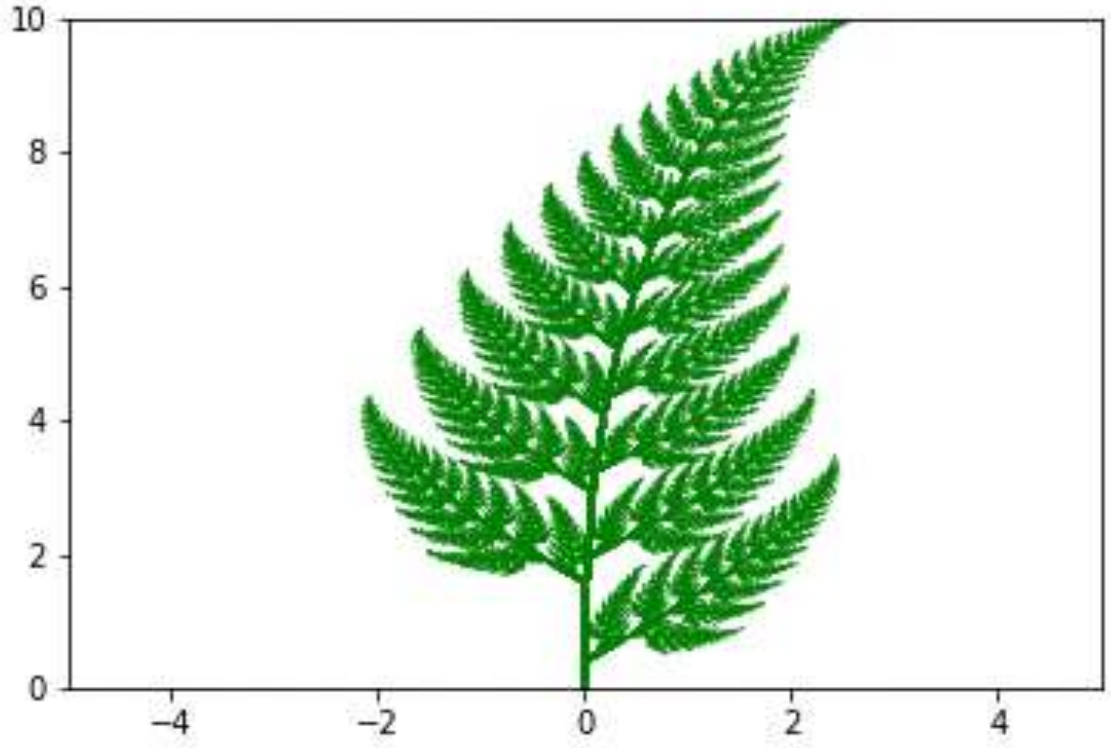
理数科のための
やさしいPython入門

天城高校物理科編

第1版

2022年7月



もくじ

巻頭言……3

1. Hello World!……4

2. やさしい計算……5

3. 繰り返し(for 文)……9

4. 繰り返し(while 文)……10

5. 条件分岐(if 文)……11

6. 関数……13

7. ファイル入出力……16

8. グラフ描画……20

9. ルンゲ・クッタ法による微分方程式の解法……24

10. データのフィッティング……30

11. ヒストグラム……34

12. ニュートン・ラプソン法による方程式の求解……35

13. シンプソン法による定積分の数値計算……37

14. モンテカルロ法……40

15. ユークリッドの互除法……41

16. フーリエ級数展開……43

17. フーリエ級数展開による熱伝導方程式の解法……48

18. 離散フーリエ変換……53

19. あとがき……64

20. 演習問題解答……65

巻頭言

「もう君たちとは逢えねえかも知れないけど、お互いに、これから、うんと勉強しよう。勉強というものは、いいものだ。代数や幾何の勉強が、学校を卒業してしまえば、もう何の役にも立たないものだと思っている人もあるようだが、大間違いだ。植物でも、動物でも、物理でも化学でも、時間のゆるす限り勉強して置かなければならん。日常の生活に直接役に立たないような勉強こそ、将来、君たちの人格を完成させるのだ。何も自分の知識を誇る必要はない。勉強して、それから、けろりと忘れてもいいんだ。覚えるということが大事なのではなくて、大事なのは、カルチベートされるということなんだ。カルチュアというのは、公式や単語をたくさん暗記している事でなくて、心を広く持つという事なんだ。つまり、愛するという事を知る事だ。学生時代に不勉強だった人は、社会に出てからも、かならずむごいエゴイストだ。学問なんて、覚えると同時に忘れてしまってもいいものなんだ。けれども、全部忘れてしまっても、その勉強の訓練の底に一つかみの砂金が残っているものだ。これだ。これが貴いのだ。勉強しなければいかん。そうして、その学問を、生活に無理に直接に役立てようとあせってはいかん。ゆったりと、真にカルチベートされた人間になれ！　これだけだ、俺の言いたいのは。」

太宰治　「正義と微笑」より

1. Hello World!

・本テキストはプログラミング未経験者を含む理数科生を対象としており，主に数値計算のやさしい具体例を通して，基礎的な処理方法に慣れてもらうことを目的としている。

・用いるアプリケーションは Google Colaboratory (Colab)であり，Web 上でプログラミング言語 Python を記述・実行するためのものである。

・プログラミングは，何より楽しみながら取り組むことが大切なので，細かな文法事項にこだわらず，エラーを恐れずに自由にプログラムを書いてほしい。

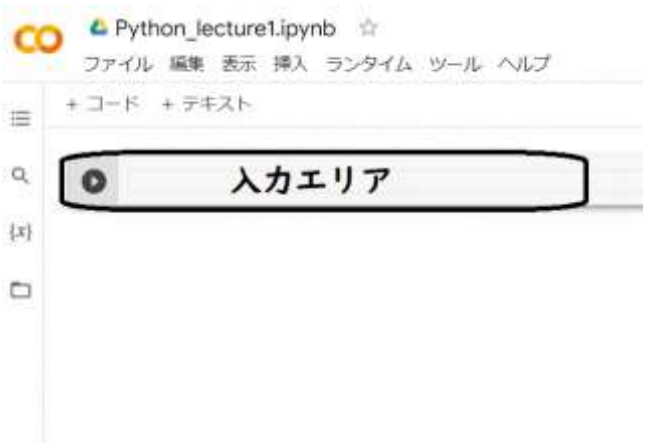
・まずはじめに，Google Chrome で「Google Colaboratory」と入力し，検索。

・検索結果の一番上のほうにある”https://colab.research.google.com/”という URL のサイトに移動。

・はじめに現れる画面で左下青文字「ノートブックを新規作成」を選択。

・Colab のノートブックが立ち上がるので，左上のファイル名”Untitled0.pynb”の”Untitled”のところを任意のファイル名に変更する。

・▷マーク右側がプログラムを入力するエリアとなっている。この入力エリアは「セル」と呼ばれる。



・セル内に試しに以下のコードをそのまま入力し，左の▷マークをクリックするか，Ctrl キーと Enter キーを同時に押す（以下，これを「実行」ということにする。）

```
print(“Hello World!”)
```

・セルのすぐ下に `Hello World!` と表示されることを確認。

・先ほど使った”`print()`”という文字列は，「`()`の中を表示しなさい」という命令を表している。

・画面左上「＋コード」をクリックすると，セルが下に追加される。 （目次へ）

2. やさしい計算

・次に，Python を電卓として用いる。セル内に以下のコードをそのまま入力し，実行する。

```
1 + 2
```

・セルのすぐ下に計算結果が表示される。

・Python で用いられる，掛け算と割り算，余り，累乗の記号はそれぞれ*，/，%，**である。

・演算子には優先順位があり，例えば，*，/は+，-よりも優先順位が高い。

・演算記号と数字の間に半角空白を入れるかどうかはいずれでもよい。

・演算をまとめると以下のようになる

加法	a + b
減法	a - b
乗法	a * b
除法	a / b
余り	a % b
累乗	a ** b
等号	a == b

【演習 1】 以下の式を入力し，結果を確かめよ。 （目次へ）

```
(1)  1 / 2 + 3
(2)  1 / (2 + 3)
(3)  365 % 7
(4)  -3 ** 2
(5)  (-3) ** 2
(6)  2 ** (-3)
(7)  3 ** 0
(8)  0 ** 0
```

```
(9)  1 == 1
(10) 1 == 0
(11) 1 == True
(12) 1 == False
```

・Python では，文字式も扱うことができる。試しに，以下のコードをそのまま入力する。

```
a = 1
b = 2
c = a + b
c
```

・ここで用いた a，b，c という文字は「変数」と呼ばれ，具体的な数字を格納できる「箱」のようなものである。

a = 1 というコードの"="という記号の意味は数学の等号とは異なり，「右辺の数字を左辺の変数に格納する」という意味である。これを「a に 1 を代入する」ともいい，変数に値を代入する行のことを代入文とよぶ。

・x と y が等しいという関係は，x == y と表す。

・先ほどのコードを実行すると，1 + 2 の計算結果である 3 が表示される。最後の c を入力しないと何も結果は表示されないので注意。

・プログラムが長くなると，後からコードを見返しても何をしていたのかがわからなくなる恐れがある。このため，プログラムにはこまめにメモを記入することをおすすめする。

・コードのすぐ左に#を記入すると，#の右側にはメモを記入できるようになる。記入したメモは実行の際に無視されるので，プログラム本体に影響はない。

・終了するときは，「ファイル」→「保存」で作業内容を保存し，ブラウザを閉じる。

・保存したファイルは「マイドライブ」内に「Colab Notebooks」というフォルダが自動作成され，その中に入っている。

・以上学習した内容を使って，底辺が 10，高さが 7 の三角形の面積を求めるプログラムを作成すると，つぎのようになる。

```
base = 10
height = 7
area = base * height / 2
print(area)
```

【演習 2】

上底 3，下底 5，高さ 7 の台形の面積を求めるプログラムを作成せよ。 （目次へ）

3. 繰り返し（for 文）

・次に，for 文を用いた繰り返し処理について学習する。例えば，1 から 10 までの自然数の和を求めるプログラムは，素直に書けば以下のようになる。

```
1 + 2 + 3 + 4 + 5 + 6+ 7 + 8 + 9 + 10
```

・10 までは上のプログラムでも可能だが，1 から 100，1 から 1000 となるとプログラムを書くのに非常に時間がかかってしまう。Python では繰り返しという操作が用意されており，作業回数が膨大な数に及ぶ場合でも簡潔なプログラムで実行できる。

・1 から 10 までの和を繰り返しを用いて計算するプログラムはつぎのように表される。


```
sum = 0
for i in range(1,11, 1):
  sum = sum + i
print(sum)
```

・このプログラムを実行すると，55 が表示される。

・初めの行の `sum = 0` は，`sum` という変数を用意し，0 を代入するという操作を表している。これを変数の初期化という。

・その次の `for` のある行で繰り返し処理を行うことを表し，文字 `i` が取りうる数の範囲を指定している。`range(1, 11, 1)` は，`range(i` の初めの値,`i` の最後の値,`i` の増分`)` という対応関係になっている。ここで気を付けなければならないのは，繰り返し処理で `i` の最後の値は含まれないということである。すなわち，10 までの和を求める場合，`i` の最後の値を 11 にしなければなら

ないことに注意。次の `sum = sum + i` という行で繰り返し処理の具体的な内容を示している。式の意味は数学とは異なり，右辺の `sum + i` という計算結果を左辺の `sum` という変数に格納しなおすということを表している。また，この行の一番左の字下げ（インデント）は，Python では繰り返しなどのまとまった処理のブロックを表す重要なものである。

・試しに，以下のプログラムを実行してみる。

```
sum = 0
for i in range(1, 11, 1):
sum = sum + i
print(sum)
```

実行結果はエラーとなる。

【演習 3】

以下の和を求めよ。 （目次へ）

$$\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{100^2}$$

4. 繰り返し（while 文）

・繰り返し処理には先ほどの for 文のほかに，while 文がある。for 文と while 文は一方から他方へ書き換えることが可能である。例えば，1 から 10 までの和を while 文を用いて求めると以下のプログラムのようになる。

```
sum = 0
i = 0
while (i < 11):
  sum = sum + i
  i = i + 1
```

```
print(sum)
```

・初めの 2 行で sum，i という変数を初期化している。

次の while がある行で繰り返し処理を行うことを示し，while(i < 11)は変数 i が 11 未満の場合はインデントされた行の命令を実行するという意味である。i が 11 になった時点で繰り返し処理が終了する。while 文を使う場合，処理の最後に繰り返し処理の回数を表す変数（カウンタという。ここでは i）を増やさなければならない。i = i + 1 がそれに対応しているが，この行がない場合，while(i < 11)は常に成り立つことになり，繰り返し処理が終了しなくなる。このプログラム上のミス（バグ）を無限ループという。

【演習 4】

1 から 1000 までの奇数の和を求めるプログラムを，while 文を用いて作成せよ。 （目次へ）

5. 条件分岐（if 文）

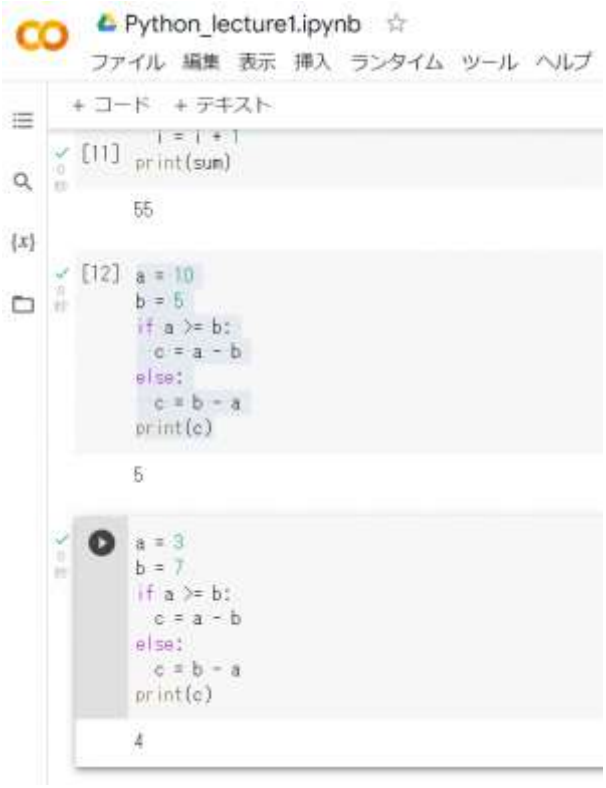
・次に，条件に応じて異なる処理を行う方法について扱う。例えば，2 つの数字 a と b の差の絶対値を求めるとき，以下のように a と b の大小関係によって処理が異なる。

a ≧ b のとき　　a − b

それ以外のとき　　b − a

この処理を行うプログラムは以下のようになる。

```
if a >= b:
  c = a - b
else:
  c = b - a
```



ここで，>=は≧と同じ意味である。if のすぐ右で条件を指定し，それが成り立つ場合に下のインデントされた行を実行する。もし条件が成り立たない場合，else の下のインデントされた行を実行する。

・試しに a に 10，b に 5 を設定すると結果は 5，a に 3，b に 7 を設定すると 4 となる。

・ここまでは変数に入れる値はプログラムの初めに指定していたが，a と b に後から自由に数値を入力できるほうがより柔軟な処理が可能になる。これを実現するには，例えば以下のようにプログラムを書けばよい。

```
str_a = input("a を入力してください")
str_b = input("b を入力してください")
a = int(str_a)
b = int(str_b)
if a >= b:
  c = a - b
else:
  c = b - a
print(c)
```

・このプログラムを実行すると，初めに以下のメッセージが表示される。

a を入力してください

・この枠内に数字を入力して Enter キーを押すと，input()という命令によって，変数 str_a に代入される（str は string の略で，文字列であることを表すために便宜上つけている）。ここで注意しなければならないのは，代入されるものが数値ではなく文字列であるということである。Python では数値と文字列は区別される。例えば 5 を入力しても，数値とはみなされず，数学的な意味とは切り離された，見た目が 5 という文字を表しているに過ぎない。これを数値とする操作が a = int(str_a)という行で表されている。int()は，カッコ内の変数を整数（英語で integer）に変換する役割を持っている。

【演習 5】

1 から 100 までの自然数の中で，3 の倍数のみを表示するプログラムを作成せよ。

【演習 6】

1 から 100 までの自然数の中で，任意の数を一つ入力できるようにし，その自然数の倍数のみを表示するプログラムを作成せよ。 （目次へ）

6. 関数

Python で関数を定義することができる。関数を用いることで，複雑な数式の処理も簡潔なコードで実現できるようになる。まず初めに簡単な例として，2 つの数字の和を求める関数 sum を次のように定義する。

```
sum(a, b) = a + b
```

これをプログラムとして書くと以下のようになる。

```
def sum(a, b):
 c = a + b
 return c
```

・def は関数を定義することを表している。その次の行で関数の具体的な処理を指定している。最後の return という命令は，何を関数の結果とするのかを指定している。ここでは，変数 a と b の和 c を結果として返している。この値のことを「戻り値」あるいは「返り値」という。また，関数に入力する変数（ここでは a と b）のことを引数（ひきすう）という。

・関数 sum を定義したので，試しに sum(1,2) を実行する。ここで，最後の行を sum(1,2) にしただけでは何も表示されないので，結果を表示するためには print(sum(1,2)) とする必要があることに注意。

・プログラミングにおける関数は数学の関数よりも広い意味を持ち，まとまった処理一般に対して使われる。例えば，入力した文字列を表示する print() や，変数に入力した値を代入する input() も，カッコ内の文字列を引数とする関数である。これらの関数は組み込み関数とよばれ，定義はあらかじめ用意されているので，あらためて定義する必要はない。

