Pythonによるデータ解析入門
―気候区分を調べよう―

第１部：気候区分の概要

１　気候帯の判定

代表的な気候区分であるケッペンの気候区分においては､月別の気温と降水量の平年値を用いて、世界を５つの気候帯に区分する。気温と降水量の基準は、植生の状態に注目して決められている。これらのうち、熱帯、温帯、亜寒帯は樹林気候、乾燥帯と寒帯は無樹林気候である。

１．１　無樹林気候の判定

ケッペンの気候区分では、まず、無樹林気候のうち、乾燥が原因で樹木のない乾燥帯と、低温が原因で樹木のない寒帯を他の気候帯と区別する。

【１】乾燥帯（Ｂ気候）の判定

気温と降水の季節性によって決まる「乾燥限界値」と年降水量を比較することによって、乾燥帯であるか判定する。

①乾燥する時期を判定する。

ｓ型（夏季乾燥型）：

$$\text{夏の最少雨月降水量} \leqq \frac{1}{3} \times \text{冬の最多雨月降水量}$$

ｗ型（冬季乾燥型）：

$$\text{冬の最少雨月降水量} \leqq \frac{1}{10} \times \text{夏の最多雨月降水量}$$

ｆ型（年中湿潤型）：　ｓ型でもｗ型でもない。

②乾燥限界値R〔ｍｍ〕を算出する。

ｓ型（夏季乾燥型）の場合：　$R=20t$

ｗ型（冬季乾燥型）の場合：　$R=20(t+14)$

ｆ型（年中湿潤型）の場合：　$R=20(t+7)$

ただし、tは年平均気温〔℃〕である。rを年降水量〔ｍｍ〕として、$r<R$ならば、乾燥帯である。

【２】寒帯（Ｅ気候）の判定

最暖月平均気温によって判定する。最暖月平均気温が１０℃未満ならば、寒帯である。

※ケッペンによる原著や、いくつかの近年の論文では、乾燥帯の判定よりも寒帯の判定を先に行なっている。この場合、乾燥かつ低温な条件の地域では気候帯の判定が変わる可能性がある。また、一般に低温な地域では降水量のデータが得られないことが多いが、寒帯の判定を先に行なえば、降水量のデータがなくても寒帯の気候区分だけは判定できる。

１．２　樹林気候の判定

次に、樹林気候を気温によって、熱帯、温帯、亜寒帯の３つの気候帯に区分する。

【３】熱帯（Ａ気候）、温帯（Ｃ気候）、亜寒帯（Ｄ気候）の判定

〇熱帯（Ａ気候）

最寒月平均気温が１８℃以上ならば、熱帯である。

〇温帯（Ｃ気候）

最寒月平均気温が－３℃以上１８℃未満ならば、温帯である。

〇亜寒帯（Ｄ気候）

最寒月平均気温が－３℃未満ならば、亜寒帯である。

２　気候区の判定

各気候帯における気候区は、気候帯ごとに決められた方法によって判定する。

２．１　熱帯

熱帯においては、月降水量から乾季の有無を判定し、熱帯雨林気候、熱帯モンスーン気候、サバナ気候に区分する。

（１）熱帯雨林気候（Ａｆ）

最少雨月降水量が６０ｍｍ以上。年を通して湿潤。

（２）熱帯モンスーン気候（Ａｍ）

最少雨月降水量が６０ｍｍ未満で、最少雨月降水量≧１００－０．０４ｒ〔ｍｍ〕。弱い乾季がある。「弱い乾季のある熱帯雨林気候」とよばれることもある。

⇒　中学校社会科では、（１）と（２）をまとめて「**熱帯雨林気候**」としている。

（３）サバナ気候（Ａｗ）

最少雨月降水量が６０ｍｍ未満で、最少雨月降水量＜１００－０．０４ｒ〔ｍｍ〕。明瞭な乾季がある。

２．２　乾燥帯

乾燥帯においては、年降水量と乾燥限界値を比較することによって、ステップ気候と砂漠気候に区分する。

（１）ステップ気候（ＢＳ）

$$\text{年降水量} \quad \geqq \quad \frac{1}{2} \times \text{乾燥限界値}$$

（２）砂漠気候（ＢＷ）

年降水量　＜　$\frac{1}{2}$×乾燥限界値

２．３　温帯

温帯においては、降水の季節性によって、地中海性気候、温暖冬季少雨気候、温暖湿潤気候・西岸海洋性気候に区分する。温暖湿潤気候と西岸海洋性気候はどちらもｆ型（年中湿潤型）であり、両者は降水ではなく気温によって区別される。

