研究ノート

Python によるコード実装について
- 公開講座と数学セミナーの報告 -

Program cording using software Python
–Report on workshop and seminar –

清水優祐*・及川大知*・大木駿介*
SHIMIZU, Yusuke*; OIKAWA, Daichi*; OKI, Shunsuke*
中原健太*・奈良彩花*・橋本赳宏*
NAKAHARA, Kenta*; NARA, Ayaka*; HASHIMOTO, Takehiro*
堀野尚之*・山口達也*
HORINO, Naoyuki*; YAMAGUCHI, Tatsuya*

概要：IoT の普及などにより，今後ますます IT 技術のニーズは高まっていくと考えられる。そこで，近年注目されているプログラミング言語“Python”によるコード実装の内容を報告する。城西大学理学部数学科が主催している，高校生向け応用数学体験講座の講演内容と，数学科の学生が行ったアクティブラーニングの内容について，実際に作成したコードを紹介する。公開講座のアンケート結果から，統計的プログラミングへの関心が高いことが判明した。今後は Python を取り入れた授業を展開することで，データサイエンスのスキルがある人材の育成が期待できる。

１．はじめに

Python は，様々な分野のアプリケーションで使用されている動的プログラミング言語であり，主に次のような特徴がある（詳細は https://www.python.jp を参照）：

非常にクリーンで読みやすい文法；強力なイントロスペクション機能；手続き型のコードによる自然な表現；広範囲に及ぶ標準ライブラリとモジュール；アプリケーションに組み込んでスクリプトインターフェースとして利用が可能

また，高度に最適化されたコンパイラとライブラリにより，多くのアプリケーションで十分な速度で実行することができる。Windows，Mac，Linux/Unix など，多くのメジャーなオペレーティング・システムで使うことができ，環境構築も容易である。城西大学では，坂戸・紀尾井町の両キャンパスに Python 3 系（2020 年 2 月現在の最新）がインストールされており，授業等で使用

※ 城西大学理学部数学科

することが可能である。

そこで本研究ノートでは，実際に Python を用いてコード実装を行った内容を報告する。2 章では，著者の一人である清水が，城西大学理学部数学科が主催している，応用数学体験講座において Python に関する講演をした際の内容を報告する。3 章では，Python に関するアクティブラーニングの一例を紹介する。具体的には，城西大学理学部数学科 4 年生の数学セミナーにおいて，学生が Python に関する実習課題を自ら見つけ，探求し学修した内容を報告する。まとめと今後の展望は４章で述べる。

２．公開講座の報告

2019 年 12 月 7 日（土）15:00-16:50 に城西大学紀尾井町キャンパスで開催された，城西大学理学部数学科主催の「高校生向け応用数学体験講座 2019」で，「AI チャットボットと話そう」というタイトルで清水が実習講演を行った（図 1)。チャットボットとは，テキストや音声を通じて人との会話を自動的に行うプログラムのことであり，様々な企業がチャットボットプラットフォームを利用し始めている。チャットボットには大きく 3 種類あり，決められたシナリオによって選択式で会話をする選択肢型のもの，登録された単語に対する応答を用意し会話をする辞書型のもの，会話ログを利用して文脈に近しい応答をするログ型のものがある。選択肢型と辞書型は比較的容易にコードが作成できるが，ログ型はディープラーニングをはじめとする統計的機械学習の理論が用いられており，高校生には多少難解であると考えられる。そこで本講座では，Python を用いて簡単な辞書型のチャットボット製作を体験してもらった。実習の流れとしては，まずチャットボットの概要を説明し，その後，(i) おうむ返しをするチャットボット (ii) 簡単な挨拶を返すチャットボット (iii) 辞書型のチャットボット の順で，用意した Python コードを実装させた。以下はそのコードと，参加者のアンケート結果である：

(i) おうむ返しをするチャットボット

```
while True:
    command = input(‘メッセージを入力> ’)
    response = ″″
    response = command
    if not response:
        response = ‘お話ししたいです(;;)’
    print(response)

    if ‘さようなら’ in command:
```

図 1:応用数学体験講座のポスター

```
        break
```

(ii) 簡単な挨拶を返すチャットボット

```
while True:
    command = input(‘メッセージを入力＞’)
    response = ””
    if ‘こんにちは’in command:
        response = ‘こんにちは(ˆωˆ)’
    elif ‘ありがとう’ in command:
        response = ‘どういたしまして m(__)m’
    elif ‘さようなら’ in command:
        response = ‘さようなら(・▽・)ノ’
    elif len(command) == 0:
        response = ‘お話ししたいです(;;)
```

```
    else:
        response = ‘何を言っているのか分かりません(><)’
    print(response)

    if ‘さようなら’in command:
        break
```

(iii) 辞書型のチャットボット

辞書ファイルの作成：

おはよう：おはようございます！

こんにちは：こんにちは！

こんばんは：こんばんは！

ありがとう：どういたしまして！

げんき？：元気です！

なんにち？：12 月 7 日です！

さようなら：さようなら！

```
#実行コード
open_file = open('index.txt', encoding='utf-8')
raw_data = open_file.read()
open_file.close()
lines = raw_data.splitlines()

bot_dict = {}
for line in lines:
    word_list = line.split(':')
    key = word_list[0]
    response = word_list[1]
    bot_dict[key] = response

import re
import webbrowser
import unicodedata
```

```
while True:
    command = input('メッセージを入力> ')
    response = ""
    for key in bot_dict:
        if key in command:
            response = bot_dict[key]
            break
        elif len(command) == 0:
            response = 'お話ししたいです！'
        else:
            response = 0
    if '+' in command or '-' in command or '*' in command or '/' in command or 'たす'
in command or 'ひく' in command or 'かける' in command or 'わる' in command or "[０ー
９]" in command:
        command = command.replace('たす', '+')
        command = command.replace('ひく', '-')
        command = command.replace('かける', '*')
        command = command.replace('わる', '/')
        #command = command.replace('１', '1')
        regex = u'[一-顫ぁ-んァ-ン？！?]'
        src = command
        dst = re.sub(regex, "", src)
        #dtr = u'dst'
        src2 = unicodedata.normalize('NFKC',dst)
        #response = src2
        try:
            str(eval(src2))
        except:
            response = 0
        else:
            response = str(eval(src2)) + 'です！'
    else:
        response = response
```

```
    if response == 0:
        x = 0
        while x != 1:
            response = input('何を言ってるかわかりません, web 検索しますか？Y or N>')
            if "Y" in response or "Y" in response:
                webbrowser.open('https://www.yahoo.co.jp')
                response = "もっとお話ししたいです！"
                x = 1
            elif "N" in response or "N" in response:
                response = '他のお話しをしましょう！'
                x = 1
            else:
                print('YかNを入力してください！')
                #print('¥n')
    else:
        response = response
    print(response)

    if 'さようなら' in command:
        break
```

アンケートの回答者は 7 名おり，“講座内容は面白かったですか？” の問いには，「面白い」と回答したものが 5 名，「どちらともいえない」と回答したものが 1 名であった。感想は，

