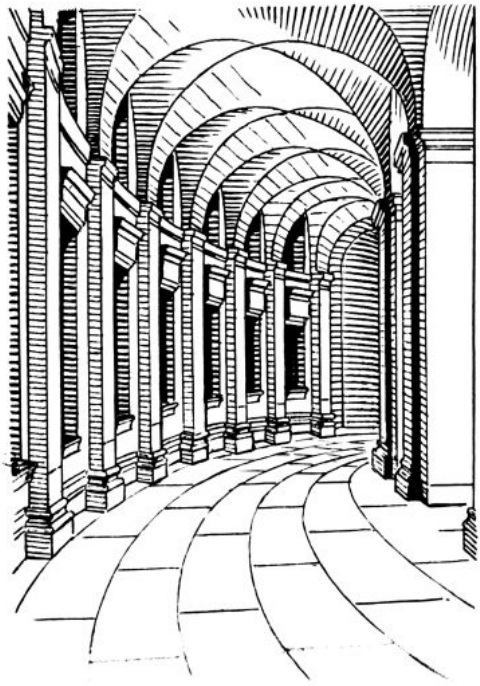
ぐんぐんわかる！

Praat スクリプトドリル



西原 史暁

ぐんぐんわかる！
Praat スクリプトドリル

2013 年 11 月 10 日　初版公開

西原 史暁

目次

第 1 課　はじめに

この課では、このドリルの目的について説明します。また、このドリルで扱うことになる音と周波数に関する情報や、Praat というソフトウェアがどのようなものなのかについても紹介します。

1.1　このドリルの目的

このドリルは、具体的なスクリプトの作成を通じ、Praat のスクリプトの文法の基礎を身につけることを目的としています。プログラミングの知識を事前に持っている必要はありません。プログラミングに関して説明すべきことが出てきたら、このドリルの中で適宜説明します。このドリルで扱う Praat のスクリプトには次のような要素が含まれています。これらの要素は、他のプログラミング言語と共通する点が多いので、今後他のプログラミング言語を学ぶ際にも役立つでしょう。

- 変数の使い方
- `if` を使った条件分け
- `for` を使った繰り返し
- `form` を使った入力欄の作成
- `pause` を使ったユーザからの入力の受け付け
- ファイルへの読み書き

このドリルでは、周波数の高い音が聞き取れるかどうかの実験を行うスクリプトを作ります。このスクリプトを作成することで、Praat のスクリプトの文法の基礎を学ぶことができます。このプログラムを一気に作るわけではありません。いくつかの部分に分けて、順を追って作っていきます。慌てずに、着実に一歩ずつ進めていきましょう。

1.2　音の高さ、周波数、年齢

このドリルでは、人間がどれだけ高い音を聞くことができるかについて実験します。というわけで、音の高さとはどういうものなのかについて簡単に見ていきたいと思います。

音には高さがあります。耳が聞こえる人ならば、音の高さに関してある程度判断がつくはずです。例えば、小さな子どもが出す声ならば高い音、大人の男の声ならば低い音という傾向があります。楽器で言えば、フルートが出す音は高いですし、コントラバスが出す音はかなり低いと言えるでしょう。

さて、私たちが音の高低を判断する際の基準は何なのでしょうか。人間の聴覚理解はかなり複雑なので、1 つの基準で音の高低が決まるわけではありません。ですが、**音の周波数**というものが、高低の判断に大きな影響を与えています。極めて単純に言うと、周波数が高い音ほど人間には高く聞こえ、周波数が低い音ほど人間には低く聞こえます。

それでは周波数とは何でしょうか。物理的に言えば、音は波です。この波には周期というものがあります。この周期がある一定の時間内に何回現れるかを示したものが周波数です。音について記述する際は、1 秒あたりの周波数を見ることが多いです。つまり、1 秒の間に何回の周期があるかということを見るわけです。1 秒間に着目したときの周波数の単位としてヘルツ (Hz) という単位があります。ある音の周期の長さが 100 分の 1 秒だったとします。このとき、1 秒間に 100 回の周期があることになります。よって、この音は 100 Hz の音になります。

周波数が高いほど、音は高く聞こえます。10000 Hz の音はかなり高く聞こえるでしょう。逆に 100 Hz 程度の音ならばかなり低く聞こえるはずです。

さて、人間はどれだけ高い音を聞くことができるのでしょうか。これは人によって違いますが、およそ 20000 Hz ぐらいの音を聞くことができます。それより高い音、例えば 50000 Hz ぐらいの音になると人間には聞き取ることができません。

参考 人間が聞き取れる音の範囲のことを可聴域（かちょういき）と言います。

今、20000 Hz ぐらいの音を聞くことができると書きましたが、実はこれは若い人に限った話です。人間は年をとればとるほど、高い音が聞こえにく

くなると言われています。年をとると、13000 Hz 程度の高さの音ですら聞き取ることが困難になります。

参考 逆に言うと、高い音は若者にしか聞こえないということになります。なお、若者にしか聞こえない高い音のことをモスキート音と呼ぶことがあります。モスキート音を利用して若者を追い払うという仕組みもあります。

先にも述べましたが、このドリルでは、高い音が聞こえるかどうかを調べるためのスクリプトを扱います。スクリプトが完成したらいろいろな年齢の人に聞かせてみて、どれぐらいの周波数の音が聞こえるかを試してみるとよいでしょう。そうすれば、どれだけ高い音を聞くことができるかと年齢との関係を見ることができるはずです。

1.3 Praat と Praat スクリプト

Praat は音声について様々な処理ができるソフトウェアです。Praat は、「プラート」と発音します。これは、元々オランダ語の単語で、「話」を意味します。

Praat を使えば、音声を録音したり、合成したりするだけでなく、音声がどういう性質を持っているのかを分析することもできます。また、音声を提示する実験を実施したり、簡単な統計分析を実行したりすることもできます。このため、Praat は人間がどのように音を産出し、どのように音を知覚するかを研究する際に、必要不可欠なソフトウェアとなっています。

参考 人間が発する言語の音の研究のことを音声学 (phonetics) と言います。Praat は音声学の研究で広く使われているソフトウェアです。

Praat は、オランダのアムステルダム大学で開発されたソフトウェアです。Windows, Macintosh, Linux などさまざまな OS で使用できます。このソフトは、以下に示した公式ウェブサイトからダウンロードできます。Praat は無料で配布されています。

- `http://www.fon.hum.uva.nl/praat/`

このウェブサイトから、自分の OS に合った Praat を選んで、ダウンロー

ドし、自分の PC で使えるようにしてください。自分の PC で使えるようにする方法については、このドリルでは特に説明しません。英語が分かるなら、Praat の公式ウェブサイトに書いてある記述を参考にすれば簡単に使えるように設定できるでしょう。英語に自信のない人は、Google などの検索サイトで、「Praat インストール」といったキーワードで検索しましょう。参考になるサイトが出てくるはずです。

参考 Praat のメニューはすべて英語で書かれています。

Praat は基本的にメニューやボタンをマウスでクリックすることで分析を行うソフトです。マウスを使った操作は分かりやすいと言えば分かりやすいのですが、大量のデータを処理したり、同じことを繰り返したりするような作業には向いていません。1 個の音声ファイルを処理するのに 5 回ボタンを押す必要があるとしたら、100 個の音声ファイルを処理するのに 500 回もボタンを押す必要が出てきます。これでは面倒です。また、途中でボタンを押し間違えるかもしれません。