（１）地中海性気候（Ｃｓ）

乾燥する時期がｓ型（夏季乾燥型）で、最少雨月降水量が３０ｍｍ未満。

☞　最少雨月降水量の基準がないと、本州の日本海側のように冬季に降水の多い地域が「地中海性気候」に判定される場合がある。

（２）温暖冬季少雨気候（Ｃｗ）

乾燥する時期がｗ型（冬季乾燥型）。

（３）温暖湿潤気候（Ｃｆａ）

乾燥する時期がｆ型（年中湿潤型）で、最暖月平均気温が２２℃以上。

⇒　中学校社会科では、（２）と（３）をまとめて「**温暖湿潤気候**」としている。

（４）西岸海洋性気候（Ｃｆｂ）

乾燥する時期がｆ型（年中湿潤型）で、最暖月平均気温が２２℃未満。

２．４　亜寒帯

亜寒帯においても、温帯と同様に、降水の季節性に基づいて、亜熱帯湿潤気候と亜熱帯冬季少雨気候に分類する。

（１）亜寒帯湿潤気候（Ｄｆ）

乾燥する時期がｆ型（年中湿潤型）。

（２）亜寒帯冬季少雨気候（Ｄｗ）

乾燥する時期がｗ型（冬季乾燥型）。

⇒　中学校社会科では、（１）と（２）をまとめて「**冷帯気候**」としている。

２．５　寒帯

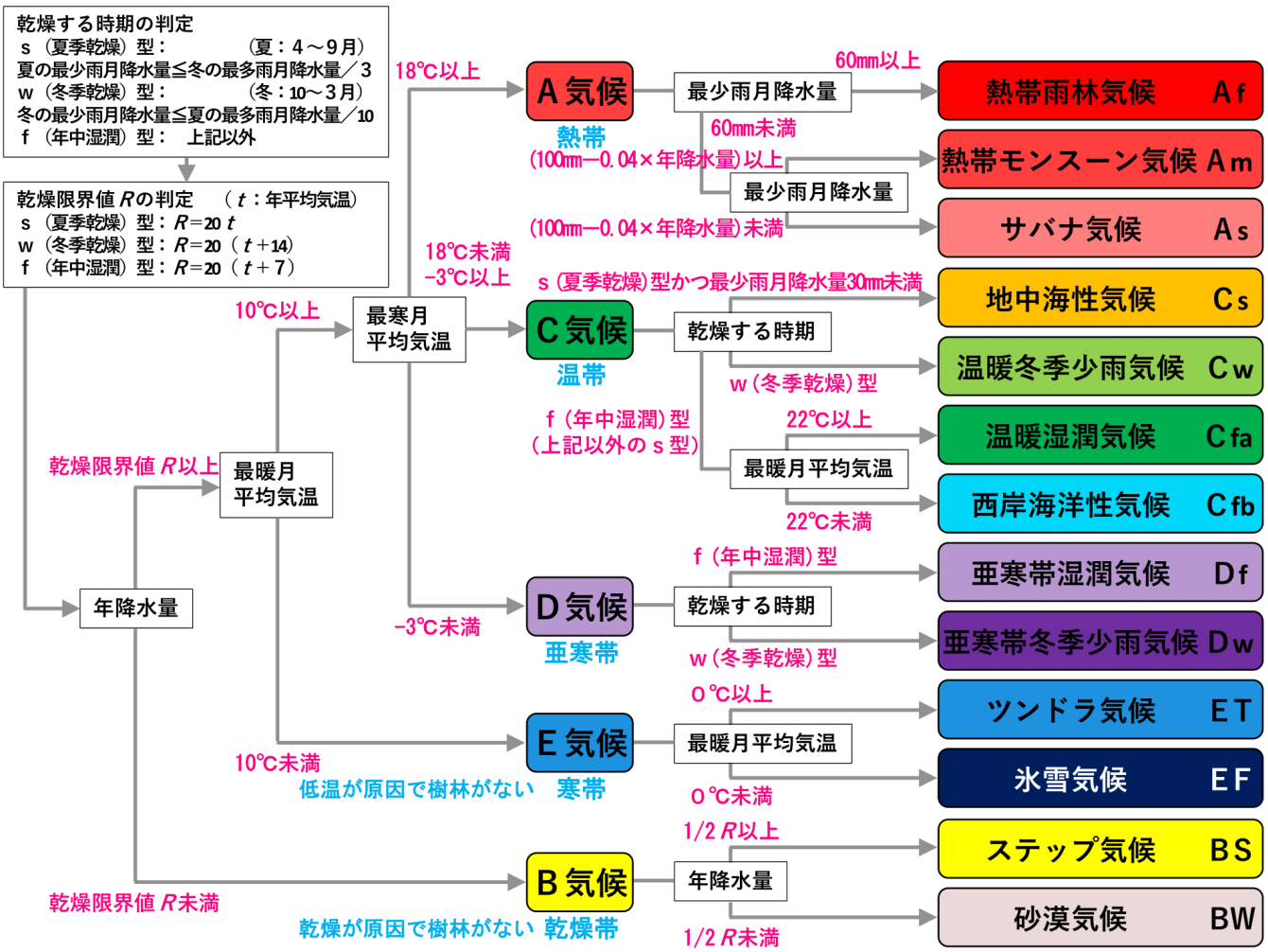
寒帯は、最暖月平均気温によって、ツンドラ気候と氷雪気候に区分される。

（１）ツンドラ気候（ＥＴ）

最暖月平均気温０℃以上。

（２）氷雪気候（ＥＦ）

最暖月平均気温０℃未満。



帝国書院「新詳地理探究」（２０２３年度用）

第２部：気候データの解析

プログラミング作業の流れ

まずターミナルを起動する。　（⇒簡単操作マニュアル１）

☞ターミナル（端末）がすべての作業の起点になります。

１．Emacs でプログラムを書く。　（⇒１－②、２－②）

２．実行する。　（⇒１－③）

３．結果を確かめる。　（⇒１－④⑤）

簡単操作マニュアル（Ｗｉｎｄｏｗｓ用）

１．ターミナルの使い方

ターミナルを起動する	デスクトップの「 msys.bat へのショートカット 」を ダブルクリック
①ファイルの一覧を表示する	$ ls ↵
ファイルをコピーする	$ cp *コピー元ファイル名* *コピー先ファイル名* ↵
ファイル名を変更する	$ mv *変更前のファイル名* *変更後のファイル名* ↵
ファイルを消去する	$ rm *ファイル名* ↵
②Ｅｍａｃｓを起動する	$ emacs *ファイル名* & ↵ （または emacs & ↵）
③実行する	$ python *ファイル名* ↵
④ファイルの中身を見る	$ less *ファイル名* ↵　矢印キーで移動、Ｑを押して終了
⑤ｇｎｕｐｌｏｔを起動する	$ gnuplot ↵
ターミナルを終了する	$ exit ↵

２．Ｅｍａｃｓの使い方

①ファイルを開く	コントロールキー＋Ｘ　コントロールキー＋Ｆ →ファイル名を入力
②ファイルを保存する	コントロールキー＋Ｘ　コントロールキー＋Ｓ
※操作の取り消し	コントロールキー＋Ｇ
Ｅｍａｃｓを終了する	コントロールキー＋Ｘ　コントロールキー＋Ｃ

３．ｇｎｕｐｌｏｔの使い方

①グラフをかく	> plot ”*ファイル名*” ↵ 複数の場合：　plot ”*ファイル名*”，　”*ファイル名*”，　… ↵
②折れ線グラフにする	> set　style　data　lines ↵
範囲を指定する	> set　xrange　[*0*:*50*] ↵ > set　yrange　[*0*:*30*] ↵
③再描画する	> replot ↵
※画像ファイルに保存する	画面上で作図したあとで： > set　term　png ↵ > set　output　”*ファイル名.png*” ↵ > replot ↵
ｇｎｕｐｌｏｔを終了する	> quit ↵

簡単操作マニュアル（Ｌｉｎｕｘ用）

１．ターミナルの使い方

ターミナルを起動する	ランチャー（画面左側）の「端末」をクリック
①ファイルの一覧を表示する	> ls
ファイルをコピーする	> cp *コピー元ファイル名* *コピー先ファイル名*
ファイル名を変更する	> mv *変更前のファイル名* *変更後のファイル名*
ファイルを消去する	> rm *ファイル名*
②Ｅｍａｃｓを起動する	> emacs & または emacs *ファイル名* &
③実行する	> python *ファイル名*
④ファイルの中身を見る	> less *ファイル名*　　矢印キーで移動、Ｑを押して終了
⑤ｇｎｕｐｌｏｔを起動する	> gnuplot
ターミナルを終了する	> exit

２．Ｅｍａｃｓの使い方

①ファイルを開く	コントロールキー＋Ｘ　コントロールキー＋Ｆ →ファイル名を入力
②ファイルを保存する	コントロールキー＋Ｘ　コントロールキー＋Ｓ
※操作の取り消し	コントロールキー＋Ｘ　コントロールキー＋Ｇ
Ｅｍａｃｓを終了する	コントロールキー＋Ｘ　コントロールキー＋Ｃ

３．ｇｎｕｐｌｏｔの使い方

①グラフをかく	> plot ”*ファイル名*” 複数の場合：　plot ”*ファイル名*”，　”*ファイル名*”，　…
②折れ線グラフにする	> set style data lines
範囲を指定する	> set xrange [*0*:*50*] > set yrange [*0*:*30*]
③再描画する	> replot
※画像ファイルに保存する	画面上で作図したあとで： > set term png > set output ”*ファイル名.png*” > replot
ｇｎｕｐｌｏｔを終了する	> quit

①メッセージを書き出そう

作業の手順：以下のサンプルプログラムを Emacs で作成しよう。

☞プログラムのファイル名は自由だが、「.py」で終わる必要がある。
　例：prog01.py なら　　⇒　　ターミナル上で　$ emacs　prog01.py　& ↵
☞書き終わったら忘れずに保存する（コントロールキー＋Ｘ　コントロールキー＋Ｓ）。
　　コントロールキーを押しながらＸを押す

```
print　("Hello, world!")　　#　print 関数でメッセージを書き出す
```

※　大文字と小文字は区別する。
※　#以降はコメントであり処理に影響しないので、書かなくてよい。

実行例：ターミナル上で実行してみよう。

```
$　ls ↵　　ファイルを確認します。

prog01.py

$ python　prog01.py ↵　　実行します。

Hello,　world!　　メッセージが出力されます。
```

☞正常に実行できることを確認したら、Emacs を終了してよい
　（コントロールキー＋Ｘ　コントロールキー＋Ｃ）。

②入力に応答しよう

作業の手順：以下のサンプルプログラムを Emacs で作成しよう。

☞プログラムのファイル名は自由だが、「.py」で終わる必要がある。
　例：prog02.py なら　　⇒　　ターミナル上で　$ emacs　prog01.py　& ↵
☞書き終わったら忘れずに保存する（コントロールキー＋Ｘ　コントロールキー＋Ｓ
　　コントロールキーを押しながらＸを押す

```
print  ("Your name?")                          #  print 関数でメッセージを書き出す

name  =  input  ()                             #  input 関数でキーボードから
                                               #     文字列を読みこむ
                                               #  読みこんだ文字列を
                                               #     変数 name に代入する

print  ("%s%s%s"  %  ("Hello, ", name, "!"))
                                               #  print 関数でメッセージを書き出す
                                               #     前半の"%s%s%s"は書式指定、
                                               #     %s は文字列という意味
```

実行例：ターミナル上で実行してみよう。

```
$  ls ↵                          ファイルを確認します。

prog01.py  prog02.py

$ python  prog02.py ↵            実行します。

Your  name?

Naoki ↵                          名前を入力します。

Hello,  Naoki!                   メッセージが出力されます。
```

☞正常に実行できることを確認したら、Emacs を終了してよい
　（コントロールキー＋Ｘ　コントロールキー＋Ｃ）。

③データを書き出そう

作業の手順：以下のサンプルプログラムを Emacs で作成しよう。

☞プログラムのファイル名は自由だが、「.py」で終わる必要がある。
　例：prog03.py なら　　⇒　　ターミナル上で　$ emacs　prog03.py　& ↵

☞書き終わったら忘れずに保存する（コントロールキー＋Ｘ　コントロールキー＋Ｓ）。
　　コントロールキーを押しながらＸを押す

```
#  データ T[0]～T[11]を準備する

T  =  [0.0]  *  12                              #  リストを宣言する

T  =  [   5.4,    6.1,    9.4,   14.3,   18.8,   21.9,   ¥
         25.7,   26.9,   23.3,   18.0,   12.5,   7.7 ]
                                                #  データを代入する

#  データ T[0]～T[11]を書き出す

print  ("%6.1f"  %  T[ 0])
print  ("%6.1f"  %  T[ 1])                      #  print 関数でデータを書き出す
```

※ここで用いているデータは、東京での月平均気温です。

実行例：ターミナル上で実行してみよう。

```
$  ls ↵                          ファイルを確認します。

prog01.py  prog02.py  prog03.py

$ python  prog03.py ↵            実行します。

   5.4                           データが出力されます。
   6.1
   9.4
  14.3
  18.8
  21.9
  25.7
  26.9
  23.3
  18.0
  12.5
   7.7
```

☞正常に実行できることを確認したら、Emacs を終了してよい
（コントロールキー＋Ｘ　コントロールキー＋Ｃ）。

④ループを使ってデータを書き出そう

作業の手順：③で作成したプログラムをコピーして、Emacs で完成させよう。

☞コピーするときには cp コマンドを使う。
　例：$ cp　prog03.py　prog04.py ↵　　（⇒簡単操作マニュアル１－①）
☞コピーしたファイルを改めて emacs で開く。
☞書き終わったら忘れずに保存する（コントロールキー＋Ｘ　コントロールキー＋Ｓ）。
コントロールキーを押しながらＸを押す

```
#  データ T[0]～T[11]を準備する

T  =  [0.0]  *  12                              #  リストを宣言する

T  =  [   5.4,    6.1,    9.4,   14.3,   18.8,   21.9,   ¥
         25.7,   26.9,   23.3,   18.0,   12.5,   7.7 ]
                                                #  データを代入する

#  データ T[0]～T[11]を書き出す

m  =  1
while  m  <=  12:                               #  変数ｍに１を代入する
                                                #  変数ｍが 12 以下であれば、
                                                #     以下の処理を反復する
    print  ("%6.1f"  %  T[m-1])                 #  print 関数でデータを書き出す

    m  =  m  +  1                               #  変数ｍの値をひとつ進める
```