- LINE の bot などもこのような仕組みだと知り，とても興味がわいた。
- 今後ますますチャットボットが普及していくと思うので，その基礎部分に触れることができて良かったです。もっと詳しく調べたいなと思いました。
- コードの通りに命令が実行されるのは興味深かった。
- 数学がどのように関係しているかについて知りたかった。音声認識はどのようにやっているのかについて講座があったら面白いと思う。
- チャットボットの仕組みがよくわかりました。応用できる範囲が広がるようになればいいと思います。
- もっと勉強してアプリを作りたいなと思いました。

といった意見があり，参加者の興味を惹くことができたと考えられる。

３．Python を用いたアクティブラーニングの展開

理学部数学科では 4 年次に数学セミナーが開講される。解析学・代数学・幾何学・統計学といった数学の分野の中から学生各自が興味のある内容を選び，課題を見つけ一年間しっかり学んでいくことを主眼としている。統計学について学ぶ清水のセミナーでは，Python に関する実習課題を自ら見つけ探求する，アクティブラーニングを取り入れた。その内容を報告する。

3.1 Word Cloud

Word Cloud を用いて，文章を解析しキーワードを視覚化するプログラムの作成を行った。Word Cloud とは，文章中で出現頻度が高い単語を複数選び出し，その頻度に応じた大きさで図示する手法であり，Python はコマンドプロンプトやターミナルでその外部ライブラリを容易に取り込むことができる。このプログラムでは，用意したテキストファイルを読み込み，文章から任意の単語を抜き出して Word Cloud を生成する。コードと画像の例を以下に記載する。

```
#モジュールのインストール(コマンドプロンプトで実行)
pip install janome
pip install matplotlib
pip install wordcloud

#cording: utf-8
from janome.tokenizer import Tokenizer
from wordcloud import WordCloud
t=Tokenizer()

#テキストファイルの読み込み
text_file = open('用意したテキストファイル名.text',encoding='utf-8')
full_text = text_file.read()
full_text = full_text.replace('¥n','')

#単語ごとに品詞の確認
tokens = t.tokenize(full_text)
word_list=[]
for token in tokens:
    word = token.surface #surface:元の単語
```

図 2：「吾輩は猫である」の冒頭部分を解析し，生成した Word Cloud

```
partOfspeech = token.part_of_speech.split(',')[0]
partOfspeech2 = token.part_of_speech.split(',')[1]

    if partOfspeech == '名詞':
        if(partOfspeech2 != '非自立') and (partOfspeech2 != '代名詞') and (partOfspeech2 != '数'):
            word_list.append(word)

#抜き出した単語の表示
word=','.join(word_list)
print(word)

#WordCloud の生成と保存
wordcloud=WordCloud(font_path='HGRGM.TTC',background_color='white',width=800,height=600).generate(word)

wordcloud.to_file('生成した画像のファイル名.png')
text_file.close()
print('End')
```

3.2 オセロの盤面の作成

Python から GUI を構築・操作するための標準ライブラリである tkinter を用いて，オセロの盤面と駒を作成した。

```
from tkinter import
import random

win = Tk()
cv = Canvas(win, width = 400, height = 400,bg="green")
cv.pack()

(i)オセロ盤の作成
cells=range(1,9)
for i in cells:
    ix=50*i
    iy=50*i
    cv.create_line(0,ix,400,ix,fill="black")
    cv.create_line(iy,0,iy,400,fill="black")

(ii)マウスポインターで駒をクリックで描く
def click1(event1):
    cv.create_oval(event1.x-20,event1.y-
20,event1.x+20,event1.y+20,fill="black",tag="oval")
cv.bind("<Button-1>",click1)

def click2(event2):
    cv.create_oval(event2.x-20,event2.y-
20,event2.x+20,event2.y+20,fill="white",tag="oval")
cv.bind("<Button-2>",click2)

(iii)間違えたときに消す
def erase(event3):
    cv.delete("oval")
```

