr1)
ave2 = sum(arr1[:,1])/len(arr1)
ave3 = sum(arr1[:,2])/len(arr1)

print(ave1)
print(ave2)
print(ave3)
```

```
2.272727272727273
1.3272727272727274
1.4
```

基礎③結果を可視化する

sum()は()内にベクトル（専門用語：1次元配列）の、要素の総和を計算してくれます

len()は()内のベクトル（専門用語：1次元配列）の長さ（何行あるか）を測定してくれます

列和を行数（人数）で割ることで平均を算出しました

ave1,ave2,ave3に代入し、print()で表示しています

実は便利なライブラリ(例えばstatisticsなど)を導入すれば、もっと楽に計算できます

基礎③結果を可視化する

得られた平均値を棒グラフで表示してみる

plt.bar([1,2,3],[ave1,ave2,ave3],tick_label=["chips","chocolate","beverage"])

plt.show()

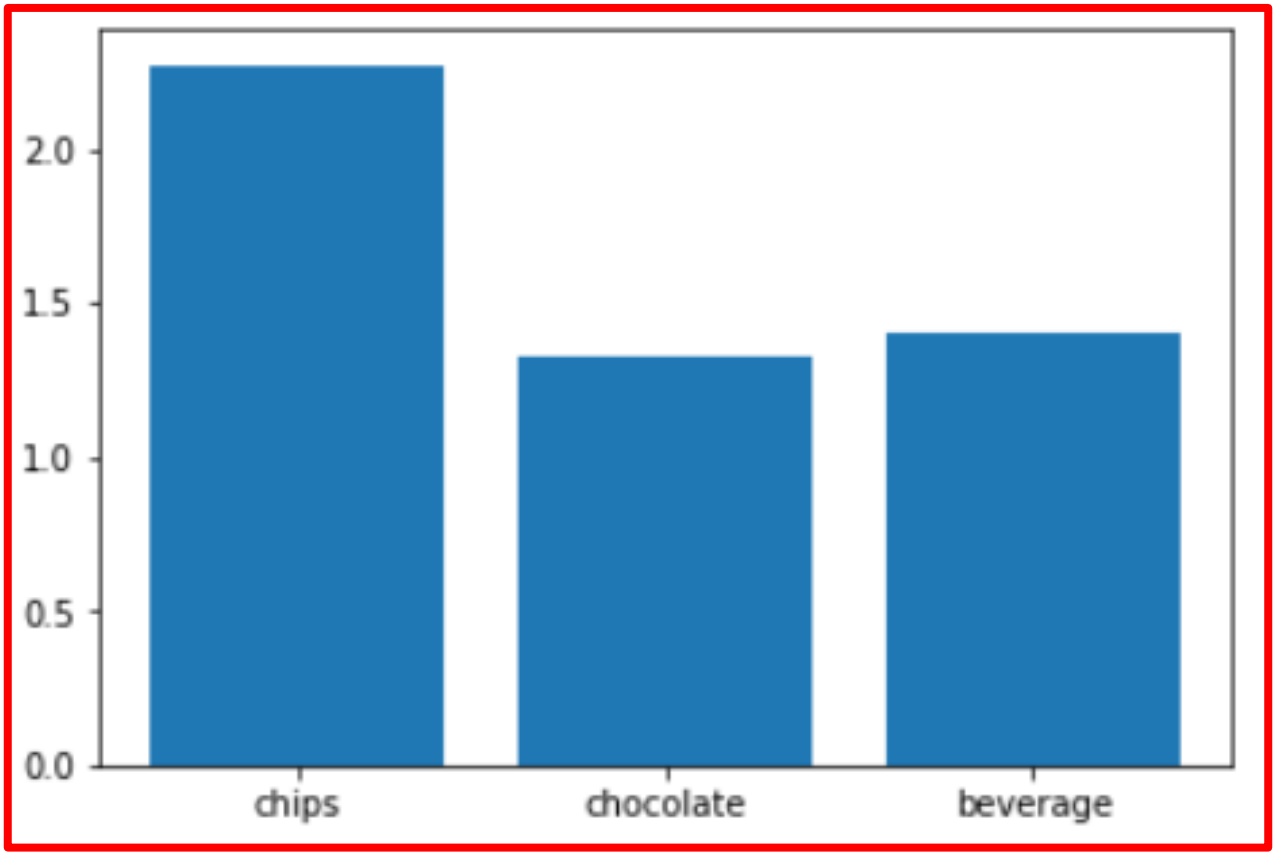
matplotlib.pyplot.barが正式名称

plt.bar([1,2,3,...],[値の配列])が基本形です

最初の[1,2,3]は実は横軸のどの座標に、棒グラフを立てるかを決めています

label=[]を用いることで、各棒に名前をつけることが可能です

基礎③結果を可視化する



基礎③結果を可視化する

円グラフで振り分けのバランスを可視化する

各要素は500円をどのように占めているのか

plt.pie([ave1,ave2,ave3],labels=["chips","chocolate","beverage"])

plt.axis('equal')