反復する範囲内では
行頭を４文字空ける

実行例：ターミナル上で実行してみよう。

```
$  ls ↵                                  ファイルを確認します。

prog01.py  prog02.py  prog03.py  prog04.py

$ python  prog04.py ↵                    実行します。

   5.4                                   データが出力されます。
   6.1
   9.4
  14.3
  18.8
  21.9
  25.7
  26.9
  23.3
  18.0
  12.5
   7.7
```

☞正常に実行できることを確認したら、Emacs を終了してよい
（コントロールキー＋Ｘ　コントロールキー＋Ｃ）。

⑤データを読みこもう

データの準備：データファイルをコピーしよう。

☞実行する前に、データディレクトリ（データフォルダ）から、data47662.txt という名前のファイルを現在の作業ディレクトリにコピーしておく。
☞コピーしたら、データファイルの中身を確認しよう。

```
$  ls ↵                                  ファイルを確認します。

data47662.txt  prog01.py  prog02.py  prog03.py  prog04.py  prog05.py

$ less  data47662.txt ↵                  データファイルの中身を見ます。 →見終わったらQで終了。

    1   5.4   59.7
    2   6.1   56.5
    3   9.4  116.0
    4  14.3  133.7
    5  18.8  139.7
    6  21.9  167.8
    7  25.7  156.2
    8  26.9  154.7
    9  23.3  224.9
   10  18.0  234.8
   11  12.5   96.3
   12   7.7   57.9
```

※data47662.txt には、東京での月平均気温と月降水量が保存されています。

※この研修で使用している世界各地の気温、降水量のデータは、気象庁のウェブサイト（https://www.data.jma.go.jp/cpd/monitor/normal/index.html）から入手しました。統計期間は１９９１～２０２０年です。

作業の手順：以下のサンプルプログラムを Emacs で作成しよう。

☞プログラムのファイル名は自由だが、「.py」で終わる必要がある。
　例：prog05.py なら　　⇒　　ターミナル上で　$ emacs　prog05.py　& ↵

☞書き終わったら忘れずに保存する（コントロールキー＋Ｘ　コントロールキー＋Ｓ）。
コントロールキーを押しながらＸを押す

```
T  =  [0.0]  *  12
R  =  [0.0]  *  12                          #  リストを宣言する

#  データを読みこむ

print  ("Input file?")
ifile  =  input  ()                         #  ファイル名を入力する
#  ファイル名を入力する手間を省きたい場合は、
#  上の 2 行の行頭に#をつけてコメントアウトし、
#  代わりに次の行の行頭の#を消してもよい。
#ifile  =  "data47662.txt"
f  =  open  (ifile,  "r")                   #  ファイルを開く

m  =  1
while  m  <=  12:                           #  同じ処理を 12 回繰り返す

    a,  b,  c  =  f.readline().split()      #  データをファイルから読みこむ

    T[m-1]  =  float  (b)                   #  気温と降水量の値を
    R[m-1]  =  float  (c)                   #    リストに代入する

    m  =  m  +  1                           #  カウントをひとつ進める

f.close  ()                                 #  ファイルを閉じる

print  ("Month Temp.  Prec.")

m  =  1
while  m  <=  12:                           #  同じ処理を 12 回繰り返す

    print  (" %4d %5.1f %6.1f"  %  (m,  T[m-1],  R[m-1]))
                                            #  データを書き出す

    m  =  m  +  1                           #  カウントをひとつ進める
```

実行例：ターミナル上で実行してみよう。

☞実行する前に、データディレクトリ（データフォルダ）から、data47662.txt という名前のファイルを現在の作業ディレクトリにコピーしておく。

```
$  ls ↵                                  ファイルを確認します。

data47662.txt  prog01.py  prog02.py  prog03.py  prog04.py  prog05.py

$ python  prog05.py ↵                    実行します。

Input  file?

data47662.txt ↵                          ファイル名を入力します。

Month Temp.   Prec.                      結果が出力されます。
    1   5.4   59.7
    2   6.1   56.5
    3   9.4  116.0
    4  14.3  133.7
    5  18.8  139.7
    6  21.9  167.8
    7  25.7  156.2
    8  26.9  154.7
    9  23.3  224.9
   10  18.0  234.8
   11  12.5   96.3
   12   7.7   57.9
```

☞正常に実行できることを確認したら、Emacs を終了してよい
（コントロールキー＋Ｘ　コントロールキー＋Ｃ）。

⑥グラフを描こう

作業の手順：⑤で作成したプログラムをコピーして、Emacs で完成させよう。

☞コピーするときには cp コマンドを使う。

例：$ cp　prog05.py　prog06.py ↵　　（⇒簡単操作マニュアル１－①）

☞コピーしたファイルを改めて emacs で開く。

☞書き終わったら忘れずに保存する（コントロールキー＋Ｘ　コントロールキー＋Ｓ）。

コントロールキーを押しながらＸを押す

```
import  matplotlib.pyplot  as  plt          #  モジュールをインポートする


T  =  [0.0]  *  12
R  =  [0.0]  *  12                          #  リストを宣言する
#  データを読みこむ
print  ("Input file?")
ifile  =  input  ()                         #  ファイル名を入力する
f  =  open  (ifile,  "r")                   #  ファイルを開く
m  =  1
while  m  <=  12:                           #  同じ処理を 12 回繰り返す
    a,  b,  c  =  f.readline().split()      #  データをファイルから読みこむ
    T[m-1]  =  float  (b)                   #  気温と降水量の値を
    R[m-1]  =  float  (c)                   #    リストに代入する
    m  =  m  +  1                           #  カウントをひとつ進める
f.close  ()                                 #  ファイルを閉じる
#  入力データを書き出す
print  ("Month Temp.  Prec.")
m  =  1
while  m  <=  12:                           #  同じ処理を 12 回繰り返す
    print  (" %4d %5.1f %6.1f"  %  (m,  T[m-1],  R[m-1]))
                                            #  データを書き出す
    m  =  m  +  1                           #  カウントをひとつ進める


#  グラフを描く


print  ("Title?")
title  =  input  ()                         #  タイトルを入力する
fig  =  plt.figure    ()                    #  描画領域全体のオブジェクトを
                                            #    作成する
ax1  =  fig.subplots  ()                    #  気温を作図するための
                                            #    オブジェクトを作成する
ax2  =  ax1.twinx     ()                    #  降水量を作図するための
                                            #    オブジェクトを作成する
month  =  list  (range  (1,  13))           #  1～12 のリストを定義する
plt.title       (title)                     #  タイトルを付ける
plt.xlim        (0.5,  12.5)                #  x 軸の範囲を指定する
plt.xticks      (month)                     #  x 軸の目盛りを指定する
ax1.set_ylim    (-40.0,  40.0)              #  y 軸(気温)の範囲を指定する
ax1.set_ylabel  ("Temperature [C]")         #  y 軸(気温)にラベルを付ける
```

```
ax1.grid        (axis="y")                      #  y 軸(気温)に目盛り線を付ける
ax1.plot        (month,  T,  color="#FF0000")
                                                #  気温の折れ線を描く
ax2.set_ylim    (0.0,  400.0)                   #  y 軸(降水量)の範囲を指定する
ax2.set_ylabel  ("Precipitation [mm]")          #  y 軸(降水量)にラベルを付ける
ax2.bar         (month,  R,  align="center",  color="#0000FF")
                                                #  降水量の棒を描く
plt.show        ()
```

☞色は、R（赤)、G（緑)、B（青）の明るさを２桁の 16 進数（00～FF）で指定します。

例）　#FF0000：R＝FF（＝255)、G＝00（＝ 0)、B＝00（＝ 0）　⇒　赤

#00FF00：R＝00（＝ 0)、G＝FF（＝255)、B＝00（＝ 0）　⇒　緑

#0000FF：R＝00（＝ 0)、G＝00（＝ 0)、B＝FF（＝255）　⇒　青

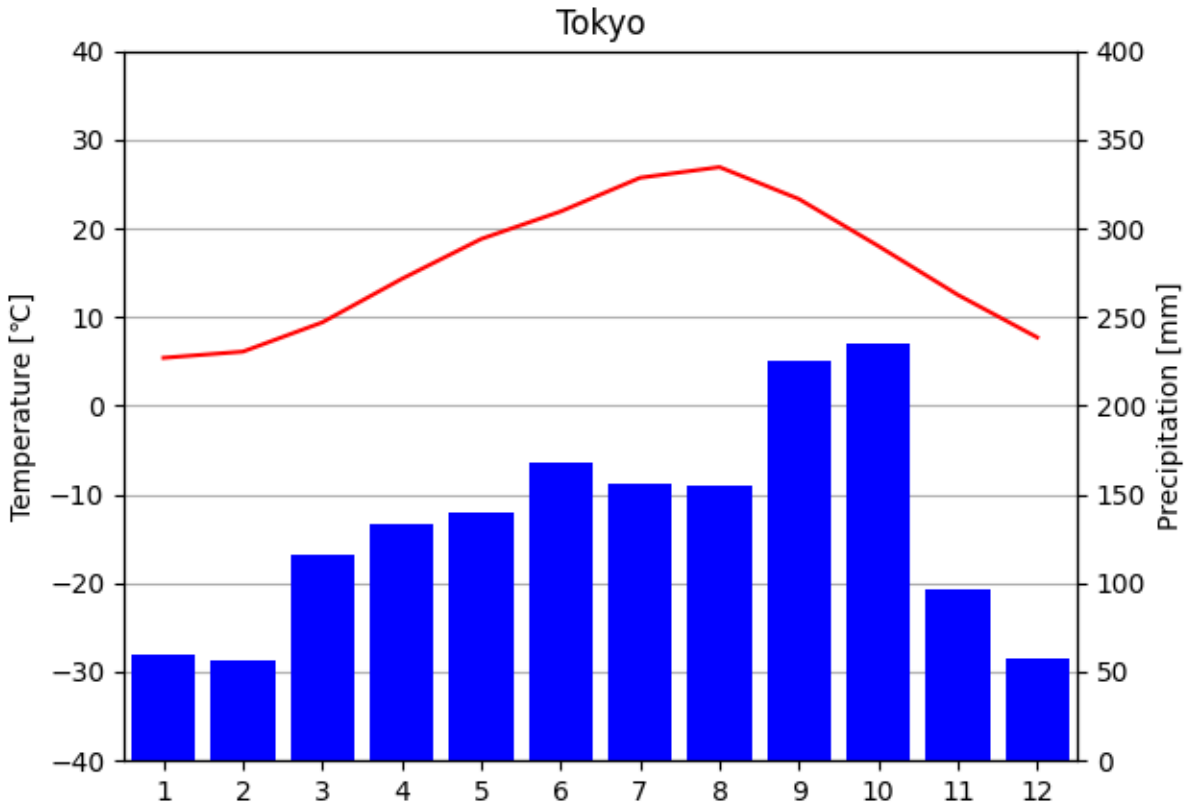
#000000：R＝00（＝ 0)、G＝00（＝ 0)、B＝00（＝ 0）　⇒　黒