このような面倒を防ぐために、Praat には**スクリプト**というものが用意されています。スクリプトを書いて実行すると、面倒な操作が一気に処理できます。

参考 スクリプト (script) という英語はもともと「台本」という意味です。劇の台本には、どういう動きをするのか、どういうセリフを言うのか、ということが書かれており、俳優はその指示に従って動きます。この俳優と台本の関係は、コンピュータとスクリプトの関係に似ています。スクリプトに命令を書くことで、コンピュータにその命令を実行させるのです。

参考 マウスなどを使ってアイコンやボタンを押して操作する方式をグラフィカルユーザインターフェース (Graphical User Interface) と言います。略して GUI と呼ぶこともあります。これと対照的なのが、キャラクタユーザインターフェース (Character User Interface, CUI) です。CUI では、キーボードを使って、文字を打ち込むことで様々な処理を行います。Praat は基本的に GUI で処理するソフトなのですが、これを CUI で使えるようにしたものがスクリプト機能なのです。

Praat のスクリプトを書くにはプログラミングの知識が必要です。もっとも、このドリルではプログラミングの経験がない人にも分かるように書いているので、安心してください。プログラミングに関して必要な知識は、このドリルの中で適宜紹介します。

参考 スクリプトはプログラムの一種です。プログラムの中でも簡易的なものがスクリプトと呼ばれると考えてください。

Praat のスクリプトを作るには次のようにします。まず、Praat を立ち上げてください。すると、ウィンドウがいくつか出てきます。その中から、Praat Objects というウィンドウを選んでください。この Praat Objects というウィンドウの上部にあるメニューから、Praat と書かれているところを選んでクリックしてください。そうすると、詳しいメニューが出てきます。その中から New Praat script を選んでクリックしてください。これで、新しいウィンドウが出てきます。これが、Praat のスクリプトエディタのウィンドウです。このウィンドウにスクリプトの文章を書きます。

スクリプトの内容を保存するには、スクリプトエディタのウィンドウのメニューから、File を選んでクリックし、さらに Save を選んでクリックしてください。

参考 Save はいわゆる「上書き保存」になります。いわゆる「名前をつけて保存」にしたければ、Save の代わりに Save as... を選んでください。

保存したスクリプトを開くには、Praat Objects ウィンドウのメニューから、Praat と書かれているところを選んでクリックし、Open Praat script... を選んでクリックしてください。スクリプトエディタから開くこともできます。スクリプトエディタのウィンドウのメニューで File を選んでクリックし、さらに Open... を選んでクリックしてください。

1.4　スクリプトを入力するときに注意すべきこと

このドリルでは、スクリプトを次のような枠に入れて表示します。

```
1 line␣one
2 line␣␣two
3 line␣three
```

枠の左側についている数字は行番号を示します。また、␣は、スペースを入力すべき場所を示します。この例で言うと、1 行目は line␣one とありますが、これは line と one の間に空白を 1 つ入力すべきことを意味しています。␣␣のように␣が 2 つ以上連続する場合がありますが、その場合は␣が出てきた回数だけ空白を入力してください。なお、␣は、アンダーバー（_）とは違いますので注意しましょう。また、各行の終わりでは必ず改行してください。

ただし、次のような長い記述があるところには気をつけてください。

```
1 This␣sentence␣is␣a␣very␣very␣very␣very␣very␣very␣long
      ␣example.
2 That␣is␣another␣example.
```

1 つの行に書かれた内容が長い場合、紙面の幅の都合により折り返されます。この例では、1 行目が long で終わって、2 行目が␣から始まっているように見えますが、これは単に紙面の幅が足りないので折り返されているだけです。あくまでも見かけ上 2 つの行に分かれているだけで、実際は合わせて 1 行です。つまり、この例を実際に入力する際は、long まで入力したら、そこで改行せずにスペースを 1 つ入れ、さらに example. と入力し、そこで改行した上で次の行に That is another example. と入力すればよいのです。枠の左側についている数字は、見た目上の行数ではなく、実質的な意味の行数を示しています。実際、This から始まる行の左に “1” とありますが、␣example. となっているところの左には数字がなく、この部分が 1 行目の続きであることが分かります。いずれにせよ、このドリルで「何行目を見てください」と書いてあるときは、見た目上の行数ではなく、枠の左側についている行番号の数字に従ってください。先ほど、行の終わりでは必ず改行せよと述べましたが、この行の終わりについても、見た目上の行ではなく、実質的な意味の行の終わりであると心得てください。

このドリルに載っているスクリプトを見るときは、形がまぎらわしい字に注意してください。特に、数字のゼロ (0) と、大文字のオー (O) は形が似ているので気をつけましょう。数字のゼロ (0) は丸の中に斜め線があります。大文字のオー (O) は丸の中に何もありません。また、数字の一 (1) と、小文字のエル (l) も形が似ているので取り違えないようにしましょう。

スクリプトで大文字と小文字は違うものとして扱われます。自分で入力するときに、このドリルで Praat と書いているものを praat としてしまったり、PRAAT としてしまったりするとうまく動かなくなります。また、スクリプトを入力するときは、いわゆる半角文字で入力してください。全角文字で入力するとうまく動かなくなります。

第 2 課　音を合成してみよう

この課では、Praat の音声合成機能を使って、簡単な音を合成します。

この課を終えると、次のようなことができるようになります。

- Praat のメニューをスクリプトから呼び出す
- Praat のスクリプトを使ってメッセージを表示する

2.1　GUI メニューから音を合成しよう

まず、Praat を使って音を合成するところから始めましょう。

音の本質は波です。単純な波であれば、それを数式で表すことができます。このため、適切な数式があれば、音を合成することができます。

問 2–1　確認しよう

Ⓐ 周波数とは何ですか。ヘルツとはどんな単位ですか。

Ⓑ 周波数が高くなると、音はどうなりますか。

Ⓒ $-2\pi \leq x \leq 2\pi$ の範囲で、正弦関数 $\sin(x)$ のグラフを描いてください。ラジアンによる角度の表し方に注意してください。普通の角度の表し方での 360 度は、ラジアンにすると 2π になります。180 度ならば π で、90 度ならば $\frac{\pi}{2}$ です。

Ⓓ 正弦波というのはどんな波ですか。音波としての正弦波はどのような性質がありますか。

理論的なことはさておき、Praat で実際に音を合成してみることにしましょう。この課では、1000 Hz の正弦波を作ります。

まず、Praat を起動してください。いくつかウィンドウが出てきますが、このうち、`Praat Objects` というウィンドウを選んでください。使っているコンピュータによって表示が微妙に異なるかもしれませんが、ウィンドウ

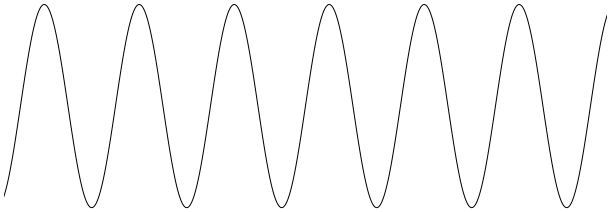


図 2.1　正弦波

の上の方にメニューが並んでいると思います。そのメニューの中から New を選んでクリックしてください。すると、詳しいメニューが開くので、Sound にマウスを合わせ、Create Sound from formula... を選んでクリックしてください。

参考 “Create Sound from formula” とは、「数式から音を作る」という意味です。

すると、色々な入力欄があるウィンドウが出てきます。このウィンドウのタイトルは Create Sound from formula となっているはずです。このウィンドウに色々入力することで、どういう音を合成するのかを Praat に伝えます。