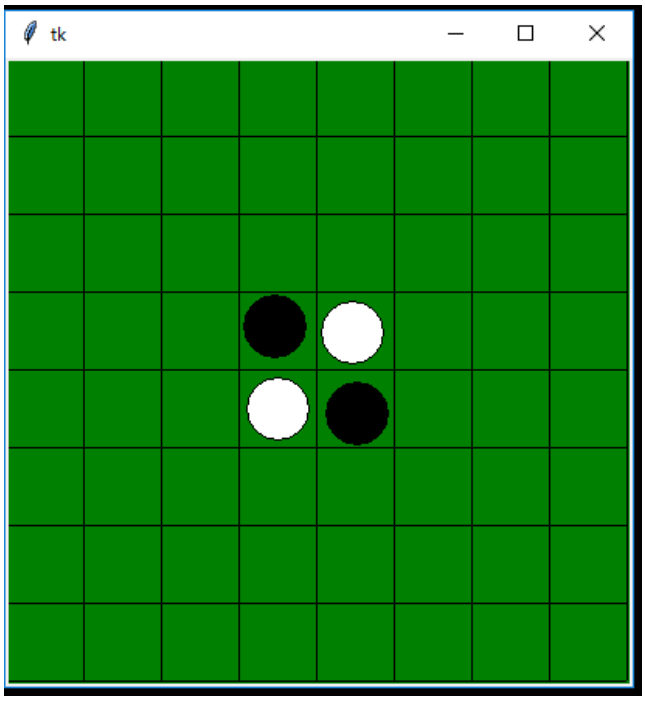

図 3：オセロの盤面

```
    cv.bind("<Button-3>",erase)
    win.mainloop()
```

3.3　条件分岐型診断プログラム

３つの質問に，”YES” or ”NO”で答えると診断結果が出るプログラムを作成した。答えた内容により質問が変化していく，条件分岐型になっている（図 4 参照)。具体例として，インドア派かアウトドア派かを診断するプログラムを作成した。tkinter を用いて，質問にはボタンで答えられるように設定し，ボタンを押すと次の質問に自動で移行する。最後の質問に答えると診断結果が表示される。

```
import tkinter as tk

root = tk.Tk()
root.geometry("1000x500")
root.title("アウトドア診断")

def A5click():
    #結果5
    q34.destroy()
```

```
    by34.destroy()
    bn34.destroy()
    A5.place(x=50,y=50)
    F.place(x=50,y=150)

def A4click():
    #結果4
    q33.destroy()
    q34.destroy()
    by33.destroy()
    bn33.destroy()
    by34.destroy()
    bn34.destroy()
    A4.place(x=50,y=50)
    F.place(x=50,y=150)

def A3click():
    #結果3
    q32.destroy()
    q33.destroy()
    by32.destroy()
    bn32.destroy()
    by33.destroy()
    bn33.destroy()
    A3.place(x=50,y=50)
    F.place(x=50,y=150)

def A2click():
    #結果2
    q31.destroy()
    q32.destroy()
    by31.destroy()
    bn31.destroy()
```

```
    by32.destroy()
    bn32.destroy()
    A2.place(x=50,y=50)
    F.place(x=50,y=150)

def A1click():
    #結果1
    q31.destroy()
    by31.destroy()
    bn31.destroy()
    A1.place(x=50,y=50)
    F.place(x=50,y=150)

def B34click():
    #3-4問目
    q22.destroy()
    by22.destroy()
    bn22.destroy()
    q34.place(x=50,y=50)
    by34.place(x=500,y=150)
    bn34.place(x=650,y=150)

def B33click():
    #3-3問目
    q22.destroy()
    by22.destroy()
    bn22.destroy()
    q33.place(x=50,y=50)
    by33.place(x=500,y=150)
    bn33.place(x=650,y=150)

def B32click():
    #3-2問目
```

```
    q21.destroy()
    by21.destroy()
    bn21.destroy()
    q32.place(x=50,y=50)
    by32.place(x=500,y=150)
    bn32.place(x=650,y=150)

def B31click():
    #3-1問目
    q21.destroy()
    by21.destroy()
    bn21.destroy()
    q31.place(x=50,y=50)
    by31.place(x=500,y=150)
    bn31.place(x=650,y=150)

def B22click():
    #2-2問目
    q1.destroy()
    by1.destroy()
    bn1.destroy()
    q22.place(x=50,y=50)
    by22.place(x=500,y=150)
    bn22.place(x=650,y=150)

def B21click():
    #2-1問目
    q1.destroy()
    by1.destroy()
    bn1.destroy()
    q21.place(x=50,y=50)
    by21.place(x=500,y=150)
    bn21.place(x=650,y=150)
```

```
def B1click():
    #1問目
    label1.destroy()
    b1.destroy()
    q1.place(x=50,y=50)
    by1.place(x=500,y=150)
    bn1.place(x=650,y=150)

label1 = tk.Label(root,text="アウトドア診断をします",font=("Helvetica",30))
label1.place(x=50,y=50)
b1=tk.Button(root,text="OK",font=("Helvetica",26),command=B1click)
b1.place(x=500,y=150)

#質問
q1 =tk.Label(root,text="少し面倒なことも楽しめますか？",font=("Helvetica",30))
q21=tk.Label(root,text="開放感のある場所は好き？",font=("Helvetica",30))
q22=tk.Label(root,text="屋内より屋外で活動したい？",font=("Helvetica",30))
q31=tk.Label(root,text="初対面の人とでも会話を続けられる？",font=("Helvetica",30))
q32=tk.Label(root,text="人が多いところは好き？",font=("Helvetica",30))
q33=tk.Label(root,text="自分の時間より他人との時間が大事？",font=("Helvetica",30))
q34=tk.Label(root,text="他人との行動は好き？",font=("Helvetica",30))

#ボタン
by1=tk.Button(root,text="YES",font=("Helvetica",26),command=B21click)
bn1=tk.Button(root,text="NO",font=("Helvetica",26),command=B22click)
by21=tk.Button(root,text="YES",font=("Helvetica",26),command=B31click)
bn21=tk.Button(root,text="NO",font=("Helvetica",26),command=B32click)
by22=tk.Button(root,text="YES",font=("Helvetica",26),command=B33click)
bn22=tk.Button(root,text="NO",font=("Helvetica",26),command=B34click)
by31=tk.Button(root,text="YES",font=("Helvetica",26),command=A1click)
bn31=tk.Button(root,text="NO",font=("Helvetica",26),command=A2click)
by32=tk.Button(root,text="YES",font=("Helvetica",26),command=A2click)
bn32=tk.Button(root,text="NO",font=("Helvetica",26),command=A3click)
```

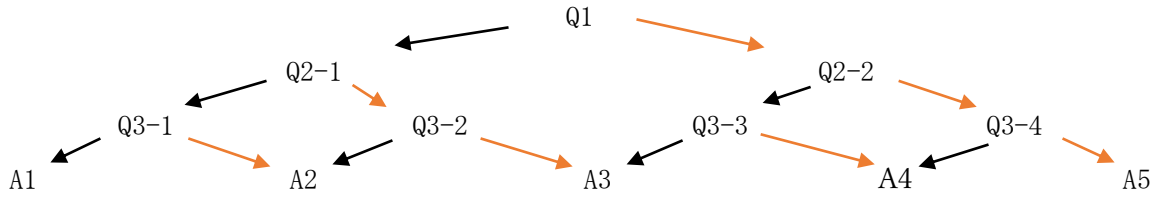

図 4：診断のアルゴリズム　　YES ⟶　NO ⟶　Q：質問　A：診断結果

```
by33=tk.Button(root,text="YES",font=("Helvetica",26),command=A3click)
bn33=tk.Button(root,text="NO",font=("Helvetica",26),command=A4click)
by34=tk.Button(root,text="YES",font=("Helvetica",26),command=A4click)
bn34=tk.Button(root,text="NO",font=("Helvetica",26),command=A5click)

#結果
A1=tk.Label(root,text="あなたは活発なアウトドア派です！",font=("Helvetica",30))
A2=tk.Label(root,text="あなたはアウトドアよりのタイプ！",font=("Helvetica",30))
A3=tk.Label(root,text="アウトドアもインドアも楽しめるタイプ？",font=("Helvetica",30))
A4=tk.Label(root,text="あなたはインドアよりのタイプ！",font=("Helvetica",30))
A5=tk.Label(root,text="あなたは生粋のインドア派です！",font=("Helvetica",30))

F=tk.Label(root,text="ウィンドウを閉じて診断を終了します",font=("Helvetica",30))

root.mainloop()
```

