プログラミング
言語実験
Python

1410055　佐藤　禎紀

課題 1-1

演習 1-3 で行った天気情報の Weather hacks を関数化してみてください. 入力としては，都市番号（前述の例題では 400040）を与えると， 本日の天気を返すような関数 Forecast を作って，テストしなさい.

関数としては,

Forecast(city='400040')

のような形で，都市コードを指定した場合に，本日の予報の文字列を返すような 関数を作成し，コードを実行させ，動作を確認しなさい.

・プログラムリスト

```
import urllib
import json

def Forecast(city=""):
    src = "http://weather.livedoor.com/forecast/webservice/json/v1?city="+city

    f=urllib.urlopen( src )
    str = f.read()
    f.close()

    q = json.loads(str)
    return q["description"]["text"]


print Forecast(city="400040")
```

・実行例

```
[s1410055@ied2 Kadai]$ python kadai1-1.py
 九州北部地方は、梅雨前線の影響により、曇りや雨で雷雨の所があります
。

 28 日の九州北部地方は、梅雨前線や湿った空気の影響により、概ね雨と
なり、雷を伴い激しく降る所があるでしょう。

 29 日の九州北部地方は、梅雨前線や湿った空気の影響により、雨となり
、雷を伴って激しく降る所があるでしょう。

 波の高さは、九州北部地方の沿岸の海域では 28 日は 1.5 メートル、2
9 日は 2 メートルでしょう。豊後水道では 28 日は 1 メートル、29 日は 1
.5 メートルでしょう。
 福岡県の内海では、28 日と 29 日は 0.5 メートルでしょう。

<天気変化等の留意点>
 筑後地方の 28 日は、概ね雨や雷雨となるでしょう。
```

・考察

演習問題 1-3 で作成した天気の情報を格納するプログラムを Forecast 関数内で定義した。そこで返り値を辞書型変数の q を返すことにより、print を関数に対して使うことにより関数化を実現し、演習 1-3 と同様の結果を表示するようなプログラムを作成した。

課題 1-2

次にオプション引数として, day を指定すると, 明日の天気を表示できるようにしなさい.

Forecast(city='400040', day='today')
のような形で定義しておいて,
Forecast(city='400040')
とすれば, 今日の天気
Forecast(city='400040', day='tomorrow')
のような形で使えば, 明日の天気を表示するように拡張しなさい.

・プログラムリスト

```
import urllib
import json

def Forecast(city="",day="today"):
    src = "http://weather.livedoor.com/forecast/webservice/json/v1?city="+city

    f=urllib.urlopen( src )
    str = f.read()
    f.close()

    q = json.loads(str)

    if day=="today":
     i=0
    elif day=="tomorrow":
     i=1

    return q["forecasts"][i]["date"],":",q["forecasts"][i]["telop"]



forecast = Forecast(city="400040")
print "Today:",forecast[0],forecast[1],forecast[2]

forecast = Forecast(city="400040",day="tomorrow")
print "Tommorow:",forecast[0],forecast[1],forecast[2]
```

・実行例

```
[s1410055@ied2 Kadai]$ python kadai1-2.py
Today: 2016-06-28： 雨時々曇
Tommorow: 2016-06-29： 雨
```

・考察

まず、課題１－１で作った関数に課題に与えられているようにオプション引数 day=”today” を増やした。その上、関数内に day の値が today なら i に 0 を、tomorrow なら i に 1 を代入し、その日の日付が格納されている q["forecasts"][i]["date"]とその日の天気が格納されている q["forecasts"][i]["telop"]を返した。

課題 2-1

演習 2−2 音信号の生成 で行ったように，音データを生成することを考えます 演習では単にスクリプトで記述しましたが，これを関数化することを考えます． 関数の形式としては，サンプリング周波数 Fs, 持続時間 dur, 生成周波数 Fc を指定したときに，データ列を返すような関数

x = GenFreq(Fc, Fs, dur)