入力欄を 1 つずつ見ていきましょう。

一番上にあるのが、Name という入力欄です。ここには、作成する音声の名前を入力します。分かりやすい名前をつけておくと、どんな音声だったか後で確認しやすくなります。実際にプログラムを書く段階にまだ入っていませんが、「分かりやすい名前をつける」ということは、プログラムを書く際の重要な心構えの 1 つです。今は、1000 Hz の正弦波 (sine wave) を作るので、sineWave1000 と入れておきましょう。間にスペースなどを入れず、一気に sineWave1000 と入力してください。

参考 sineWave1000 のように、単語の区切りを大文字で表す手法のことをキャメルケースと言います。プログラミングでは、単語を区切るためにスペースを入れられない場合にキャメルケースが使われることがあります。キャメルケースを使う場合、2 番目以降の単語の最初の

文字を大文字にします。1 番目の単語の最初の文字は大文字にする場合もありますが、ここでは小文字にします。例えば、“first name” なら firstName に、“special blend tea” なら specialBlendTea になります。

次の入力欄を見てみましょう。Number of channels (e.g. 2 = stereo) とあります。これは、作成する音声のチャンネル数を指定する欄です。チャンネル数が 2 の場合、ステレオ音声となり、イヤホンで聴いた場合、右と左とで違った音声が流れることになります。これに対して、チャンネル数が 1 の場合は、モノラルとなります。このときは、右と左とで音が異なるようなことはなく、音が 1 種類だけになります。このドリルではモノラル音声でかまわないので、この欄には 1 と入れておいてください。

その次の Start time (s) は音声の始まる時間を秒単位で入力します。とりあえず 0.0 と入れておきます。

また、その次の End time (s) は音声の終わる時間を秒単位で入力します。とりあえず 1.0 と入れておきます。開始時間を 0.0 秒、終了時間を 1.0 秒と設定しましたから、音声の持続時間はちょうど 1 秒になります。

次に、Sampling frequency (Hz) という欄で作成する音声のサンプリング周波数を指定します。サンプリング周波数とは、音をどれだけきめ細かに合成するかを示す値です。値が大きければ大きいほど、きめ細かなものになります。ここでは 44100 と入れておきます。1 秒を 44100 の点で表現することになります。

参考 ここでのサンプリング周波数と、これから合成する正弦波の周波数は、周波数という名前が付いている点では同じですが、意味合いが異なります。サンプリング周波数はあくまでも合成される音のきめ細かさを示すものです。

最後に、Formula の欄に、合成したい音声がどのようなものなのかを数式で書き入れます。1000 Hz の正弦波の数式は、sin(2*pi*1000*x) となりますので、これをそのまま入力してください。なお、数式の中のアスタリスク（*）はかけ算の記号です。例えば、3*2 は 3×2 という意味で、答えは 6 になります。また、pi は円周率の π を示します。

かけ算の記号としてアスタリスク（*）を用いるのは、Praat に限ったこ

とではありません。他のプログラミング言語でも同様です。

かけ算に限らず、プログラミング言語での数式の書き表し方は、日常生活での数式の書き表し方と異なる点があります。この書き表し方を覚えておくと、Praat に限らず、他のプログラミング言語を使う際にも役立ちます。

かけ算　かけ算は $\times$ の代わりに、アスタリスク（*）を用いて表します。例えば、4×7 ならば、`4*7` と表記します。

割り算　割り算は $\div$ の代わりに、スラッシュ（/）を用いて表します。例えば、$9 \div 3$ ならば、`9/3` と表記します。

足し算と引き算　足し算と引き算に関しては、日常生活と同じように書き表します。例えば、足し算なら `2+1` のように、引き算なら `6-5` のように書き表します。

累乗　8^4 のような累乗は、サーカムフレックス（^）を用いて、`8^4` のように表記します。なお、$8^4 = 8 \times 8 \times 8 \times 8$ です。

すべて入力し終わったら、ウィンドウの下の方にある OK というボタンを押してください。すると、先ほどまでの入力ウィンドウが消え、新しい音声が加わります。Praat Objects ウィンドウを見ると、`1. Sound sineWave1000` というのが選択された状態になっているはずです。

参考　操作をやり直した場合、1 以外の数字になっているかもしれません。別の数字になっていても特に問題はありません。

さて、合成した音声を聞いてみましょう。`1. Sound sineWave1000` が選択された状態のままで、Praat Objects ウィンドウの Play というボタンを押します。これで音声が再生されます。高く澄んだ音が聞こえましたか。もし、うまく音が聞こえないようでしたら、コンピュータ、スピーカーあるいはイヤホンの音量を調節してみてください。それでもうまくいかないようでしたら、最初からやり直してください。

音が無事聞こえたら、合成した音声はとりあえず不要となります。そこで、`1. Sound sineWave1000` が選択された状態のままで、Praat Objects ウィンドウの Remove というボタンを押してください。すると、先ほど作成した音声が消えます。

2.2 スクリプトから音を合成しよう

さて、先ほど GUI メニューで処理したことを今度はスクリプトを使って処理してみましょう。

Praat のスクリプトウィンドウを出して、次のように入力してください。大文字と小文字を間違えるとうまくいかないので注意してください。

```
1 do("Create␣Sound␣from␣formula...",␣"sineWave1000",␣1,
      ␣0.0,␣1.0,␣44100,␣"sin(2*pi*1000*x)")
2 do("Play")
3 do("Remove")
```

入力し終わったら、早速実行しましょう。Praat のスクリプトを実行するには次のようにします。まず、Praat のスクリプトウィンドウの上の方にあるメニューから Run を選んでクリックします。すると、詳しいメニューが出てくるので、そこから Run を選んでクリックしてください。こうすると、1000 Hz の正弦波の音が聞こえるはずです。もし聞こえなかったり、エラーメッセージが出たりした場合は、自分の入力したスクリプトがこのドリルで掲げたものと違っていないか一字一句確認してください。コンマやダブルクォーテーションを入れる場所が間違っていたり、大文字と小文字を取り違えたりするだけでも、うまく動かなくなります。そのような誤りがないかよく見てください。

参考 メニューからスクリプトを実行する代わりに、キーボードショートカットを使って実行することもできます。Windows の場合は、Ctrl キーを押しながら R キーを押すとスクリプトが実行されます。Macintosh の場合は、⌘ キーを押しながら R キーを押してください。

今入力したスクリプトについて解説したいと思います。このスクリプトでは、どの行も do という記述から始まっています。そして、do の後に "Create Sound from formula..."や"Play"といったダブルクォーテーションで囲まれた記述があります。これらは、先ほどマウスを使って操作したときのメニューあるいはボタンの名前です。Praat のスクリプトでは、do の後に、GUI メニューの名前を並べることで、そのメニューを実行することができます。ですので、マウスを使ってメニューから操作しなくても、スクリプトで do を使って記述することで同じことができるのです。

重要 Praat のスクリプトでは、do を使って、GUI メニューでの操作と同じことが実行できる。

さて、do についてもう少し詳しく説明します。まず、do("……") というのが基本構造ですので、これをよく覚えておきましょう。do の後に丸括弧をおき、その後に GUI メニューの名前をダブルクォーテーションに囲んで入れるのです。つまり、do("……") というのが基本構造で、…… にメニューの名前が入ります。先ほどのスクリプトの 2 行目にあったように、Play ならば do("Play") となります。同様に、Remove をスクリプトから実行したければ、do("Remove") とします。

先ほどのスクリプトの 1 行目は、do("……") という基本構造よりずっと複雑に見えます。しかし、基本構造をしっかりおさえれば理解は難しくありません。

GUI メニューから音を合成したときは、メニューの中から、Create Sound from formula... というものを選びました。スクリプトから実行する場合は、do("……") という基本構造に従えば、do("Create Sound from formula...") となりそうです。しかし、これではうまくいきません。先ほどメニューから音を合成したときは、音を設定するためのウィンドウが出てきて、そこで色々と設定を行いました。どういう音なのか設定しないかぎり、Praat はどんな音を合成していいか分かりません。ですから、スクリプトで do を使うときも、音の設定を Praat に教える必要があります。