3.4　肥満度測定ツール

身長と体重を入力し，肥満度を計算するプログラムを作成した。一般的な肥満度計算に用いられるのは BMI であるが，満 3 ヵ月から 5 歳児対象の“カウプ指数”と学童期対象の“ローレル指数”も考慮した。tkinter を用いて，入力をボタン形式にし，操作の簡便性を図った。入力値に対して，その人が痩せているか肥満体かといった結果が出力される。

```
import math
from tkinter import *

root = Tk()
```

```
cv = Canvas(root,width = 900,height = 500)
cv.pack()
root.title("肥満度診断")

#選択画面
def B1click():
    label1.destroy()
    b1.destroy()
    q1.place(x=50,y=150)
    b2.place(x=350,y=100)
    b3.place(x=350,y=250)
    b4.place(x=350,y=400)

#カウプ指数の身長と体重の入力
def A1click():
    q1.destroy()
    b2.destroy()
    b3.destroy()
    b4.destroy()
    q11.place(x=50,y=50)
    EditBox1.pack()
    EditBox2.pack()
    EditBox1.place(x=300,y=100)
    EditBox2.place(x=300,y=130)
    b20.place(x=300,y=300)

#ローレル指数の身長と体重の入力
def A2click():
    q1.destroy()
    b2.destroy()
    b3.destroy()
    b4.destroy()
    q11.place(x=50,y=50)
```

```
    EditBox3.pack()
    EditBox4.pack()
    EditBox3.place(x=300,y=100)
    EditBox4.place(x=300,y=130)
    b30.place(x=300,y=300)

#BMIの身長と体重の入力
def A3click():
    q1.destroy()
    b2.destroy()
    b3.destroy()
    b4.destroy()
    q11.place(x=50,y=50)
    EditBox5.pack()
    EditBox6.pack()
    EditBox5.place(x=300,y=100)
    EditBox6.place(x=300,y=130)
    b40.place(x=300,y=300)

#カウプ指数の結果
def C1click():
    q11.destroy()
    b20.destroy()
    height1 = float(EditBox1.get())
    weight1 = float(EditBox2.get())
    KAUP = weight1/((height1/100)**2)
    K=Label(root,text= KAUP,font=("Helvetica",30))
    K.place(x=300,y=170)
    if KAUP < 15:
        K1.place(x=300,y=220)
    elif 15 <= KAUP <= 17:
        K2.place(x=300,y=220)
    else:
```

```
        K3.place(x=300,y=220)
    q21.place(x=50,y=50)
    F.place(x=300,y=350)
    E.place(x=300,y=400)

#ローレル指数の結果
def C2click():
    q11.destroy()
    b30.destroy()
    height2 = float(EditBox3.get())
    weight2 = float(EditBox4.get())
    ROH = (weight2*10)/((height2/100)**3)
    R=Label(root,text= ROH,font=("Helvetica",30))
    R.place(x=300,y=170)
    if ROH < 100:
        R1.place(x=300,y=220)
    elif 100 <= ROH < 115:
        R2.place(x=300,y=220)
    elif 115 <= ROH < 145:
        R3.place(x=300,y=220)
    elif 145 <= ROH < 160:
        R4.place(x=300,y=220)
    else:
        R5.place(x=300,y=220)
    q22.place(x=50,y=50)
    F.place(x=300,y=350)
    E.place(x=300,y=400)

#BMIの結果
def C3click():
    q11.destroy()
    b40.destroy()
    height3 = float(EditBox5.get())
```

```
    weight3 = float(EditBox6.get())
    BMI = weight3/((height3/100)**2)
    B=Label(root,text= BMI,font=("Helvetica",30))
    B.place(x=300,y=170)
    if BMI < 18.5:
        B1.place(x=300,y=220)
    elif 18.5 <= BMI < 25:
        B2.place(x=300,y=220)
    elif 25 <= BMI < 30:
        B3.place(x=300,y=220)
    elif 30 <= BMI < 35:
        B4.place(x=300,y=220)
    elif 35 <= BMI < 40:
        B5.place(x=300,y=220)
    else:
        B6.place(x=300,y=220)
    q23.place(x=50,y=50)
    F.place(x=300,y=350)
    E.place(x=300,y=400)

#初めのラベルとボタン
labell = Label(root,text="肥満度測定を始めます",font=("Helvetica",30))
labell.place(x=50,y=50)
b1=Button(root,text="OK",font=("Helvetica",26),command=B1click)
b1.place(x=500,y=150)

#ボタンの一覧
b2=Button(root,text="生後3ヶ月から5歳",font=("Helxetica",26),command=A1click)
b3=Button(root,text="小学生",font=("Helvetica",26),command=A2click)
b4=Button(root,text="上記以外の方",font=("Helvetica",26),command=A3click)
b20=Button(root,text="OK",font=("Helvetica",26),command=C1click)
b30=Button(root,text="OK",font=("Helvetica",26),command=C2click)
b40=Button(root,text="OK",font=("Helvetica",26),command=C3click)
```

```
#終了のラベル
F=Label(root,text="これで肥満度測定を終わります。",font=("Helvetica",30))
E=Label(root,text="ウィンドウを閉じて下さい。",font=("Helvetica",30))

#診断結果のラベル
K1=Label(root,text="お子様は痩せ気味です",font=("Helvetica",30))
K2=Label(root,text="お子様の体型は普通です",font=("Helvetica",30))
K3=Label(root,text="お子様は太り気味です",font=("Helvetica",30))
R1=Label(root,text="あなたは痩せ過ぎです",font=("Helvetica",30))
R2=Label(root,text="あなたは痩せ気味です",font=("Helvetica",30))
R3=Label(root,text="あなたの体型は普通です",font=("Helvetica",30))
R4=Label(root,text="あなたは太り気味です",font=("Helvetica",30))
R5=Label(root,text="あなたは太り過ぎです",font=("Helvetica",30))
B1=Label(root,text="あなたは低体重(やせ型)です",font=("Helvetica",30))
B2=Label(root,text="あなたは普通体重です",font=("Helvetica",30))
B3=Label(root,text="あなたは肥満 (1度) です",font=("Helvetica",30))
B4=Label(root,text="あなたは肥満 (2度) です",font=("Helvetica",30))
B5=Label(root,text="あなたは肥満 (3度) です",font=("Helvetica",30))
B6=Label(root,text="あなたは肥満 (4度) です",font=("Helvetica",30))

#その他のラベル
q1 =Label(root,text="対象者",font=("Helvetica",30))
q11=Label(root,text="身長と体重を入力してください",font=("Helvetica",30))
q21=Label(root,text="お子様のカウプ指数",font=("Helvetica",30))
q22=Label(root,text="ローレル指数",font=("Helvetica",30))
q23=Label(root,text="BMI",font=("Helvetica",30))

#身長と体重を入力
EditBox1 = Entry(width=50)
EditBox1.insert(END,"身長(cm)")
EditBox2 = Entry(width=50)
EditBox2.insert(END,"体重(kg)")
EditBox3 = Entry(width=50)
```