実行例：ターミナル上で実行してみよう

省略

☞⑤と同じように prog06.py を実行する。

作図例：



【追加】

⑦雨温図（ハイサーグラフ）を描こう

作業の手順：⑥で作成したプログラムをコピーして、Emacs で完成させよう。

☞コピーするときには cp コマンドを使う。

例：$ cp　prog06.py　prog07.py ↵　　（⇒簡単操作マニュアル１－①）

☞コピーしたファイルを改めて emacs で開く。

☞書き終わったら忘れずに保存する（コントロールキー＋Ｘ　コントロールキー＋Ｓ）。

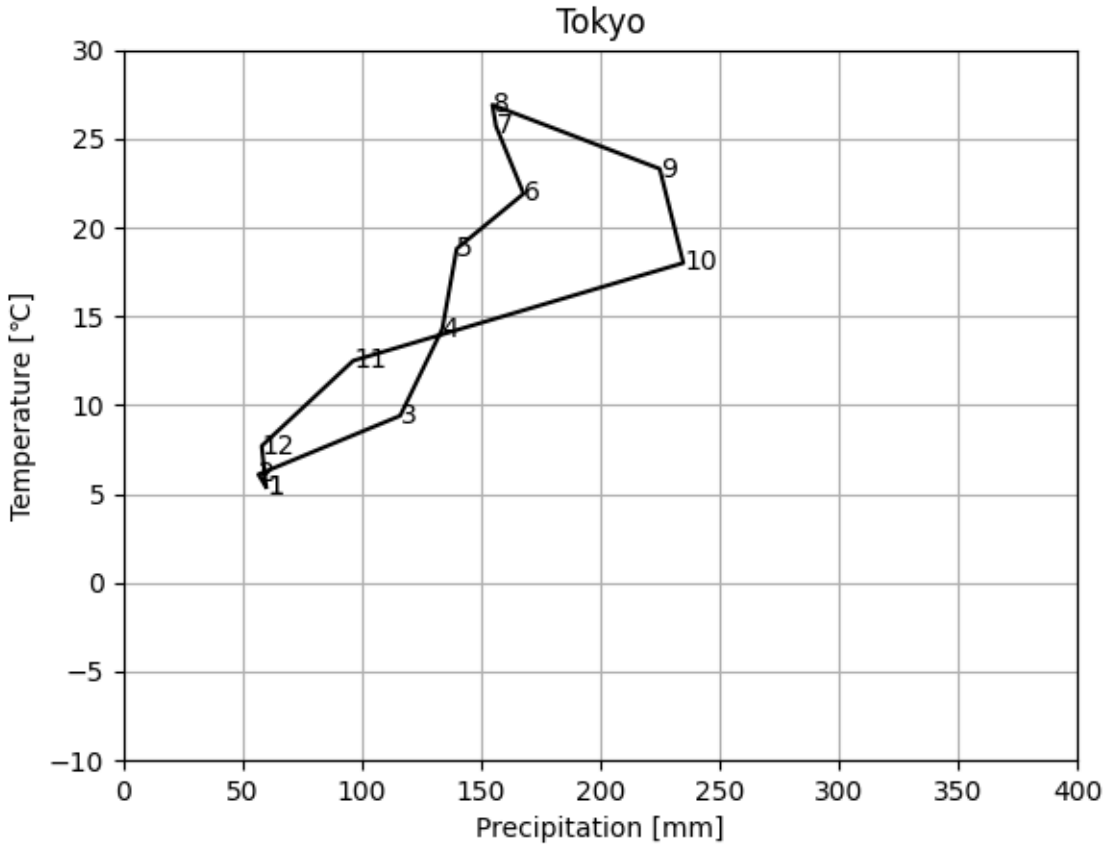
コントロールキーを押しながらＸを押す

```
import  matplotlib.pyplot  as  plt          #  モジュールをインポートする
T  =  [0.0]  *  12
R  =  [0.0]  *  12                          #  リストを宣言する
#  データを読みこむ
print  ("Input file?")
ifile  =  input  ()                         #  ファイル名を入力する
f  =  open  (ifile,  "r")                   #  ファイルを開く
m  =  1
while  m  <=  12:                           #  同じ処理を 12 回繰り返す
    a,  b,  c  =  f.readline().split()      #  データをファイルから読みこむ
    T[m-1]  =  float  (b)                   #  気温と降水量の値を
    R[m-1]  =  float  (c)                   #    リストに代入する
    m  =  m  +  1                           #  カウントをひとつ進める
f.close  ()                                 #  ファイルを閉じる
#  入力データを書き出す
print  ("Month Temp.  Prec.")
m  =  1
while  m  <=  12:                           #  同じ処理を 12 回繰り返す
    print  (" %4d %5.1f %6.1f"  %  (m,  T[m-1],  R[m-1]))
                                            #  データを書き出す
    m  =  m  +  1                           #  カウントをひとつ進める

#  グラフを描く

print  ("Title?")
title  =  input  ()                         #  タイトルを入力する
month  =  list  (range  (1,  13))           #  1～12 のリストを定義する
month.append    (month[0])
T.append        (T[0])                      #  12 月と 1 月をつなぐため、
R.append        (R[0])                      #    12 月の後に 1 月を追加する
plt.title       (title)                     #  タイトルを付ける
plt.xlim        (  0.0,  400.0)             #  x 軸の範囲を指定する
plt.ylim        (-10.0,   30.0)             #  y 軸の範囲を指定する
plt.xlabel      ("Precipitation [mm]")      #  x 軸にラベルを付ける
plt.ylabel      ("Temperature [C]")         #  y 軸にラベルを付ける
plt.grid        ()                          #  目盛り線を付ける
plt.plot        (R,  T,  color="#000000")   #  折れ線を描く
for  i,  label  in  enumerate  (month):
    plt.text  (R[i],  T[i],  label,  ha="left",  va="center")
                                            #  データにラベルを付ける
plt.show        ()
```

実行例：ターミナル上で実行してみよう

省略
☞⑤と同じように prog07.py を実行する。

作図例：



※高等学校地理では、⑥で作成した図を「雨温図」、⑦で作成した図を「ハイサーグラフ」と呼んでいるが、専門的な観点からは必ずしも同意が得られているわけではなく、本テキストでは、⑥で作成した図には特には名称を与えず、⑦を「雨温図」または「ハイサーグラフ」と呼んでいる。

⑧年平均気温を計算しよう

作業の手順：⑤で作成したプログラムをコピーして、Emacs で完成させよう。

☞コピーするときには cp コマンドを使う。

例：$ cp　prog05.py　prog08.py ↵　　（⇒簡単操作マニュアル１－①）

☞コピーしたファイルを改めて emacs で開く。

☞書き終わったら忘れずに保存する（コントロールキー＋Ｘ　コントロールキー＋Ｓ）。

コントロールキーを押しながらＸを押す

```
T  =  [0.0]  *  12
R  =  [0.0]  *  12                          #  リストを宣言する
#  データを読みこむ
print  ("Input file?")
ifile  =  input  ()                         #  ファイル名を入力する
f  =  open  (ifile,  "r")                   #  ファイルを開く
m  =  1
while  m  <=  12:                           #  同じ処理を 12 回繰り返す
    a,  b,  c  =  f.readline().split()      #  データをファイルから読みこむ
    T[m-1]  =  float  (b)                   #  気温と降水量の値を
    R[m-1]  =  float  (c)                   #    リストに代入する
    m  =  m  +  1                           #  カウントをひとつ進める
f.close  ()                                 #  ファイルを閉じる
#  入力データを書き出す
print  ("Month Temp.  Prec.")
m  =  1
while  m  <=  12:                           #  同じ処理を 12 回繰り返す
    print  (" %4d %5.1f %6.1f"  %  (m,  T[m-1],  R[m-1]))
                                            #  データを書き出す
    m  =  m  +  1                           #  カウントをひとつ進める

#  年平均気温を計算する                      ここで平均値を計算します。

s  =  0.0                                   #  合計値にゼロを代入する

m  =  1
while  m  <=  12:                           #  同じ処理を 12 回繰り返す

    s  =  s  +  T[m-1]                      #  月平均気温の値を加える

    m  =  m  +  1                           #  カウントをひとつ進める

Tave  =  s  /  12.0                         #  平均値を計算する

#  計算結果を書き出す

print  ("Tave  =  %5.1f"      %  Tave)      #  年平均気温
```