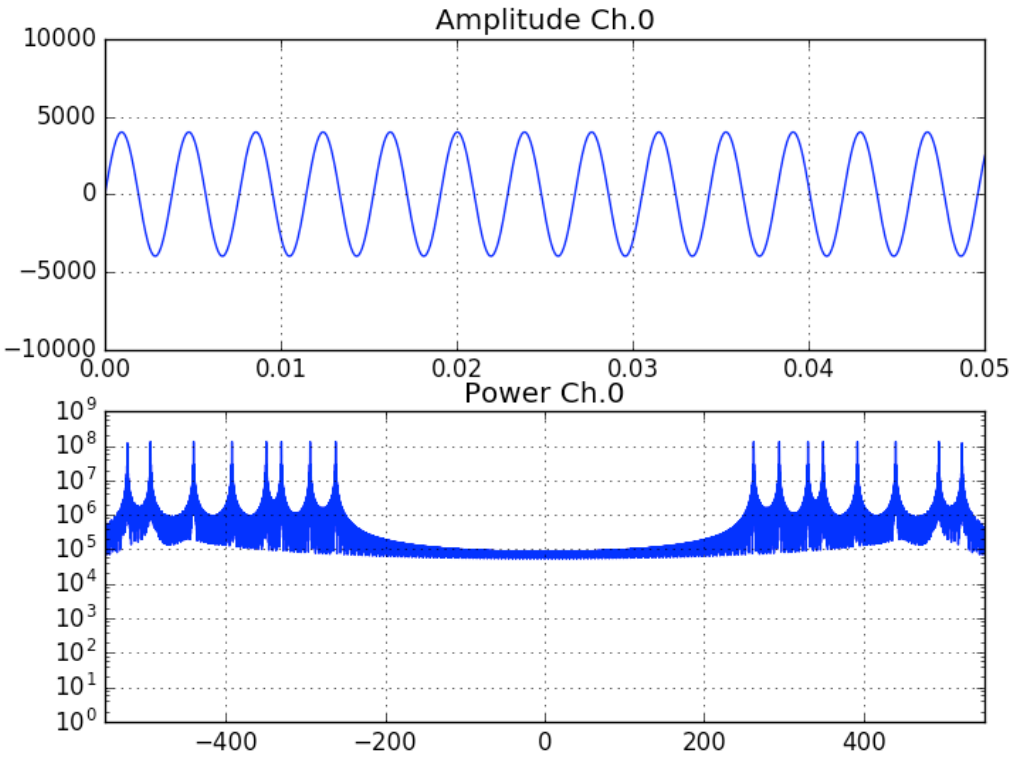
のように用いることが出来る関数を作成しなさい． この関数を用いて，音階ドレミファソラシド(CDEFGAB) を 各音 3 秒ずつ鳴らしていくデータを生成してみなさい． 但し，CDEFGAB の各音の周波数 Fc はそれぞれ (262, 294, 330, 349, 392, 440, 494, 523) [Hz] を用いても 良いものとします.

・プログラムリスト

```
import numpy as np
import scipy as sp
from scipy.io.wavfile import write

def GenFreq(Fc,Fs,dur):
    Amp=4000.
    N = Fs*dur
    delta = 1./Fs
    t = np.arange(N)*delta
    x = Amp*np.sin(2*np.pi*Fc*t)
    return x

Fs = 22050.
Fc =(262,294,330,349,392,440,494,523)
d = 3.
```

```
N = Fs*d

for i in Fc:
    x = GenFreq(i,Fs,d)
    if i==Fc[0]:
        list=[x]
    else:
        list+=[x]

write("Generated.wav",Fs,np.int16(list).reshape(N*8,1))
```

・実行例(作った音源の波形)



・考察

まず、関数はサンプリング周波数と持続時間、生成周波数を与えればその音を生成する関数を作った。この関数内では振幅は 4000 とし、与えられた値からサンプリング数やサン

プリング間隔、時間と波を求め、その波を返し、音を生成した。これをすべての音階において生成し、それぞれをリストである list に代入した。これを音源として wav ファイルに出力することによりすべての音階を３秒ずつ鳴らす音源を生成することができた。また、すべての音階がこの音源に入っている証拠としてこの音源のパワースペクトルを見ると Fc で指定したすべての値付近において波形を確認することができる。

課題 2-2

演習 2-3 音声の読み込みとフーリエ変換 では音声データの読み込みとフーリエ変換を行って パワースペクトル表示を行った． 次に，短時間フーリエ変換 を用いたスペクトログラムの作成を行う． スペクトログラムは，音声データを，短い区間でデータ区切り， 各区間に FFT をかけ，パワースペクトルを計算する． 各区間のスペクトルデータは行方向にならべ， 列方向には区切られた時間順にデータを並べていくと， ちょうど２次元の行列状のデータを造ることができる． このデータ（の対数をとったもの）をスペクトログラムと呼ぶ.