plt.show()

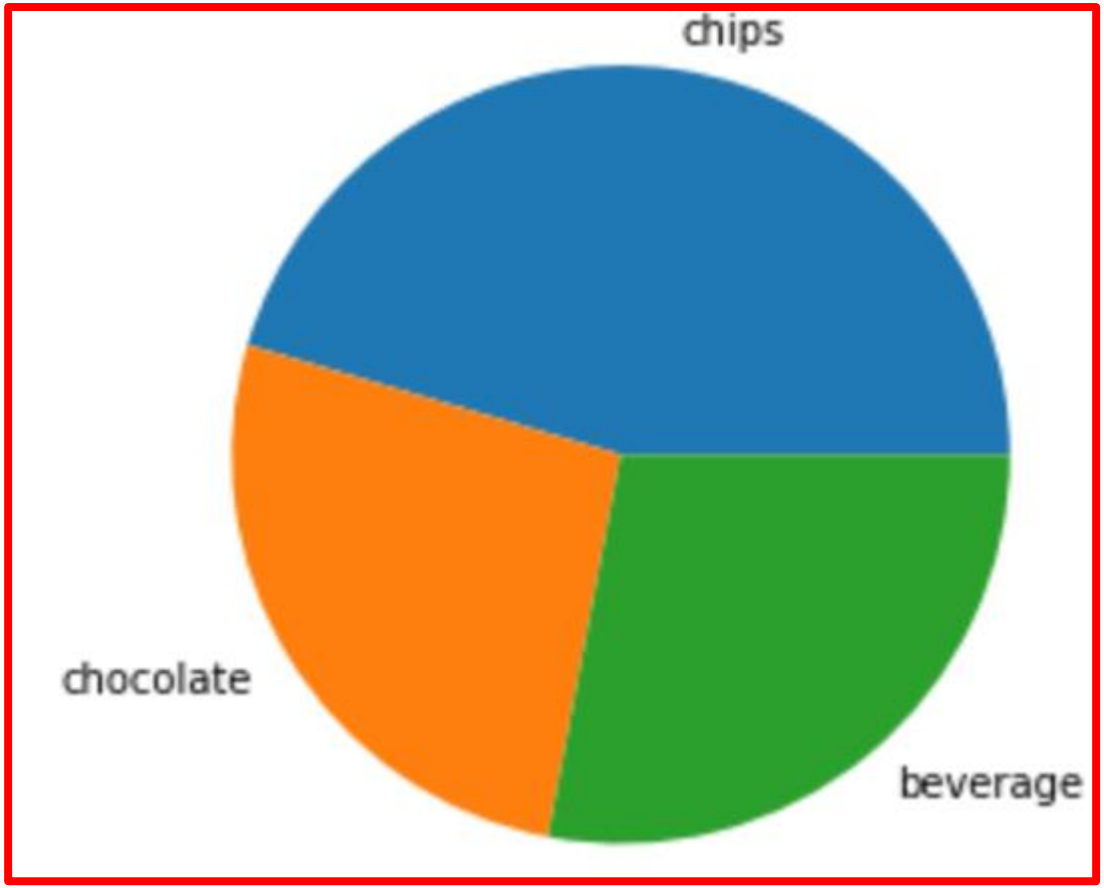
matplotlib.pyplot.pieが正式名称

plt.pie([値の配列],labels=[要素名の配列])が基本形です

細かいですが、labels=とsがつくことに注意してください

plt.axis('equal')がないと、Jupyter Notebookでは楕円になってしまいます

基礎③結果を可視化する

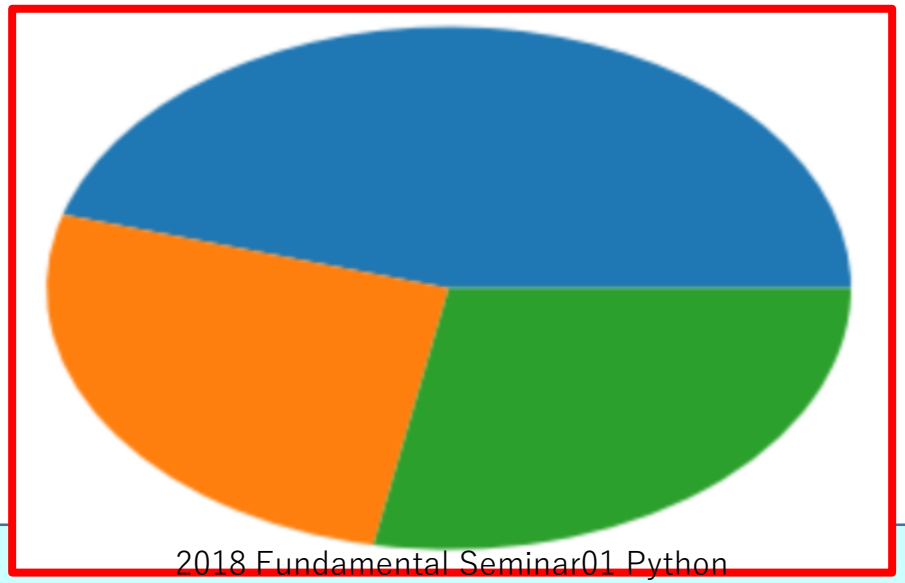


基礎③結果を可視化する

実はシンプルな設定で円グラフを書いてみるとイマイチだった

plt.pie([ave1,ave2,ave3])

plt.show()



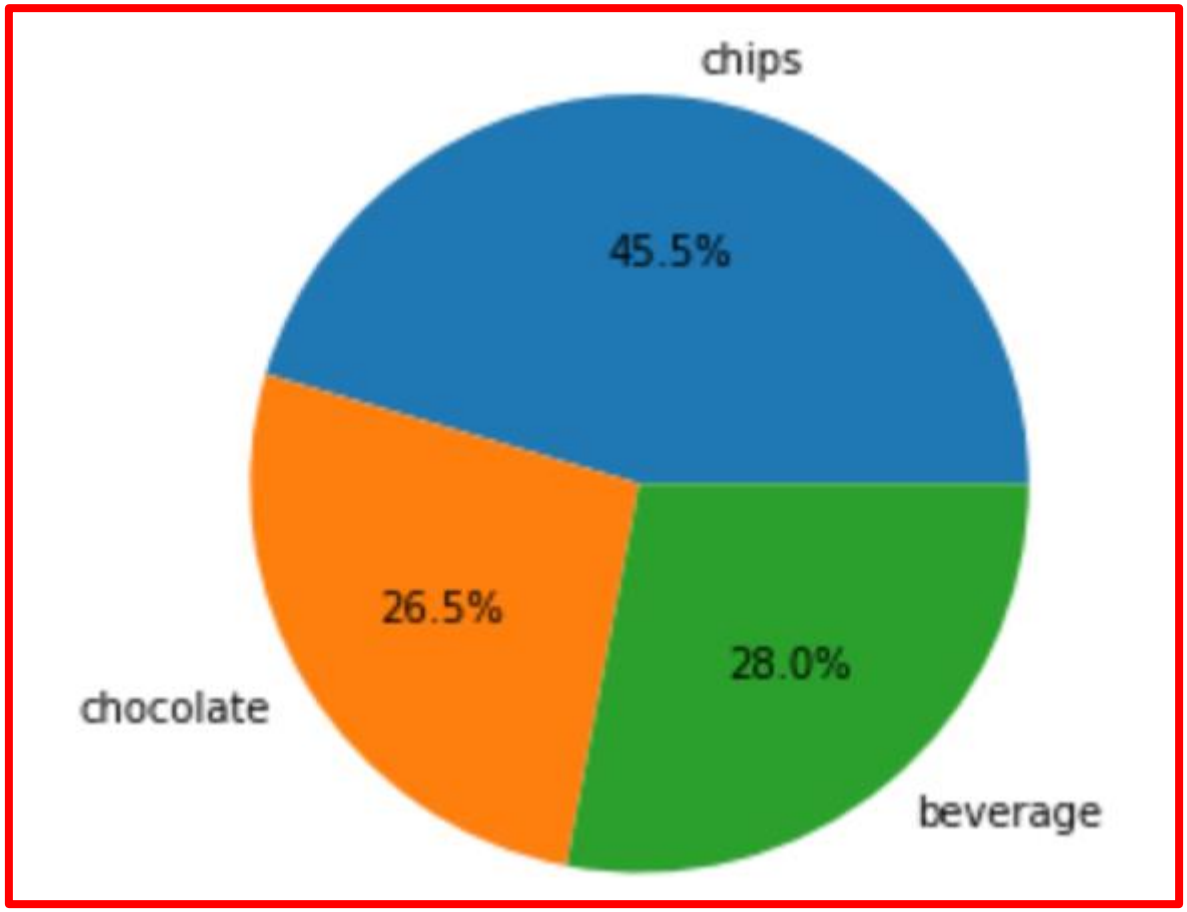
基礎③結果を可視化する

具体的な割合を表示する