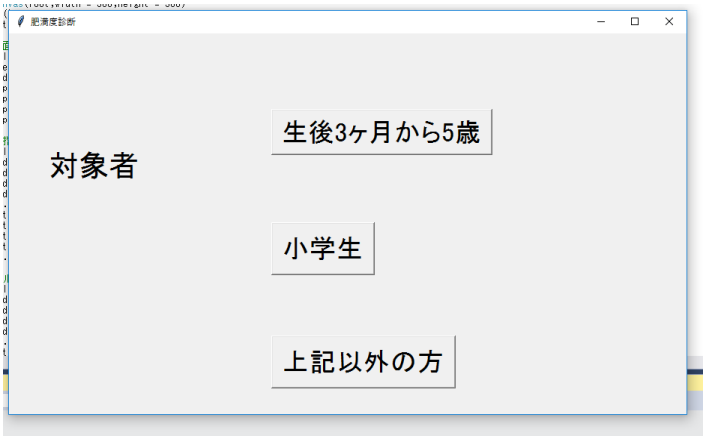

図5：対象者の選択画面

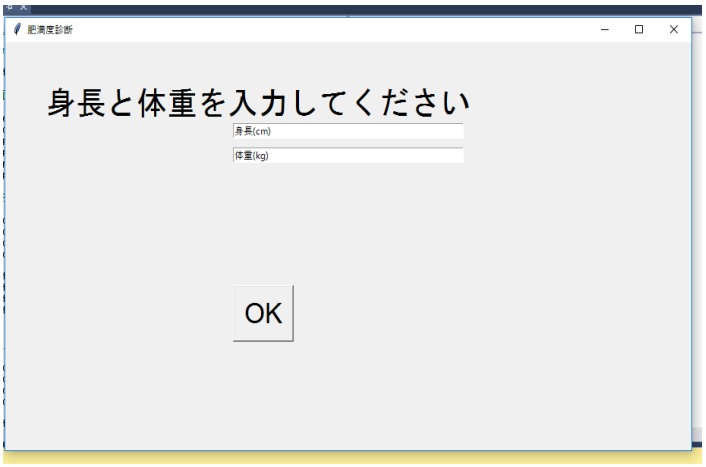

図 6：入力画面

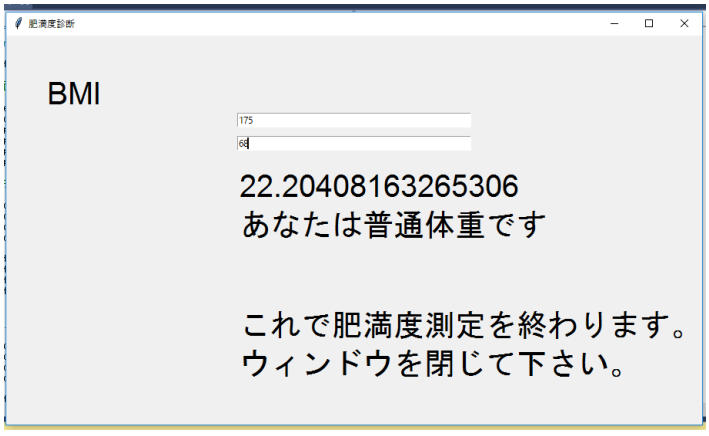

図 7：診断結果

```
EditBox3.insert(END,"身長(cm)")
EditBox4 = Entry(width=50)
EditBox4.insert(END,"体重(kg)")
EditBox5 = Entry(width=50)
EditBox5.insert(END,"身長(cm)")
EditBox6 = Entry(width=50)
EditBox6.insert(END,"体重(kg)")
root.mainloop()
```