【演習 7】

2 つの数値 a と b の差の絶対値を返り値とする関数 dif(a, b) を作成せよ。

【演習 8】 （目次へ）

1 から n までの自然数の和を返り値とする関数 sum(n) を，繰り返し処理を用いて作成せよ。

・Python を用いて自然現象などをシミュレーションするとき，三角関数や指数関数など，数学で用いられる関数が必要になる。これらの関数は，math と呼ばれる，関数や特別な値をまとめたファイル（これを一般にモジュールという）を呼び出すことで使うことができる。モジュールの呼び出し方は，次のように import コマンドを用いる。

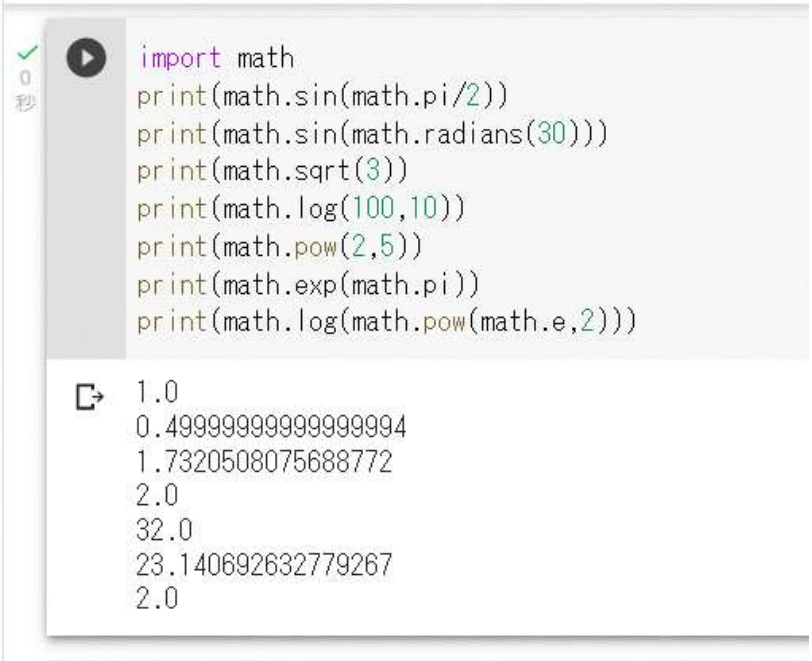
import モジュール名

・Python の数学関数，定数一覧

関数（定数）	Python での表記
$\sin x$	math.sin(x)
$\cos x$	math.cos(x)
$\tan x$	math.tan(x)
e^x	math.exp(x)
a^x	math.pow(a,x)
$\log x$	math.log(x)
$\log_{10} x$	math.log10(x)
$\log_2 x$	math.log2(x)
$\log_a x$	math.log(x,a)
$\sqrt{x}$	math.sqrt(x)
$\|x\|$	math.fabs(x)
$x!$	math.factorial(x)
${}_nC_k$	math.comb(n,k)
x [rad] → x [°]	math.degrees(x)
x [°] → x [rad]	math.radians(x)
π	math.pi
e	math.e

・具体的な使い方は以下のようになる。

```
import math
print(math.sin(math.pi/2)) #sin(π/2)
print(math.sin(math.radians(30))) #sin(30°)
print(math.sqrt(3))  #√3
print(math.log(100,10))  #log10 100
print(math.pow(2,5))  #2^5
print(math.exp(math.pi))  #e^π
print(math.log(math.pow(math.e,2)))  #log e^2
```



【演習 9】

地面から小球を初速度 v [m/s]，地面からの角度 theta [°] で斜方投射するとき，着地点と投射点の間の距離を求める関数 L [m] を v, theta を引数として作成せよ。重力加速度を 9.8 [m/s^2]とする。 （目次へ）

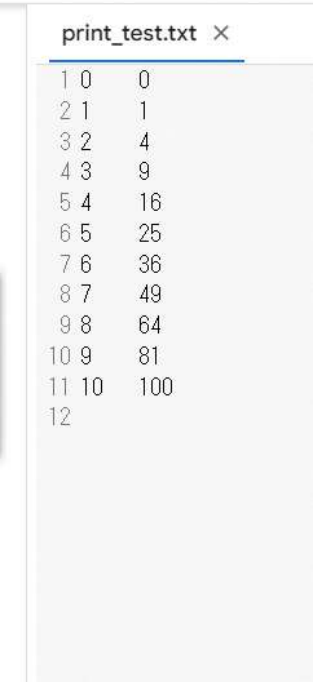
7. ファイル入出力

・数値計算を行うとき，計算結果をファイルに書き出すと，データをグラフ描画や解析などに再利用できて便利である。一例として，xとx^2の値をタブで区切ってファイル print_test.txt に

書き出すプログラムは以下のようになる。各行の行頭の位置にも気を付けてもらいたい。

```
with open('print_test.txt', 'w') as f: #書き込みモードでファイルを新規作成
  x = 0
  while (x < 11):
    print(x,'¥t' ,x**2, file = f)
    i = i + 1
```

・初めの行は，print_test.txt という名前のテキストファイルを作成することを表している。'w'は書き込みモードでファイルを扱うという意味である。その次の行はファイルに書き出す具体的な内容である。書き込みは print()関数を用いる。引数内の¥t はタブを表す。''で囲っていることに注意。タブと半角スペースが並んだものは異なる。¥t のように，¥で始まる記号はエスケープシーケンスと呼ばれ，タブのほかに改行を表す¥n などがある。プログラムを実行してできたファイルは，左の「ファイル」内に保存される。「ファイル」をクリックすると，print_test.txt があるので，それをダブルクリックすると，右側にファイルの中身が表示される。「ファイル」内に作成したデータは，一定時間が経過すると自動的に削除される。ファイルを保存しておきたいときは，ファイル名の右側に丸印が縦に 3 つ並んだマークをクリック→ダウンロードを選択すると，PC 上に保存される。



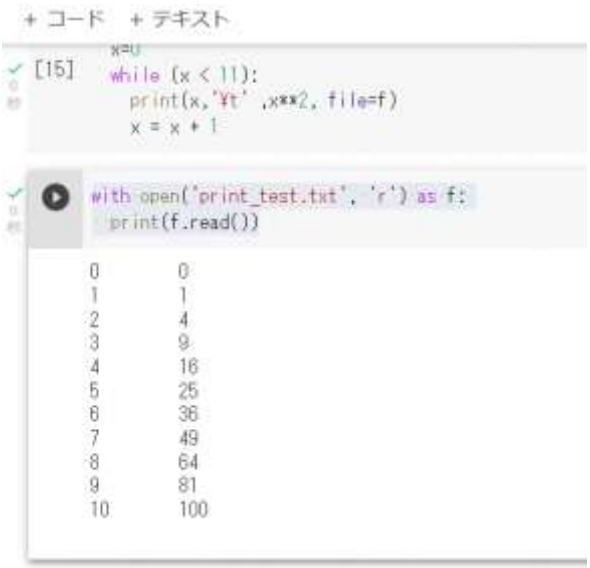
【演習 10】

原点 O から小球を初速度 10 [m/s]，x 軸正の向きからの角度 45°で斜方投射するとき，時刻 t にお

ける小球の x 座標，y 座標をタブ区切りで左から「時刻」，「x 座標」，「y 座標」の順で新規テキストファイル projectile.txt に書き出すプログラムを作成せよ。重力加速度を 9.8 m/s^2 とし，時刻は 3.0 秒までで間隔は 0.1 秒とする。 （目次へ）

・print_test.txt 内のデータを読み込んで表示するには次のようなプログラムを書けばよい。

```
with open('print_test.txt', 'r') as f:
  print(f.read())
```



・1 行目は print_test.txt を開くことを表している。'r'は読み込みモードでファイルを扱うという意味である。read()で読み込んだ内容を，print()で表示している。

・テキストファイルのデータを読み込んで，平均などの演算処理を行うためには，データを配列に格納すればよい。配列とは，データを収めることができる箱が並んだものと考えることができる。

・print_test.txt 内のデータを配列に格納するには，次のようなプログラムを書けばよい。

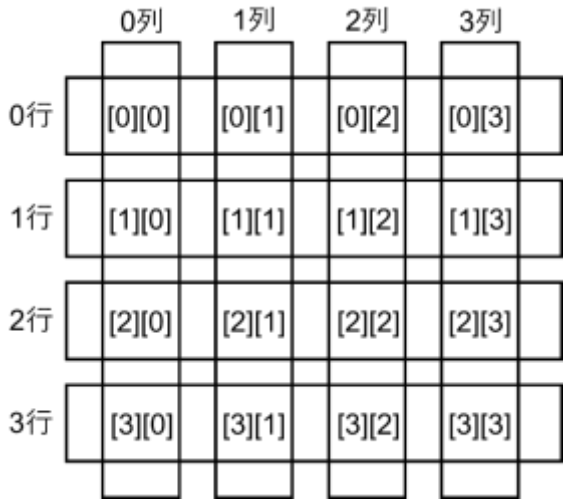
```
import csv
with open('print_test.txt') as f:
  data = csv.reader(f, delimiter = '¥t')
  A = [row for row in data]
```

・3 行目で print_test.txt 内のデータをタブ（¥t）区切りで読み込み，4 行目で配列 A に格納している。配列 A 内のデータは整数 m，n を用いて A[m][n]の形式で取り出すことができる。

m，n は 0 から始まることに注意。A[m][n]は変数が 2 次元的に並べられたものと考えることができ，2 次元配列と呼ばれる。このとき，横の変数の並びを行，縦の変数の並びを列という。

・print_test.txt 内のデータは次の順序で格納されている。この具体例を通じて，データが配列のどの位置に収められるかを読み取ってもらいたい。

```
A[0][0] = 0,   A[0][1] = 0
A[1][0] = 1,   A[1][1] = 1
A[2][0] = 2,   A[2][1] = 4
A[3][0] = 3,   A[3][1] = 9
A[4][0] = 4,   A[4][1] = 16
A[5][0] = 5,   A[5][1] = 25
```



・例えばこのデータの第 1 列の平均を求めるプログラムはつぎのようになる。

```
import csv

with open('print_test.txt') as f:
  data = csv.reader(f, delimiter = '¥t')
  A = [row for row in data]

sum = 0
for i in range (0, len(A), 1):
  sum = sum + int(A[i][1])

average = sum/len(A) #len(A)は配列 A の行数

print(average)
```

・配列内のデータは，文字列として扱われるため，数値データに変換するために int(A[i][1])のようにする必要がある。配列 A のデータの行数は len(A)で求められる。

・配列 A の第 1 列のデータの最大値と，そのときの第 0 列の値（すなわち，上から何行目か）を求めるプログラムは次のようになる。

```
import csv

with open('print_test.txt') as f:
  data = csv.reader(f, delimiter = '¥t')
  A = [row for row in data]

max = 0 #最大値を記憶しておくための変数
index = 0 #最大値を与える行の番号
for i in range (0, len(A), 1):
  if max < int(A[i][1]):
    max = int(A[i][1])
    index = i + 1
print('上から', index, '行目のときに最大値', max, 'をとる')
```

・配列 A のデータが例えば 2.78 や 3.14 のように整数でない場合，int()の代わりに float()を用いて float(A[i][1])のようにする必要がある。

【演習 11】

地面から斜方投射を行う場合，投射角が45°のときに投射点と着地点の水平距離（飛距離）が最大となる。ここで，高さが 2.0 m の位置から水平方向に対して上方に角度 theta [°]，初速度の大きさ 10 [m/s]で投げ上げたとき，投射角が何度のときに飛距離は最大となるか。また，飛距離の最大値は何 m か。 （目次へ）

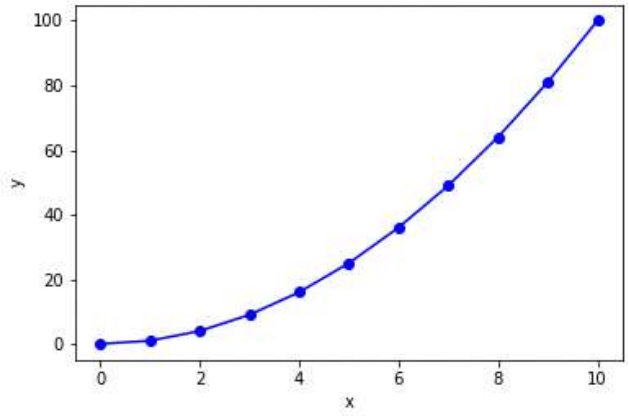
8. グラフ描画

・数値計算の結果をわかりやすく表示するためには，データをグラフで示すことが不可欠である。ここでは，グラフを作成して画像ファイルに出力するまでを扱う。print_test.txt 内のデータをグラフに表示し，ファイル dataplot.png に出力するためのプログラムは以下のよう

になる。

```
import numpy as np
import matplotlib.pyplot as plt

data = np.loadtxt('print_test.txt')

plt.plot(data[:,0], data[:,1], '-ob') #1 列目を x 軸, 2 列目を y 軸, -は実線, o は丸印, b は青

plt.xlabel('x')#x 軸名を x にする
plt.ylabel('y')

plt.savefig('dataplot.png')#ファイル名を指定して保存
```

・出力されたグラフは右のようになる。



`plt.plot(data[:,0], data[:,1], '-ob')` という命令で 1 列目を x 軸，2 列目を y 軸としてグラフを作成している。列番号は 0 から始まることに注意。

・'-ob'はグラフのスタイルを指定しており，-は実線，o は丸印，b は青を表している。もし実線のみにしたいときは'-b'のようにすればよい。黒線は k，赤線は r で指定する。

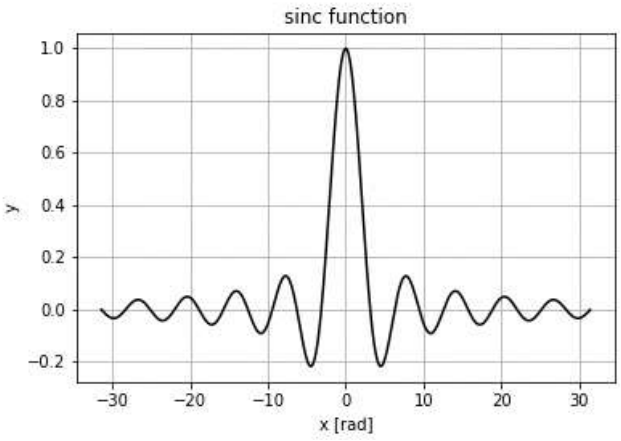
・点の形は'o'以外に'v'，'^'，'d'などがあり，線の種類は'-'以外に'-.'，'—'，':'などが指定できる。

・プロットの色指定の記号一覧

k	黒

r	赤
b	青
y	黄色
g	緑
c	シアン
m	マゼンタ

・もしテキストファイルからのデータではなく，数式をプロットするには，例えば次のようにすればよい。例として示しているのは sinc 関数と呼ばれるもので，信号処理などで使われている。




```
import math
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

x = np.arange(-10*np.pi, 10*np.pi, 0.1)
plt.plot(x, np.sin(x)/x, '-k')#黒の実線
plt.title('sinc function')
plt.xlabel('x [rad]')
plt.ylabel('y')
plt.grid(True)#グリッド線を入れる
plt.savefig('sinc.png')
```

・上のプログラムの `x = np.arange(-10*np.pi, 10*np.pi, 0.1)` は$-10\pi \leqq x \leqq 10\pi$の範囲で$x$軸間隔が 0.1 おきにプロットすることを表している。

【演習 12】

以下の関数$I(x)$をプロットし，グラフをファイルに出力せよ。これは単スリットによる干渉で生じた干渉縞の強度分布を表している。

$$I(x) = \left(\frac{\sin(\pi x)}{\pi x}\right)^2$$

【演習 13】

演習 11 で飛距離が最大となるときの小球の軌跡を描画し，グラフをファイルに出力せよ。小球の投射点の座標は$(0, 2)$とする。 （目次へ）

本授業で主に用いる python の関数一覧

`print()`	引数の文字列や数字を表示
`input()`	引数の文字列を表示したのち，入力を受け付ける
`int()`	引数の文字列を数値に変換（整数）
`float()`	引数の文字列を数値に変換
`open()`	引数で指定したファイル名のファイルを開く
`read()`	ファイル全体を文字列として読み込む
`len()`	引数で指定した配列の長さを求める

append()	引数で指定した配列の末尾に要素を追加する

9. ルンゲ・クッタ法による微分方程式の解法

・関数$f(x)$の導関数$f'(x)$（または，dy/dx）は以下の式で定義される。≔は左辺を右辺で定義する，$\lim_{h\to 0}$はhを 0 とは異なる値をとりながら 0 に限りなく近づけるという意味である。

$$f'(x) = \frac{dy}{dx} \coloneqq \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}$$

・高校数学でよく用いられる関数$f(x)$とその導関数$f'(x)$の一覧は以下のようになる。

$f(x)$	$f'(x)$
$\sin x$	$\cos x$
$\cos x$	$-\sin x$
$\tan x$	$\dfrac{1}{\cos^2 x}$
e^x	e^x
$\log x$	$\dfrac{1}{x}$
$x^n \left(n\text{は実数}\right)$	nx^{n-1}