例えば，x[0]～X[1024] までのデータがあった場合， 256 個ずつの区切りを考えると下記の４つの部分を考えることが出来る

x[0] ～ x[255]
x[256] ～ x[511]
x[512] ～ x[767]
x[768] ～ x[1023]
それぞれに対して，FFT を書けて並べると 256 行 4 列の行列を作成できる． これを図示化したものがスペクトログラムである.

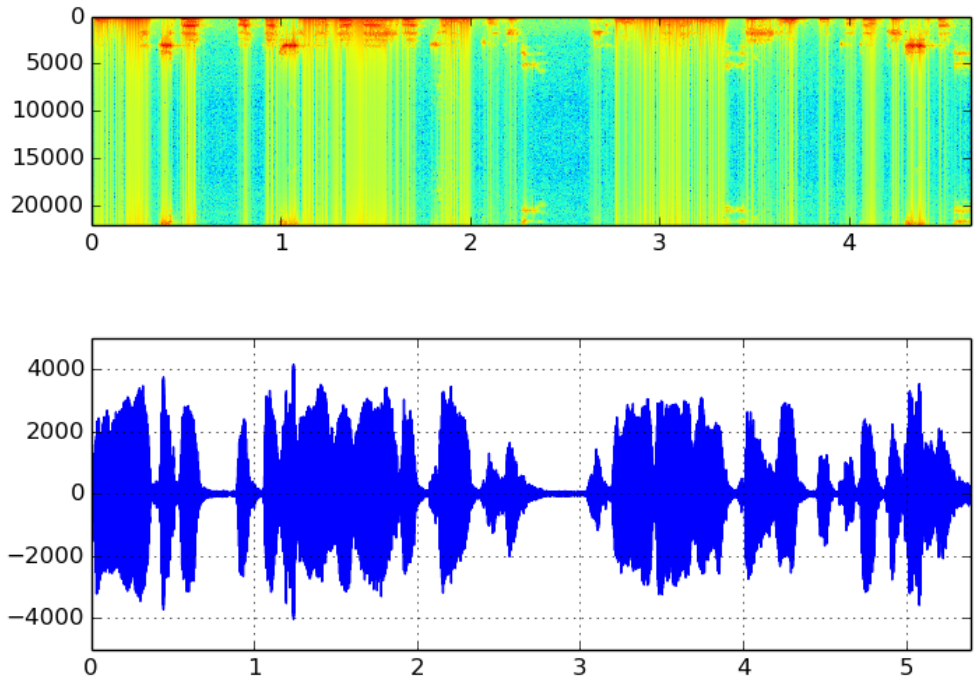
いまモノラル音声信号の配列 x が与えられた時, 256 個ずつのデータで区切ったスペクトログラムを返す関数 SpecGram を作成し， Voice Example のデータを表示せよ.

・プログラムリスト

```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import scipy.fftpack as spf
from scipy.io.wavfile import read
```

```
y = read("KyokoSampling.wav")

Fs = y[0]
delta = 1./Fs
Nmax =238152
fdelta = 1./(Nmax*delta)
t = np.arange(Nmax)*delta
f = np.arange(-Nmax/2,Nmax/2)*fdelta



def SpecGram(x):
    yl = x[:256,0]
    Yl = spf.fft(yl)
    return np.abs(np.log(Yl))



z = [SpecGram(y[1][:256])]
for i in range(Nmax/256-1):
    z += [SpecGram(y[1][256*(i+1):256*(i+2)])]

z2 = np.array(z)
z3 = z2.transpose()

plt.figure()
plt.subplot(2,1,1)
plt.ylim(Fs/200,0)
plt.xticks([0,200,400,600,800],[0,1,2,3,4])
plt.yticks([0,50,100,150,200],[0,5000,10000,15000,20000])
plt.imshow(z3)

plt.subplot(2,1,2)
plt.plot(t,y[1][:Nmax,0])
plt.xlim(0,t[-1])
plt.grid()

plt.show()
```