実行例：ターミナル上で実行してみよう。

```
$ python  prog08.py ↵
Input  file?
data47662.txt ↵
Month Temp.   Prec.
    1   5.4   59.7
    2   6.1   56.5
    3   9.4  116.0
    4  14.3  133.7
    5  18.8  139.7
    6  21.9  167.8
    7  25.7  156.2
    8  26.9  154.7
    9  23.3  224.9
   10  18.0  234.8
   11  12.5   96.3
   12   7.7   57.9
Tave  =   15.8
```

実行します。

ファイル名を入力します。

結果が出力されます。

⑨月平均気温の最大値と最小値を求めよう

作業の手順：⑧で作成したプログラムをコピーして、Emacs で完成させよう。

```
T  =  [0.0]  *  12
R  =  [0.0]  *  12                          #  リストを宣言する
#  データを読みこむ
<中略>
#  入力データを書き出す
<中略>
#  年平均気温を計算する
s  =  0.0                                   #  合計値にゼロを代入する
m  =  1
while  m  <=  12:                           #  同じ処理を 12 回繰り返す
    s  =  s  +  T[m-1]                      #  月平均気温の値を加える
    m  =  m  +  1                           #  カウントをひとつ進める
Tave  =  s  /  12.0                         #  平均値を計算する

#  月平均気温の最大値と最小値を求める

Tmax  =  T[1-1]                             #  変数 Tmax と Tmin に
Tmin  =  T[1-1]                             #     1 月の月平均気温の値を代入する

m  =  2
while  m  <=  12:                           #  2～12 月の月平均気温を調べる
    if  T[m-1]  >  Tmax:                    #  月平均気温が Tmax より大きい場合は
        Tmax  =  T[m-1]                     #     Tmax の値を更新する

    if  T[m-1]  <  Tmin:                    #  月平均気温が Tmin より小さい場合は
        Tmin  =  T[m-1]                     #     Tmin の値を更新する

    m  =  m  +  1                           #  カウントをひとつ進める

#  計算結果を書き出す

print  ("Tave  =  %5.1f"      %  Tave)      #  年平均気温
print  ("Tmax  =  %5.1f"      %  Tmax)      #  最暖月平均気温
print  ("Tmin  =  %5.1f"      %  Tmin)      #  最寒月平均気温
```

条件分岐の範囲内では行頭を４文字空ける

平均値に加えて最大値と最小値を書き出します。

実行例：ターミナル上で実行してみよう。

```
$ python  prog09.py ↵

Input  file?

data47662.txt ↵

Month Temp.   Prec.
    1   5.4   59.7
    2   6.1   56.5
    3   9.4  116.0
    4  14.3  133.7
    5  18.8  139.7
    6  21.9  167.8
    7  25.7  156.2
    8  26.9  154.7
    9  23.3  224.9
   10  18.0  234.8
   11  12.5   96.3
   12   7.7   57.9
Tave  =   15.8
Tmax  =   26.9
Tmin  =    5.4
```

実行します。

ファイル名を入力します。

結果が出力されます。

⑩気候帯を判定しよう

最暖月平均気温 Tmax＝１０℃以上：
　　最寒月平均気温 Tmin＝１８℃以上　　⇒　熱帯（Ａ気候）
　　最寒月平均気温 Tmin＝－３℃以上１８℃未満　⇒　温帯（Ｃ気候）
　　最寒月平均気温 Tmin＝－３℃未満　　⇒　亜寒帯（Ｄ気候）
最暖月平均気温 Tmax＝１０℃未満　　⇒　寒帯（Ｅ気候）

作業の手順：⑨で作成したプログラムをコピーして、Emacs で完成させよう。

```
T  =  [0.0]  *  12
R  =  [0.0]  *  12                          #  リストを宣言する
#  データを読みこむ
<中略>
#  入力データを書き出す
<中略>
#  年平均気温を計算する
<中略>
#  月平均気温の最大値と最小値を求める
<中略>
#  計算結果を書き出す
print  ("Tave  =  %5.1f"      %  Tave)      #  年平均気温
print  ("Tmax  =  %5.1f"      %  Tmax)      #  最暖月平均気温
print  ("Tmin  =  %5.1f"      %  Tmin)      #  最寒月平均気温

#  気候区分を判定する

if  Tmax  >=  10.0:      #  最暖月平均気温 Tmax が 10℃以上の場合

    if  Tmin  >=  18.0:    #  最寒月平均気温 Tmin が 18℃以上の場合
        Class  =  "A"

    elif  Tmin  >=  -3.0:  #  最寒月平均気温 Tmin が-3℃以上 18℃未満の場合
        Class  =  "C"

    else:                  #  最寒月平均気温 Tmin が-3℃未満の場合
        Class  =  "D"

else:                      #  最暖月平均気温 Tmax が 10℃未満の場合
    Class  =  "E"

print  ("Type  =  %s"  %  Class)             #  結果を書き出す
```

実行例：ターミナル上で実行してみよう。

```
$ python  prog10.py ↵

Input  file?

data47662.txt ↵

Month Temp.   Prec.
    1   5.4   59.7
    2   6.1   56.5
    3   9.4  116.0
    4  14.3  133.7
    5  18.8  139.7
    6  21.9  167.8
    7  25.7  156.2
    8  26.9  154.7
    9  23.3  224.9
   10  18.0  234.8
   11  12.5   96.3
   12   7.7   57.9
Tave  =   15.8
Tmax  =   26.9
Tmin  =    5.4
Type  =  C
```

実行します。

ファイル名を入力します。

結果が出力されます。

【追加】

⑪降水量を調べよう

作業の手順：⑩で作成したプログラムをコピーして、Emacs で完成させよう。

```
T  =  [0.0]  *  12
R  =  [0.0]  *  12                          #  リストを宣言する
#  データを読みこむ
<中略>
#  入力データを書き出す
<中略>
#  年平均気温を計算する
<中略>
#  月平均気温の最大値と最小値を求める
<中略>

#  年降水量を計算する

Rtotal  =  0.0                              #  合計値にゼロを代入する

m  =  1
while  m  <=  12:                           #  同じ処理を 12 回繰り返す

    Rtotal  =  Rtotal  +  R[m-1]            #  月降水量の値を加える

    m  =  m  +  1                           #  カウントをひとつ進める

#  乾燥する季節を判定する

Rmin     =  R[1-1]                          #  変数 Rmin に
MonRmin  =  1                               #    1 月の月降水量の値を代入する

m  =  2
while  m  <=  12:                           #  同じ処理を 11 回繰り返す

    if  R[m-1]  <  Rmin:                    #  月降水量が Rmin より小さい場合は
        Rmin     =  R[m-1]                  #    Rmin の値を更新する
        MonRmin  =  m

    m  =  m  +  1                           #  カウントをひとつ進める

#  計算結果を書き出す
print  ("Tave  =  %5.1f"      %  Tave)      #  年平均気温
print  ("Tmax  =  %5.1f"      %  Tmax)      #  最暖月平均気温
print  ("Tmin  =  %5.1f"      %  Tmin)      #  最寒月平均気温
print  ("Rtotal  =  %6.1f"    %  Rtotal)    #  年降水量
print  ("Rmin    =  %6.1f"    %  Rmin)      #  最少雨月降水量
```

```
print  ("Mon.(Rmin)  =  %2d"  %  MonRmin)    #  最少雨月


#  気候区分を判定する
if  Tmax  >=  10.0:      #  最暖月平均気温 Tmax が 10℃以上の場合
    if  Tmin  >=  18.0:     #  最寒月平均気温 Tmin が 18℃以上の場合
        Class  =  "A"
    elif  Tmin  >=  -3.0:  #  最寒月平均気温 Tmin が-3℃以上 18℃未満の場合
        Class  =  "C"
    else:                   #  最寒月平均気温 Tmin が-3℃未満の場合
        Class  =  "D"
else:                       #  最暖月平均気温 Tmax が 10℃未満の場合
    Class  =  "E"
print  ("Type  =  %s"  %  Class)             #  結果を書き出す
```

実行例：ターミナル上で実行してみよう。

```
$ python  prog11.py ↵

Input  file?

data47662.txt ↵

Month Temp.   Prec.
    1   5.4   59.7
    2   6.1   56.5
    3   9.4  116.0
    4  14.3  133.7
    5  18.8  139.7
    6  21.9  167.8
    7  25.7  156.2
    8  26.9  154.7
    9  23.3  224.9
   10  18.0  234.8
   11  12.5   96.3
   12   7.7   57.9
Tave  =   15.8
Tmax  =   26.9
Tmin  =    5.4
Rtotal  =  1598.2
Rmin    =    56.5
Mon.(Rmin)  =   2
Type  =  C
```