・ここで，$f(x) = g(x)h(x)$と表されるとき，$f'(x)$はつぎの式で求められる。

$$f'(x) = g'(x)h(x) + g(x)h'(x)$$

・また，$f(x) = g(x)/h(x)$ と表されるとき，$f'(x)$はつぎの式で求められる。

$$f'(x) = \frac{g'(x)h(x) - g(x)h'(x)}{\left(h(x)\right)^2}$$

・ここでは，以下の方程式を 4 次のルンゲ・クッタ法と呼ばれる方法を用いて解く。この式のように，関数の微分を含む方程式を微分方程式と呼び，微分方程式から元の微分する前の関数（原始関数）を求めることを微分方程式を解くという。

$$\frac{dx}{dt} = f(x,t)$$

・数値計算を行う前に，まずやさしい例としてつぎの微分方程式を初期値$x(0) = 0$という条件で解く。

$$\frac{dx}{dt} = t$$

上式を変形すると

$$dx = tdt$$

両辺を積分して

$$\int dx = \int tdt$$

$$x = \frac{1}{2}t^2 + C \ \ (C\text{は定数})$$

$t = 0$のとき$x = 0$だから，$C = 0$

したがって，求める解は

$$x = \frac{1}{2}t^2$$

・この例では微分方程式の解を求めることができたが，実際の研究の場面では微分方程式を立てることができても，解を明確な式で表せないことが多い。この場合，数値計算に頼る必要があり，広く用いられている手法が 4 次のルンゲ・クッタ法である。

・4 次のルンゲ・クッタ法の手順は次のようになる。

・初期条件を$x(t_0) = x_0$とし，$t_0 < t < t_1$までの範囲をN分割する。

・刻み幅hと，分割された各時刻t_iは以下の式で表される。

$$h = \frac{t_1 - t_0}{N}$$

$$t_i = t_0 + ih$$

・$x_i = x(t_i)$として，次の手順でx_{i+1}を定める。

$$k_1 = hf(x_i, t_i)$$

$$k_2 = hf\left(x_i + \frac{1}{2}k_1, t_i + \frac{1}{2}h\right)$$

$$k_3 = hf\left(x_i + \frac{1}{2}k_2, t_i + \frac{1}{2}h\right)$$

$$k_4 = hf(x_i + k_3, t_i + h)$$

$$x_{i+1} = x_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

・一例として，以下の微分方程式を$x(0) = 0$の初期条件で解く。

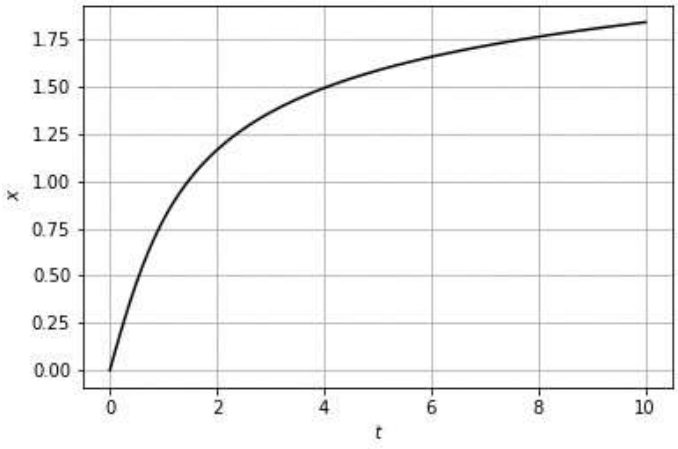
$$\frac{dy}{dx} = e^{-x^2}$$

・この微分方程式の解となる式を求めることはできないが，ルンゲ・クッタ法を用いることで，

十分な精度で数値計算ができる。

・解を求めるプログラムと，その結果を示すと次のようになる。

```
import math
import matplotlib.pyplot as plt
#関数の定義
def f(x,t):
  return math.exp(-x**2)

#初期条件
t0 = 0
x0 = 0
#終状態の時刻
t1 = 10
#分割数を 100 に設定
N = 100
#刻み幅
h = (t1 - t0)/N
#t, x を初期化
t = t0
x = x0
#配列 T, X を初期化
T = []
X = []
T.append(t)#append()は配列の末尾に要素を追加する関数
```



```
X.append(x)
```

#4 次のルンゲ・クッタ法

```
for i in range(1, N + 1):
  k1 = h*f(x,t)
  k2 = h*f(x+0.5*k1, t+0.5*h)
  k3 = h*f(x+0.5*k2, t+0.5*h)
  k4 = h*f(x+k3, t+h)
  x = x + (k1 + 2*k2 + 2*k3 + k4)/6
  t = t + h
  T.append(t)
  X.append(x)
```

#グラフを作成してファイルに保存

```
plt.plot(T,X, '-k');
plt.xlabel('t', fontstyle = 'italic')
plt.ylabel('x', fontstyle = 'italic')
plt.grid(True)

plt.savefig('diff_eq.png')
```

【演習 14】

以下の 1 階常微分方程式を$x(0) = 0.1$の初期条件で解き，グラフを出力せよ。

$$\frac{dx}{dt} = (1 - x)x$$

・$f'(x)$をさらにxで微分した関数は$f''(x)$あるいはd^2f/dx^2と表され，2 階微分と呼ばれる。

・力学や電気回路などで振動現象を扱う際，以下の形式で表される 2 階の常微分方程式が現れることがある。

$$\frac{d^2x}{dt^2} = f\left(x, \frac{dx}{dt}, t\right)$$

・このとき，$v \coloneqq dx/dt$とおけば次の連立 1 階常微分方程式に帰着できる。

$$\frac{dx}{dt} = F_1(x, v, t) = v$$

$$\frac{dv}{dt} = F_2(x, v, t) = f(x, v, t)$$

・初期条件を$x(t_0) = x_0, v(t_0) = v_0$とし，$t_0 < t < t_1$を$N$分割して解く。

・刻み幅hと，分割された各時刻t_iは以下の式で表される。

$$h = \frac{t_1 - t_0}{N}$$

$$t_i = t_0 + ih$$

・$x_i = x(t_i)$，$v_i = v(t_i)$として，次の手順でx_{i+1}，v_{i+1}を定める。

$$k_1 = hF_1(x_i, v_i, t_i)$$
$$m_1 = hF_2(x_i, v_i, t_i)$$
$$k_2 = hF_1\left(x_i + \frac{1}{2}k_1, v_i + \frac{1}{2}m_1, t_i + \frac{1}{2}h\right)$$
$$m_2 = hF_2\left(x_i + \frac{1}{2}k_1, v_i + \frac{1}{2}m_1, t_i + \frac{1}{2}h\right)$$
$$k_3 = hF_1\left(x_i + \frac{1}{2}k_2, v_i + \frac{1}{2}m_2, t_i + \frac{1}{2}h\right)$$
$$m_3 = hF_2\left(x_i + \frac{1}{2}k_2, v_i + \frac{1}{2}m_2, t_i + \frac{1}{2}h\right)$$
$$k_4 = hF_1(x_i + k_3, v_i + m_3, t_i + h)$$
$$m_4 = hF_2(x_i + k_3, v_i + m_3, t_i + h)$$
$$x_{i+1} = x_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$v_{i+1} = v_i + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4)$$

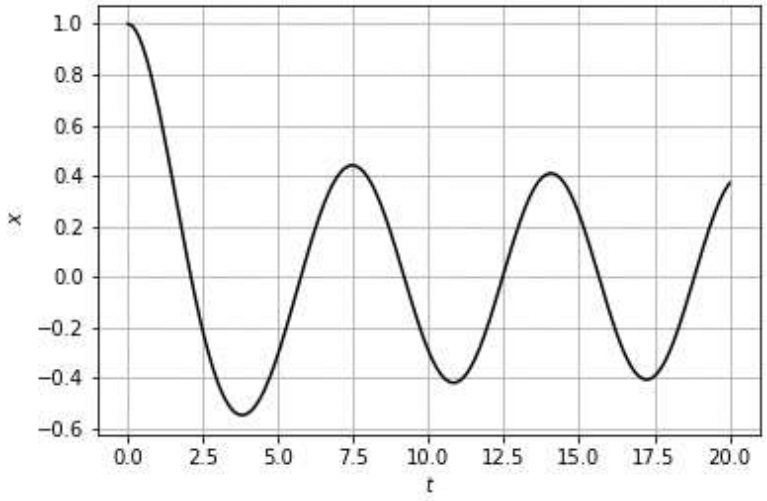
・例として，次の 2 階常微分方程式を$x(0) = 10, x'(0) = 0$の初期条件で解く。

$$\frac{d^2x}{dt^2} = -\frac{1}{2}\frac{dx}{dt} - x + \frac{1}{5}\cos t$$

・プログラムとグラフは以下のようになる。

```
import math
import matplotlib.pyplot as plt

def F1(x, v, t):
  return v
```

```
def F2(x, v, t):
  return -0.5*v - x + 0.2*math.cos(t)

t0 = 0.0
x0 = 1.0
v0 = 0.0
t1 = 20.0

N = 400
h = (t1 - t0)/N

T0 = []
X0 = []
V0 = []

t = t0
x = x0
v = v0

T0.append(t)
X0.append(x)
V0.append(v)
```




```
for i in range(1, N + 1):
  k1 = h*F1(x, v, t)
  m1 = h*F2(x, v, t)
  k2 = h*F1(x + 0.5*k1, v + 0.5*m1, t + 0.5*h)
  m2 = h*F2(x + 0.5*k1, v + 0.5*m1, t + 0.5*h)
  k3 = h*F1(x + 0.5*k2, v + 0.5*m2, t + 0.5*h)
  m3 = h*F2(x + 0.5*k2, v + 0.5*m2, t + 0.5*h)
  k4 = h*F1(x + k3, v + m3, t + h)
  m4 = h*F2(x + k3, v + m3, t + h)
  x = x + (k1 + 2*k2 + 2*k3 + k4)/6
  v = v + (m1 + 2*m2 + 2*m3 + m4)/6
  t = t + h
  T0.append(t)
  X0.append(x)
  V0.append(v)
```

```
plt.plot(T0,X0, '-k');
plt.xlabel('t', fontstyle = 'italic')
plt.ylabel('x', fontstyle = 'italic')
plt.grid(True)

plt.savefig('diff_eq_2.png')
```

【演習 15】

以下の 2 階常微分方程式を$x(0) = 1, x'(0) = 0$の初期条件で解き，グラフを出力せよ。この方程式は単振動と呼ばれる現象を表している。ω_0は正の定数であり，計算の際に適当な数値を設定すること。ω_0が変化するとグラフがどのように変化するか観察してみるのもよい。 （目次へ）

$$\frac{d^2x}{dt^2} + \omega_0^2 x = 0$$

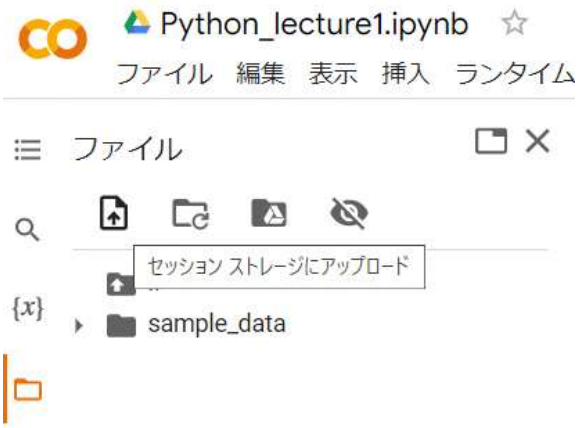
10.データのフィッティング

・実験データを一次関数で近似して，傾きなどをデータ解析に利用することがある。元のデータを一次関数などの理論曲線で近似することをフィッティングという。

・ここでは，あらかじめ以下のデータをテキストファイル `sampledata_fit.txt` に作成して保存し，Colab にアップロードする必要がある。

```
0     0.3   1.0
1     1.2   1.0
2     1.7   1.0
3     2.9   1.0
4     4.3   1.0
5     4.8   1.0
6     6.4   1.0
7     6.9   1.0
8     8.3   1.0
9     9.1   1.0
10    10.4  1.0
```

・sampledata_fit.txt をマイドライブに作成したのち，Colab 画面左「ファイル」の「セッションストレージにアップロード」を選択。



・マイドライブ内の sampledata_fit.txt を選択し，「開く」をクリック。

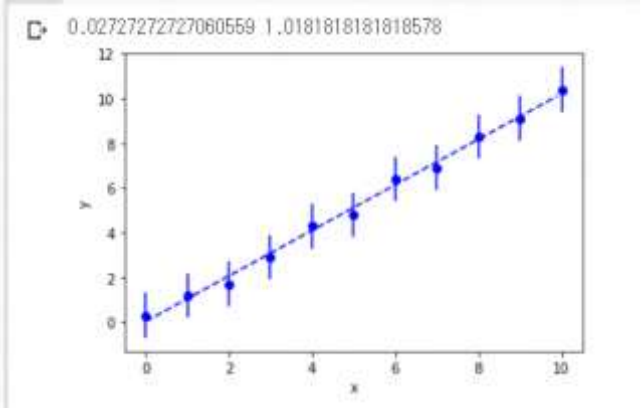
・sampledata_fit.txt 内の，1 列目は x 座標，2 列目は y 座標，3 列目は誤差である。

・このデータを一次関数でフィッティングするプログラムは以下のようになる。

```
import numpy as np

data = np.loadtxt('sampledata_fit.txt')
x_data = data.T[0]
y_data = data.T[1]
error_data = data.T[2]

def f(x, a, b):
  return a + b*x

from scipy.optimize import curve_fit
import math

par, cov = curve_fit(f, x_data, y_data, sigma = error_data)
import matplotlib.pyplot as plt

x_func = np.arange(0, 10, 0.1)#x の範囲指定
y_func = par[0] + par[1]*x_func#フィッティング曲線の式

plt.plot(x_func, y_func, '--b')#フィッティング曲線を表示
plt.errorbar(x_data, y_data, error_data, fmt='ob')#エラーバーと元データを表示

plt.xlabel('x')
```

```
plt.ylabel('y')

print(par[0], par[1])#a = par[0], b = par[1]
plt.savefig('fitting.png')
```

・このプログラムを実行すると，データを一次関数$y = a + bx$で近似したときのa, bの値と，エラーバー，データ，フィッティング曲線が出力される。



・もしデータが直線的とならない場合でも，対数を用いれば一次関数のフィッティングが応用できる。

・例えば実験データ(x, y)が次のような理論曲線で表されているとする。ここで，n, Cは未知の定数であり，これをフィッティングで求める。

$$y = Cx^n$$

・このとき，両辺の対数をとると次のように，$\log y$と$\log x$に関する一次式となる。

$$\log y = n(\log x) + \log C$$

・実験データ(x, y)を$(\log x, \log y)$に置き換えて一次式でフィッティングを行い，$n, \log C$の値を求めればよい。

・実験データyがxに対して指数関数的な変化をする場合がある。Cとaは定数である。このとき，以下の関係式でフィッティングできる。

$$y = Ca^x$$

・両辺の対数をとると

$$\log y = (\log a)x + \log C$$

・上式より，実験データ(x, y)を$(x, \log y)$に置き換えて一次式でフィッティングを行い，$\log a, \log C$の値を求めればよい。

・補足：対数

・３つの数a, b, c $(a > 0, b \neq 1)$の間に以下の関係が成り立っているとする。

$$a = b^c$$

・ここで，b^cはbをc回かけたものである。cは自然数でなくてもよく，例えば$2^{0.6}$という数字は 10 乗すれば$2^6 = 64$になる数のことであり，それは存在して，およそ 1.516 となる。

・この関係は，対数を表す記号logを用いてつぎのように表される。

$$c = \log_b a$$

・このとき，aを「真数」といい，cを「bを底とするaの対数」という。

・特にネイピア数$e := \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n \fallingdotseq 2.718281828\cdots$を底とする対数を自然対数といい，$\log a$のようにlog記号右下の底の表記はふつう省略される。

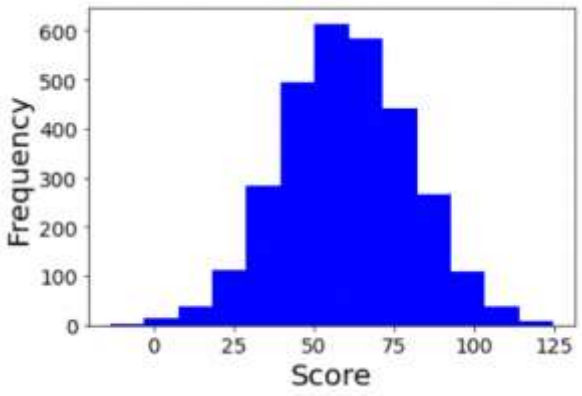
【演習 16】

以下のデータの組$(x, y,$誤差$)$が与えられたとする。このデータの理論曲線をフィッティングにより求めよ　（目次へ）

```
0     0.30  1.0
1     0.90  1.0
2     4.20  1.0
3     8.70  1.0
4     16.7  1.0
5     26.1  1.0
6     35.5  1.0
7     49.9  1.0
8     63.2  1.0
9     80.1  1.0
10    101.3 1.0
```

11. ヒストグラム

・データの統計処理を行う上で，ヒストグラムとして分布を可視化する作業が必須である。例として，平均 60，標準偏差 20 の乱数データを 3000 個生成し，それをヒストグラムとして表す。



・ヒストグラムと棒グラフは似て非なるものである。ヒストグラムでは，隣り合う縦棒は必ず接していなければならない。

・ヒストグラムの作成の際に，階級数を決める必要があるが，多すぎても少なすぎてもデータの特徴をうまく表すことができず，難しいところである。ここでは，階級数の目安としてスタージェスの公式を利用する。データ数がNのとき，階級数は次の式の最も近い整数値とする。

$$1 + \log_2 N$$

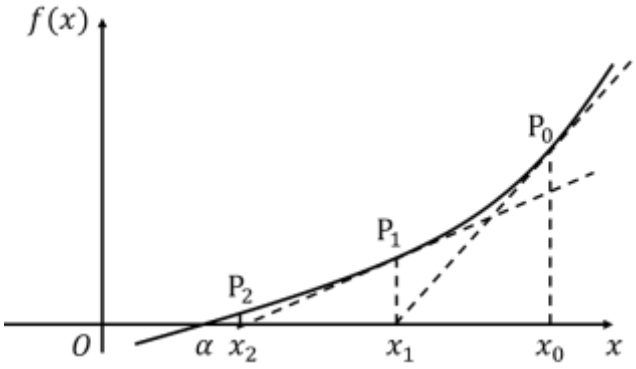
・例えば$N = 3000$のとき，$1 + \log_2 3000 \fallingdotseq 12.55$であるので，階級数を 13 に設定する。