・実行例



・考察

まず与えられた音源を読み込んだ。ここからサンプリング周波数を読み込み、これからサンプリング間隔、時間、周波数などを計算した。その上で関数 SpecGram では信号の配列データを 256 個ずつ取得し、これらをフーリエ変換し、log の絶対値で返すように定義した。この関数を Nmax/2-1 まで実行し、z に格納した。この格納した z を行列化して画像で表示することによりスペクトログラムの表示に成功した。

課題 3-1

ensyu-1-3 で行った天気情報の Weather hacks をオブジェクト化することを 考えます. オブジェクトの定義は自由にして良いものとしますが, 一例として, WeatherHacks では, クラスの初期化関数 __init__() に対して都市番号 (cityid) を与えることで, Weather hacks のサイトから 返ってくる文字列（もしくは JSON を解析したデータ構造）を保持するようにし, オブジェクト内部に保持させるといった実装例が考えれます.

```
WeatherHacks(object):
    def __init__( self, cityid ):
        self.cityid = cityid
        # weather hacks からの文字の読み込み，クラス内変数 wstring に保持
        self.wstring = ...

    def Forecast( self, day='today'):
        # すでに保持されている wstring を解析することで天気を予報

    def Description( self ):
        # すでに保持されている wstring を解析することで概況を表示
```

のような形でクラスを定義したファイル ‘Weather3.py’ を用意して，以下のような形で実行出来るようなモノを想定しています.

```
In [1]: run 'Weather3.py'
In [2]: Kurume = WeatherHacks( '400040' )
In [3]: Tokyo = WeatherHacks( '130010' )
In [4]: Kurume.Forecast()
Out[4]: 久留米地方の天気: 晴
In [5]: Tokyo.Description()
....
```

さらに，余力がある場合，この保持した内容を解析し， 今日の天気や，温度を表示させるような メソッド関数を定義し，テストしたものを提出しなさい.

・プログラムリスト

```
import urllib
import json

class WeatherHacks(object):
    def __init__(self,cityid):
        self.cityid = cityid

        src ="http://weather.livedoor.com/forecast/webservice/json/v1?city="+cityid
```

```
        f = urllib.urlopen(src)
        self.wstring = f.read()
        f.close()

    def Forecast(self,day="today"):

        self.q = json.loads(self.wstring)
        if(day=="today"):
            i=0
        elif(day=="tomorrow"):
            i = 1

        print self.q["title"],":",self.q["forecasts"][i]["telop"]

    def Description(self):
        print json.loads(self.wstring)["description"]["text"]
```

・実行例

```
In [1]: run "Weather3.py"

In [2]: Kurume = WeatherHacks( "400040" )

In [3]: Tokyo = WeatherHacks( "130010" )

In [4]: Kurume.Forecast()
福岡県 久留米 の天気 : 曇のち晴

In [5]: Tokyo.Description()
 梅雨前線が九州から本州の南岸に停滞しています。

 【関東甲信地方】
 関東甲信地方は、曇りで雨の降っている所があります。

 30 日は、前線の影響により、おおむね曇りで雨の降る所があるでしょう
。
```

```
 7月 1 日は、前線の活動が弱まるため、曇りや晴れですが、午後は大気の
状態が不安定となり、にわか雨や雷雨の所がある見込みです。

 関東近海では、30 日から 7月 1 日にかけて、波がやや高いでしょう。
また、所々で霧が発生しています。船舶は視程障害に注意してください。

【東京地方】
 30 日は、曇りで、所により雨となる見込みです。
 7月 1 日は、曇り時々晴れでしょう。
```

・考察

まず、クラスの初期化関数ではその地方の URL を src に格納し、self.wstring に読み込んだデータを格納しておく。次に Forecast 関数では self.q に初期化関数で読み込んだ self.wstring をロードしたものを代入する。そしてオプション引数の day が today なら i に 0、tomorrow なら i に 1 を代入する。そして self.q["title"]と self.q["forecasts"][i]["telop"]を返すことにより地方と天気を表示した。最後に Description 関数では json.loads(self.wstring)["description"]["text"]を print で表示することにより概況を表示した。

