提出日：2024/5/15

プログラミング演習　第５回演習レポート

担当教員：杉本　千佳先生

　　　　　　　　　　　　　　　　所属：理工学部　数物・電子情報系学科

　電子情報システムEP

　　　　　　　　　　　　学年・クラス：２年　Fe1

　　　　　　　　　　　　　　学籍番号：2364092

　　　　　　　　　　　　　　　　氏名：熊田　真歩

（１）課題番号：基本課題３

　　　　課題名：逆行列を求める関数の作成

（２）プログラムのフローチャート（関数ごと）

ダイアグラム

自動的に生成された説明　・逆行列を計算するrever(A,revA)関数

ダイアグラム, 概略図

自動的に生成された説明・行列の入力を行うinin()関数

ダイアグラム, 概略図

自動的に生成された説明・行列の出力を行うoutout()関数

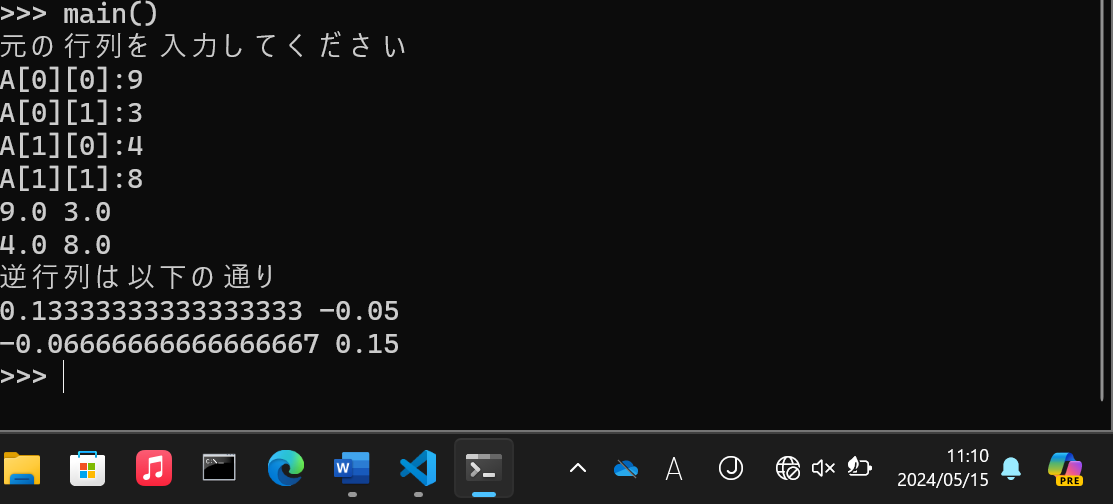
ダイアグラム

自動的に生成された説明・行列式が０であるかを判定し逆行列存在の有無を判定するmain関数

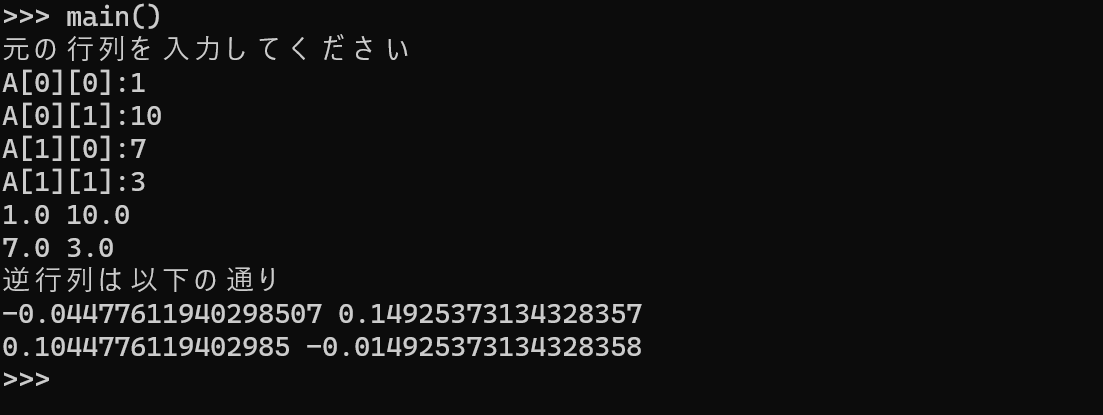
（３）アルゴリズム「正しいこと」である説明

本プログラムでは、どのような要素を持つ２×２行列であってもその逆行列を出力するプログラムを作成した。ここで行列式の値が０となってしまう場合には逆行列の定義式からも明らかな用に、分母に０が来てしまうため逆行列を持たない。このつまり、本プログラムでは逆行列は存在しないというメッセージが出力される。以下に、行列式の値が正の時、負の時、零の時、更には各要素が少数値である時の出力結果を示す。