3.5　一筆書きゲーム

白い通路を桜の花びらのアイコンを動かし，一筆書きで全ての通路をピンク色で塗ればクリアするという一筆書きプログラムを作成した。操作はキーボードの矢印キーを用いる。クリア後は次のステージに自動的に進み，7 ステージまで設定した。

```
import tkinter
import tkinter.messagebox

idx = 0  #ゲーム進行を管理するインデックス
tmr = 0  #ゲーム進行を管理するタイマー
stage = 1  #ステージを管理する変数
ix = 0  #横座標
iy = 0  #縦座標
key = 0  #キーの値を入れる変数

#キーを押した時に実行する関数の定義
def key_down(e):  #キーを押した時の関数
    global key
    key = e.keysym

def key_up(e):  #キーを離した時の関数
    global key
    key = 0
```

```
#空のリストで迷路定義
maze = [[],[],[],[],[],[],[],[]]

#関数内で7ステージ分の迷路を変数stageで定義。通路が0，通った部分が1，壁が9
def stage_data():
    global ix,iy
    global maze #リスト全体を変更するのでglobal宣言必須
    if stage == 1:
        ix = 1
        iy = 1
        maze =[
        [9,9,9,9,9,9,9,9,9,9,9,9,9,9,9],
        [9,0,9,0,0,0,9,0,0,9,9,9,0,9],
        [9,0,9,0,9,0,9,0,0,9,9,0,0,9],
        [9,0,9,0,9,0,9,0,0,0,0,0,0,9],
        [9,0,9,0,9,0,9,0,0,9,0,9,0,9],
        [9,0,9,0,9,0,9,9,0,9,0,9,0,9],
        [9,0,9,0,9,0,0,0,0,9,0,9,0,9],
        [9,0,9,0,9,9,9,9,9,0,0,9,0,9],
        [9,0,0,0,9,9,9,9,9,0,0,0,0,9],
        [9,9,9,9,9,9,9,9,9,9,9,9,9,9,9],
        ]
    if stage == 2:
        ix = 10
        iy = 7
        maze =[
        [9,9,9,9,9,9,9,9,9,9,9,9,9,9,9],
        [9,0,0,0,0,0,0,0,0,0,0,0,0,0,9],
        [9,0,0,0,0,0,0,9,0,0,0,0,0,0,9],
        [9,0,0,9,9,0,0,9,0,0,0,0,0,0,9],
        [9,0,0,9,9,0,0,9,0,0,0,0,0,0,9],
        [9,9,9,9,9,0,0,9,0,0,0,0,0,0,9],
        [9,9,9,9,9,0,0,0,0,0,0,0,0,0,9],
```

```
    [9,0,0,0,0,0,0,0,9,0,0,9,0,9],
    [9,0,0,0,9,9,9,9,9,0,0,0,0,9],
    [9,9,9,9,9,9,9,9,9,9,9,9,9,9],
    ]
if stage == 3:
    ix = 7
    iy = 2
    maze =[
    [9,9,9,9,9,9,9,9,9,9,9,9,9,9],
    [9,9,9,0,0,0,0,9,9,0,0,0,0,9],
    [9,9,0,0,9,9,0,0,9,0,0,0,0,9],
    [9,0,0,0,0,0,0,0,0,0,0,0,0,9],
    [9,0,9,0,0,0,9,0,0,9,0,0,0,9],
    [9,0,0,0,0,9,0,0,9,0,0,0,0,9],
    [9,9,0,0,0,0,0,9,9,0,0,0,0,9],
    [9,0,0,0,9,9,9,9,9,0,0,9,9,9],
    [9,0,0,0,9,9,9,9,9,0,0,9,9,9],
    [9,9,9,9,9,9,9,9,9,9,9,9,9,9],
    ]
if stage == 4:
    ix = 7
    iy = 8
    maze =[
    [9,9,9,9,9,9,9,9,9,9,9,9,9,9],
    [9,0,0,0,0,0,0,0,0,0,0,0,9,9],
    [9,0,0,0,0,0,0,0,0,0,0,9,9,9],
    [9,9,0,0,0,0,0,0,0,9,0,0,0,9],
    [9,0,0,0,0,0,0,9,9,0,0,0,0,9],
    [9,0,0,0,0,0,0,0,0,0,9,0,0,9],
    [9,0,0,9,9,0,0,9,0,0,0,0,0,9],
    [9,0,0,0,9,0,0,9,0,0,0,0,0,9],
    [9,9,9,0,0,0,0,0,9,0,0,0,0,9],
    [9,9,9,9,9,9,9,9,9,9,9,9,9,9],
```

```
    ]
if stage == 5:
    ix = 12
    iy = 1
    maze =[
    [9,9,9,9,9,9,9,9,9,9,9,9,9,9],
    [9,9,0,0,0,0,0,0,0,0,0,0,0,9],
    [9,0,0,0,0,9,0,0,0,0,0,9,0,9],
    [9,0,0,0,0,9,0,0,0,0,0,9,0,9],
    [9,9,9,0,0,0,0,0,9,0,0,0,0,9],
    [9,0,0,0,9,0,0,0,0,0,0,0,0,9],
    [9,0,0,0,9,9,0,9,0,0,0,0,0,9],
    [9,0,0,0,9,0,0,0,0,0,0,0,0,9],
    [9,0,0,0,0,0,9,0,0,0,0,0,0,9],
    [9,9,9,9,9,9,9,9,9,9,9,9,9,9],
    ]
if stage == 6:
    ix = 9
    iy = 3
    maze =[
    [9,9,9,9,9,9,9,9,9,9,9,9,9,9],
    [9,0,0,9,0,0,9,9,0,0,0,0,0,9],
    [9,0,0,9,0,0,0,0,0,0,0,0,0,9],
    [9,0,0,0,0,0,0,0,9,0,0,0,0,9],
    [9,0,9,0,0,9,9,0,0,0,0,0,0,9],
    [9,0,0,0,0,9,0,0,0,0,0,0,0,9],
    [9,0,0,9,0,9,0,0,0,0,0,0,0,9],
    [9,0,0,0,0,0,0,9,9,0,0,9,0,9],
    [9,0,0,0,0,0,9,0,0,0,0,0,0,9],
    [9,9,9,9,9,9,9,9,9,9,9,9,9,9],
    ]
if stage == 7:
    ix = 2
```

```
        iy = 3
        maze =[
        [9,9,9,9,9,9,9,9,9,9,9,9,9,9],
        [9,9,9,9,9,9,0,0,0,0,0,0,9,9],
        [9,0,0,9,9,0,0,0,0,0,0,0,0,9],
        [9,0,0,0,0,0,0,0,9,0,0,0,0,9],
        [9,0,0,0,0,0,0,0,9,0,9,0,0,9],
        [9,0,0,9,0,0,0,0,0,0,0,0,0,9],
        [9,0,0,0,0,9,0,9,0,9,0,0,0,9],
        [9,0,0,0,0,0,0,0,0,0,9,0,0,9],
        [9,0,0,0,0,0,0,0,0,0,0,0,0,9],
        [9,9,9,9,9,9,9,9,9,9,9,9,9,9],
        ]
    maze[iy][ix] = 1

def draw_bg():
    #10×14 の迷路の幅 40 のマスを定義し,
    for y in range(10):
        for x in range (14):
            gx = 40*x
            gy = 40*y
            #通路 (0) の四角を白色に, 壁 (9) の四角に画像イメージ hana を描画
            if maze[y][x] == 0:
                cvs.create_rectangle(gx,gy,gx+40,gy+40,fill="white",width=0,tag="BG")
            if maze[y][x] == 9:
                cvs.create_image(gx+20,gy+20,image=hana,tag="BG")
    #ステージ数を表示する
    cvs.create_text(60,15,text="STAGE "+str(stage),fill="darkorchid",font=("Segoe
UI",20,"bold"),tag="BG")
    gx = 40*ix
    gy = 40*iy
    #塗った場所をピンク色で塗る
    cvs.create_rectangle(gx,gy,gx+40,gy+40,fill="pink",width=0,tag="BG")
```

```
    #sakura 画像を表示する
    cvs.create_image(gx+20,gy+20,image=sakura,tag="SAKURA")

#"BG"と"SAKURA"のタグが付いたものを消す
def erase_bg():
    cvs.delete("BG")
    cvs.delete("SAKURA")

#sakura 画像を動かす関数を定義
def move_sakura():
    global idx,tmr,ix,iy,key
    bx = ix
    by = iy
    if key == "Left" and maze[iy][ix-1] == 0: #←が押され，かつ，現在位置の左が通路の
時
        ix = ix-1                              #左に移動。以下省略
    if key == "Right" and maze[iy][ix+1] == 0:
        ix = ix+1
    if key == "Up" and maze[iy-1][ix] == 0:
        iy = iy-1
    if key == "Down" and maze[iy+1][ix] == 0:
        iy = iy+1
    if ix != bx or iy != by: #移動した後に
        maze[iy][ix] = 2
        gx = 40*ix
        gy = 40*iy
        #移動する前の座標を通路と同じ色で塗り
        cvs.create_rectangle(gx,gy,gx+40,gy+40,fill="pink",width=0,tag="BG")
        #移動する前の"SAKURA"タグのある sakura 画像をいったん消す
        cvs.delete("SAKURA")
        #移動した後の座標に sakura 画像を表示
        cvs.create_image(gx+20,gy+20,image=sakura,tag="SAKURA")
    #g か大文字の G か左の Shift を押した時,
```

```
    if key == "g" or key == "G" or key == "Shift_L":
        #キーを初期化して yesno メッセージを表示。yes を押す（ret == TRUE）とステージを
初期化して最初からやり直せる
        key = 0
        ret = tkinter.messagebox.askyesno("初めから","やり直しますか？")
        if ret == True:
            stage_data()
            erase_bg()
            draw_bg()

#ステージ内の全ての四角の通路の数を数える関数を定義（通路の数 cnt が 0 になったらクリ
ア）
def count_tile():
    cnt = 0
    for y in range(10):
        for x in range(14):
            if maze[y][x] == 0:
                cnt = cnt + 1
    return cnt

#idx の値で処理を分ける
def game_main():
    global idx, tmr, stage
    #ゲームを開始する処理
    if idx == 0:
        stage_data()
        draw_bg()
        idx = 1
    #ステージクリア判定処理
    if idx == 1:
        move_sakura()
        if count_tile() == 0:
            txt = "STAGE CLEAR"
```

```
            if stage == 7:
                txt = "ALL STAGE CLEAR!"
            cvs.create_text(270,180,text=txt,fill="deeppink",font=("Segoe
UI",40,"bold"),tag="BG")
            idx = 2
            tmr = 0
    #次のステージに進む
    if idx == 2:
        tmr = tmr + 1
        if tmr == 30:
            if stage < 7:
                stage = stage + 1
                stage_data()
                erase_bg()
                draw_bg()
                idx = 1
    #0.2秒後再びこの関数を実行
    root.after(200,game_main)

root = tkinter.Tk()  #ウインドウのオブジェクトを作る
root.title("一筆書きゲーム")  #タイトル指定
root.resizable(False,False)
root.bind("<KeyPress>",key_down)  #キーを押した時に実行する関数を指定
root.bind("<KeyRelease>",key_up)  #キーを離した時に実行する関数を指定
cvs = tkinter.Canvas(root,width=560,height=400)  #キャンパスの部品を作る
cvs.pack()  #キャンパスを配置
img = tkinter.PhotoImage(file="sakura013.png")  #画像を変数imgに読み込む
sakura = img.subsample(8)  #画像を1/8のサイズにする
img2 = tkinter.PhotoImage(file="sakura022.png")
hana = img2.subsample(18)
game_main() #game_main関数を実行
root.mainloop() #ウインドウ表示
```