課題 3-2

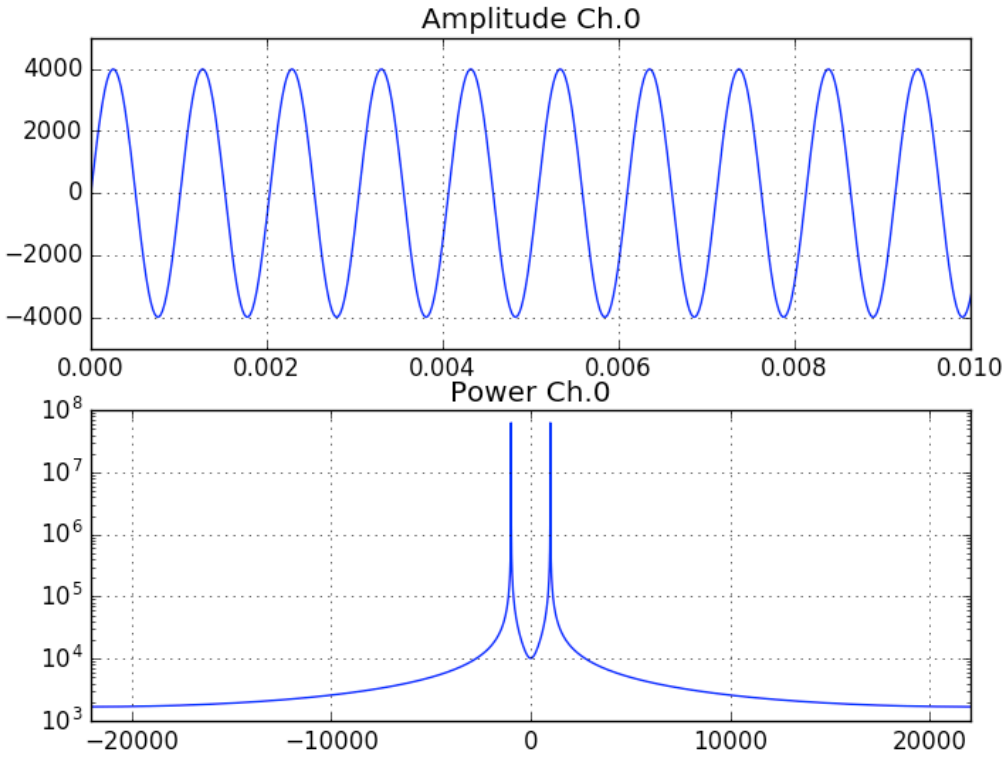
課題 2-1 で行った周波数による音データの生成部のクラス化を考えます． 基本機能としては GenFreq() 関数の拡張を行い， 和音をだせるようなクラスを考えます． 上手くクラスが設計できないようなら 演習 2−2 音信号の生成 を参照してみてください．クラスを定義して，ド・ミ・ソの和音を作ってみるサンプルコードが 下記のようにかけるクラスを課題として想定しています.

・プログラムリスト

```
import numpy as np
import scipy as sp
from scipy.io.wavfile import write

class GenFreq( object ):
    def __init__( self, Fs=44100., dur=1. ):

        self.Fcs={"C":262.,"D":294.,"E":330.,"F":349.,"G":392.,"A":440.,"B":494.}
```

```
        self.Amp = 4000.
        self.N = Fs*dur
        self.delta = 1./Fs
        self.t = np.arange(self.N)*self.delta
        self.tone = 0

    def getSound( self ):
        self.x = self.Amp*np.sin(2*np.pi*self.tone*self.t)
        return self.x

    def getLen( self ):
        return self.N

    def addTone( self, code='C' ):
        self.tone+=self.Fcs[code]


Fs = 44100.
s = GenFreq( Fs, dur=1. )
s.addTone( 'C' )
s.addTone( 'E' )
s.addTone( 'G' )
N = s.getLen()

write( 'hogehoge.wav', Fs, s.getSound().reshape( (N, 1) ) )
```

・実行例(作った音源の波形)



・考察

まずクラス GenFreq の中身だが最初、初期関数で音生成に必要な変数を定義している。次に関数 getSound で波形を作り、その波形を返すように定義している。そして getLen では音の長さである self.N を返している。最後に addTone 関数ではオプション引数 code を指定し、ここで指定された値を変数 self.tone に加える。クラスの外では C と E と G の音を addTone でクラス内の self.tone に足し合わせ和音を生成している。最後に write で音源を出力している。

課題 4-1

課題は周波数フィルタリングを使ってノイズ除去をしてもらいます.　３つのファイルのノイズを除去してなんと言っているかを報告しなさい.