実行します。

ファイル名を入力します。

結果が出力されます。

【追加】

⑫乾季を判定しよう

【１】乾季の判定

最少雨月＝夏（４～９月）：

夏（４～９月）の最少月降水量 Rmin ≦ $\frac{1}{3}$×冬（10～３月）の最多月降水量 Rmax

⇒ ｓ型（夏季乾燥型）

夏（４～９月）の最少月降水量 Rmin ＞ $\frac{1}{3}$×冬（10～３月）の最多月降水量 Rmax

⇒ ｆ型（年中湿潤型）

最少雨月＝冬（10～３月）：

冬（10～３月）の最少月降水量 Rmin ≦ $\frac{1}{10}$×夏（４～９月）の最多月降水量 Rmax

⇒ ｗ型（冬季乾燥型）

冬（10～３月）の最少月降水量 Rmin ＞ $\frac{1}{10}$×夏（４～９月）の最多月降水量 Rmax

⇒ ｆ型（年中湿潤型）

注意：北半球を前提にしてプログラムを作成しています。南半球では季節の設定を逆にする必要があります。

【２】乾燥限界値の計算

乾季の型＝ｓ型（夏季乾燥型） ⇒ 乾燥限界値 Rdry＝20×年平均気温 Tave

乾季の型＝ｗ型（冬季乾燥型） ⇒ 乾燥限界値 Rdry＝20×（年平均気温 Tave＋14）

乾季の型＝ｆ型（年中湿潤型） ⇒ 乾燥限界値 Rdry＝20×（年平均気温 Tave＋７）

【３】気候区分の判定

年降水量 Rtotal≧乾燥限界値 Rdry：

最暖月平均気温 Tmax＝１０℃以上：

最寒月平均気温 Tmin＝１８℃以上 ⇒ 熱帯（Ａ気候）

最寒月平均気温 Tmin＝－３℃以上１８℃未満 ⇒ 温帯（Ｃ気候）

最寒月平均気温 Tmin＝－３℃未満 ⇒ 亜寒帯（Ｄ気候）

最暖月平均気温 Tmax＝１０℃未満 ⇒ 寒帯（Ｅ気候）

年降水量 Rtotal＜乾燥限界値 Rdry ⇒ 乾燥帯（Ｂ気候）

作業の手順：⑪で作成したプログラムをコピーして、Emacs で完成させよう。

```
T = [0.0] * 12
R = [0.0] * 12                          #  リストを宣言する
#  データを読みこむ
<中略>
#  年平均気温を計算する
<中略>
#  月平均気温の最大値と最小値を求める
<中略>
#  年降水量を計算する
<中略>
#  乾燥する季節を判定する
Rmin    = R[1-1]                        #  変数 Rmin に
MonRmin = 1                             #    1 月の月降水量の値を代入する
m = 2
while m <= 12:                          #  同じ処理を 11 回繰り返す
    if R[m-1] < Rmin:                   #  月降水量が Rmin より小さい場合は
        Rmin    = R[m-1]                #    Rmin の値を更新する
        MonRmin = m
    m = m + 1                           #  カウントをひとつ進める

if MonRmin >= 4 and MonRmin <= 9:       #  最少雨月が夏（4～9 月）の場合

    Rmax = R[1-1]                       #  変数 Rmax に
                                        #    1 月の月降水量の値を代入する
    m = 2
    while m <= 3:                       #  2～3 月の月降水量の値を調べる
        if R[m-1] > Rmax:               #  月降水量が Rmax より大きい場合は
            Rmax = R[m-1]               #    Rmax の値を更新する
        m = m + 1                       #  カウントをひとつ進める

    m = 10
    while m <= 12:                      #  10～12 月の月降水量の値を調べる
        if R[m-1] > Rmax:               #  月降水量が Rmax より大きい場合は
            Rmax = R[m-1]               #    Rmax の値を更新する
        m = m + 1                       #  カウントをひとつ進める

    if Rmin <= Rmax / 3.0:
        dry = "s"
    else:
        dry = "f"

if MonRmin <= 3 or MonRmin >= 10:       #  最少雨月が冬（10～3 月）の場合
```

```
    Rmax  =  R[4-1]                          #  4月の月降水量の値を代入する

    m  =  5
    while  m  <=  9:                         #  5～9月の月降水量の値を調べる
        if  R[m-1]  >  Rmax:
            Rmax  =  R[m-1]
        m  =  m  +  1                        #  カウントをひとつ進める

    if  Rmin  <=  Rmax  /  10.0:
        dry  =  "w"
    else:
        dry  =  "f"

#  乾燥限界値を計算する

if  dry  ==  "s":
    Rdry  =  20.0  *  Tave
if  dry  ==  "w":
    Rdry  =  20.0  *  (Tave  +  14.0)
if  dry  ==  "f":
    Rdry  =  20.0  *  (Tave  +  7.0)


#  計算結果を書き出す
print  ("Tave  =  %5.1f"      %  Tave)      #  年平均気温
print  ("Tmax  =  %5.1f"      %  Tmax)      #  最暖月平均気温
print  ("Tmin  =  %5.1f"      %  Tmin)      #  最寒月平均気温
print  ("Rtotal  =  %6.1f"    %  Rtotal)    #  年降水量
print  ("Rmax    =  %6.1f"    %  Rmax)      #  雨季最多降水量
print  ("Rmin    =  %6.1f"    %  Rmin)      #  最少雨月降水量
print  ("Mon.(Rmin)  =  %2d"  %  MonRmin)   #  最少雨月
print  ("Dry season  =  %s"  %  dry)
print  ("Rdry    =  %6.1f"    %  Rdry)      #  乾燥限界値

#  気候区分を判定する

if  Rtotal  >=  Rdry:   #  年降水量Rtotalが乾燥限界値Rdry以上の場合

    if  Tmax  >=  10.0:     #  最暖月平均気温Tmaxが10℃以上の場合

        if  Tmin  >=  18.0:     #  最寒月平均気温Tminが18℃以上の場合
            Class  =  "A"                    #  熱帯

        elif  Tmin  >=  -3.0:   #  最寒月平均気温Tminが-3℃以上18℃未満の場合
```

```
            Class  =  "C"                    #  温帯

        else:                       #  最寒月平均気温 Tmin が-3℃未満の場合
            Class  =  "D"                    #  亜寒帯

    else:                       #  最暖月平均気温 Tmax が 10℃未満の場合
        Class  =  "E"                        #  寒帯

else:                       #  年降水量 Rtotal が乾燥限界値 Rdry 未満の場合
    Class  =  "B"                            #  乾燥帯

print  ("Type  =  %s"  %  Class)             #  結果を書き出す
```

実行例：ターミナル上で実行してみよう。

```
$ python  prog12.py ↵

Input  file?

data47662.txt ↵

Month Temp.   Prec.
    1   5.4   59.7
    2   6.1   56.5
    3   9.4  116.0
    4  14.3  133.7
    5  18.8  139.7
    6  21.9  167.8
    7  25.7  156.2
    8  26.9  154.7
    9  23.3  224.9
   10  18.0  234.8
   11  12.5   96.3
   12   7.7   57.9
Tave  =   15.8
Tmax  =   26.9
Tmin  =    5.4
Rtotal  =  1598.2
Rmax    =   224.9
Rmin    =    56.5
Mon.(Rmin)  =   2
Dry season  =   f
Rdry    =   456.7
Type  =  C
```

実行します。

ファイル名を入力します。

結果が出力されます。

【追加】

⑬気候区分を判定しよう

作業の手順：⑫で作成したプログラムをコピーして、Emacs で完成させよう。

```
T = [0.0] * 12
R = [0.0] * 12                          #  リストを宣言する
#  データを読みこむ
<中略>
#  年平均気温を計算する
<中略>
#  月平均気温の最大値と最小値を求める
<中略>
#  年降水量を計算する
<中略>
#  乾燥する季節を判定する
<中略>
#  乾燥限界値を計算する
<中略>
#  計算結果を書き出す
<中略>

#  気候区分を判定する

if  Rtotal  >=  Rdry:   #  年降水量 Rtotal が乾燥限界値 Rdry 以上の場合

    if  Tmax  >=  10.0:     #  最暖月平均気温 Tmax が 10℃以上の場合

        if  Tmin  >=  18.0:     #  最寒月平均気温 Tmin が 18℃以上の場合

            if  Rmin  >=  60.0:
                Class  =  "Af"              #  熱帯雨林気候

            elif  Rmin  >=  100.0  -  0.04  *  Rtotal:
                Class  =  "Am"              #  熱帯モンスーン気候

            else:
                Class  =  "Aw"              #  サバナ気候

        elif  Tmin  >=  -3.0:   # 最寒月平均気温 Tmin が-3℃以上 18℃未満の場合

            if  dry  ==  "s"  and  Rmin  <  30.0:
                Class  =  "Cs"              #  地中海性気候

            elif  dry  ==  "w":
                Class  =  "Cw"              #  温暖冬季少雨気候

            else:
```

```
                if  Tmax  >=  22.0:
                    Class  =  "Cfa"          #  温暖湿潤気候

                else:
                    Class  =  "Cfb"          #  西岸海洋性気候

        else:                    #  最寒月平均気温 Tmin が-3℃未満の場合

            if  dry  !=  "w":
                Class  =  "Df"               #  亜寒帯湿潤気候

            else:
                Class  =  "Dw"               #  亜寒帯冬季少雨気候

    else:                    #  最暖月平均気温 Tmax が 10℃未満の場合

        if  Tmax  >=  0.0:
            Class  =  "ET"                   #  ツンドラ気候

        else:
            Class  =  "EF"                   #  氷雪気候

else:                    #  年降水量 Rtotal が乾燥限界値 Rdry 未満の場合

    if  Rtotal  >=  Rdry  /  2.0:
        Class  =  "BS"                       #  ステップ気候

    else:
        Class  =  "BW"                       #  砂漠気候

print  ("Type  =  %s"  %  Class)             #  結果を書き出す
```

実行例：ターミナル上で実行してみよう。

```
$ python  prog13.py ↵

Input  file?

data47662.txt ↵

Month Temp.   Prec.
    1   5.4   59.7
    2   6.1   56.5
    3   9.4  116.0
    4  14.3  133.7
    5  18.8  139.7
    6  21.9  167.8
    7  25.7  156.2
    8  26.9  154.7
    9  23.3  224.9
   10  18.0  234.8
   11  12.5   96.3
   12   7.7   57.9
Tave  =   15.8
Tmax  =   26.9
Tmin  =    5.4
Rtotal  =  1598.2
Rmax    =   224.9
Rmin    =    56.5
Mon.(Rmin)  =   2
Dry season  =   f
Rdry    =   456.7
Type  =  Cfa
```