・ヒストグラム作成のプログラム例を以下に示す。 （目次へ）

```
import numpy as np
import matplotlib.pyplot as plt

x = np.random.normal(60,20,3000)#平均 60，標準偏差 20 の乱数データを 3000 個生成

plt.hist(x, bins = 13, color = 'blue')#ヒストグラムを作成
plt.xlabel('Score', fontsize = 20)
plt.ylabel('Frequency', fontsize = 20)
plt.tick_params(labelsize = 14)
plt.savefig('histogram.png')
```

12. ニュートン・ラプソン法による方程式の求解

・方程式$f(x)=0$の解の公式が分からないような場合，ニュートン・ラプソン法と呼ばれる方法を用いて数値解を求めることができる。手順は次のとおりである。



・はじめにxの初期値x_0を任意に定める。

・点$\mathrm{P}_0\big(x_0, f(x_0)\big)$を通り，$\mathrm{P}_0$で$y=f(x)$のグラフに接する直線を求めると，次の式で表される。

$$y=f'(x_0)(x-x_0)+f(x_0)$$

・この直線がx軸と交わる点のx座標をx_1とすると

$$0=f'(x_0)(x_1-x_0)+f(x_0)$$

$$x_1=x_0-\frac{f(x_0)}{f'(x_0)}$$

・点$\big(x_1, f(x_1)\big)$を点P_1とし，P_1で$y=f(x)$のグラフに接する直線が，x軸と交わる点のx座標をx_2とする。

・以下，同様の手順で数列$x_0, x_1, x_2, \cdots$を定め，$|x_{n+1}-x_n|$が十分小さな値になったら，すなわちx_nがある値αに収束したら$x=\alpha$を方程式$f(x)=0$の解とする。

・例として$f(x)=x^2-x-1$として，$f(x)=0$を求めるプログラムを書くと次のようになる。

```
import math

x = float(input('初期値を入力してください'))
delta = 0.00001

def f(x):
  return x**2 - x - 1
```

```
def f_1(x):
  return 2*x - 1

while True:
  xp = x - f(x)/f_1(x)
  if abs(xp - x) < delta:
    break #収束したらwhileループを終了
  x = xp

print('x = ', xp)
```

・$x^2 - x - 1 = 0$を解の公式を用いて解くと

$$x = \frac{1 \pm \sqrt{5}}{2} \fallingdotseq 1.618, -0.618$$

となるが，初期値を 1 として上のプログラムを実行すると，1 に近いほうの 1.618 という解が得られる。

・もし初期値を -1 とすると，今度は -0.618 という解になる。

・このように，ニュートン・ラプソン法では初期値に近い解を 1 つだけ求めることができる。

【演習 17】

以下の方程式のすべての解を，ニュートン・ラプソン法を用いて求めよ。ここで，解の存在範囲と解の個数に関する次の定理を参考にしてもよい。 （目次へ）

$$x^3 - 6x^2 - 9x + 14 = 0$$

◆中間値の定理

「連続関数$f(x)$が区間$a \leqq x \leqq b$で連続であり，かつ$f(a) \cdot f(b) < 0$ならば方程式$f(x) = 0$は$a < x < b$の範囲に少なくとも一つの解をもつ。」

◆代数学の基本定理

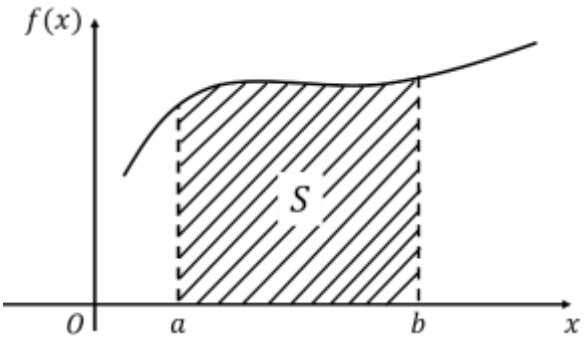
「n次方程式$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0 \ (a_n \neq 0)$は重根を含めて$n$個の解をもつ。」

13.シンプソン法による定積分の数値計算

・ここでは，区間$a \leqq x \leqq b$における関数$f(x)$の定積分Sの値を数値計算によって求める。Sは次の式で表される。記号$\int$はインテグラルと呼ばれる。

$$S = \int_a^b f(x)dx$$

・もし区間$a \leqq x \leqq b$で常に$f(x) \geqq 0$ならば，Sは直線$x = a, b$と$y = 0$，そして$y = f(x)$のグラフで囲まれた領域の面積に等しい。



・例

$$\begin{aligned} S &= \int_0^1 x dx \\ &= \left[\frac{1}{2}x^2\right]_0^1 \\ &= \frac{1}{2} \cdot 1^2 - \frac{1}{2} \cdot 0^2 \\ &= \frac{1}{2} \end{aligned}$$

・Sの値は，シンプソン法と呼ばれる方法を用いると次の式で求められる。

$$S = \frac{1}{3}h\left(f_0 + f_{2n} + 2\sum_{i=1}^{n-1} f_{2i} + 4\sum_{i=1}^{n} f_{2n-1}\right)$$

ここで

$$h = \frac{b-a}{2n}$$
$$f_i = f(a+ih)$$

・例として，以下の定積分の値を求める。

$$S = \int_0^{\frac{\pi}{2}} \sin x \, dx$$

・この式は数値計算に頼らなくとも以下のように理論的に求める（解析解を得る）ことができて，$S = 1$である。

$$\begin{aligned} S &= \int_0^{\frac{\pi}{2}} \sin x \, dx \\ &= [-\cos x]_0^{\frac{\pi}{2}} \\ &= -\left(\cos\frac{\pi}{2} - \cos 0\right) \\ &= -(0-1) \\ &= 1 \end{aligned}$$

・数値計算を行うプログラムはつぎのようになる。

```
import math

def f(x):
  return math.sin(x) #被積分関数の定義

a = 0 #積分区間の下端
b = math.pi / 2 #積分区間の上端
n = 20
N = 2 * n #積分区間を 2*20 等分に区切る
h = (b - a) / N #区切り幅

def f_i(i):
  return f(a + i * h)

S = 0

for i in range(2, N + 1, 2):
  S = S + (h / 3) * (f_i(i - 2) + 4 * f_i(i - 1) + f_i(i)) #シンプソンの公式
```

```
print('S = ', S)
```

・このプログラムを実行した結果は

```
1.0000000132143794
```
となり，理論値に近い値が得られている。

```
import math

def f(x):
  return math.sin(x) #被積分関数の定義

a = 0 #積分区間の下端
b = math.pi / 2 #積分区間の上端
n = 20
N = 2 * n #積分区間を2*20等分に区切る
h = (b - a) / N #区切り幅

def f_i(i):
  return f(a + i * h)

S = 0

for i in range(2, N + 1, 2):
  S = S + (h / 3) * (f_i(i - 2) + 4 * f_i(i - 1) + f_i(i)) #シンプソンの公式

print('S = ', S)
```

```
S =  1.0000000132143794
```

【演習 18】

次の定積分Sの値をシンプソン法を用いて求めよ。 （目次へ）

$$S = \int_0^{\pi} \sin(\sin x)\, dx$$

・関数の積分一覧（Cは定数）

$f(x)$	$\int f(x)dx$
$x^n \ \ (n \neq -1)$	$\frac{1}{n+1}x^{n+1} + C$
$\frac{1}{x}$	$\log\|x\| + C$
$\sin x$	$-\cos x + C$
$\cos x$	$\sin x + C$
$\tan x$	$-\log\|\cos x\| + C$
$\frac{1}{\cos^2 x}$	$\tan x + C$
$\frac{1}{\sin^2 x}$	$-\frac{1}{\tan x} + C$
$\frac{1}{\sin x}$	$\frac{1}{2}\log\left(\frac{1-\cos x}{1+\cos x}\right) + C$
$\frac{1}{\cos x}$	$\frac{1}{2}\log\left(\frac{1+\sin x}{1-\sin x}\right) + C$
$\log x$	$x\log x - x + C$
e^x	$e^x + C$
a^x	$\frac{a^x}{\log a} + C$

$\frac{1}{\sqrt{x^2+a^2}}$	$\log\left(x+\sqrt{x^2+a^2}\right)+C$
$\sqrt{x^2+a^2}$	$\frac{1}{2}\left(x\sqrt{x^2+a^2}+a^2\log\left(x+\sqrt{x^2+a^2}\right)\right)+C$

14．モンテカルロ法

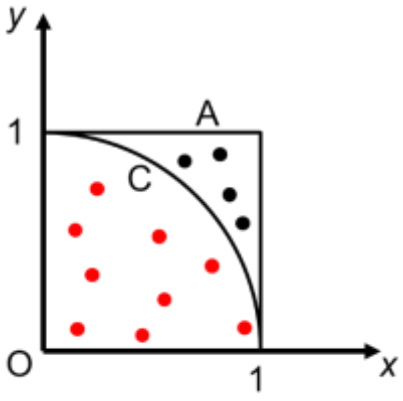
・シミュレーションを行う際に，モデルを定式化し，その式に従って現象を予測する方法がとられるが，定式化が困難な場合もある。そのときに効果的な手段が，乱数を用いるモンテカルロ法と呼ばれる方法である。ここでは一例として，モンテカルロ法を用いて円周率を求める。

・xy座標平面に，中心が原点で半径が 1 の円 C があるとする。平面上に位置が確率的に定まる点 P があるとして，そのx座標，y座標がともに 0 から 1 までの範囲をとる乱数（すなわち，でたらめな値をとり，予測ができない）とすると，点 P は$0 \leqq x \leqq 1, 0 \leqq y \leqq 1$で表される 1 辺が 1 の正方形 A 内に存在する。

・x座標，y座標を変更して点 P を何回も繰り返し打つと，点 P のx座標，y座標は乱数なので，点 P の集合は正方形 A 内のランダムな位置に分布する。

・点 P が円 C 内に存在する確率pは，正方形 A の面積 1 に対する，円 C の第一象限内の面積$\pi/4$の割合に等しい。すなわち，以下の関係が成り立つ。

$$p = \frac{\pi/4}{1}$$



・上の式より，円周率πはpを用いて次のように表される。

$$\pi = 4p$$

・乱数を用いて多数回の試行を行うことでpを求め，それを 4 倍することでπの近似値を推定するプログラムは以下のようになる。

```
import random #乱数を使うためのモジュール
import math

n = 100000 #総試行回数
count = 0
for i in range(n):
  x = random.random() #0 から 1 までの乱数を発生させる
  y = random.random()
  dist = math.sqrt(x*x + y*y) #点 P と原点との距離
  if dist <= 1: #もし点 P が円 C 内に存在するならば
    count = count + 1
p = count/n #4 分円の中に点が入る確率
pi = 4*p
print(pi)
```

例えば点 P を打つ試行を 100000 回繰り返すと，得られた円周率は 3.1438 となる。

【演習 19】

モンテカルロ法を用いて，半径 1 の 4 次元球$x^2+y^2+z^2+w^2 \leqq 1$の体積を求めよ。

同様にして半径 1 の 5 次元球，6 次元球…の体積を求め，どのように変化するか調べよ。

ここで，1 辺が 1 のd次元立方体の体積は 1 であり，d次元球においてすべての座標が正となる部分の割合は，球全体の$1/2^d$であることを用いてもよい。 （目次へ）

15. ユークリッドの互除法

・2 つの自然数$a,b\ (a>b)$が与えられたとき，aとbの最大公約数gを求める方法としてユークリッドの互除法を扱う。

・aをbで割ったときの商をq，余りをr $(0 \leqq r \leqq b-1)$とすると，次の式が成り立ち，qとrはただ一通りに決まる。

$$a = qb + r$$

・ここで，互いに素な自然数α, βを用いて$a = \alpha g, b = \beta g$と表されるから，上の式より

$$\alpha g = q\beta g + r$$
$$r = (\alpha - q\beta)g$$

・α, β, qは自然数なのでgはrの約数であり，かつ α–$q\beta$ とβは互いに素である。なぜなら，もし α–$q\beta$ とβが公約数を持つと仮定すると，qが一意に定まることと矛盾するからである。よって，gはbとrの最大公約数である。

・この事実より，2 つの自然数a, bの最大公約数を求める関数を$\gcd(a, b)$とすると，$\gcd(a, b) = \gcd(b, r)$という関係が成り立つ。この操作を余りが 0 になるまで繰り返すことで最大公約数を求める方法をユークリッドの互除法といい，単純に 2 つの自然数を素因数分解するよりも効率よく最大公約数を得ることができる。

・例えば，84 と 60 の最大公約数をユークリッドの互除法を用いて求めると次のようになる。

$$84 \div 60 = 1 \times 60 + 24$$
$$60 \div 24 = 2 \times 24 + 12$$
$$24 \div 12 = 2 \times 12 + 0$$

よって，最大公約数は 12 である。

・2 つの自然数a, bを入力して，その最大公約数を求めるプログラムを書くと次のようになる。

```
a = int(input('整数 a を入力してください:'))
b = int(input('整数 b ( < a)を入力してください:'))

def gcd(p,q):
  if (q == 0):
    return p
  else:
```

```
    r = p%q
    return gcd(q,r)

print(gcd(a,b))
```

・関数 gcd()の定義には，自分自身の関数 gcd()が含まれている。このような関数の定め方を再帰的な定義とよぶ。再帰的な定義を用いる利点としては，関数を簡潔に記述できることが挙げられるが，実行速度が遅くなるということに注意が必要である。ただし，ユークリッドの互除法は効率がよく，再帰的な定義でも問題なく動作する。

【演習 20】

次の整数の組(a)~(c)の最大公約数をそれぞれ求めよ。 （目次へ）

(a) 99041863, 69811

(b) 15032385529, 10737418235

(c) 1143767162282257, 470962949175047

16. フーリエ級数展開

・関数$f(x)$が周期$2L$の周期関数であるとする。すなわち，以下の関係が成り立つ。

$$f(x+2L)=f(x)$$

・このとき，$f(x)$は(A)式のように，三角関数の無限個の和（無限級数）として表すことができる。これをフーリエ級数展開という。

$$f(x)=\frac{a_0}{2}+\sum_{n=1}^{\infty}\left(a_n\cos\frac{n\pi x}{L}+b_n\sin\frac{n\pi x}{L}\right) \quad \text{(A)}$$

・ここで，∑は次の式のように，数列の和を表す記号である。

$$\sum_{k=1}^{n}a_k=a_1+a_2+a_3+\cdots+a_n$$

・参考に，自然数に関する和の公式を例として挙げる。

$$\sum_{k=1}^{n} k = \frac{1}{2}n(n+1)$$

$$\sum_{k=1}^{n} k^2 = \frac{1}{6}n(n+1)(2n+1)$$

$$\sum_{k=1}^{n} k^3 = \left(\frac{1}{2}n(n+1)\right)^2$$

・(A)式の両辺を$0 \leqq x \leqq 2L$の範囲で積分すると

$$\int_0^{2L} f(x)dx = \int_0^{2L} \frac{a_0}{2}dx + \int_0^{2L} \left(\sum_{n=1}^{\infty} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)\right)dx$$

$$\int_0^{2L} f(x)dx = a_0 L + \sum_{n=1}^{\infty} \int_0^{2L} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)dx$$

$$\int_0^{2L} f(x)dx = a_0 L$$

$$a_0 = \frac{1}{L}\int_0^{2L} f(x)dx$$

・(A)式の両辺に$\cos\frac{m\pi x}{L}$ $(m \neq 0)$を掛け，$0 \leqq x \leqq 2L$の範囲で積分すると

$$\int_0^{2L} f(x)\cos\frac{m\pi x}{L}dx = \int_0^{2L} \frac{a_0}{2}\cos\frac{m\pi x}{L}dx + \int_0^{2L} \left(\sum_{n=1}^{\infty} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)\right)\cos\frac{m\pi x}{L}dx$$

$$\int_0^{2L} f(x)\cos\frac{m\pi x}{L}dx = \sum_{n=1}^{\infty} \int_0^{2L} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)\cos\frac{m\pi x}{L}dx$$

$$\int_0^{2L} f(x)\cos\frac{m\pi x}{L}dx = a_m \int_0^{2L} \cos^2\frac{m\pi x}{L}dx$$

$$\int_0^{2L} f(x)\cos\frac{m\pi x}{L}dx = a_m L$$

$$a_m = \frac{1}{L}\int_0^{2L} f(x)\cos\frac{m\pi x}{L}dx$$

・この式変形の過程で，以下の事実を用いた。

n, mが自然数のとき

$$\int_0^{2L} \cos\left(\frac{n\pi}{L}x\right)\cos\left(\frac{m\pi}{L}x\right)dx = 0 \ \ (n \neq m)$$

$$\int_0^{2L} \sin\left(\frac{n\pi}{L}x\right)\sin\left(\frac{m\pi}{L}x\right)dx = 0 \ \ (n \neq m)$$

$$\int_0^{2L} \sin\left(\frac{n\pi}{L}x\right)\cos\left(\frac{m\pi}{L}x\right)dx = 0$$

$$\int_0^{2L} \cos^2\left(\frac{n\pi}{L}x\right)dx = \int_0^{2L} \sin^2\left(\frac{n\pi}{L}x\right)dx = L$$

・ここで，参考までに，kを定数として，$\cos^2 k\theta$の不定積分の求め方を示す。

$$\int \cos^2 k\theta \, d\theta = \frac{1}{2}\int (1+\cos 2k\theta)d\theta$$

$$= \frac{1}{2}\left(\theta + \frac{1}{2k}\sin 2k\theta\right) + C \ \ (C\text{は定数})$$