1.　KyokoColored.wav

・切った周波数：4000Hz

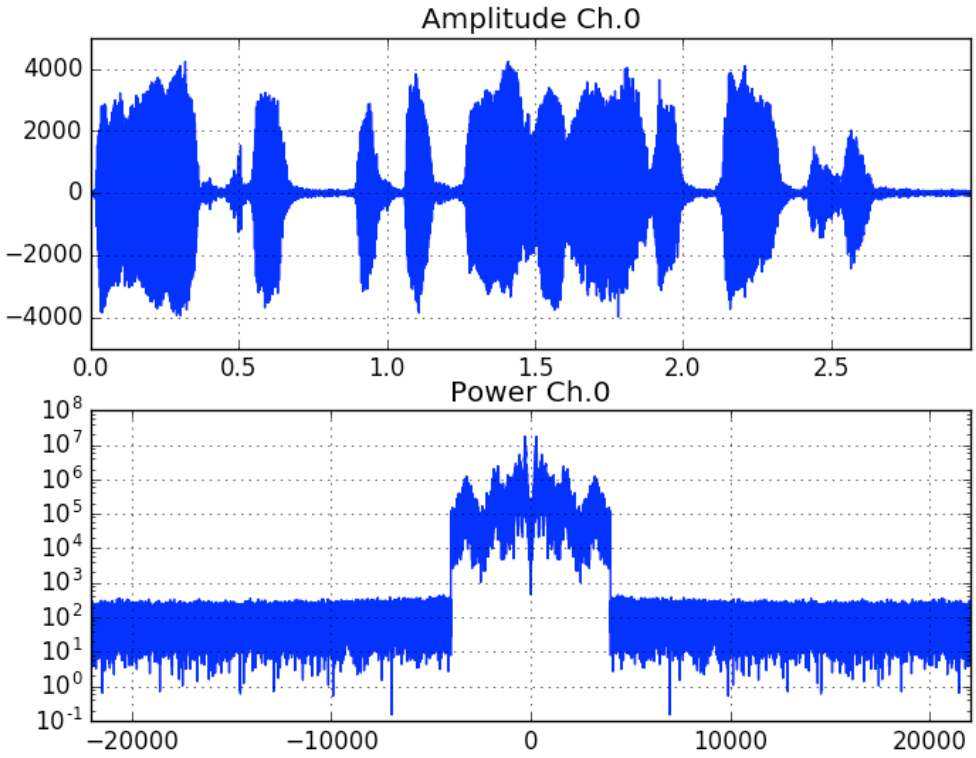
ShowWav クラスを用いてスペクトルを表示させるとノイズの波形はすべて 4000Hz

以上を示していたため、4000Hz 以上の周波数を切って表示した。

・プログラムリスト

```
import numpy as np
import scipy as sp
import scipy.fftpack as spf
from scipy.io.wavfile import read,write
import matplotlib.pyplot as plt

def LowPass(x,f,Fs):
    N = len(x)
    fdelta = float(Fs)/N

    X =spf.fft(x)
    flt = np.zeros(N)
    flt[0:f/fdelta] = 1
    flt[-f/fdelta+1:0] = 1
    Xflt = X*flt

    return spf.ifft(Xflt).real*100

ifname = "KnoiseColored.wav"
olfname = "RemoveKnoise.wav"

y = read(ifname)

Fs = y[0]
Nmax = 65536*2
yl = y[1][:Nmax]
fc = 4000.

yflt = LowPass(yl,fc,Fs)

write(olfname,Fs,np.int16(np.real(yflt)).reshape(Nmax,1))
```