実行します。

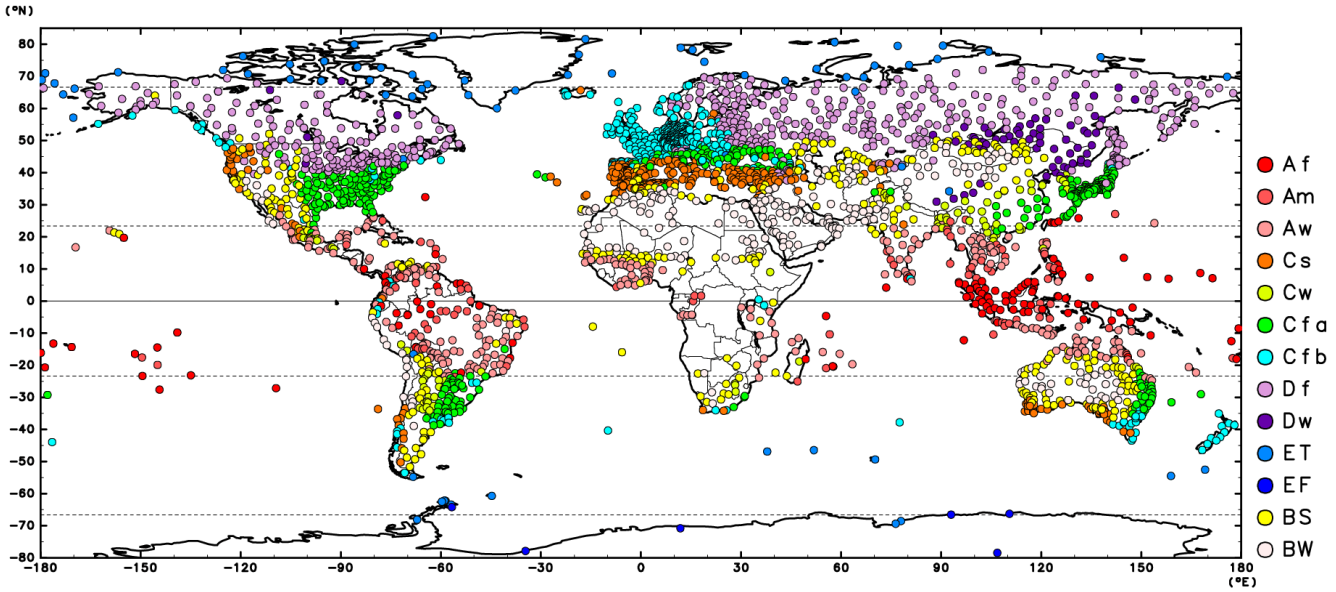
ファイル名を入力します。

結果が出力されます。

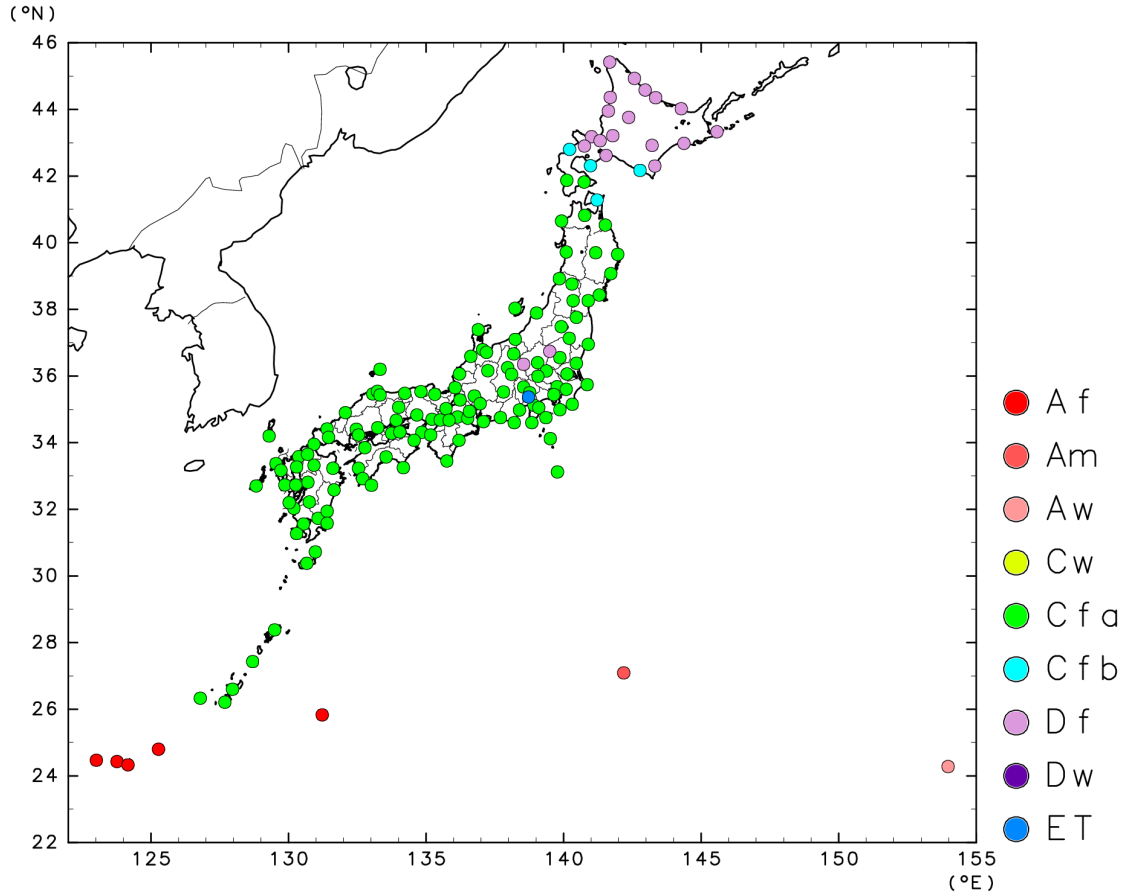
【参考】世界の代表的な地点の一覧

地点番号	地点	北緯〔度〕	東経〔度〕	標高〔m〕	気候区分
3772	ロンドン	51.48	-0.45	24	西岸海洋性気候（Cfb）
16245	ローマ	41.66	12.45	12	地中海性気候（Cs）
27612	モスクワ	55.85	37.62	147	亜寒帯湿潤気候（Df）
30710	イルクーツク	52.27	104.32	467	亜寒帯冬季少雨気候(Dw)
41640	ラホール	31.55	74.33	214	ステップ気候（BS）
47662	東京	35.68	139.77	6	温暖湿潤気候（Cfa）
48455	バンコク	13.73	100.56	3	サバナ気候（Aw）
48698	シンガポール	1.37	103.98	5	熱帯雨林気候（Af）
59287	コワンチョウ	23.22	113.48	72	温暖冬季少雨気候（Cw）
62378	カイロ	29.86	31.35	139	砂漠気候（BW）
70026	バロー	71.29	-156.78	12	ツンドラ気候（ET）
89606	ボストーク基地	-78.46	106.87	3488	氷雪気候（EF）
98425	マニラ	14.59	120.98	13	熱帯モンスーン気候(Am)

【参考】世界の気候区分



【参考】日本の気候区分


(°N)
46
44
42
40
38
36
34
32
30
28
26
24
22
125
130
135
140
145
150
155
(°E)
Af
Am
Aw
Cw
Cfa
Cfb
Df
Dw
ET

補遺：文法の解説

【変数の宣言】

Python では、適当な名前の変数に値を代入することによって、自動的に変数が宣言される。整数を代入すれば整数型の変数、実数を代入すれば浮動小数点型の変数（小数点以下を含む実数を表現するための変数）として宣言される。

例：
```
i = 1
x = 1.0
```
（実際には左側に空白を入れず行頭から書く。）

変数名の大文字と小文字は区別される。変数名は２文字以上の長さでもよい。プログラムの中では、順序や個数などを表すための整数と、連続的な数量を表すための実数（浮動小数点）は区別される。i = 1 と x = 1.0 は別の数である。

Python では、整数型どうしの四則演算の結果は整数型、浮動小数点型どうし、または整数型と浮動小数点型の四則演算の結果は浮動小数点型になる。ただし、割り算は整数型どうしであっても、演算の結果は浮動小数点型になる（割り切れる場合も含む)。整数型で割り算を実行するときは「 / 」の代わりに「 // 」を使う。この場合、小数点以下は切り捨てになる。

注意：以下、見やすくするために、プログラムの行頭に空白（インデント）を入れているが、実際にプログラムを書くときは、反復や条件分岐などの範囲を示す場合を除き、空白は入れない。

【メッセージを書き出す】

メッセージをターミナルに書き出すためには、print を使う。固定されたメッセージを書き出す場合は、
```
print ("Hello, world!")
```
のようにする。二重引用符で囲むことに注意。

【数値を書き出す】

ターミナルに数値を書き出すためには、
```
print ("i = %9d, x = %9.3f" % (i, x))
```
のようにする。「 %9d 」や「 %9.3f 」は書式指定子とよばれる。この例では、１番目の書式指定子%9d に i の値が、２番目の書式指定子%9.3f に x の値が代入される。「 %9d 」は９桁の整数、「 %9.3f 」は９桁の実数で小数点以下は３桁であることを示している。桁数の指定を省略して、「 %d 」、「 %f 」と書いてもよい。