・この式変形では，余弦関数の 2 倍角の公式を用いた。

$$\cos 2\theta = 2\cos^2\theta - 1 = 1 - 2\sin^2\theta$$

・また，関数$f(x)$の原始関数の一つを$F(x)$とすると，以下の関係が成り立つ。

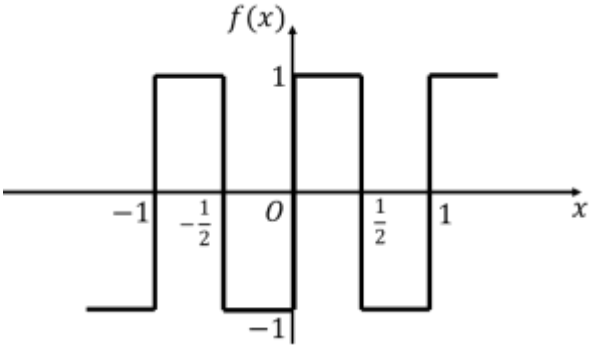
$$\int f(kx)\,dx = \frac{1}{k}F(kx)$$

・(A)式の両辺に$\sin\frac{m\pi x}{L}$ $(m \neq 0)$を掛け，$0 \leqq x \leqq 2L$の範囲で積分すると，同様にして

$$b_m = \frac{1}{L}\int_0^{2L} f(x)\sin\frac{m\pi x}{L}dx$$

・周期関数$f(x)$が与えられたときに，数列$\{a_n\},\{b_n\}$の一般項を求めることによって，フーリエ級数展開を行うことができる。

・ここでは，例として図のような周期 1 の矩形波（くけいは：四角い波形）をフーリエ級数展開で表す。



・数列$\{a_n\},\{b_n\}$の一般項は次のように求められる。

$$a_0 = 2\int_0^{\frac{1}{2}} dx - 2\int_{\frac{1}{2}}^{1} dx$$

$$= 0$$

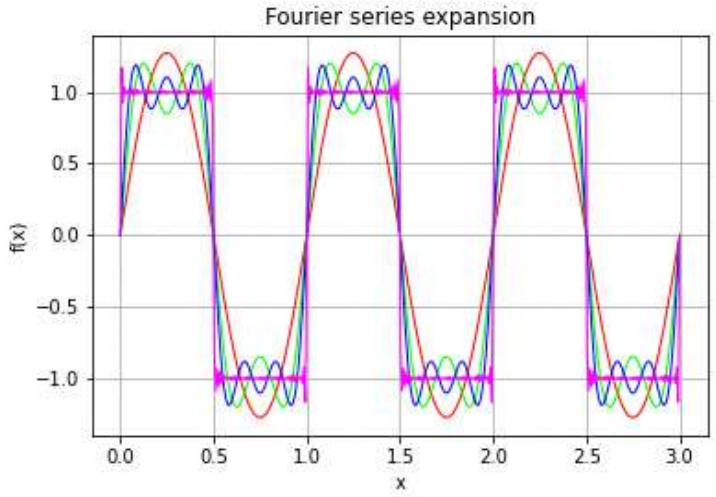
$$a_n = 2\int_0^{\frac{1}{2}} \cos 2n\pi x\, dx - 2\int_{\frac{1}{2}}^{1} \cos 2n\pi x\, dx$$

$$= 0$$

$$b_n = 2\int_0^{\frac{1}{2}} \sin 2n\pi x\, dx - 2\int_{\frac{1}{2}}^{1} \sin 2n\pi x\, dx$$

$$= \frac{2}{n\pi}(1-(-1)^n)$$

・このフーリエ級数を 1 次（橙），3 次（緑），5 次（青），そして 59 次（赤）までで打ち切ったグラフは次のようになる。



・プログラムは次のようになる。

```
import numpy as np
import math
import matplotlib.pyplot as plt

x = np.linspace(0, 3, 400) #0<x<3 の範囲を 400 等分する。

f1 = 4/(1*np.pi)*np.sin(2*np.pi*1*x)
f3 = 4/(1*np.pi)*np.sin(2*np.pi*1*x) + 4/(3*np.pi)*np.sin(2*np.pi*3*x)
f5 = 4/(1*np.pi)*np.sin(2*np.pi*1*x) + 4/(3*np.pi)*np.sin(2*np.pi*3*x) +
 4/(5*np.pi)*np.sin(2*np.pi*5*x)

N = 60
```

```
fn = np.zeros(shape = (x.shape[0],)) #配列 fn の中身をすべて 0 で初期化
for n in range(1, N, 2):
  fn = fn + 4/(n*np.pi)*np.sin(2*np.pi*n*x)

plt.title('Fourier series expansion')
plt.plot(x, f1, color = (1.0, 0.0, 0.0), linewidth = 1.0)
plt.plot(x, f3, color = (0.0, 1.0, 0.0), linewidth = 1.0)
plt.plot(x, f5, color = (0.0, 0.0, 1.0), linewidth = 1.0)
plt.plot(x, fn, color = (1.0, 0.0, 1.0), linewidth = 1.0)
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(True)
plt.savefig('fourier.png')
```

・フーリエ級数の展開の次数が増えるほど，もとの矩形波に近づいていくことが分かる。

・展開次数を増やしたとき，不連続点（矩形波の角など）で細かい振動が生じているが，これをギブス現象という。

【演習 21】（バーゼル問題）

関数$f(x)$を次の式で定義する。

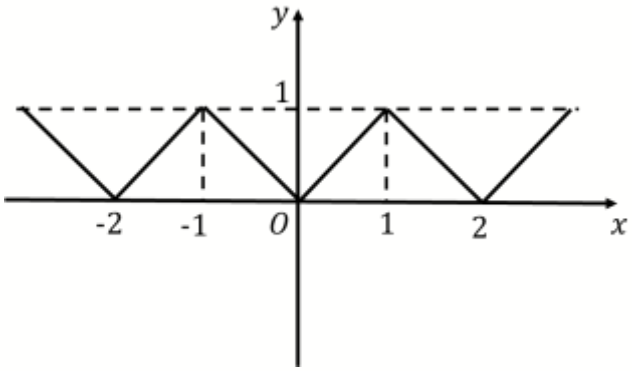
$$f(x) = \frac{1}{4}x^2 \ (-\pi \leqq x \leqq \pi)$$

$f(x)$をフーリエ級数展開することによって，以下の等式を証明せよ。

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

【演習 22】

次のグラフで表された三角波（周期 2）をフーリエ級数展開して，次数を増やしていったときグラフがどのように変化するのかを調べよ。 （目次へ）



17. フーリエ級数展開による熱伝導方程式の解法

・関数fが 2 つの変数x, yの関数として$f(x,y)$と表されるとき，一方の変数を固定したまま他方の変数のみでfを微分することを偏微分という。

・$f(x,y)$をxで偏微分することを次のように表す。

$$\frac{\partial f}{\partial x}$$

・記号∂は微分記号のdを丸めたもので，「ラウンドディー」，「デル」などと読む。

・$f(x,y)$をxで 2 回偏微分することを次のように表す。

$$\frac{\partial^2 f}{\partial x^2}$$

・例として，$f(x,y)$が次の式で定義される場合を考える。

$$f(x,y) = x^2 + xy + 2y^2$$

・このときの偏微分は次のようになる。

$$\frac{\partial f}{\partial x} = 2x + y$$
$$\frac{\partial f}{\partial y} = x + 4y$$
$$\frac{\partial^2 f}{\partial x^2} = 2$$
$$\frac{\partial^2 f}{\partial y^2} = 4$$

・ここで，位置xと時間tの関数$u(x,t)$の偏微分を含む次の方程式(B)について考える。この型の方程式を熱伝導方程式といい，一般に偏微分方程式と呼ばれるものの一種である。

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2} \tag{B}$$

・x,t,uに関する条件を次のように設定する。

$$0<x<L$$
$$t>0$$
$$u(0,t)=u(L,t)=0$$
$$u(x,0)=1$$

・$u(0,t)=u(L,t)=0$のように，端点の値を固定した条件設定を，ディリクレ境界条件と呼ぶ。一方，端点の微分係数の値を固定した条件設定を，ノイマン境界条件という。

・$u(x,t)$がxのみの関数$X(x)$とtのみの関数$T(t)$の積として$u(x,t)=X(x)T(t)$という形で表される と仮定する（変数分離型という）。このとき，(B)式は次のようになる。

$$X(x)T'(t)=X''(x)T(t)$$
$$\frac{T'(t)}{T(t)}=\frac{X''(x)}{X(x)}$$

・この式の左辺はtのみに依存し，右辺はxのみに依存する。これらが等しいということは，左辺，右辺ともに定数であることを意味する。この定数を，正の数μを用いて$-\mu$とおくと

$$\frac{T'(t)}{T(t)}=\frac{X''(x)}{X(x)}=-\mu$$

・よって，つぎの 2 つの微分方程式が得られる。

$$T'(t)=-\mu T(t)$$
$$X''(x)=-\mu X(x)$$

・$X(x)$の解は三角関数となる。微分することで確かめてもらいたい。

$$X(x)=c_1\cos\sqrt{\mu}\,x+c_2\sin\sqrt{\mu}x$$

・境界条件より，$X(0)=0$ かつ$X(L)=0$であるから

$$c_1=0$$

$$\sqrt{\mu}L = n\pi \ \left(n\text{は整数}\right)，\text{すなわち}\mu = \frac{n^2\pi^2}{L^2}$$

・したがって，$X(x)$の解は

$$X(x) = c_2 \sin\left(\frac{n\pi}{L}x\right)$$

・$T(t)$の解は指数関数となる。$\exp(x)$はe^xのことである。指数の表記が煩雑になるときに用いられる。

$$\begin{aligned} T(t) &= c_3 e^{-\mu t} \\ &= c_3 \exp\left(-\frac{n^2\pi^2}{L^2}t\right) \end{aligned}$$

・以上より，ある自然数nの値を選んだときの解は，aを定数として次の式で表される。

$$u(x,t) = a\exp\left(-\frac{n^2\pi^2}{L^2}t\right)\sin\left(\frac{n\pi}{L}x\right)$$

・ここで，nがどの自然数の値をとっても上の形の関数は，与えられた熱伝導方程式の解であり，異なるnの関数を足し合わせた関数も解となる。これを重ね合わせの原理という。

・重ね合わせの原理より，$u(x,t)$の解は以下のnに関する無限級数となる。

$$u(x,t) = \sum_{n=1}^{\infty} a_n \exp\left(-\frac{n^2\pi^2}{L^2}t\right)\sin\left(\frac{n\pi}{L}x\right)$$

・初期条件より，$u(x,0) = 1$であるから

$$1 = \sum_{n=1}^{\infty} a_n \sin\left(\frac{n\pi}{L}x\right)$$

・この式の両辺に$\sin\left(\frac{m\pi}{L}x\right)$をかけて$0 \leqq x \leqq L$の範囲で積分すると

$$\int_0^L \sin\left(\frac{m\pi}{L}x\right)dx = \int_0^L \left(\sum_{n=1}^{\infty} a_n \sin\left(\frac{n\pi}{L}x\right)\right)\sin\left(\frac{m\pi}{L}x\right)dx$$

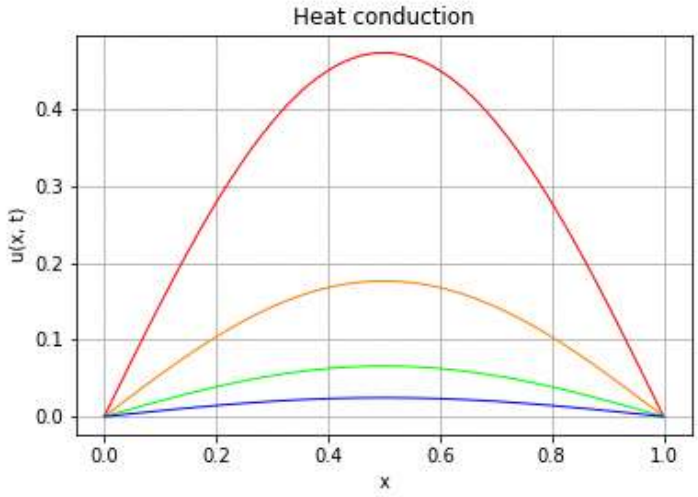
$$\frac{L}{m\pi}(1-(-1)^m) = \sum_{n=1}^{\infty}\int_0^L a_n \sin\left(\frac{n\pi}{L}x\right)\sin\left(\frac{m\pi}{L}x\right)dx$$

$$\frac{L}{m\pi}(1-(-1)^m) = \int_0^L a_m \sin^2\left(\frac{m\pi}{L}x\right)dx$$

$$\frac{L}{m\pi}(1-(-1)^m) = \frac{1}{2}La_m$$

$$a_m = \frac{2}{m\pi}(1-(-1)^m)$$

・ここでは，$L=1$として 59 次まで展開を行い，$t=0.1$（赤）から$t=0.4$（青）まで 0.1 ごとの結果を示すと次のグラフのようになる。



・プログラムは次のようになる。

```
import numpy as np
import math
import matplotlib.pyplot as plt

x = np.linspace(0, 1, 400) #0<x<1 の範囲を 400 等分する。

N = 60
t1 = 0.1
t2 = 0.2
t3 = 0.3
t4 = 0.4

fn1 = np.zeros(shape = (x.shape[0],))
for n in range(1, N, 2):
  fn1 = fn1 + 4/(n*np.pi)*np.exp(-((n*np.pi)**2)*t1)*np.sin(n*np.pi*x)
```

```
fn2 = np.zeros(shape = (x.shape[0],))
for n in range(1, N, 2):
  fn2 = fn2 + 4/(n*np.pi)*np.exp(-((n*np.pi)**2)*t2)*np.sin(n*np.pi*x)

fn3 = np.zeros(shape = (x.shape[0],))
for n in range(1, N, 2):
  fn3 = fn3 + 4/(n*np.pi)*np.exp(-((n*np.pi)**2)*t3)*np.sin(n*np.pi*x)

fn4 = np.zeros(shape = (x.shape[0],))
for n in range(1, N, 2):
  fn4 = fn4 + 4/(n*np.pi)*np.exp(-((n*np.pi)**2)*t4)*np.sin(n*np.pi*x)

plt.title('Heat conduction')
plt.plot(x, fn1, color = (1.0, 0.0, 0.0), linewidth = 1.0)
plt.plot(x, fn2, color = (1.0, 0.5, 0.0), linewidth = 1.0)
plt.plot(x, fn3, color = (0.0, 1.0, 0.0), linewidth = 1.0)
plt.plot(x, fn4, color = (0.0, 0.0, 1.0), linewidth = 1.0)
plt.xlabel('x')
plt.ylabel('u(x, t)')
plt.grid(True)
plt.savefig('heat.png')
```

【演習 23】

以下の熱伝導方程式（ノイマン境界条件）を解き，適当に展開次数を設定して，時間変化を調べよ。 （目次へ）

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$
$$0 < x < 1$$
$$t > 0$$
$$\frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(1,t) = 0$$
$$u(x,0) = 4x(1-x)$$

必要があれば，以下の部分積分の公式を用いよ。

$$\int_a^b f(x)g'(x)dx = [f(x)g(x)]_a^b - \int_a^b f'(x)g(x)dx$$

$$= \big(f(b)g(b) - f(a)g(a)\big) - \int_a^b f'(x)g(x)dx$$

18. 離散フーリエ変換

・虚数単位と呼ばれる数iを次の式で定義する。iは実数とは異なる数である。

$$i \coloneqq \sqrt{-1}$$

・2 つの実数aとbを用いて次の形で表される数zを複素数という。aを実部，bを虚部と呼ぶ。

$$z = a + bi$$

・$z = a + bi$に対して$a - bi$を$\bar{z}$と表し，これをzの共役複素数という。

・zと$\bar{z}$の積は次のように実数となる。

$$\begin{aligned} z\bar{z} &= (a + bi)(a - bi) \\ &= a^2 - b^2 i^2 \\ &= a^2 + b^2 \end{aligned}$$

・$\sqrt{z\bar{z}} = \sqrt{a^2 + b^2}$を$z$の絶対値といい，$|z|$と表す。例えば，$i$の絶対値は 1 である。

・複素数の演算も，実数と同様に，交換法則，結合法則，分配法則が成り立つ。演算の途中でi^2が現れたら，$i^2 = -1$と置き換えて計算を行えばよい。

・例

$$\begin{aligned} (2 + 3i)(1 + 4i) &= 2 + 8i + 3i + 12i^2 \\ &= 2 + 11i - 12 \\ &= -10 + 11i \end{aligned}$$

・複素数zの逆数は，分母と分子に$\bar{z}$をかけることで実部と虚部がわかる。

・例

$$\begin{aligned} \frac{1}{1 + i} &= \frac{1 - i}{(1 + i)(1 - i)} \\ &= \frac{1 - i}{1 - i^2} \\ &= \frac{1}{2} - \frac{1}{2}i \end{aligned}$$

・複素数には大小関係が定義されない。例えば，$3 > 2$，$i = i$だからといって$3 + i > 2 + i$とするのは誤りである。

・複素数を引数とする関数を複素関数という。xが実数のとき，例えばe^xであればその求め方はわかりやすい（$x = 2$であれば$e \fallingdotseq 2.718$を 2 回かけた数であり，$x = 0.1$であれば 10 乗してeになる数である，といった具合である）。

・しかし，複素関数の場合は求め方は自明ではない。例えばe^{1+2i}はどのような数なのかは，実数を引数にとる関数との類推では理解しにくい。

・複素関数の値は，多項式による展開（テイラー展開）で求められる。よく使う関数として，zが複素数であるときのe^z，sin$z$，cos$z$はそれぞれ次の式で定義される。

$$e^z \coloneqq \sum_{n=0}^{\infty} \frac{z^n}{n!}$$

$$\cos z \coloneqq \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} z^{2n}$$

$$\sin z \coloneqq \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} z^{2n+1}$$