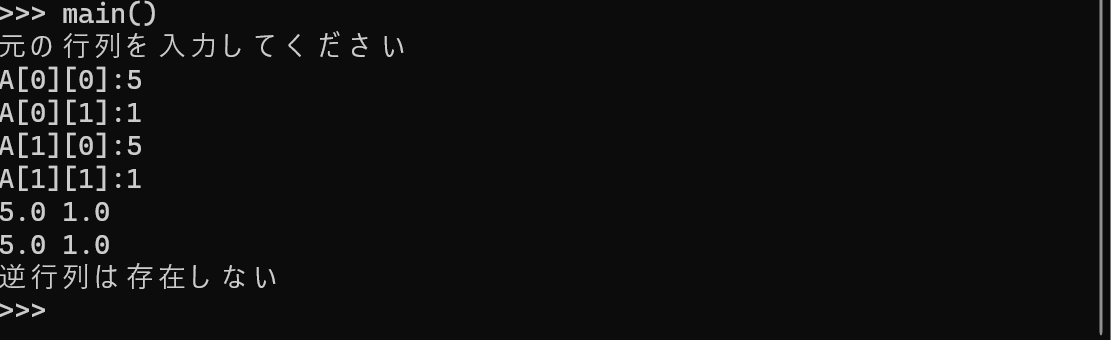
（ⅰ）行列式の値が正の時



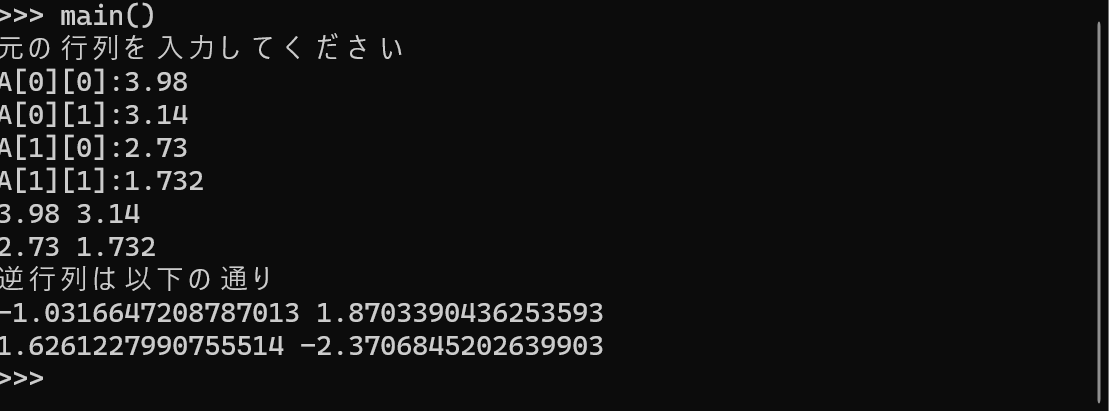
（ⅱ）行列式の値が負の時



（ⅲ）行列式の値が０の時



（ⅳ）各要素が少数値である時



このように、いかなる２×２行列においても要素に数字を入れる限り、正しく逆行列を出力するという観点から本プログラムにおけるアルゴリズムは正しいと言える。

（４）ソース・プログラムの説明

・逆行列を計算するrever(A,revA)関数について

def rever(A,revA): **##関数の定義**

detA = float(A[0][0])\*float(A[1][1])-float(A[0][1])\*float(A[1][0])  **##行列式の計算**

if detA != 0:　　　　　　　　**##行列式の値が０でない時（逆行列が存在する時）**

revA=[[(1/detA)\*A[1][1],-(1/detA)\*A[0][1]],[-(1/detA)\*A[1][0],(1/detA)\*A[0][0]]]