図 8：一筆書きゲームの STAGE1。左上の桜アイコンを移動させて，白い部分を塗り潰す。

図 9:：STAGE7 クリア後の画面

3.6　時系列データの解析と予測

時系列データを解析し，将来のデータを予測するツールを Python で実装した。用いたツールは，”Prophet”という Facebook の技術者が開発した API であり，オープンソースソフトウェアである。開発環境は Google Colaboratory を使用し，pandas を用いてデータの整理を行い，ipywidgets モジュールで視覚化を行った。時系列データを正確に分析するのには，そのデータの専門的知識や経験が必要であるため，時系列データを活用するのは少々難解であったが，Prophet はこの問題を解決するツールである。Prophet は機械学習で解析を行う。解析したい元のデータを学習データとして入力し，そこから予測データを出力する。また，Prophet にはグラフ描画機能も備わっており，直感的な分析も可能である。以下は Prophet を組み込んだプログラムのソースコードと実行例である。

```
import pandas as pd
from google.colab import files # ファイルのアップロード
import io

print("CSV ファイルを選択してください。")
uploaded = files.upload()# データを変数に読み込む

file_name = uploaded.keys()
file_name = ''.join(list(file_name))

# parse_dates : datetime 型で読み込む column 名
# index_col : index とする column 名
df = pd.read_csv(io.BytesIO(uploaded[file_name]),
                 parse_dates = [0],
                 index_col = [0])
df = df.astype(float)

import numpy as np
from matplotlib import pyplot as plt
import warnings
import ipywidgets as widgets
from IPython.display import display
from fbprophet import Prophet

warnings.filterwarnings("ignore") # warning を表示させないようにする

# データと学習データを選択するウィジェット
column_options = df.columns.astype(str)
column_options = column_options.tolist()

data_dropdown = widgets.Dropdown(
    options=column_options,
    description='データ:'
```

```
)
df_index = df.index.astype(str)
df_index = df_index.tolist()

trainperiod_dropdown = widgets.Dropdown(
    options = df_index,
    description='学習データ:'
)
# 三角関数の数と予測期間のウィジェット
seasonality_slider = widgets.IntSlider(
    value=10,
    min=1,
    max=20,
    step=1,
    description='三角関数の数:'
)
predictperiod_slider = widgets.IntSlider(
    value=24,
    min=12,
    max=120,
    step=1,
    description='予測期間(月):'
)
@widgets.interact_manual(data=data_dropdown)
def view_the_graph(data):
    plt.figure(figsize=(15.0, 8.0))
    plt.plot(df[data])
    plt.ylabel(data)
# 予測実行する関数

@widgets.interact_manual(data=data_dropdown,
                         train_period=trainperiod_dropdown,
                         seasonality=seasonality_slider,
```

```
                        predict_period=predictperiod_slider)
def data_reset(data,train_period,seasonality,predict_period):
    # データのセットと地域選択
    train = df[:train_period]
    as_dsname = df.index.name
    train = train.reset_index().rename(columns={as_dsname:'ds',data:'y'})
    # return train
    print("データを整形しました。")
    # Prophet で未来予測

    # 1.Prophet():モデルオブジェクトの作成・モデルの詳細の決定
    # モデルオブジェクトの作成・詳細の設定
    m = Prophet(growth="linear",
                yearly_seasonality = seasonality,
                weekly_seasonality = False,
                daily_seasonality = False,
                seasonality_mode='multiplicative').fit(train)
    print("Prophet で分析中...")

    # 2.make_future_dataframe():予測期間の指定, 推定された値を入れるデータフレームの用
意
    # 予測期間の指定
    future = m.make_future_dataframe(periods=predict_period,freq='MS')
    # 日次データの場合
    #future2 = m.make_future_dataframe(periods=365*4 + 1,freq='D')
    # 3.predict():予測
    global forecast
    forecast = m.predict(df = future)
    print("分析が終わりました！")

    fig1 = plt.figure(figsize=(15.0, 8.0))
    fig1 = m.plot(forecast)
    fig2 = plt.figure(figsize=(15.0, 8.0))
```

```
fig2 = m.plot_components(forecast)
fig3 = plt.figure(figsize=(15.0, 8.0))
fig3 = plt.scatter(df.index,df[data],color = "black",s= 10)
fig3 = plt.plot(forecast["ds"],forecast["yhat"])
fig3 = plt.xlim(future.ds.iloc[-30], future.ds.iloc[-1])
```

実行例：

埼玉県の各市町村の人口データを Prophet で解析し，その精度を調べた。図 10 はさいたま市の実際のデータのグラフである。図 11 は Prophet でさいたま市の人口データを 1973 年から 2019 年までを学習データとして解析し，その後 24 ヶ月の人口を予測したグラフである。Prophet の予測精度を調べるため，埼玉県全体の人口を Prophet で解析し，得られた予測結果と実際のデータの比較も行った。まず，1973 年から 2009 年までのデータを学習データとし，そこから 2010 年から 2019 年の約 10 年間のデータを予測した。そして，実際のデータと Prophet で得られた予測データの差の絶対値を誤差として計算し，グラフにした。比較するために，従来の基本的な人口予測モデルである，マルサスの人口予測の微分方程式で得られた予測データを計算し，同じ 10 年間を比べた。図 12 は，Prophet が得た予測の誤差とマルサスモデルで得た予測の誤差をまとめたグラフである。従来と比べて制度がよくなったと言える。