・これらの式は，zが実数のときにも適用できる。

・記号!は階乗と呼ばれ，nを自然数とすると$n!$は次の式で定義される。

$$n! \coloneqq n(n-1)(n-2)\cdots 2 \cdot 1$$

・ここでe^zに$z = ix$を代入する。

$$e^{ix} = \sum_{n=0}^{\infty} \frac{i^n}{n!} x^n$$

$$= \sum_{n=0}^{\infty} \frac{i^{2n}}{(2n)!} x^{2n} + \sum_{n=0}^{\infty} \frac{i^{2n+1}}{(2n+1)!} x^{2n+1}$$

$$= \sum_{n=0}^{\infty} \frac{(-1)^{2n}}{(2n)!} x^{2n} + i \sum_{n=0}^{\infty} \frac{(-1)^{2n}}{(2n+1)!} x^{2n+1}$$

$$= \cos x + i \sin x$$

・指数関数と三角関数は複素数の範囲で考えると，オイラーの公式と呼ばれる次の関係が成り立つ。

$$e^{ix} = \cos x + i \sin x$$

・特に，$x = \pi$のときは次の関係が成り立つ。

$$e^{i\pi} + 1 = 0$$

・任意の周期関数はフーリエ級数展開によって三角関数の和に分解することができるが，周期関数以外の一般的な関数に対して拡張したものがフーリエ変換である。

・関数$x(t)$のフーリエ変換$X(f)$は次の式で定義される。

$$X(f) \coloneqq \int_{\infty}^{\infty} x(t) e^{-2\pi i f t} dt$$

・$X(f)$は，もとの関数$x(t)$に含まれている周波数fの成分の大きさを表している。

・以下，フーリエ変換を python で実行するにあたり，複素数の扱いについて述べる。

・python では複素数も扱うことができるが，虚数単位iに対応する数は python の表記では `j` となるため注意が必要である。

・虚数単位iは `1j` と表し，`j` のみではエラーとなる。

・複素数$a + bi$は `a + bj` と表し，`b` と `j` の間にスペースや積の記号`*`を入れてはいけない。

・複素数zを `z = a + bj` と python で定義したとき，実部aは `z.real`，虚部bは `z.imag` で取得できる。

・複素数zの共役複素数$\bar{z}$は `z.conjugate()`，絶対値$|z|$は `abs(z)` で取得できる。

・複素数を引数とする数学関数を扱う場合，`cmath` モジュールを使う。プログラムの最初に

```
import cmath
```

と記せばよい。例えば，$\sin(z)$の場合は

```
cmath.sin(z)
```

と表す。

・フーリエ変換をコンピュータで実行する際には，離散フーリエ変換という手法を用いる。

・$x(t)$ (tは整数)を時間変化する信号とすると，$x(t)$から周波数fの関数$X(f)$を得るような，以下の変換を離散フーリエ変換という。離散フーリエ変換によって，時間変化する信号がどの周波数成分を持っているかを知ることができる。

$$X(f) \coloneqq \sum_{t=0}^{N-1} x(t) \exp\left(-i \frac{2\pi f t}{N}\right)$$

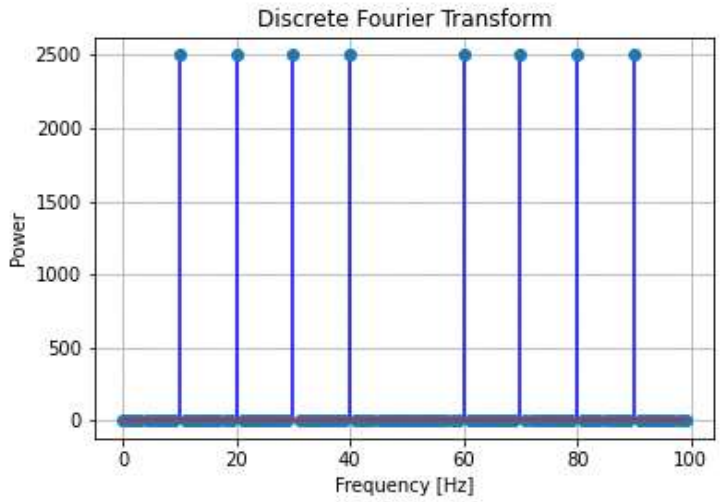
・$X(f)$は複素数の値をとるが，$|X(f)|^2$は周波数fの信号がどれだけのエネルギーを持っているのかを表しており，パワースペクトルと呼ばれる。

・任意の三角関数を設定し，離散フーリエ変換を行ってパワースペクトルを求めるプログラムを以下に示す。

```
%matplotlib inline

import functools
import matplotlib.pyplot as plt
import cmath
import random
import numpy as np

delta = 0.01 #サンプリング周波数 100Hz
t = np.arange(0, 1, delta) #0 から 1 まで 0.01 ステップの配列を作成
# 以下，適当な三角関数を定義
y1 = np.sin(2 * cmath.pi * 10 * t)
y2 = np.sin(2 * cmath.pi * 20 * t)
y3 = np.sin(2 * cmath.pi * 30 * t)
```

```
y4 = np.sin(2 * cmath.pi * 40 * t)

y = y1 + y2 + y3 + y4

#離散フーリエ変換の定義（パワースペクトル）
def dft_(f):
  n = len(f)
  Y = []
  for x in range(n):
    y = 0 + 0j
    for t in range(n):
      a = 2 * cmath.pi * t * x / n
      y = y + f[t] * cmath.exp(-1j * a)
    Y.append(y*y.conjugate())
  return Y

F = dft_(y)

plt.title('Discrete Fourier Transform')
plt.xlabel('Frequency [Hz]')
plt.ylabel('Power')
plt.grid(True)
plt.stem(F, linefmt = "blue") #グラフを縦棒にする
plt.savefig('DFT.png')
```

・この変換によって得られたパワースペクトルのグラフは以下のようになる。離散フーリエ変換を行う前の関数では 10 Hz, 20 Hz, 30 Hz, 40 Hz の周波数をもっていたが，グラフのピークも同じ周波数の位置にあることがわかる。



・このプログラムでは，三角関数の信号を 0.01 秒間隔で取り出している（サンプリングをしている）が，この間隔の逆数をサンプリング周波数といい，f_s [Hz]と表す。

・サンプリング周波数がf_s[Hz]のとき，離散フーリエ変換後のデータで意味を持つのは$f_s/2$ [Hz]までであり，この周波数をナイキスト周波数という。上の例では，$f_s = 100$ [Hz]であるためナイキスト周波数は 50 Hz であり，これより高周波数の領域にあるピークはデータ解析の際に意味を持たないので注意が必要である。

【演習 24】

時間の関数$f(t)$を次の式で定義する。

$$f(t) = e^{-1000t^2}$$

$0 \leqq t \leqq 1$の範囲で 0.01 秒間隔でサンプリングをし，その信号を離散フーリエ変換することによってパワースペクトルを図示せよ。 （目次へ）

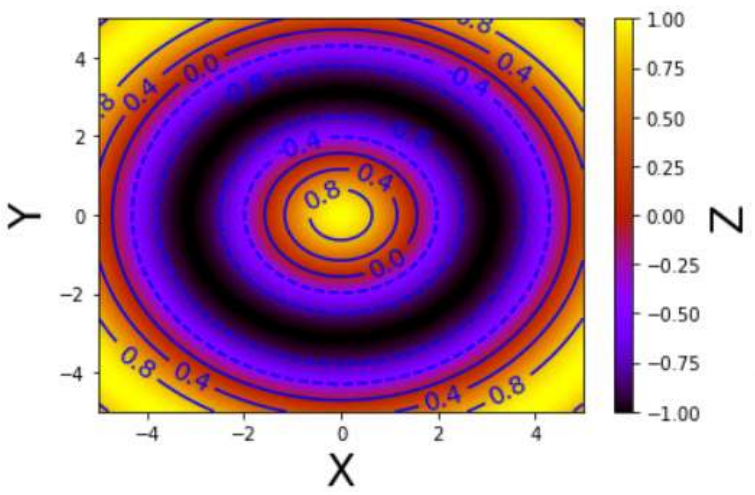
【補充演習 1】（等高線プロット）

以下のプログラムは，一例として関数

$z(x,y) = \cos\left(\sqrt{x^2+y^2}\right)$の等高線図を作成

するものである。

関数zを変えてみて，出力を確認せよ。




```
import numpy as np

import matplotlib.pyplot as plt

x = np.arange(-5, 5, 0.01) #-5 から 5 まで 0.01 間隔
y = np.arange(-5, 5, 0.01)

X, Y = np.meshgrid(x, y) #グリッドの作成
Z = np.cos(np.sqrt(X**2 + Y**2))

cont = plt.contour(X, Y, Z, 5, vmin = -1, vmax = 1, colors = 'blue') #等高線の作成
cont.clabel(fmt = '%1.1f', fontsize = 14) #小数点以下 1 桁まで表示でフォントサイズは 14 ポイント

plt.xlabel('X', fontsize = 24)
plt.ylabel('Y', fontsize = 24)

plt.pcolormesh(X, Y, Z, cmap = 'gnuplot')
pp = plt.colorbar(orientation = 'vertical')
pp.set_label('Z', fontsize = 24)

plt.show()
```

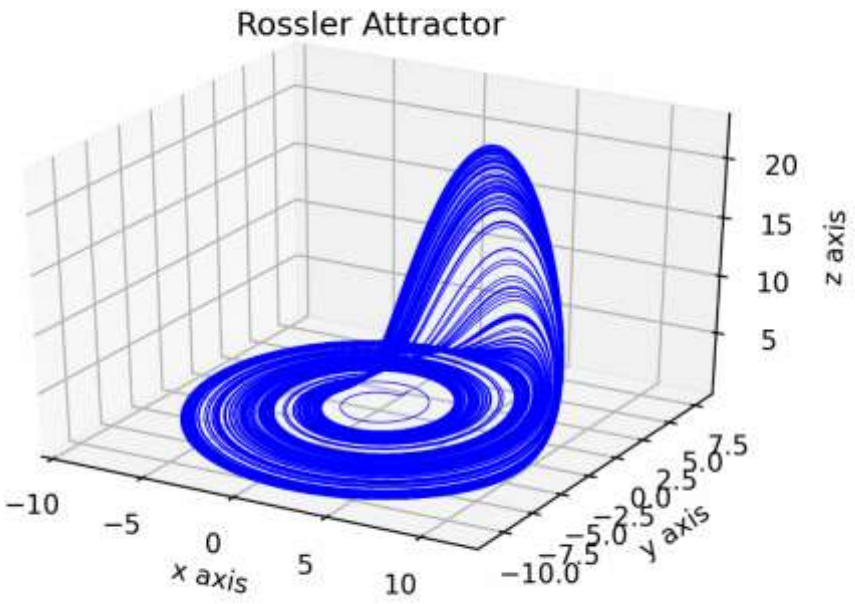
【補充演習 2-1】（レスラーアトラクター）

以下のx, y, zに関する連立微分方程式をレスラー方程式という。

$$\frac{dx}{dt} = -y - z$$
$$\frac{dy}{dt} = x + ay$$
$$\frac{dz}{dt} = b + xz - cz$$

この微分方程式のふるまいは，3 つの定数a, b, cの組み合わせがわずかに異なるだけで，予測できないような変化をする。一例として，$a = 0.2, b = 0.2, c = 5.7$のときの$(x, y, z)$は次の軌跡を描く。これをレスラーアトラクターという。

パラメータa, b, cをいくつか試してみて，レスラー方程式のふるまいを観察せよ。



以下にプログラムの一例を示す。

```
import numpy as np
import matplotlib.pyplot as plt
```

```
from mpl_toolkits.mplot3d import axes3d, Axes3D

def rossler(x, y, z, a = 0.2, b = 0.2, c = 5.7):
  dx = -y - z
  dy = x + a*y
  dz = b + x*z - c*z
  return dx, dy, dz

dt = 0.001
num_steps = 1000000

xs = np.empty(num_steps + 1)
ys = np.empty(num_steps + 1)
zs = np.empty(num_steps + 1)

xs[0], ys[0], zs[0] = (1., 1., 1)

for i in range(num_steps):
 dx, dy, dz = rossler(xs[i], ys[i], zs[i])
 xs[i + 1] = xs[i] + (dx*dt)
 ys[i + 1] = ys[i] + (dy*dt)
 zs[i + 1] = zs[i] + (dz*dt)

ax = plt.figure(figsize = (6,4), dpi = 300).add_subplot(projection = '3d
')
ax.plot(xs, ys, zs, lw = 0.5, color = 'blue')
ax.set_xlabel("x axis", size = 10)
ax.set_ylabel("y axis", size = 10)
ax.set_zlabel("z axis", size = 10)
ax.set_title("Rossler Attractor", size = 12)

plt.savefig('rossler_attractor.png')
```

【補充演習 2 - 2】（ローレンツアトラクター）

以下のx, y, zに関する連立微分方程式をローレンツ方程式という。

$$\frac{dx}{dt} = -sx + sy$$

$$\frac{dy}{dt} = -xz + rx - y$$

$$\frac{dz}{dt} = xy - bz$$

$s = 10, r = 28, b = 8/3$ のときの(x, y, z)の軌跡をグラフで表せ。この軌跡をローレンツアトラクターという。

【補充演習 3 - 1】（シェルピンスキーのギャスケット）

以下のプログラムを実行し，何が表示されるか確認せよ。このように，図形の一部に全体と同じ図形が現れるものをフラクタルという。

```
import numpy as np
import matplotlib.pyplot as plt
from random import random, randrange
from math import floor

fig = plt.figure()
anim = []

a = np.zeros([3])
b = np.zeros([3])

X1, Y1 = 0, 1.73
X2, Y2 = 1, 0
X3, Y3 = -1, 0

a[0], a[1], a[2] = X1, X2, X3
b[0], b[1], b[2] = Y1, Y2, Y3

plt.plot(a[0], b[0], 'ob', markersize = 1)
plt.plot(a[1], b[1], 'ob', markersize = 1)
plt.plot(a[2], b[2], 'ob', markersize = 1)

x = 2.5
y = 2.5

for i in range(10000):
```

```
  n = randrange(3)
  x = (x + a[n])/2
  y = (y + b[n])/2
  plt.plot(x, y, 'ob', markersize = 1)

plt.savefig('sierpinski_gasket.png')
```

【補充演習 3 - 2】（バーンズリーのシダ）

以下のプログラムを実行し，何が表示されるか確認せよ。

```
import matplotlib.pyplot as plt
import numpy as np
import random

x, y = 0, 0
N = 200000

X_list = []
Y_list = []

X_list.append(0)
Y_list.append(0)

for i in range(N):
  s = random.randint(1, 100)
  if s <= 10:
    X = 0.0
    Y = 0.16*y

  elif 10 < s <= 20:
    X = 0.2 * x - 0.26 * y
    Y = 0.23 * x + 0.22 * y + 1.6

  elif 20 < s <= 30:
    X = -0.15 * x + 0.28 * y
    Y = 0.26 * x + 0.24 * y + 0.44

  else:
    X = 0.85 * x + 0.04 * y
```

```
    Y = -0.04 * x + 0.85 * y + 1.6

  X_list.append(X)
  Y_list.append(Y)

  x = X
  y = Y

plt.scatter(X_list, Y_list, s = 0.01, c = 'green')
plt.xlim(-5, 5)
plt.ylim(0, 10)
plt.savefig('barnsley_fern.png')
```

（目次へ）

あとがき

多くの方々に支えられながら本テキストは完成いたしました。プログラムの作成に際しての参考 URL の一部を記載いたしますが，この他にも多くの方々のプログラムを参考にさせていただきました。感謝いたします。ここで触れた内容は，理工学の分野では必須となる知識ですが，焦る必要はありません。本当に必要な知識はゆっくりと身につくものです。もし研究で困ったときは，このテキストを読み返してみてください。何か手助けになるヒントが見つかれば幸いです。

参考 URL
・「Python ドキュメント」
https://docs.python.org/ja/3/index.html

・「Python プログラミング入門」
https://utokyo-ipp.github.io/

・「Python で学ぶコンピュテーショナル・シンキングとデータ科学（2022 年度版）」
https://wagtail.cds.tohoku.ac.jp/coda/python/

演習問題解答

【演習 1】

(1)　3.5
(2)　0.2
(3)　1
(4)　−9
(5)　9
(6)　0.125
(7)　1
(8)　1
(9)　True
(10)　False
(11)　True
(12)　False

【演習 2】

```
upper = 3
lower = 5
height = 7
area = (upper + lower) * height / 2
print(area)
```

（答）28

【演習 3】

```
sum = 0
for i in range(1, 101, 1):
  sum = sum + 1 / (i ** 2)
print(sum)
```

（答）1.635

【演習 4】

```
sum = 0
```

```
i = 1
while (i < 1000):
  sum = sum + i
  i = i + 2
print(sum)
```

（答）250000

【演習 5】

```
i = 0
while (i < 100):
  if i % 3 == 0:
    print(i)
  i = i + 1
```

【演習 6】

```
str_n = input("自然数を一つ入力してください")
n = int(str_n)
i = 1
while (i < 101):
  if i % n == 0:
    print(i)
  i = i + 1
```

【演習 7】

```
def dif(a,b):
  if a >= b:
    c = a - b
  else:
    c = b - a
  return c
```

【演習 8】

```
def sum(n):
  sigma = 0
  for i in range(1, n + 1, 1):
```

```
    sigma = sigma + i
  return sigma
```

【演習 9】

```
import math
g = 9.8
def L(v, theta):
  length = (v ** 2) * math.sin(2 * math.radians(theta)) / g
  return length
```