参考 Praat のメニューには、Create Sound from formula... のように、末尾にピリオドが 3 つあるものと、Play のように末尾に特に何もないものがあります。末尾のピリオドの有無には、ちゃんとした意味があります。末尾にピリオドが 3 つあるメニューを実行すると、細かな設定をするためのウィンドウが出現します。これに対して、末尾に特に何もないものは、設定のためのウィンドウが出現しません。ただし、スクリプトの場合は、末尾にピリオドが 3 つあるメニューを実行しても、細かな設定をするためのウィンドウが出現しません。このため、設定内容をあらかじめ書いておく必要があります。

先ほど、スクリプトの1行目に、do("Create Sound from formula...", "sineWave1000", 1, 0.0, 1.0, 44100, "sin(2*pi*1000*x)") と記述しました。こう記述することで、1000 Hz の正弦波が合成されます。この部分をもっと詳しく見てみましょう。まず、do があります。これは「GUI メニューでのコマンドを実行せよ」という意味でした。しかし、do だけではどんなコマンドを実行するか分かりません。ここの do がすることの詳細は、丸括弧〔"(" と ")"〕の中に書かれています。というわけで、丸括弧の中身を見ましょう。丸括弧の中身はコンマによって区切られています。

参考 理論的な話をすると、Praat のスクリプトにおいて、do は**関数**の一種です。プログラミングにおける「関数」は数学における「関数」とは意味が若干異なります。極めて単純にいえば、プログラミングでの関数とは、何かしらの作業を実行して結果を出すものです。「関数」の代わりに「命令」と言う場合もあります。Praat のスクリプトでは関数を並べることで様々な作業を行います。関数は、引数というものをとることがほとんどです。引数を用いることで関数がする作業を細かく設定することができます。Praat では、関数の後に丸括弧をおいて、その中に引数を記述します。例えば、sqrt という平方根を求める関数があります。この関数ではどんな数の平方根を求めるのかということを引数の形で指定する必要があります。3 の平方根を求めたければ、sqrt(3) とするのです。括弧の中にあるのが引数というのはこういうことです。また、引数を複数持つ関数もあります。このような場合、ある引数と別の引数との境界をはっきりしめす必要があります。ここではコンマを使って引数と引数を区切ります。引数がコンマによって区切るというのは do に限りません。Praat のスクリプトではみなそうです。なお、他のプログラミング言語でもコンマで区切るという場合が多いです。

コンマによって区切られているわけですから、do("Create Sound from formula...", "sineWave1000", 1, 0.0, 1.0, 44100, "sin(2*pi*1000*x)") の丸括弧の中身は、次に掲げるように 7 つの部分に分かれることになります。

- "Create Sound from formula..."
- "sineWave1000"
- 1
- 0.0
- 1.0
- 44100
- "sin(2*pi*1000*x)"

参考 要するに、ここでは関数の do が、7 つの引数を持っているということになります。

最初の"Create Sound from formula..."は、GUI メニューの音を作るコマンドである Create Sound from formula... を実行するということを示しています。2 番目の"sineWave1000"から 7 番目の"sin(2*pi*1000*x)"までがどのような音を作るかの設定を示しています。この設定内容は、メニューから選んだときに出てくる細かな設定をするためのウィンドウの入力欄に対応しています。細かな設定をするためのウィンドウの入力欄に入れるべきものを順に並べているのです。細かな設定をするためのウィンドウで一番上にあったのは Name という入力欄でした。この入力欄の内容が"sineWave1000"に対応します。次に Number of channels (e.g. 2 = stereo) という入力欄が 1 に、Start time (s) という入力欄が 0.0 にといったように順に対応しています。

ところで、do の中身は、ダブルクォーテーションに囲まれているものとそうでないものがあります。"Create Sound from formula..."や"sineWave1000"はダブルクォーテーションで囲まれていますが、0.0 や 44100 は囲まれていません。単純に言うと、数値はダブルクォーテーションで囲みません。文字列はダブルクォーテーションで囲みます。0.0 や 44100 というのは数値ですからダブルクォーテーションで囲まなかったのです。

重要 数値はダブルクォーテーションで囲まず、文字列はダブルクォーテーションで囲む。

色々書いてきましたが、まとめに入りたいと思います。GUI メニューでの操作は、スクリプトから do を使って実行することができます。メニューの名前を do の後の括弧の中にそのまま書けばよいのです。詳細設定を行う場合も、do の後の括弧の中に記述します。ですから、先ほどのスクリプトの 1 行目では、`do("Create Sound from formula...", "sineWave1000", 1, 0.0, 1.0, 44100, "sin(2*pi*1000*x)")` と入力し、音声の合成を行いました。その次に、メニューから実行したときは、Play というボタンを押して音声を再生しました。これをスクリプトで実行するには、スクリプトに `do("Play")` と書けばよいのです。実際、先ほどのスクリプトでは 2 行目に `do("Play")` と書きました。さらに、音を消すときに Remove というボタンを押しました。これをスクリプトから実行するには、Play の例と同様に、`do("Remove")` とスクリプトに書けばよいのです。つまり、メニューから行った操作を 1 つずつスクリプトに書いていけば、メニューから実行した場合と同じことができるのです。

問 2–2　確認しよう

Ⓐ 先ほどのスクリプトの 1 行目の do の後の括弧の中にある要素が、GUI メニューで合成したときの細かな設定をするためのウィンドウの入力欄とどう対応しているか述べてください。

Ⓑ のこぎり波の音はどんな音でしょうか。調べてみましょう。

Ⓒ 正弦波の合成の例にならって、Praat のメニューから、1000 Hz ののこぎり波を合成してみてください。なお、数式として `1000*x-floor(1000*x)` と入力してください。

Ⓓ 今度は、Praat のスクリプトから、1000 Hz ののこぎり波を合成してみてください。

2.3 メッセージを表示しよう

先ほど作った 1000 Hz の正弦波を合成するためのスクリプトは次のようなものでした。

```
1 do("Create␣Sound␣from␣formula...",␣"sineWave1000",␣1,
      ␣0.0,␣1.0,␣44100,␣"sin(2*pi*1000*x)")
2 do("Play")
3 do("Remove")
```

これで特に問題はないのですが、音を合成して再生するだけでは無愛想な感じがします。そこで、音を合成したら、メッセージを表示するようにしてみましょう。

Praat のスクリプトでメッセージを表示するには、writeInfoLine という命令を使います。writeInfoLine は、writeInfoLine の後の括弧の中身を表示する命令です。先ほどのスクリプトの途中に、writeInfoLine("Created sineWave1000") という命令を書いた行を付け加えてください。付け加える際に、writeInfoLine の大文字と小文字の使い分けに気をつけてください。writeinfoline のように全部小文字にしてしまうとうまく動きません。

修正されたスクリプトは次のようになるはずです。

```
1 do("Create␣Sound␣from␣formula...",␣"sineWave1000",␣1,
      ␣0.0,␣1.0,␣44100,␣"sin(2*pi*1000*x)")
2 writeInfoLine("Created␣sineWave1000")
3 do("Play")
4 do("Remove")
```

このスクリプトを実行すると、Praat Info というウィンドウに *Created sineWave1000* と表示されるはずです。ここで、writeInfoLine を使う際に、Created sineWave1000 をダブルクォーテーションで囲んでいることに注意してください。これは do の時と同じ話で、文字列なのでダブルクォーテーションで囲んでいます。