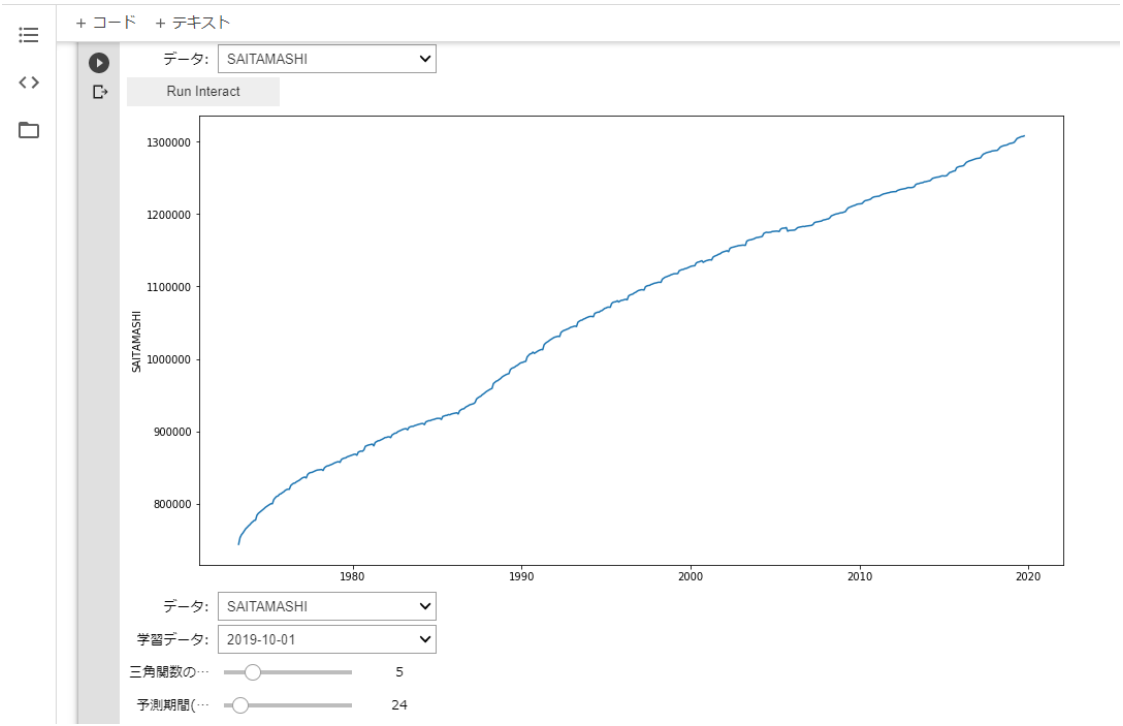

図 10：さいたま市の人口データ。横軸は西暦で縦軸は人口を表す。

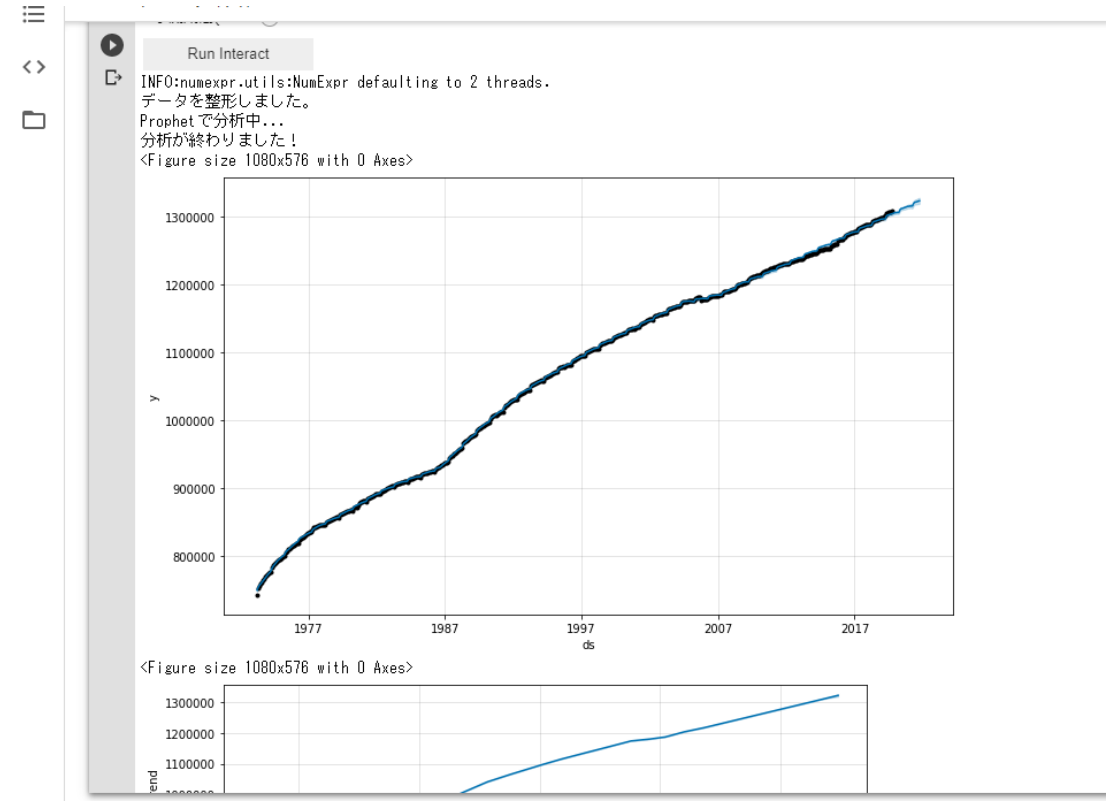

図 11：機械学習による人口の予想（青線）

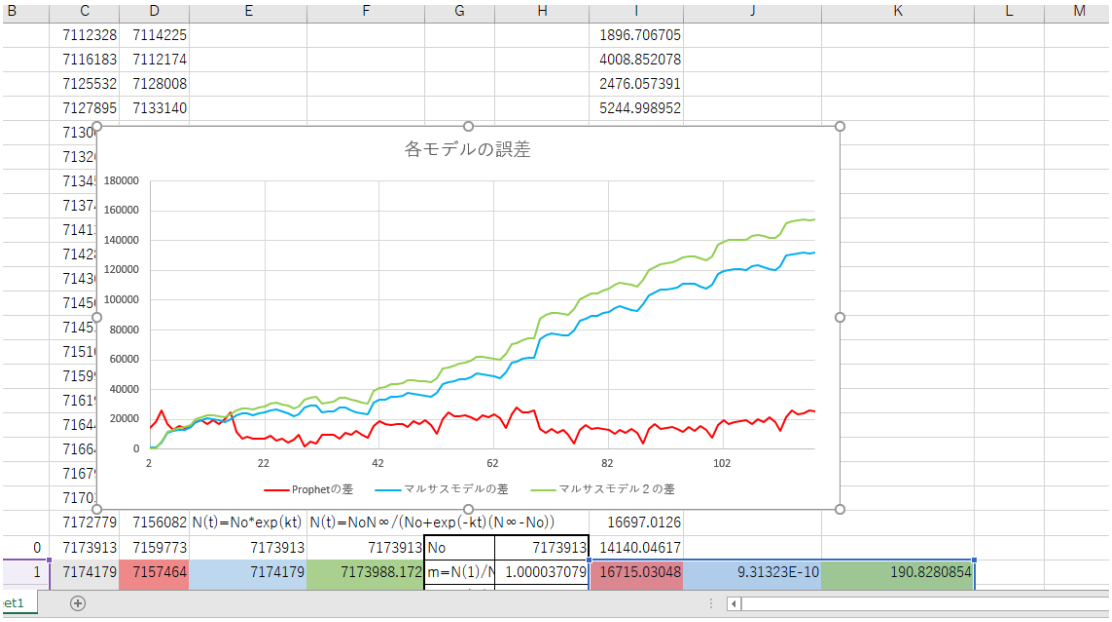

図 12：Prophet が得た予測の誤差とマルサスモデルで得た予測の誤差

4．まとめ・今後の展望

プログラミング言語 Python によるコード実装の内容を報告した。理学部数学科が主催している高校生向け応用数学体験講座での講演内容と，4 年次の数学セミナーで実施したアクティブラ

ーニングの内容を，ソースコードとともに紹介した。高校生向け応用数学体験講座で実施したアンケート結果より，Python を用いたプログラミング実習は，学生の興味を惹く内容であることが分かった。さらに，数学セミナーでのアクティブラーニングを通して，テキストデータや時系列データのような独立性のないデータの解析や，複雑な条件分岐・クラス処理・視覚化といった，非常に応用的なプログラミング教育が展開できたと思われる。今後もデータサイエンス教育の一環として機械学習を取り入れるなど，より充実した授業を行っていきたい。

謝辞

セミナーに積極的に参加し，実習に協力してくれた，岩佐はるかさん，岩本孝平君，金子恵里奈さん，高木咲希さん，二瓶哲君には，心から感謝いたします。

参考文献

[1] 池内孝啓，鈴木たかのり，石本敦夫，小坂健二郎，真嘉比愛，Python ライブラリ厳選レシピ，技術評論社，2015

[2] 大澤文孝，いちばんやさしい Python 入門教室，ソーテック社，2017

[3] 金城俊哉，世界でいちばん簡単な Python プログラミングの e 本，秀和システム，2017

[4] 鈴木たかのり，杉谷弥月，いちばんやさしい Python の教本 人気講師が教えるサーバサイド開発まで，インプレス，2017

[5] 高橋麻奈，やさしい Python，SB クリエイティブ，2018

[6] 株式会社ビープラウド著，スラスラ読める Python ふりがなプログラミング，インプレス， 2018