【演習 10】

```
import math
g = 9.8
v = 10.0
theta  = math.pi / 4.0
start = 0
stop = 3
step = 0.1

def x(t):
  horizontal = (v * math.cos(theta)) * t
  return horizontal

def y(t):
  vertical = (v * math.sin(theta)) * t - (1.0/2.0) * g * (t ** 2)
  return vertical

with open('projectile.txt', 'w') as f:
  t = start
  while (t < stop + step):
    print(t, '¥t', x(t), '¥t', y(t), file = f)
    t = t + step
```

【演習 11】

```
import math

g = 9.8
```

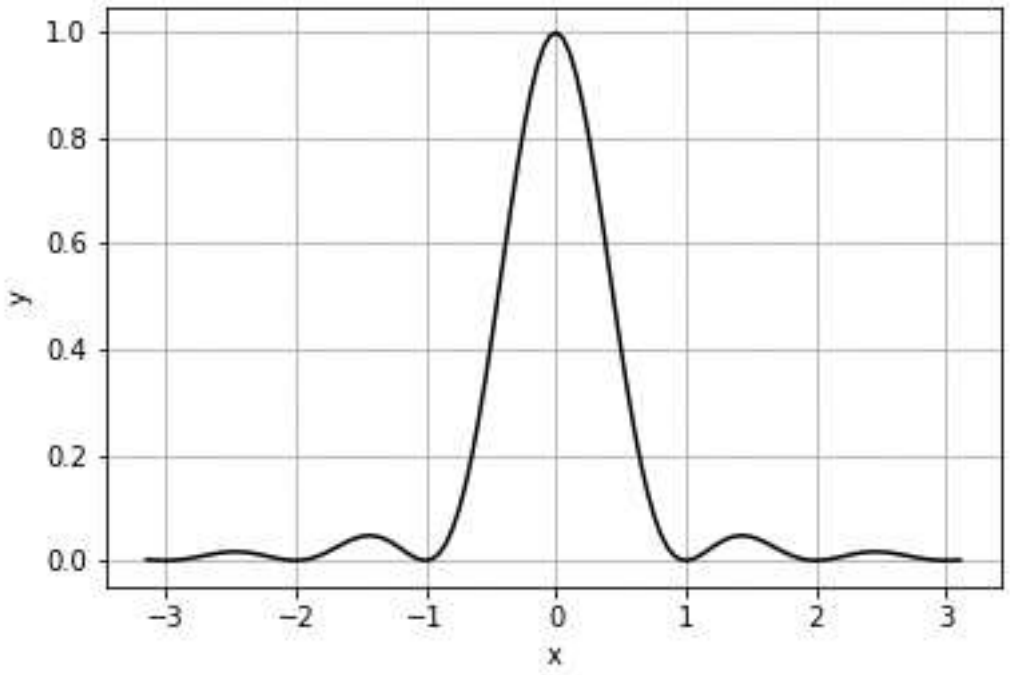
```
start = 1
stop = 90
step = 1

def T(theta):
  vy = 10 * math.sin(math.radians(theta))
  time = (vy + math.sqrt((vy ** 2) + 4.0 * g)) / g
  return time

def L(theta):
  vx = 10 * math.cos(math.radians(theta))
  horizontal = vx * T(theta)
  return horizontal

with open('practice_11.txt', 'w') as f:
  theta = start
  while (theta < stop):
    print(theta, '¥t', L(theta), file = f)
    theta = theta + step

import csv

with open('practice_11.txt') as f:
  data = csv.reader(f, delimiter = '¥t')
  A = [row for row in data]

max = 0
index = 0
for i in range(0, len(A), 1):
  if max < float(A[i][1]):
    max = float(A[i][1])
    index = i + 1
print('投射角', index, '°のときに最大値', max, 'mをとる')
```

（答）投射角40°のときに最大値12.04 mをとる。

【演習 12】

```
import math
```

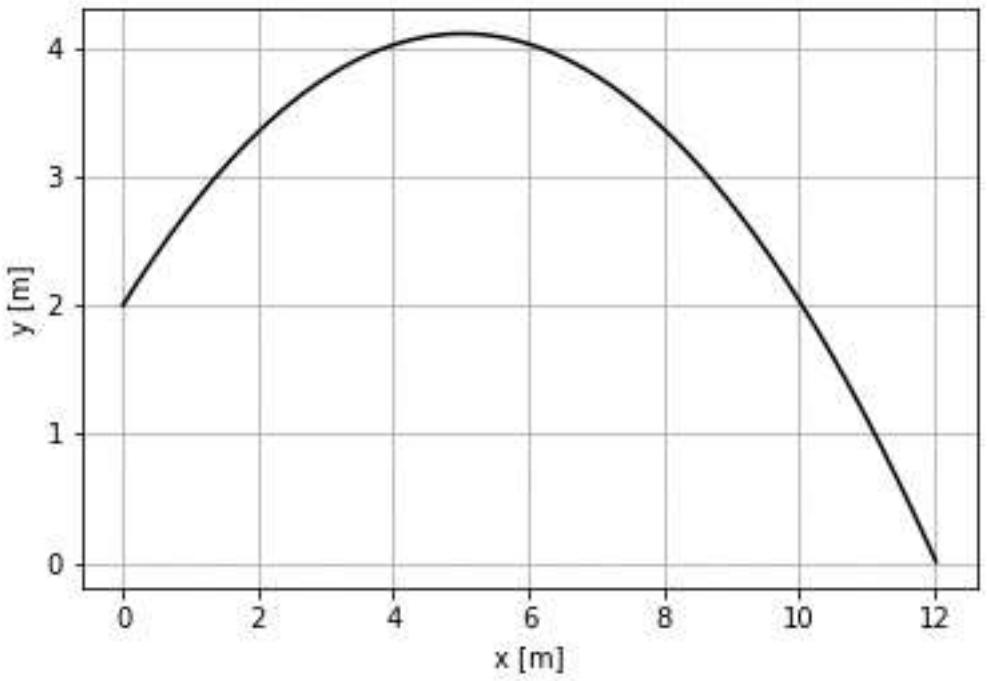
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

x = np.arange(-np.pi, np.pi, 0.05)
def I(x):
  a = np.sin(np.pi * x)
  b = np.pi * x
  return (a / b)**2

plt.plot(x, I(x), '-k')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.savefig('practice_12.png')
```



【演習 13】

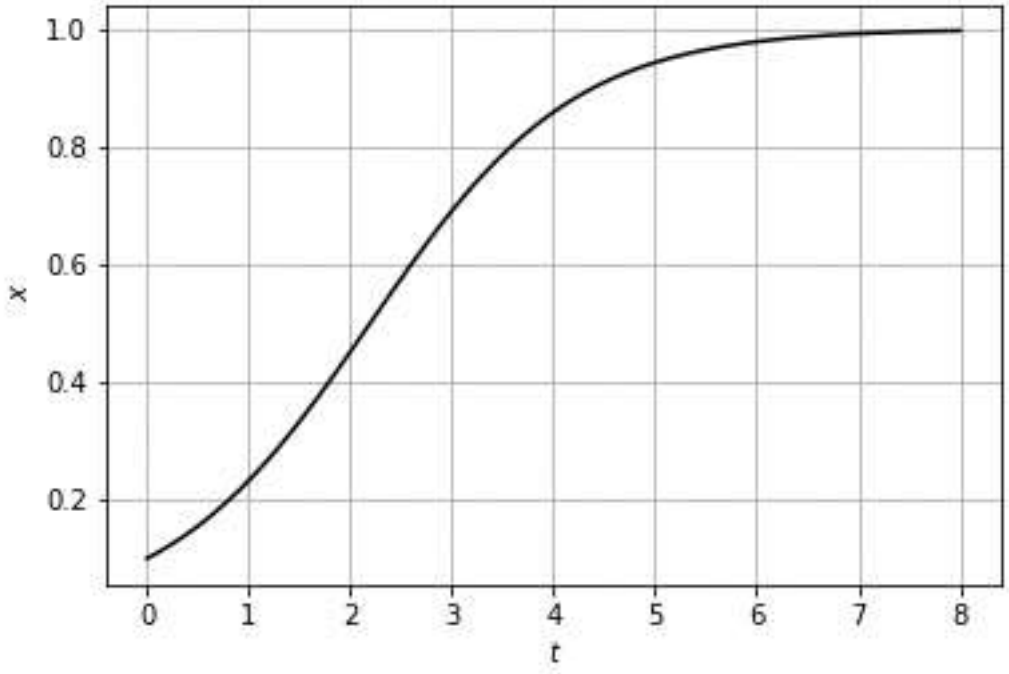
```
import math
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
g = 9.8
v = 10.0
theta = 40
h = 2.0
tan = math.tan(math.radians(theta))
ini = 0
fin = 12.04
step = 0.01

def y(x):
  return h + tan * x - (1.0 / 2.0) * g * (1 + tan **2) * (x ** 2) / (v *
* 2)

x = np.arange(ini, fin, step)
plt.plot(x, y(x), '-k')
plt.xlabel('x [m]')
plt.ylabel('y [m]')
plt.grid(True)
plt.savefig('practice_13.png')
```



【演習 14】

```
import math
import matplotlib.pyplot as plt

def f(x,t):
  return (1 - x) * x


t0 = 0
x0 = 0.1

t1 = 8

N = 100

h = (t1 - t0) / N

t = t0
x = x0

T = []
X = []
T.append(t)
X.append(x)


for i in range(1, N + 1):
  k1 = h * f(x,t)
  k2 = h * f(x + 0.5 * k1, t + 0.5 * h)
  k3 = h * f(x + 0.5 * k2, t + 0.5 * h)
  k4 = h * f(x + k3, t + h)
  x = x + (k1 + 2 * k2 + 2 * k3 + k4) / 6
  t = t + h
  T.append(t)
  X.append(x)

plt.plot(T,X, '-k');
plt.xlabel('t', fontstyle = 'italic')
plt.ylabel('x', fontstyle = 'italic')
plt.grid(True)

plt.savefig('practice_14.png')
```



【演習 15】

$\omega_0 = 1.0$としたときのプログラムとグラフは次のようになる。

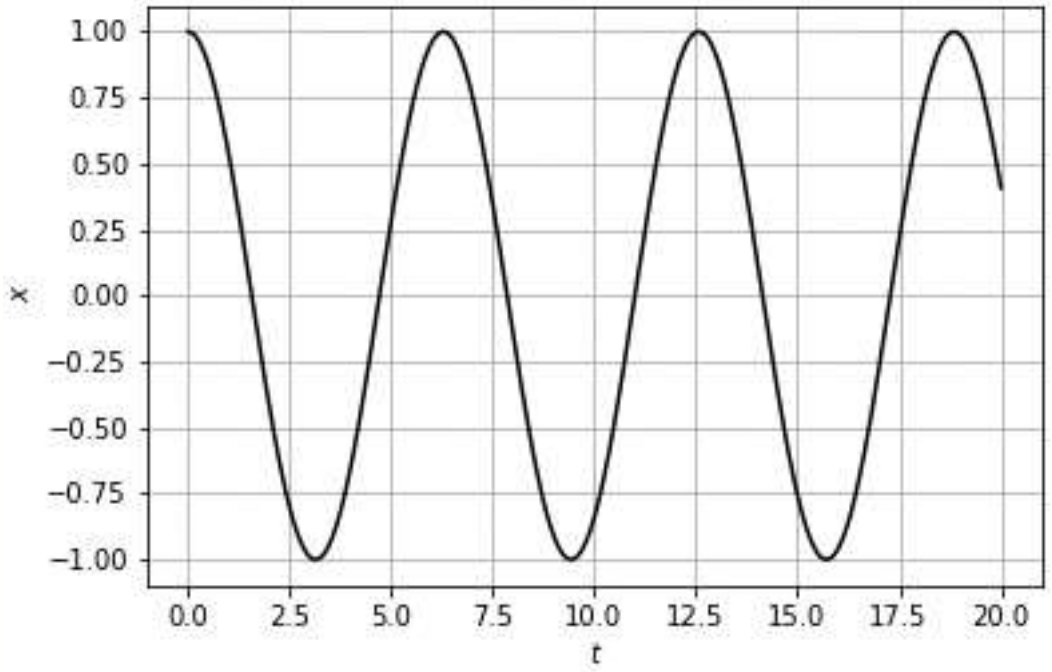
このとき，グラフは正弦曲線となり，その周期をTとすると

$$T = \frac{2\pi}{\omega_0} \fallingdotseq 6.28$$

```
import math
import matplotlib.pyplot as plt

omega0 = 1.0

def F1(x, v, t):
  return v

def F2(x, v, t):
  return   -(omega0 ** 2) * x

t0 = 0.0
```

```
x0 = 1.0
v0 = 0.0
t1 = 20.0

N = 400
h = (t1 - t0) / N

T0 = []
X0 = []
V0 = []

t = t0
x = x0
v = v0

T0.append(t)
X0.append(x)
V0.append(v)

for i in range(1, N + 1):
  k1 = h * F1(x, v, t)
  m1 = h * F2(x, v, t)
  k2 = h * F1(x + 0.5 * k1, v + 0.5 * m1, t + 0.5 * h)
  m2 = h * F2(x + 0.5 * k1, v + 0.5 * m1, t + 0.5 * h)
  k3 = h * F1(x + 0.5 * k2, v + 0.5 * m2, t + 0.5 * h)
  m3 = h * F2(x + 0.5 * k2, v + 0.5 * m2, t + 0.5 * h)
  k4 = h * F1(x + k3, v + m3, t + h)
  m4 = h * F2(x + k3, v + m3, t + h)
  x = x + (k1 + 2 * k2 + 2 * k3 + k4) / 6
  v = v + (m1 + 2 * m2 + 2 * m3 + m4) / 6
  t = t + h
  T0.append(t)
  X0.append(x)
  V0.append(v)

plt.plot(T0,X0, '-k');
plt.xlabel('t', fontstyle = 'italic')
plt.ylabel('x', fontstyle = 'italic')
plt.grid(True)
```

```
plt.savefig('practice_15.png')
```



【演習 16】

理論曲線を以下の形と仮定する。

$$y = Cx^n$$

正のxの値のデータに対して，x，yの対数をとると以下のデータとなる。

x	log$x$	log$y$
1	0	-0.105
2	0.693	1.435
3	1.099	2.163
4	1.386	2.815
5	1.609	3.262
6	1.792	3.570
7	1.946	3.910
8	2.079	4.146
9	2.197	4.383
10	2.303	4.618

$(\log x, \log y)$のデータを practice_16_data.txt に保存してフィッティングを行い，nと$\log C$の値を求めると$n = 2.03$，$\log C = -0.04$であるから，理論曲線は

$$y = 0.96x^{2.03}$$

プログラム例

```
import numpy as np
import matplotlib.pyplot as plt

data = np.loadtxt('practice_16_data.txt')
x_data = data.T[0]
y_data = data.T[1]

def f(x, a, b):
  return a + b * x

from scipy.optimize import curve_fit
import math

par, cov = curve_fit(f, x_data, y_data)
import matplotlib.pyplot as plt

x_func = np.arange(0.0, 3.0, 0.1)
y_func = par[0] + par[1]*x_func

plt.plot(x_func, y_func, '--b')
plt.plot(data[:,0], data[:,1], 'ob')

plt.xlabel('x')
plt.ylabel('y')

print(par[0], par[1])
plt.savefig('practice_16_data.png')
```

【演習 17】

関数$f(x)$を次の式で定義する。

$$f(x) = x^3 - 6x^2 - 9x + 14$$

xに適当な値を代入して符号を調べると

$$f(-3) = -40 < 0$$
$$f(0) = 14 > 0$$
$$f(2) = -20 < 0$$
$$f(10) = 324 > 0$$

$f(x) = 0$の 3 つの解をx_1，x_2，x_3 $(x_1 < x_2 < x_3)$とすると，それぞれ次の範囲内にある。

$$-3 < x_1 < 0$$
$$0 < x_2 < 2$$
$$2 < x_3 < 10$$

ニュートン・ラプソン法を用いて，初期値をそれぞれ次のように設定すると，解が得られる。

初期値-3のとき$x_1 = -2$

初期値2のとき$x_2 = 1$

初期値10のとき$x_3 = 7$

ニュートン・ラプソン法は以下のプログラムで実行できる。

```
import math

x = float(input('初期値を入力してください:'))
delta = 0.00001

def f(x):
  return x ** 3 - 6 * (x ** 2) - 9 * x + 14

def f_1(x):
  return 3 * (x ** 2) - 12 * x - 9

while True:
  xp = x - f(x)/f_1(x)
  if abs(xp - x) < delta:
    break
  x = xp

print('x = ', xp)
```

【演習 18】

```
import math

def f(x):
  return math.sin(math.sin(x))

a = 0
b = math.pi
n = 20
N = 2 * n
h = (b - a) / N

def f_i(i):
  return f(a + i * h)