メッセージをもう 1 つ追加してみましょう。

```
1 do("Create␣Sound␣from␣formula...",␣"sineWave1000",␣1,
      ␣0.0,␣1.0,␣44100,␣"sin(2*pi*1000*x)")
2 writeInfoLine("Created␣sineWave1000")
```

```
3 do("Play")
4 writeInfoLine("Played␣sineWave1000")
5 do("Remove")
```

このスクリプトを実行すると、Praat Info ウィンドウにまず *Created sineWave1000* と表示され、正弦波の再生が終わると、*Created sineWave1000* に代わって *Played sineWave1000* と表示されるはずです。

ところで、*Played sineWave1000* が表示された後、*Created sineWave1000* というメッセージはどうなってしまったのでしょうか。このスクリプトでは、*Played sineWave1000* が表示されると、*Created sineWave1000* というメッセージは消去され、どこにも残されません。

実は、writeInfoLine という命令は、それまで Praat Info ウィンドウに表示していたものを消した上で新しいメッセージを表示する命令です。このため、一度 writeInfoLine を使うと、今まで Praat Info ウィンドウに表示されていたものはなくなってしまいます。

今まで表示していたものを消さずに新しいメッセージを追加するには、writeInfoLine の代わりに、appendInfoLine を使います。

参考 英語で “append” は「付け加える」という意味です。

以下は、先ほどのスクリプトの 4 行目の writeInfoLine を appendInfoLine に置き換えたものです。このスクリプトを実行すると、Praat Info ウィンドウには、*Created sineWave1000* がまず表示され、その下の行に *Played sineWave1000* が表示されるはずです。

```
1 do("Create␣Sound␣from␣formula...",␣"sineWave1000",␣1,
      ␣0.0,␣1.0,␣44100,␣"sin(2*pi*1000*x)")
2 writeInfoLine("Created␣sineWave1000")
3 do("Play")
4 appendInfoLine("Played␣sineWave1000")
5 do("Remove")
```

重要 Praat のスクリプトでは、メッセージを新しく出すときには writeInfoLine を、その後に追加でメッセージを出すときには appendInfoLine を使う。

多くの場合、スクリプトの最初のところだけ writeInfoLine を用い、他は appendInfoLine を用いることになると思います。

参考 Praat のスクリプトでメッセージを表示する命令には、他に writeInfo と appendInfo というのがあります。先ほど扱った writeInfoLine と appendInfoLine は、メッセージを表示した後、改行を行います。これに対して、writeInfo と appendInfo は改行を行いません。つまり、writeInfoLine("foo") を実行した上で、appendInfoLine("bar") を実行すると、1 行目に *foo* が表示され、2 行目に *bar* が表示されます。これに対し、writeInfo("foo") を実行した上で、appendInfo("bar") を実行すると、1 行に *foobar* と表示されます。なお、writeInfo は writeInfoLine と同様に、Praat Info ウィンドウに表示していたものをすべて消してからメッセージを表示します。これに対し、appendInfo は appendInfoLine と同様に今まで表示されているものの後にメッセージを表示します。

復習

1. Praat のメニューで、ピリオド 3 つ（...）で終わっているものとそうでないものの違いは何ですか。
2. 正弦波とは何ですか。Praat ではどのようにして正弦波の音を作ることができますか。
3. のこぎり波とは何ですか。Praat ではどのようにしてのこぎり波の音を作ることができますか。
4. writeInfoLine と appendInfoLine の違いは何ですか。

第 3 課　周波数を変えよう その 1

この課では、周波数を変えると音がどのように変わるかについて試します。

この課を終えると、次のようなことができるようになります。

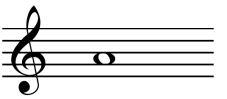
- Praat のスクリプトで変数を用いる
- プログラムの内容を示すためにコメントを書く
- if 構造を用いて、プログラムの条件分岐を行う

3.1　周波数の違いで音がどう変わるか確かめよう

前の課では、次のようにして、1000 Hz の正弦波を作成しました。

```
1 do("Create␣Sound␣from␣formula...",␣"sineWave1000",␣1,
      ␣0.0,␣1.0,␣44100,␣"sin(2*pi*1000*x)")
2 writeInfoLine("Created␣sineWave1000")
3 do("Play")
4 appendInfoLine("Played␣sineWave1000")
5 do("Remove")
```

このスクリプトの 1 行目の `sin(2*pi*1000*x)` というのが、この正弦波の形を示した数式です。`sin(2*pi*1000*x)` の `1000` が、この正弦波が 1000 Hz であるということを示しています。正弦波の周波数を変えたい場合、`sin(2*pi*1000*x)` の `1000` を別の数字に変えます。



440 Hz の正弦波を作ってみましょう。実は、440 Hz というのは、音楽で言うと、一点イ (A4) の音の基本周波数に当たります。上に掲げた楽譜で、どの音かを確認してみてください。数式は `sin(2*pi*440*x)` です。スクリプトは次のようになります。1000 Hz の正弦波を作るスクリプトを少し書き換えるだけです。どこが書き換えられたのか確認しながら、次のスクリプト

を見てください。

```
1 do("Create␣Sound␣from␣formula...",␣"sineWave440",␣1,␣
      0.0,␣1.0,␣44100,␣"sin(2*pi*440*x)")
2 writeInfoLine("Created␣sineWave440")
3 do("Play")
4 appendInfoLine("Played␣sineWave440")
5 do("Remove")
```

1000 Hz の正弦波を作るスクリプトからの変更点は次の 4 つです。1 つずつ直しましょう。

- 1 行目の数式が、`sin(2*pi*1000*x)` から、`sin(2*pi*440*x)` に。
- 1 行目で付ける音の名前が、`sineWave1000` から、`sineWave440` に。
- 2 行目で表示するメッセージが、`sineWave1000` から、`sineWave440` に。
- 4 行目で表示するメッセージが、`sineWave1000` から、`sineWave440` に。

問 3–1　確認しよう

A 1000 Hz や 440 Hz 以外の周波数の正弦波を先ほどの例にならって作成してみてください。例えば、262 Hz にするとどういう音ができるか確かめてみましょう。

3.2　変数を使って楽をしよう

1000 Hz の正弦波を作るスクリプトから 440 Hz の正弦波を作るスクリプトに変更しましたが、その際に同じ変更をした場所がありました。`sineWave1000` から、`sineWave440` に変更するということを 3 回も繰り返しているのです。3 回ぐらいならば大したことはないと思うかもしれません。しかし、もっと複雑なスクリプトを書く場合は、何十回も書き換える必要が出てきます。そうなると大変な作業になります。また、ある場所は変更したのに、別の場所は変更するのを忘れたということも出てきます。

表 3.1　音名と基本周波数の関係。A4 の基本周波数を 440.0 Hz とし、その他の音の周波数は十二平均律によって算出した。小数点第二位以下は四捨五入した。

音名	(日本語)	基本周波数 (Hz)
F3	(へ)	174.6
F#3	(嬰へ)	185.0
G3	(ト)	196.0
G#3	(嬰ト)	207.7
A3	(イ)	220.0
A#3	(嬰イ)	233.1
B3	(ロ)	246.9
C4	(一点ハ)	261.6
C#4	(一点嬰ハ)	277.2
D4	(一点ニ)	293.7
D#4	(一点嬰ニ)	311.1
E4	(一点ホ)	329.6
F4	(一点ヘ)	349.2
F#4	(一点嬰ヘ)	370.0
G4	(一点ト)	392.0
G#4	(一点嬰ト)	415.3
A4	(一点イ)	440.0
A#4	(一点嬰イ)	466.2
B4	(一点ロ)	493.9
C5	(二点ハ)	523.3
C#5	(二点嬰ハ)	554.4
D5	(二点ニ)	587.3
D#5	(二点嬰ニ)	622.3
E5	(二点ホ)	659.3
F5	(二点ヘ)	698.5
F#5	(二点嬰ヘ)	740.0
G5	(二点ト)	784.0
G#5	(二点嬰ト)	830.6
A5	(二点イ)	880.0