**##逆行列を計算**

outout(revA)　　　　　　　　　　　　**##上の式によって計算した逆行列をoutout(revA)関数により出力**

return detA　　　　　　　　　　　　　　　**##戻り値の値を行列式の値とする。**

・行列に値を入力するinin()関数について

def inin():　　　　　　　　　　　　　　　　**##関数inin()の定義**

print("元の行列を入力してください")　　　**##行列の入力を促すメッセージの出力**

A[0][0] = float(input("A[0][0]:"))　　　　**##１行1列の値の入力**

A[0][1] = float(input("A[0][1]:"))　　　　**##1行2列の値の入力**

A[1][0] = float(input("A[1][0]:"))　　　　**##2行1列の値の入力**

A[1][1] = float(input("A[1][1]:"))　　　　**##2行2列の値の入力**

print(A[0][0],A[0][1])　　　　　　　　　**##1行目の行列の値を出力**

print(A[1][0],A[1][1])　　　　　　　　　**##２行目の行列の値を出力**

・逆行列の値を出力するoutout()関数について

def outout(revA):　　　　　　　　　**##関数outout()の定義**

print("逆行列は以下の通り")　　　**##逆行列は以下の通りというメッセージの出力**

print(revA[0][0],revA[0][1])　　　**##1行目の逆行列を出力**

print(revA[1][0],revA[1][1])　　　**##2行目の逆行列を出力**

・行列式が０か否か判定し逆行列の出力を行うmain（）関数について

def main():　　　　　　　　　　　**##main（）関数の定義**

inin()　　　　　　　　　　　　**##行列の入力を行うinin()関数の呼び出し**

if rever(A,revA) == 0:　　　　**## rever(A,revA)の値を呼び出し、行列式の値が０の時**

print("逆行列は存在しない")　　　**##逆行列は存在しないというメッセージの出力**

（５）考察

本プログラムではあらゆる２×２行列に対して逆行列の出力を行うというプログラムを作成した。今回は、逆行列の計算を行う関数、行列の入力を行う関数、逆行列の出力を行う関数、そして行列式が０であるかを判定し逆行列存在の有無を判定する関数と、４つの関数を用いたプログラムとなった。関数はプログラムを書く上で非常に重要となるが今回のような場合においても複数の関数を適宜利用することによって人間が読んでも見やすく、理解しやすいプログラムとなった。その観点から、４つの関数を採用したアルゴリズムを採用したことは良い選択であったと言える。

また、ここで、数はどのような四則演算においても式は同じであることが知られているがPython等のプログラミング言語によるプログラムにおいて、計算する数が整数であるか少数であるかという事が非常に重要となる。これはPythonの変数型の特性のためである。int型である整数値とfloat型である少数値を同じ式において混在させることはできない。すなわち式においてはすべての型を統一させる必要がある。今回のように計算式に割り算がある時はint型のみの計算であってもその結果がfloat型となることは大いに考えられるため、すべての値をfloat型としてから行列式の計算を行った。このように、割り算を含む計算はfloat型となっても対応できるよう、int型変数をfloat型とすることが望ましい。この選択は本アルゴリズムを正しいものとするために不可欠である。

今回は２×２行列を表現するために二次元のリストを採用した。行列を表現するうえでｎ次正方行列に対応するにはn次元のリストを使うと簡単にその行列の計算が可能となることが本プログラムから理解できる。

（６）参考文献

・２次元リスト作成のために参照したサイト

GeekBlocks　「【Python】2次元リストの使い方(初期化/追加/参照など)を詳しく解説」

https://af-e.net/python-two-dimensional-list/　　2024/05/15アクセス

・フローチャートを作成したサイト

draw.io 「Security-first diagramming for teams.」

https://www.drawio.com/　　　2024/05/15アクセス

（７）感想

　今回は２次元リスト以外、よく精通しているコードのみを用いてコードを書くことができた。そういった観点からは比較的短時間でプログラムを書くことができたが、加点対象である出力を関数とするということに少し時間を要した。関数を分けるとその分見やすく理解しやすいプログラムになる一方で、コードを書く際には関数の引数や戻り値をしっかり意識する必要があり、時間がかかってしまう。無論、これは私の勉強、実践不足であろう。更に多くの経験を積んでどんなコードをも短時間で正しくかける人材を目指して奮闘していきたい。