S = 0

for i in range(2, N + 1, 2):
  S = S + (h / 3) * (f_i(i - 2) + 4 * f_i(i - 1) + f_i(i))

print('S = ', S)
```

（答）1.7865

【演習 19】

半径 1 の 4 次元球$x^2 + y^2 + z^2 + w^2 \leqq 1$の体積を求めるプログラムは次のようになる。

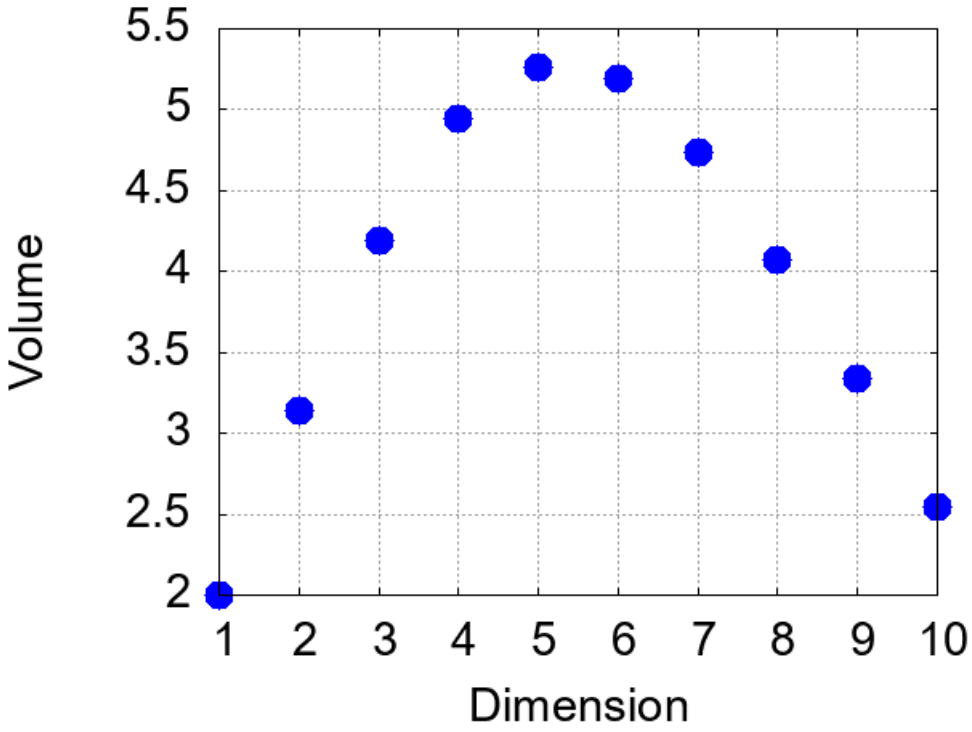
```
import random
import math

n = 1000000
d = 4
count = 0
for i in range(n):
  x = random.random()
  y = random.random()
```

```
  z = random.random()
  w = random.random()
  dist = math.sqrt(x * x + y * y + z * z + w * w)
  if dist <= 1:
    count = count + 1
V = (2 ** d) * (count / n)
print(V)
```

（答）半径 1 の四次元単位球の体積は 4.943

同様にして，半径を 1 に固定して 10 次元球までの体積を求め，グラフで表すと

次元	体積
1	2
2	3.14
3	4.189
4	4.943
5	5.265
6	5.193
7	4.738
8	4.074
9	3.335
10	2.545



【演習 20】

(a) 9973
(b) 2147483647
(c) 67280421310721

これらはいずれも素数である。

プログラム例

```
a = int(input('整数 a を入力してください:'))
b = int(input('整数 b ( < a)を入力してください:'))
1
def gcd(p,q):
  if (q == 0):
    return p
  else:
    r = p%q
    return gcd(q,r)

print(gcd(a,b))
```

【演習 21】

$f(x)$を以下の式で定義する。

$$f(x) = \frac{1}{4}x^2 \ \ (-\pi \leqq x \leqq \pi)$$

このとき，$f(x)$のフーリエ級数展開における係数を求めると

$$\begin{aligned} a_0 &= \frac{1}{\pi}\int_{-\pi}^{\pi} \frac{1}{4}x^2\, dx \\ &= \frac{\pi^2}{6} \\ a_n &= \frac{1}{\pi}\int_{-\pi}^{\pi} \frac{1}{4}x^2 \cos nx\, dx \\ &= \frac{(-1)^n}{n^2} \\ b_n &= 0 \end{aligned}$$

したがって，$f(x)$のフーリエ級数展開は次の式で表される。

$$\frac{1}{4}x^2 = \frac{\pi^2}{12} + \sum_{n=1}^{\infty} \frac{(-1)^n}{n^2}\cos nx$$

両辺に$x = \pi$を代入すると

$$\frac{\pi^2}{4} = \frac{\pi^2}{12} + \sum_{n=1}^{\infty} \frac{1}{n^2}$$

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

【演習 22】

$f(x)$は次の式で表される。

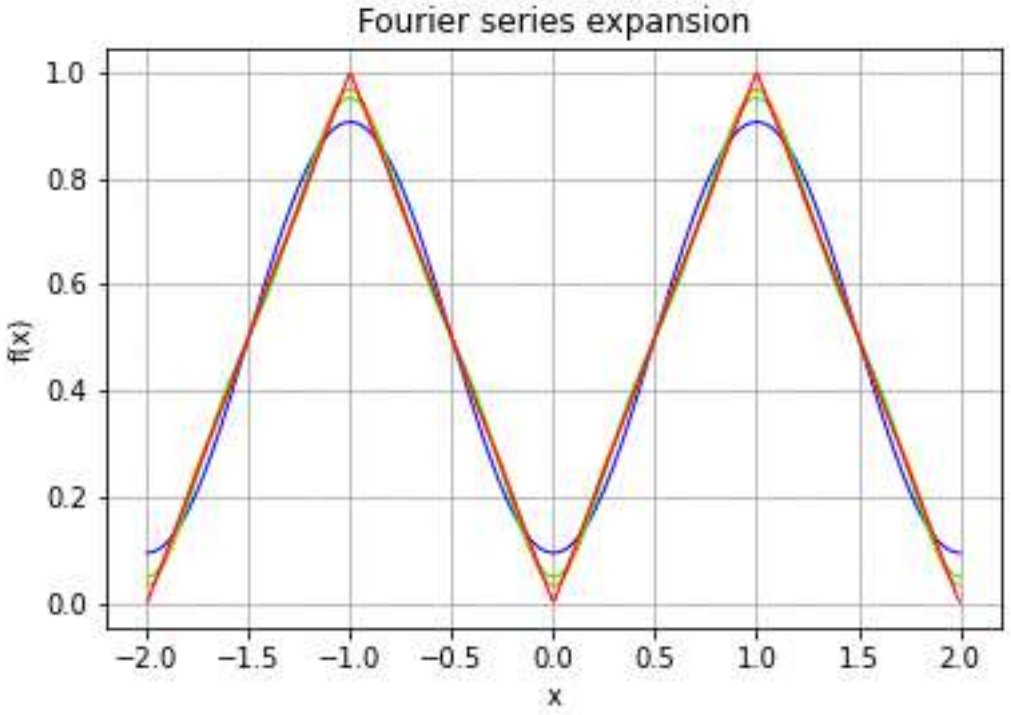
$$f(x) = \begin{cases} -x, & -1 \leqq x < 0 \\ x, & 0 \leqq x \leqq 1 \end{cases}$$

フーリエ級数展開における係数を求めると

$$\begin{aligned} a_0 &= \int_{-1}^{1} f(x)dx \\ &= 1 \end{aligned}$$

$$
\begin{aligned}
a_n &= \int_{-1}^{1} f(x)\cos n\pi x\, dx \\
&= \int_{-1}^{0} (-x)\cos n\pi x\, dx + \int_{0}^{1} x\cos n\pi x\, dx \\
&= \frac{2}{(n\pi)^2}\left((-1)^n - 1\right) \\
b_n &= 0
\end{aligned}
$$

$f(x)$の 1 次（青），3 次（緑），5 次（橙），59 次（赤）までのフーリエ級数展開をグラフで表すと次のようになる。



【演習 23】

$u(x,t)$がxのみの関数$X(x)$とtのみの関数$T(t)$を用いて次の形で表されると仮定する。

$$u(x,t) = X(x)T(t)$$

これを熱伝導方程式に代入すると

$$
\begin{aligned}
X(x)T'(t) &= X''(x)T(t) \\
\frac{T'(t)}{T(t)} &= \frac{X''(x)}{X(x)}
\end{aligned}
$$

上の式は定数であり，これを$-\mu$ $(\mu > 0)$と表すと

$$\frac{T'(t)}{T(t)} = \frac{X''(x)}{X(x)} = -\mu$$

よって，つぎの微分方程式が得られる。

$$T'(t) = -\mu T(t)$$
$$X''(x) = -\mu X(x)$$

c_1，c_2を定数として，$X(x)$は以下の三角関数で表される。

$$X(x) = c_1 \cos\sqrt{\mu}x + c_2 \sin\sqrt{\mu}x$$

xで微分すると

$$X'(x) = -c_1\sqrt{\mu}\sin\sqrt{\mu}x + c_2\sqrt{\mu}\cos\sqrt{\mu}x$$

境界条件より，$X'(0) = X'(1) = 0$であるから

$$c_2 = 0$$

$$\mu = (n\pi)^2 \quad (n\text{は整数})$$

よって

$$X(x) = c_1 \cos n\pi x$$

次に，c_3を定数として$T(t)$を求めると

$$T(t) = c_3 e^{-\mu t}$$
$$= c_3 \exp(-n^2\pi^2 t)$$

重ね合わせの原理より$u(x,t)$は以下の無限級数として表される。

$$u(x,t) = \sum_{n=0}^{\infty} a_n \exp(-n^2\pi^2 t)\cos n\pi x$$

初期条件より，$u(x,0) = 4x(1-x)$であるから

$$4x(1-x) = \sum_{n=0}^{\infty} a_n \cos n\pi x$$

この式の両辺に$\cos m\pi x$をかけて$0 \leqq x \leqq 1$の範囲で積分する。

(i)　$m = 0$のとき

$$\int_0^1 4x(1-x)dx = \int_0^1 \sum_{n=0}^{\infty} a_n \cos n\pi x \, dx$$

$$a_0 = \frac{2}{3}$$

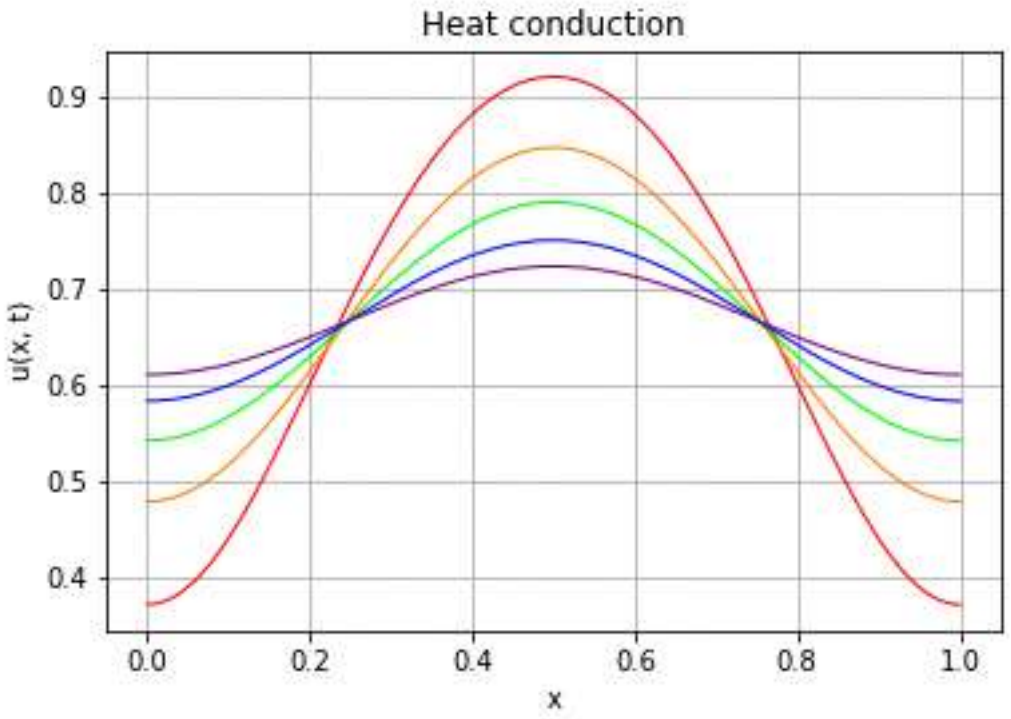
(ii)　$m > 0$のとき

$$\int_0^1 4x(1-x)\cos m\pi x dx = \int_0^1 \sum_{n=0}^{\infty} a_n \cos n\pi x \cos m\pi x dx$$

$$\frac{-4}{(m\pi)^2}((-1)^m + 1) = \frac{1}{2}a_m$$

$$a_m = \frac{-8}{(m\pi)^2}((-1)^m + 1)$$

60 次までの展開を行い，$t = 0.01$（赤）から$t = 0.05$（紫）まで 0.01 ごとの計算結果を示すと次のグラフのようになる。



プログラムは以下のようになる。

```
import numpy as np
import math
import matplotlib.pyplot as plt

x = np.linspace(0, 1, 400)
```

```
N = 60
t1 = 0.01
t2 = 0.02
t3 = 0.03
t4 = 0.04
t5 = 0.05

fn1 = np.zeros(shape = (x.shape[0],))
fn1 = 2.0 / 3.0
for n in range(2, N, 2):
  fn1 = fn1 -16.0 / ((n * np.pi) ** 2) * np.exp(-
((n * np.pi) ** 2) * t1) * np.cos(n * np.pi * x)

fn2 = np.zeros(shape = (x.shape[0],))
fn2 = 2.0 / 3.0
for n in range(2, N, 2):
  fn2 = fn2 -16.0 / ((n * np.pi) ** 2) * np.exp(-
((n * np.pi) ** 2) * t2) * np.cos(n * np.pi * x)

fn3 = np.zeros(shape = (x.shape[0],))
fn3 = 2.0 / 3.0
for n in range(2, N, 2):
  fn3 = fn3 -16.0 / ((n * np.pi) ** 2) * np.exp(-
((n * np.pi) ** 2) * t3) * np.cos(n * np.pi * x)

fn4 = np.zeros(shape = (x.shape[0],))
fn4 = 2.0 / 3.0
for n in range(2, N, 2):
  fn4 = fn4 -16.0 / ((n * np.pi) ** 2) * np.exp(-
((n * np.pi) ** 2) * t4) * np.cos(n * np.pi * x)

fn5 = np.zeros(shape = (x.shape[0],))
fn5 = 2.0 / 3.0
for n in range(2, N, 2):
  fn5 = fn5 -16.0 / ((n * np.pi) ** 2) * np.exp(-
((n * np.pi) ** 2) * t5) * np.cos(n * np.pi * x)

plt.title('Heat conduction')
plt.plot(x, fn1, color = (1.0, 0.0, 0.0), linewidth = 1.0)
plt.plot(x, fn2, color = (1.0, 0.5, 0.0), linewidth = 1.0)
```

```
plt.plot(x, fn3, color = (0.0, 1.0, 0.0), linewidth = 1.0)
plt.plot(x, fn4, color = (0.0, 0.0, 1.0), linewidth = 1.0)
plt.plot(x, fn5, color = (0.5, 0.0, 0.5), linewidth = 1.0)
plt.xlabel('x')
plt.ylabel('u(x, t)')
plt.grid(True)
plt.savefig('practice_23.png')
```

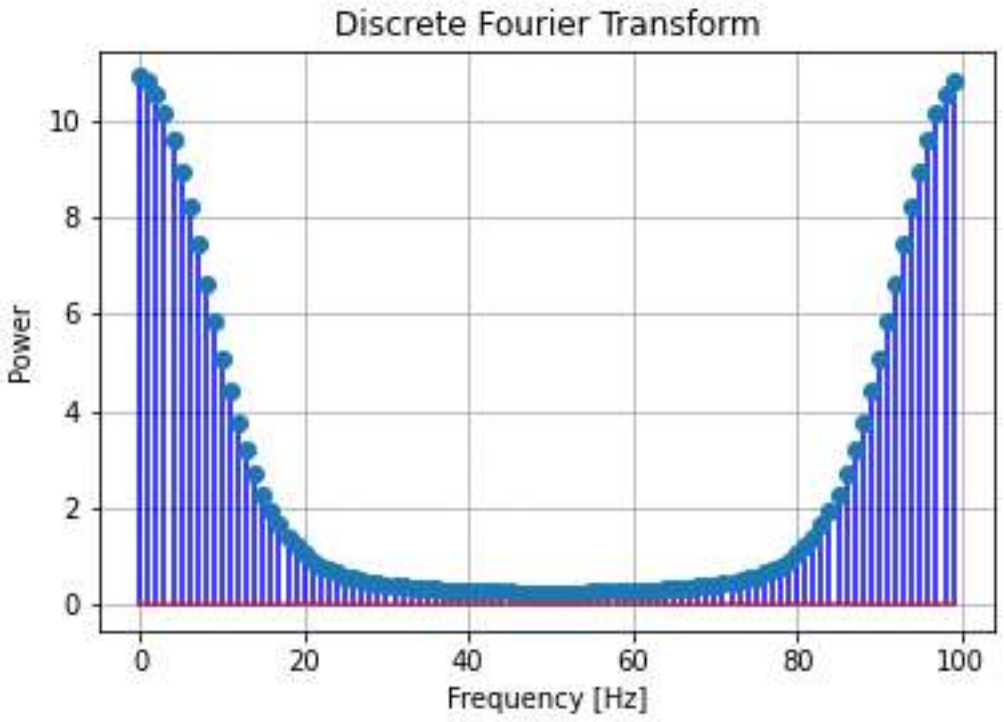
【演習 24】

プログラム例

```
%matplotlib inline

import functools
import matplotlib.pyplot as plt
import cmath
import random
import numpy as np

delta = 0.01
t = np.arange(0, 1, delta)

y = np.exp(-1000 * t * t)

def dft_(f):
  n = len(f)
  Y = []
  for x in range(n):
    y = 0 + 0j
    for t in range(n):
      a = 2 * cmath.pi * t * x / n
      y = y + f[t] * cmath.exp(-1j * a)
    Y.append(y * y.conjugate())
  return Y

F = dft_(y)

plt.title('Discrete Fourier Transform')
plt.xlabel('Frequency [Hz]')
```

```
plt.ylabel('Power')
plt.grid(True)
plt.stem(F, linefmt = "blue")
plt.savefig('practice_24.png')
```

グラフは次の図のようになる。



（目次へ）

MEMO