・抽出した音声信号の図



・なんといっているか

こんにちは、わたしのなまえはきょうこです。

2. HAL0001noiseColored.wav

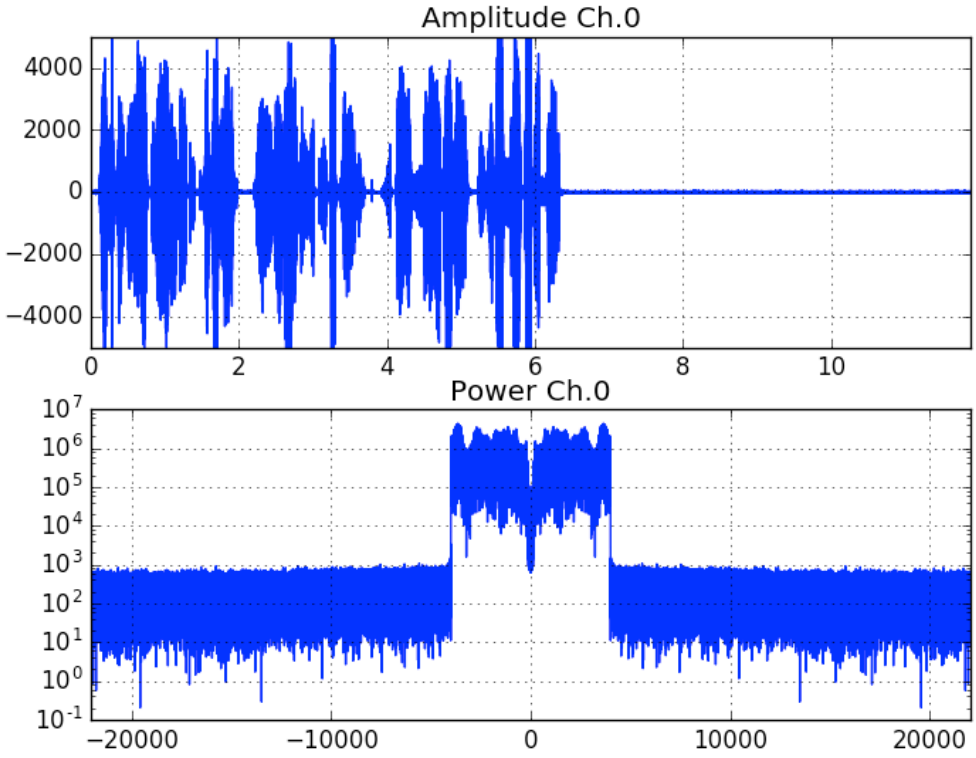
・切った周波数：4000Hz

KnoiseColored.wav の時と同様、ShowWav クラスを用いてスペクトルを表示させるとノイズの波形はすべて 4000Hz 以上を示していたため、4000Hz 以上の周波数を切って表示した。

・プログラムリスト

```
import numpy as np
import scipy as sp
import scipy.fftpack as spf
from scipy.io.wavfile import read,write
import matplotlib.pyplot as plt

def LowPass(x,f,Fs):
    N = len(x)
    fdelta = float(Fs)/N

    X =spf.fft(x)
    flt = np.zeros(N)
    flt[0:f/fdelta] = 1
    flt[-f/fdelta+1:0] = 1
    Xflt = X*flt

    return spf.ifft(Xflt).real*100

ifname = "HAL0001noiseColored.wav"
olfname = "RemoveHAL0001.wav"

y = read(ifname)

Fs = y[0]
Nmax = 65536*8
yl = y[1][:Nmax]
fc = 4000

yflt = LowPass(yl,fc,Fs)

write(olfname,Fs,np.int16(np.real(yflt)).reshape(Nmax,1))
```