こういった問題が起きないように、プログラムを作るときは、同じ作業はまとめて処理できるように組んでいきます。

> **重要** プログラムを作る場合、同じ作業はまとめられないか常に考える。

さて、変更する量を減らすには、sineWave1000 や sineWave440 と書く場所を 1 つに限定してしまうのが一番楽です。このようなときに便利なのが**変数**です。

Praat のスクリプトの変数には、2 種類あります。**数値変数**と、**文字列変数**です。数値変数は文字通り数値を格納する変数です。これに対して、文字列変数は文字列を格納します。

Praat のスクリプトでは、英語の大文字・小文字、それに数字を使って、変数の名前を自由につけることができます。ただし、変数の名前の最初の 1 文字は英小文字でなくてはなりません。このため、userA や user1 というのは問題ないのですが、Auser や 1user というのは変数の名前として許されません。

文字列変数は、末尾にドル記号を付けなくてはなりません。数値変数の末尾にはドル記号を付けてはいけません。例えば、ant, student1 は数値変数となりますし、elephant$, teacher1$は文字列変数になります。

変数の名前で、大文字と小文字は区別されます。例えば、dog, dOg, dOG, doG はそれぞれ別の変数と見なされます。

まず、**数値変数**から見てみましょう。次のスクリプトを見てください。

```
1 a␣=␣50
2 b␣=␣24
3 c␣=␣a+b
4 writeInfoLine(c)
5 d␣=␣a-b
6 appendInfoLine(d)
```

まず 1 行目で、a という数値変数に 50 という数値を代入しています。Praat では、変数に代入を行うときに、イコール (=) を使います。イコール

の右側にあるものを、イコールの左側にあるものに格納すると考えてください。つまり、ここでは a の中身を 50 にしているわけです。同様に 2 行目では、b という数値変数に 24 という数値を代入しています。

こうやって代入すれば、色々なところの計算で使えます。例えば、3 行目では a と b の和を別の数値変数 c に代入しています。a には 50 が、b には 24 が格納されているわけですから、a+b は $50 + 24 = 74$ となります。つまり、c には 74 という数値が代入されます。a や b に代入する数値が変われば、a+b の結果も変わるので、c の中身も変わります。同様に 5 行目では a と b の差を数値変数 d に代入しています。

4 行目や 6 行目のように、writeInfoLine や appendInfoLine を使うと、数値変数に格納されている数値を表示することができます。以前、writeInfoLine や appendInfoLine を使ったときは、writeInfoLine("Created sineWave1000") のように、表示したい内容をダブルクォーテーションで囲みました。これに対して、変数の中身を表示する場合は、変数をダブルクォーテーションで囲む必要はありません。ダブルクォーテーションで囲んで writeInfoLine("c") としてしまうと、変数の中身ではなく、*c* という文字そのものが出力されてしまいます。

> 問 3–2　確認しよう
>
> Ⓐ 先ほどのスクリプトで、4 行目が writeInfoLine で、6 行目が appendInfoLine が用いられているのはなぜでしょうか。
>
> Ⓑ 先ほどのスクリプトで、1 行目で a に代入する数値を変更して、結果がどう変わるか確かめてください。また、2 行目で b に代入する数値を変更してみてください。
>
> Ⓒ 先ほどのスクリプトで、4 行目と 5 行目の間に a = 30 という行を挿入するとどうなりますか。結果を確かめてください。

さて、正弦波を合成するスクリプトの話に戻りましょう。今度は**文字列変数**を使います。次のスクリプトでは、soundName$という文字列変数に、*sineWave1000* という文字列を代入しています。先ほど述べたように、Praat では、変数に代入を行うときに、イコール（=）を使います。イコールの右

側にあるものをイコールの左側にあるものに格納するというのは、数値変数でも文字列変数でも同じです。ここで *sineWave1000* が文字列であるということをはっきりさせるために、ダブルクォーテーションで囲っています。Praat で文字列を文字列変数に代入する際は、このようにダブルクォーテーションで囲う必要があります。スクリプトの 2 行目では、writeInfoLine を使って、soundName$に格納されているものを表示しています。ここでは、*sineWave1000* と表示されるはずです。

```
1 soundName$ = "sineWave1000"
2 writeInfoLine(soundName$)
```

writeInfoLine を使って変数の中身を表示させるときは、変数をダブルクォーテーションで囲んでしまわないようにしてください。ダブルクォーテーションで囲んでしまったら、*soundName$*という文字列そのものが表示されてしまいます。

問 3–3　確認しよう

Ⓐ 先ほどのスクリプトで、代入する内容を *sineWave1000* から *sineWave440* にすると、結果はどうなりますか。どうなるか想像した上で、実際に試してみましょう。

Ⓑ 先ほどのスクリプトで、soundName$ = "sineWave1000" のダブルクォーテーションを外して、soundName$ = sineWave1000 とするとどうなるか実際に試してみましょう。

Ⓒ 上で writeInfoLine(soundName$) でダブルクォーテーションを加えて、writeInfoLine("soundName$") とするとどうなりますか。どうなるか想像した上で、実際に試してみましょう。

文字列変数を使って、正弦波を合成するスクリプトを作り直してみましょう。作り直した後のスクリプトは次のようになります。作り直したスクリプトがちゃんと動くか確認してください。

```
1 soundName$␣=␣"sineWave1000"
2 do("Create␣Sound␣from␣formula...",␣soundName$,␣1,␣
      0.0,␣1.0,␣44100,␣"sin(2*pi*1000*x)")
3 writeInfoLine("Created␣",␣soundName$)
4 do("Play")
5 appendInfoLine("Played␣",␣soundName$)
6 do("Remove")
```

基本的に言えば、今まで sineWave1000 と書いていたところを soundName$に置き換えています。ただ、3 行目と 5 行目の書き方は今まで出てこなかったものなので、ここで説明したいと思います。3 行目では、*Created* という文字列と soundName$という変数の中身とを表示する必要があります。文字列そのものを出力する場合は writeInfoLine("hoge") のようにダブルクォーテーションで囲めばよいのでした。これに対して、変数の中身を出力する場合は、ダブルクォーテーションで囲まずに、writeInfoLine(soundName$) のようにしなくてはならないのでした。ここではダブルクォーテーションで囲むものと囲まないものを合わせて表示する必要があります。合わせるために、コンマで区切ります。だから、writeInfoLine("Created ", soundName$) のようにコンマで 2 つの部分に分けたのです。5 行目も同様です。

さて、先ほど 1000 Hz の正弦波を作るスクリプトからの 440 Hz のスクリプトにしたときは変更点が 4 つもありました。しかし、今のスクリプトでは 2 つ直すだけで済みます。

- 1 行目で代入する文字列を、sineWave1000 から、sineWave440 にする。
- 2 行目の数式を、sin(2*pi*1000*x) から、sin(2*pi*440*x) にする。

しかし、もっと楽にできるはずです。この 2 つの変更点は、いずれも 1000 を 440 に変えるだけのものです。1000 を 440 に変えるという点で共通しているので、これらをまとめて処理することができるはずです。