```
plt.pie([ave1,ave2,ave3],labels=["chips","chocolate","beverage"],autopct="%.1f%%")
plt.axis('equal')
plt.show()
```

autopct="%.1f%%"を加筆することで、小数点第1位までのパーセント表示ができます

基礎③結果を可視化する



基礎③結果を可視化する

```
# せっかくなので考察しましょう
print("遠足での支出振り分けの傾向について、基礎ゼミ参加者からの意見を頂いて分かったこと")
print("""
- （ヒストグラムより）チョコと飲み物には100円を投じる人が最も多い
- （ヒストグラムより）ポテトチップスには400円を投じる人が最も多い
- （ヒストグラムより）分布は三者三様である
- （円グラフより）チョコと飲み物への予算はほぼ同じである
- （円グラフより）ポテトチップスに最も多く予算が投じられており、それは半分弱である
- （円グラフより）ひとつだけに500円全て投じる人は少なく、多少バランスを考慮している
""")
# '''で囲むと、改行ありの文字列を記述できます
```

基礎③結果を可視化する

```
# せっかくなので考察しましょう

print("遠足での支出振り分けの傾向について、基礎ゼミ参加者からの意見を頂いて分かったこと")
print("""
- （ヒストグラムより）チョコレートと飲み物には100円を投じる人が最も多い
- （ヒストグラムより）ポテトチップスには400円を投じる人が最も多い
- （ヒストグラムより）分布は三者三様である
- （円グラフより）チョコレートと飲み物への予算はほぼ同じである
- （円グラフより）ポテトチップスに最も多く予算が投じられており、それは半分弱である
- （円グラフより）ひとつだけに500円全て投じる人は少なく、多少バランスを考慮している
""")
```

遠足での支出振り分けの傾向について、基礎ゼミ参加者からの意見を頂いて分かったこと

- （ヒストグラムより）チョコレートと飲み物には100円を投じる人が最も多い
- （ヒストグラムより）ポテトチップスには400円を投じる人が最も多い
- （ヒストグラムより）分布は三者三様である
- （円グラフより）チョコレートと飲み物への予算はほぼ同じである
- （円グラフより）ポテトチップスに最も多く予算が投じられており、それは半分弱である
- （円グラフより）ひとつだけに500円全て投じる人は少なく、多少バランスを考慮している

④で紹介する道具

- pandas
- Dataframe.head()
- np.array(Dataframe)
- for文
- range(n)
- sum
- len()
- plt.bar
- plt.pie

基礎④実データを用いた分析

1. 外からcsvファイルを読み込み
2. 計算できる形式にデータ変換
3. インポートしたデータの大きさの確認
4. (行和・列和の計算)
5. (列平均の計算)
6. (ヒストグラムの作成)
7. Sum関数を使わない行和・列和(for文)
8. ポテチに400円投じた人の数え上げ(if文)

基礎④実データを用いた分析

9. (ポテチに400円投じた人の割合の計算)
10. (ポテチに400円投じた人が残り100円を投じた対象物の割合の比較)

④-1外からcsvファイルを読み込み

```
import pandas as pd
df1 = pd.read_csv("limit_over.csv",header=None)
print(df1.head(10))
```

pandas.read_csvが正式名称です

dfはデータフレーム（Dataframe）の略です

header=Noneは元のデータに列名（専門用語：ヘッダー）がないときに書き込みます

読み込むファイル内に日本語が入っていた場合、エンコードの設定が必要ですが、それは応用編で

pd.read_csvで読み込んだデータの型をデータフレームと言います

データフレームに".head(n)"をつけると、データの先頭n行が表示できます

④-1外からcsvファイルを読み込み