・抽出した音声信号の図



・なんといっているか

あしのはやいちゃいろのきつねがのろまないぬをとびこえる。スペインのあめはおもにへいちにふる。

3. HAL0002noiseColored.wav

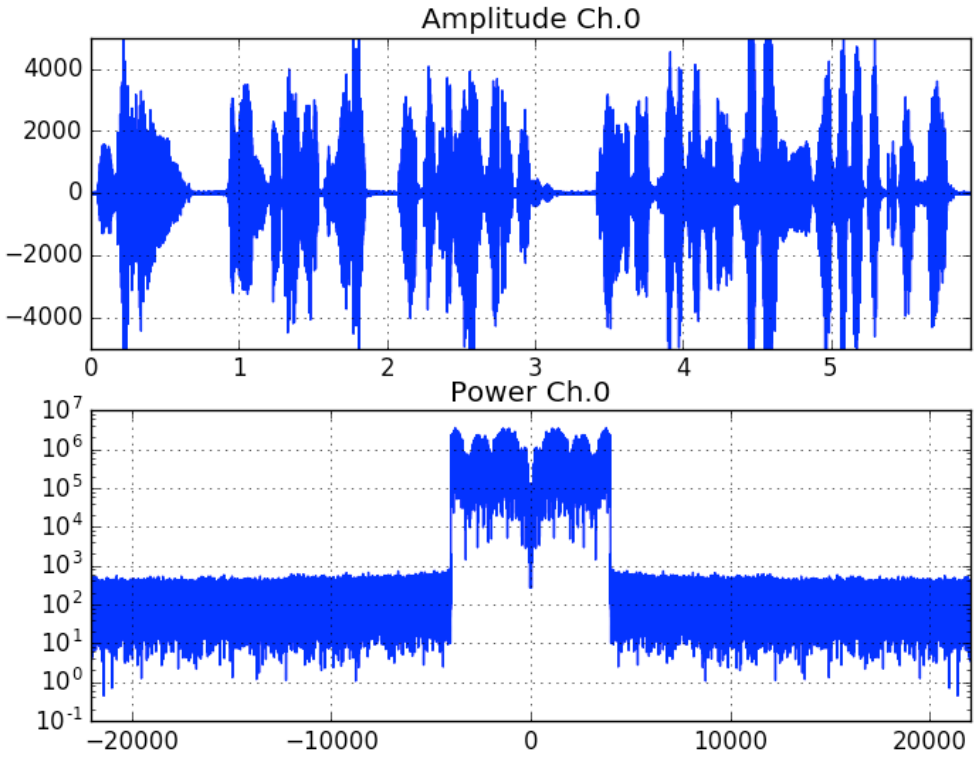
・切った周波数：4000Hz

1., 2.の時と同様、ShowWav クラスを用いてスペクトルを表示させるとノイズの波形はすべて 4000Hz 以上を示していたため、4000Hz 以上の周波数を切って表示した。

・プログラムリスト

```
import numpy as np
import scipy as sp
import scipy.fftpack as spf
from scipy.io.wavfile import read,write
import matplotlib.pyplot as plt

def LowPass(x,f,Fs):
    N = len(x)
    fdelta = float(Fs)/N

    X =spf.fft(x)
    flt = np.zeros(N)
    flt[0:f/fdelta] = 1
    flt[-f/fdelta+1:0] = 1
    Xflt = X*flt

    return spf.ifft(Xflt).real*100

ifname = "HAL0002noiseColored.wav"
olfname = "RemoveHAL0002.wav"

y = read(ifname)

Fs = y[0]
Nmax = 65536*4
yl = y[1][:Nmax]
fc = 4000

yflt = LowPass(yl,fc,Fs)

write(olfname,Fs,np.int16(np.real(yflt)).reshape(Nmax,1))
```

・抽出した音声信号の図



・なんといっているか

　おはよう、チャンドラはくし、わたしははるです。きょうのさいしょのじゅぎょうをはじめてください。

課題 4 全体の考察

　すべての音源で用いたプログラムは総サンプリング数と入力、出力ファイル名が違うだけで他は全く同じプログラムを使っている。まず、関数 LowPass で与えられた値からサンプリグ数と周波数間隔を計算、そして x をフーリエ変換した。そして指定された値より小さい周波数のものを 0 年とし、逆フーリエ変換したものを返すように定義した。値を返す際、ノイズ除去によって音が小さくなってしまったため、振幅を 100 倍にして返した。そして、切った周波数の項目でも明示したようにノイズ除去前の音源の波形を見たところノイズの波形と思われる波形が全て 4000Hz 以上だったことから fc は 4000 とした。