変数を使って楽をします。スクリプトを次のようにすると、1 行目の 1000 という数値を書き換えるだけで、音の名前も数式も一気に変更されます。

```
1 frequency␣=␣1000
2 soundType$␣=␣"sineWave"
3 soundName$␣=␣soundType$␣+␣string$(frequency)
4 do("Create␣Sound␣from␣formula...",␣soundName$,␣1,␣
      0.0,␣1.0,␣44100,␣"sin(2*pi*frequency*x)")
5 writeInfoLine("Created␣",␣soundName$)
6 do("Play")
7 appendInfoLine("Played␣",␣soundName$)
8 do("Remove")
```

ここで一番ややこしいのは 3 行目の記述なので、これについて説明したいと思います。3 行目では、*sineWave* と *1000* をくっつけて、*sineWave1000* という文字列を作っています。*sineWave* は soundName$という文字列変数に格納されています。また、1000 は frequency という数値変数に格納されています。文字列と数値をくっつけて、*sineWave1000* という文字列を作るのが目標です。しかし、Praat では、文字列と数値を厳密に区別します。このため、そのままでは、文字列と数値をくっつけることができません。よって、まず数値を文字列に置き換えます。ここで役に立つのが、string$という関数です。これには数値を文字列に変換する働きがあります。例えば、string$(42) とすると、数値の 42 を文字列の *42* に変換します。人間からすると見た目は 42 のままで変わらないのですが、Praat からすると数値から文字列へと大きく変わったことになります。何はともあれ、string$を使うことで、文字列扱いされる *1000* が出てきます。これで、文字列である *sineWave* と対等の立場となり、くっつけることができます。文字列と文字列をくっつけるには、プラス (+) を使います。プラスの左側の文字列の後に、プラスの右側の文字列をつなげるのです。ここでのプラス (+) は数値の足し算をしているわけではないのです。

3.3 コメントを書いて分かりやすくしよう

さて、スクリプトを修正していくうちに、内容がどんどん長くなってしまいました。今このドリルを解いている間は、内容をちゃんと理解していても、後で見直したときには内容が分からなくなってしまうかもしれません。このため、後から見直す際の覚え書きがあるとよいでしょう。

こういったときに便利なのが**コメント**です。コメントとして書いた記述は、Praat には無視されますので、覚え書きとして自由に使用できます。Praat では、行頭にシャープ (#) を入れることで、その行がコメントと見なされます。スクリプトでの他の英数字と同様に、コメントのためのシャープはいわゆる半角文字の方を入力してください。

> **重要** プログラムを書くときはコメントをしっかり書こう。

次のスクリプトの 1 行目と 4 行目がコメントです。

```
1  # Input the frequency
2  frequency = 1000
3  soundType$ = "sineWave"
4  # Make the name of the sound
5  soundName$ = soundType$ + string$(frequency)
6  do("Create Sound from formula...", soundName$, 1,
       0.0, 1.0, 44100, "sin(2*pi*frequency*x)")
7  writeInfoLine("Created ", soundName$)
8  do("Play")
9  appendInfoLine("Played ", soundName$)
10 do("Remove")
```

コメントを書き入れておけば、後で見直すときに助かります。また、コメントをしっかり書いておくと、別の人にプログラムを見てもらうときにも役立ちます。

3.4 条件分けをしよう

先ほど作ったスクリプトでは、frequency という数値変数に、作りたい音の周波数の値を格納しました。ところで、frequency は数値変数ですから、数値なら何でも入れることができます。例えば、-440 のような負数や 620000 のようなとても大きな数を入れることすらできます。

問 3–4　確認しよう

Ⓐ frequency に 440 と指定した場合と、-440 を指定した場合の音を比べてみてください。

Ⓑ frequency に 620000 のような大きな数値を指定した場合、音はどうなると思いますか。考えた上で、実際に試してみましょう。

人間が聞くことができる音の範囲は 50 Hz から 20000 Hz と言われていますから、この範囲に当てはまらない周波数を frequency に入れる意味はありません。ですので、frequency に 50 未満の値、あるいは 20000 を超えた値が代入された場合は音を合成しないようにしましょう。

スクリプトではどう処理すればよいのでしょうか。今実行したいことは、frequency に格納された値によって音を合成するかしないかを決めるということです。こういうときに便利なのが、if を使った条件分けです。

次の例では、a に格納されている数値が正なら b$に *positive* を、負なら b$に *negative* を、さもなければ、b$に *zero* を代入します。そして、最後に、b$の中身を表示します。

```
1  a␣=␣-52
2  if␣a␣>␣0
3  ␣␣b$␣=␣"positive"
4  elsif␣a␣<␣0
5  ␣␣b$␣=␣"negative"
6  else
7  ␣␣b$␣=␣"zero"
8  endif
9  writeInfoLine(b$)
```

1 行目で a に代入する数値を変更した場合にどうなるか、色々試してみてください。

if 構造では、if と同じ行に条件を書き、それ以降の行に条件に当てはまる場合の命令を書きます。どこまでが if で条件分けしている範囲かを分かりやすくするため、このスクリプトでは行頭にスペースを入れて読みやすくしています。このことをインデントをつけると言います。

重要 インデントをつけることで `if` 構造を見やすくする。

なお、Praat は行頭に空白があってもなくても気にしません。あくまでも人間にとって見やすくするためにインデントをつけるのです。

問 3–5　確認しよう

Ⓐ 先ほどのスクリプトで 1 行目で `a` に代入する値を変えたらどうなるでしょうか。色々な数値を代入してみましょう。

Ⓑ 先ほどのスクリプトに適切なコメントを加えてください。

さて、50 未満の数値や 20000 を超えた数値を排除しましょう。なお、`exit` は、後に続く文字列を出力して、その場で終了する命令です。ここでは、`frequency` が 20000 より大きければ、4 行目の `exit` により *Too big!* というメッセージを出して終了します。また、`frequency` が 50 未満ならば、6 行目の `exit` により *Too small!* というメッセージを出して終了します。

```
1  # Input the frequency
2  frequency = 1000
3  if frequency > 20000
4  	exit Too big!
5  elsif frequency < 50
6  	exit Too small!
7  else
8  	soundType$ = "sineWave"
9  	# Make the name of the sound
10 	soundName$ = soundType$ + string$(frequency)
11 	do("Create Sound from formula...", soundName$, 1,
       0.0, 1.0, 44100, "sin(2*pi*frequency*x)")
12 	writeInfoLine("Created ", soundName$)
13 	do("Play")
14 	appendInfoLine("Played ", soundName$)
15 	do("Remove")
16 endif
```

参考 `exit` の書き方は将来の Praat のバージョンアップで、`writeInfoLine` などの書き方に合わせて、`exit("Too big!")` のように変更される可能性が高いと考えられます。

> 問 3–6　確認しよう
>
> A 先ほどのスクリプトで、`frequency` の値が 30 ならどうなりますか。180 ならどうなりますか。
>
> B 50 Hz でなく 70 Hz 未満の音を排除したい場合は、先ほどのスクリプトをどう変更すればよいですか。

復習

1. 一点イ (A4) の音の基本周波数は何 Hz ですか。一点ハ (C4) の音の基本周波数は何 Hz ですか。
2. 周波数が 2 倍になると音はどのように変わりますか。220 Hz, 440 Hz, 880 Hz の音を作って聞き比べてみましょう。
3. なぜプログラムを書くときは、コメントを書く必要があるのですか。
4. Praat のスクリプトで文字列と文字列をつなぐ場合にはどうすればよいですか。
5. `exit` とは何をする命令ですか。
6. Praat では、`if` 構造をどのように書きますか。
7. なぜ `if` 構造を書くときには、インデントをつけるのですか。