```
import pandas as pd
df1 = pd.read_csv("limit_over.csv",header=None)
print(df1.head(10))
```

型とは、例えば"1"を数字の1と見るか、文字として１と見るか、という定義をするものです

例えば、(文字としての1)+(文字としての1)=(文字としての11)となります

(数字としての1)+(数字としての1)=(数字としての2)ですね

④-2計算できる形式にデータ変換

ndarr1 = np.array(df1)
print(ndarr1)

np.array()の()内にデータフレームを入れると型を変換できます
numpy型にすると便利で、所謂数字の計算がしやすくなります

④-3インポートしたデータの大きさの確認

print(len(ndarr1))

len()でデータの長さ（行数）を取得します

実はExcelで読み込める最大の行数がExcel2016では1,048,576行です

つまりimportしたcsvファイルはExcel2016(最新版)では読み込めません

従ってpythonの活躍の場が発生するわけです

実はこの1,048,577行のデータはpythonで平林が作成しました

④-7 Sum関数を使わない行和と列和(for文)

```
total = 0
for i in range(3):
    total = total + ndarr1[0,i]
print(total)
```

total=0は変数の定義と同時に初期値を設定しています

なぜならば、for文の中は異世界であり、変数の定義はできないためです

for文とは、決まった作業を繰り返すコードのことです（繰り返し文）

for文の内部は、for文よりも行頭が右(インデントが深い)場所になります

range(n)とは、[0,1,2,...n-2,n-1]という配列を表します

for i in range(n):とは、i=0,i=1,...i=(n-1)まで作業するよ、という意味です

④-7 Sum関数を使わない行和と列和(for文)

```
total = 0
for i in range(3):
    total = total + ndarr1[0,i]
print(total)
```

```
# total = total + aは、(新total)に((現total)+a)を代入するという意味です
# aだけ足すよ、という意味と同義です
# total += aという特殊な書き方が実はメジャーです
# コロン(:)で閉じましょう
# インデントを揃えましょう
```

④-8ポテチに400円投じた人の数え上げ(if文)

```
total = 0
for i in range(len(ndarr1)):
    if ndarr1[i,0] == 4:
        total = total + 1
print(total)
```

range()にlen()を入れることにより、行列の長さ回数ぴったりに作業を繰り返せます

if文というのは条件によって作業を実行するコードのことです（条件文）

==は同値を意味します。

この場合は、i番目のデータにおいて、ポテチに400円充てた人であれば==が成立します

==が成立したとき、合計に1を加えることで、人数を数えているのです

その他、少し便利な道具（コード）の紹介

① パスの設定
② 2次元配列の作成
③ ファイルの読み込み
④ ファイルの出力

※hを押すとショートカット一覧出現！

少し便利な道具（コード）①

- パスの設定

Import os

path1 = os.path.abspath('')

- 絶対パスを毎回取得すると、ipynbと異なる場所（深層など）にあるデータファイルをインポートしやすくなる
- 研究室PCのパスと、自分のPCのパスの違いを解消できる

少し便利な道具（コード）②

- 2次元配列の作成

arr1 = np.zeros((5,3))

list1 = [[0 for i in range(3)] for j in range(5)]

- np.arrayを用いる方法は、5行3列の0行列を作る方法である
- 後者は、リスト型の2次元配列である
- 例えば2行5列を示す場合、arr1[1,4]とlist1[1][4]と表記が異なる
- np.arrayは文字列を含むことができないが、listでは可能である

少し便利な道具（コード）③

- ファイルの読み込み

変数A = pd.read_csv(“ファイルA.csv”, encoding=“SHIFT-JIS,header=None)

変数B = 変数A.drop([1,2], axis = 0)

変数C = 変数A.drop([“Unnamed: 0”, “列名1”, “列名2”, “列名3”], axis = 1)

- 日本語を含んだファイル、列名がないファイルにも対応できる
- 不要な行と列をpython上で削除できる
- 列名がない場所は”Unnamed: 0”となる

少し便利な道具（コード）④

- ファイルの出力

変数B = pd.DataFrame(変数A)

変数B.to_csv(“ファイルA.csv”, **index = False, header = False, encoding=“SHIFT-JIS”**)

- 出力の際、indexとヘッダーの有無を決められる
- エンコード設定も可能であり、日本語も交え出力できる