【数値を読み取る】

ターミナルから数値を読み取るためには、input を使う。たとえば、整数型の変数 i に値を入れる場合は、
```
i = int (input())
```
浮動小数点型の変数 x に値を入れる場合は、

```
x = float (input())
```
のようにする。int や float は入力された文字列を整数型や浮動小数点型に変換するための関数である。

【処理の反復】

同じ処理を反復するためには、次のようにループを作成する。たとえば

```
i = 1
while i <= 1000:
    .....
    .....
    i = i + 1
```
（この範囲を４文字インデントする）

のようにすれば、４文字インデントされている行の処理が反復される。上の例では、初め変数 i の値は 1 であり、i が 1000 以下であれば同じ処理を繰り返す。１回の処理が終わるごとに変数 i に 1 を加えているので、処理は 1000 回反復することになる。Python では、反復処理を行う範囲をインデントによって示している点に注意する。したがって、単にプログラムを見やすくするという理由でインデントを用いることはできない。

【条件分岐】

一定の条件を満たす場合だけ処理を実行するためには、次のように if 文を用いる。たとえば

```
if x >= 0.0:
    .....
    .....
    .....
```
（この範囲を４文字インデントする）

のようにすれば、変数 x の値が 0.0 以上の場合のみ、４文字インデントされている行の処理が実行される。また、

```
if x >= 0.0:
    .....
else:
    .....
```
のようにすれば、「 x >= 0.0 」という条件を満たす場合、つまり変数 x の値が 0.0 以上の場合には else より前に書かれた処理が、そうでない場合には else より後に書かれた処理が実行される。さらに、

```
if x >= 0.0:
    .....
elif x >= -10.0:
    .....
else:
    .....
```
のように、最初の条件を満たさなかった場合に、次の条件でさらに分岐することも可能である。上の例では、変数 x が 0.0 以上ではなかった場合には-10.0 以上かどうかでさらに分岐している。反復処理と同様、条件分岐を行う範囲をインデントによって示している点に注意する。

【ファイルに書き出す】

数値をターミナルではなくファイルに書き出すためには、まず open で出力ファイルを開く。

```
f = open ("output.txt", "w")
```

f はファイルオブジェクトとよばれ、開いたファイルを示す目印である。open ではファイル名とモードを指定する。モードは今回の場合"w"であり書き込み可能であることを示している。すでに存在するファイルを指定すると上書きされる。open で開いたファイルに数値を書き出すためには、ファイルオブジェクト f の write メソッドを用いて、

```
f.write ("%9d %9.3f\n" % (x, z))
```

のようにする。基本的には print と同じ使い方である。「ファイルオブジェクト. メソッド」の形になっている点が異なっている。print とは異なり write メソッドでは、行末の改行は自動では付かないので行末に改行を意味する「 \n 」を付ける。ファイルへの書き出しが終わったら、

```
f.close ()
```

でファイルを閉じる。上記の例では、ファイルオブジェクトの名前を f としているが、他の名前であってもよい。

アプリケーションのインストール

※ここでは、WindowsＰＣ上にプログラミング環境を構築する方法を解説します。おもにWindows 10/11を想定した説明になっていますが、Windows 7やXPでも同様にインストールできます。

以下のインストール作業の解説の内容には十分に注意を払っておりますが、環境によっては指示通りに作業をしても正常にインストールできない場合があります。その場合のサポートには応じられませんので、あらかじめご了承ください。また、アプリケーションのインストールや利用の際に生じたトラブルや損害についても責任を負うことはできませんので、この点も理解のうえご利用ください。

【アカウントの準備】

ＰＣ上での自分のユーザ名（ログインするときの名前）を確認する。ユーザ名が半角英数字でない場合はインストールできない。ユーザアカウントを作成したときには半角英数字でなかったが、後で半角英数字に変更した場合も同様である。ユーザ名が半角英数字でない場合は、半角英数字のユーザ名で新しいアカウントを作成し、そのアカウントでインストールやプログラミングを行なう必要がある。

例：○ hgakugei　　× 学芸花子　　× ｈｇａｋｕｇｅｉ　（←全角英数字）

× 学芸花子からhgakugeiに変更

また、インストール作業では管理者権限が必要である。

【インストール作業】

ＣＤ－ＲＯＭの中のフォルダ「プログラミング環境」に保存されている圧縮フォルダmingw.zipをＰＣにコピーした後、すべて展開して、中身を確認する。

☞mingw.zipを右クリックして「すべて展開」を選択する。

単にダブルクリックしただけでは一時展開なので以下の作業がうまくいかない。

①フォルダMinGWをC:¥にコピーする。

☞画面左下のスタートボタンからWindowsシステムツール、エクスプローラーを選び、ローカルディスク(C:)をクリックすると、C:¥を開くことができる。

②フォルダemacs-24.3をC:¥にコピーする。

③フォルダgnuplotをC:¥にコピーする。

④システム環境変数を設定する：

スタートボタン→Windowsシステムツール→コントロールパネル

→システムとセキュリティ→システム→システムの詳細設定→詳細設定→環境変数

「システム環境変数」の中の「Path」を選択し、「編集」をクリックする。
「新規」をクリックして１行に１項目ずつ、「 C:¥MinGW¥bin 」、「 C:¥emacs-24.3¥bin 」、
「 C:¥gnuplot¥bin 」の合計３項目を追加する。
※Windows 7、XP では、「編集」をクリックした後、「 ; 」で区切りながら、
「 C:¥MinGW¥bin;C:¥emacs-24.3¥bin;C:¥gnuplot¥bin 」を追加する。
☞すでに書かれている内容の末尾にセミコロンを付け加え、その後にかぎ括弧の内容を追記する。

注意：すでに設定されている内容を絶対に変えてはいけない。大文字と小文字の違い、コロンとセミコロンの違いにも注意すること。この部分は特に慎重に行なう必要がある。

以上の設定変更を反映するため、①～③の作業で開いていたウィンドウを閉じて、新たに別のウィンドウを開いて⑤以下の作業を進める。

⑤C:¥MinGW¥msys¥1.0¥msys.bat (msys | Windows バッチファイル)をダブルクリックして実行する。
ターミナルが出てきたら以下のコマンドを実行する。

```
$ exit ↵          ☞exit とタイプして、エンターキーを押す
```

⑥ショートカットを作成する：
C:¥MinGW¥msys¥1.0¥msys.bat へのショートカットと
C:¥MinGW¥msys¥1.0¥home¥*username* へのショートカットを
デスクトップ上に作成する。
☞*username* は(半角英数字の)ユーザ名のことである。

⑦フォルダ files の中のファイル profile と emacs を
C:¥MinGW¥msys¥1.0¥home¥*username*¥の中にコピーする。

⑧ショートカットから msys を開始する。
☞「 msys.bat へのショートカット 」をダブルクリックする。
ターミナルが出てきたら以下のコマンドを実行する。

```
$ mv   profile   .profile ↵          ☞３つめの単語の語頭はピリオドである
$ mv   emacs   .emacs ↵
$ exit ↵
```

以上で基本的なプログラミング環境のインストールは完了である。再び「 msys.bat へのショートカット 」をダブルクリックすればターミナルが起動し、Emacs や gnuplot を利用できるはずである。Python をインストールする場合は、さらに⑨を実行する。

⑨フォルダ python の中の python-3.6.8-amd64.exe をダブルクリックして実行する：
まず以下の２つのオプションにチェックを入れる：

☑ Install launcher for all users (recommended)
☑ Add Python 3.6 to PATH
その後で「 Install Now 」をクリックする。
もし最後の段階で「 Disable path length limit 」というボタンが現れたら、
クリックしてから終了する。

以上で、Python 本体のインストールは完了である。

⑩フォルダ pythonlib を「ドキュメント」にコピーする：
☞エクスプローラーから「ドキュメント」を開き、フォルダ pythonlib を「ドキュメント」にコピーする。
画面左下のスタートボタンから Windows システムツール（または「Windows ツール」）を選び、「コマンドプロンプト」を起動する。
コマンドプロンプトで以下のコマンドをこの通りの順序で実行する：

```
>cd Documents
>cd pythonlib
```
☞以下の pip を実行したときに黄色の警告メッセージが出る場合があるが無視してよい。また、「successfully installed...」という表示の後であれば、赤色のエラーメッセージも無視してよい。
```
>pip install numpy-1.19.5-cp36-cp36m-win_amd64.whl
```
☞少し時間がかかります。
```
>pip install scipy-1.5.4-cp36-cp36m-win_amd64.whl
```
☞少し時間がかかります。
```
>pip install Pillow-8.4.0-cp36-cp36m-win_amd64.whl
>pip install cycler-0.11.0-py3-none-any.whl
>pip install kiwisolver-1.3.1-cp36-cp36m-win_amd64.whl
>pip install pyparsing-3.0.6-py3-none-any.whl
>pip install six-1.16.0-py2.py3-none-any.whl
>pip install python_dateutil-2.8.2-py2.py3-none-any.whl
>pip install matplotlib-3.3.4-cp36-cp36m-win_amd64.whl
```
☞少し時間がかかります。
```
>python sample.py
```
☞少し時間がかかります。
コマンドプロンプトに
Matplotlib is building the font cache; this may take a moment.
という表示が出て（出ない場合もある）、グラフが表示されたら、右上のボタンで閉じてよい。
```
>exit
```
と打って、コマンドプロンプトを終了する。

以上で、Python の追加ライブラリのインストールは完了である。

アンインストールについて

①～③で作成したフォルダを削除し、④で行なった環境変数の変更を元に戻せば、インストール前の状態に戻すことができる（Python を除く）。