第 4 課　周波数を変えよう その 2

この課では、周波数の異なった音を一気に作成する方法について紹介します。

この課を終えると、次のようなことができるようになります。

- for 構造を用いて、繰り返しの作業を行う

4.1　繰り返しの方法を覚えよう

今までのスクリプトでは、音を 1 回しか再生しませんでした。複数回再生したい場合はどうすればよいでしょうか。スクリプトの一部をコピーしていけば、同じような作業を繰り返してみせることが一応できます。例えば、次のスクリプトでは 1000 Hz の正弦波を作成して 3 回再生した上で、作成した音を削除します。

```
1 do("Create␣Sound␣from␣formula...",␣"sineWave1000",␣1,
      ␣0.0,␣1.0,␣44100,␣"sin(2*pi*1000*x)")
2 do("Play")
3 do("Play")
4 do("Play")
5 do("Remove")
```

しかし、プログラミングでコピーは禁物です。もし、先ほどのスクリプトで、最初に Play の代わりに、Pray と誤って書いてしまったらどうなるでしょうか。誤ったのに気づかずコピーをしてしまったとします。すると、次のようになるはずです。

```
1 do("Create␣Sound␣from␣formula...",␣"sineWave1000",␣1,
      ␣0.0,␣1.0,␣44100,␣"sin(2*pi*1000*x)")
2 do("Pray")
3 do("Pray")
4 do("Pray")
5 do("Remove")
```

これではスクリプトがちゃんと動きませんので、Pray を Play に直す必要が出てきます。コピーを繰り返したので、3 回も直さないといけません。これでは修正が面倒です。また、今は誤っているところがまとまって出ているので修正しやすいのですが、複雑なスクリプトでは修正する場所があちらこちらに分散してしまい、こちらは直したがあちらは直していないということになりかねません。

参考 先ほどのスクリプトにはもう 1 つ問題があります。それは繰り返す回数を変更しにくいということです。先ほどのスクリプトでは、再生回数は 3 回と決まっています。再生回数を 5 回とか 9 回とかに変えたい場合は、do("Play") を何度もコピーしなくてはなりません。それよりも、もっと汎用的なスクリプトにすべきです。例えば、再生回数を格納する数値変数を用意しておき、そこに 5 という数値を入れれば 5 回再生し、9 という数値を入れれば 9 回再生するといったようにした方が望ましいのです。

コピーを使わずに繰り返したい場合、どうすればよいでしょうか。Praat で繰り返しの作業を行うには、for 構造を使うのが便利です。例えば、次のスクリプトでは、for 構造を用いて 1 から 5 までの整数の和がいくつになるかを求めています。

```
1 sum␣=␣0
2 for␣i␣from␣1␣to␣5
3 ␣␣sum␣=␣sum␣+␣i
4 endfor
5 writeInfoLine(sum)
```

このスクリプトの 1 行目では、合計を入れるための箱として、sum という数値変数を用意しています。この sum に、1 を加え、2 を加え、3 を加え、4 を加え、最後に 5 を加えることで、1 から 5 までの整数の和を求めています。

2 行目から 4 行目が繰り返しを示す for 構造です。for 構造は、for で始まり、endfor で終わります。そして、繰り返したい作業内容は、for と endfor の間に記述します。if のときと同様に、for と endfor に挟まれたものはインデントをつけると見やすくなります。

重要 作業を繰り返し行う際には、繰り返したい作業を for と endfor で挟み込む。

このスクリプトの 2 行目では、for i from 1 to 5 と書いてあります。これは、数値変数 i が 1 の時から始め、5 になる時まで同じ作業を繰り返せという意味です。つまり、i が 1 の場合、i が 2 の場合、i が 3 の場合、i が 4 の場合、i が 5 の場合の計 5 回の繰り返しが行われることになります。ここでは i という名前の変数を設定しましたが、名前は別に i でなくともかまいません。例えば、for abcde from 1 to 5 とすれば、abcde が 1 の場合から 5 の場合まで計 5 回繰り返されます。また、この例では、1 から 5 としましたが、他の数でも問題ありません。for i from 18 to 24 ならば、i が 18 の場合から 24 の場合まで繰り返すことになります。

参考 この課では、繰り返し作業を行う手法として、for 構造を紹介しました。Praat には、他にも while と endwhile で囲んで作る繰り返しや、repeat と until で囲んで作る繰り返しがあります。

問 4–1　確認しよう

Ⓐ 先ほどのスクリプトの 3 行目の sum = sum + i がどういう操作なのか説明してください。等号 (=) の左側の sum と右側の sum に入っている数がどうなっているかについて着目してください。また、i が 1 の場合、3 行目ではどんな計算が行われますか。i が 2, 3, 4, 5 の場合はどうですか。

Ⓑ 先ほどのスクリプトを変更して、1 から 15 までの整数の和を求めてください。

Ⓒ 先ほどスクリプトを変更して、38 から 74 までの整数の和を求めてください。

次のスクリプトを実行すると、1000 Hz の正弦波の音が 3 回流れます。

参考 ここでは先ほどのスクリプトと異なり、for で値が変化する変数（ここでは i）が for と endfor の間で使われていません。こういうこともあります。

```
do("Create␣Sound␣from␣formula...",␣"sineWave1000",␣1,
    ␣0.0,␣1.0,␣44100,␣"sin(2*pi*1000*x)")
for␣i␣from␣1␣to␣3
␣␣do("Play")
endfor
do("Remove")
```

問 4–2　確認しよう

A このスクリプトを変更して、音を再生する回数を 5 回にしてみてください。

4.2　周波数の数列を作ろう

100 Hz, 200 Hz, 300 Hz, 400 Hz, 500 Hz という 5 つの正弦波の音を作ることを考えましょう。つまり、周波数の数値を 100 ずつ増やしていくのです。

100 ずつ増やせれば楽なのですが、for 構造では数値を 1 つずつしか増やすことができません。このため、少し工夫をする必要があります。次のスクリプトを実行すると、100, 200, 300, 400, 500 という数列が出力されます。

```
writeInfo("Frequencies:␣")
for␣i␣from␣1␣to␣5
␣␣frequency␣=␣100␣*␣i
␣␣appendInfo(frequency)
␣␣appendInfo("␣")
endfor
appendInfo(newline$)
```

問 4–3　確認しよう

Ⓐ writeInfo と writeInfoLine にはどのような違いがありますか。このスクリプトの 1 行目を writeInfoLine に書き換えることで違いを確認してください。

Ⓑ このスクリプトの for 構造（2 行目から 6 行目）が何をしているか順を追って説明してください。

Ⓒ このスクリプトの 4 行目は、なぜ writeInfo でなく、appendInfo なのでしょうか。第 2 課で学んだことを踏まえて説明してください。

Ⓓ このスクリプトの 4 行目と 5 行目を 1 つにまとめてください。すなわち、appendInfo を 1 回しか使わない形に改めてください。

Ⓔ このスクリプトの最終行はどんなことをしていますか。この行を取り払うことで、この行の働きを確かめてください。取り払う前と取り払う後で、スクリプト実行終了時の Praat Info ウィンドウでのカーソルの位置に着目してください。その結果を踏まえて newline$がどういう意味を持つか説明してください。

Ⓕ このスクリプトを書き換えて、200, 400, 600, 800, 1000 という数列を出力するようにしてください。

Ⓖ このスクリプトを書き換えて、1000, 1100, 1200, 1300, 1400 という数列を出力するようにしてください。

Ⓗ 110, 220, 440, 880, 1760, 3520 という 2 倍になっていく数列をこのスクリプトのように出力するには、どうすればよいでしょうか。for 構造をうまく使って処理してください。

4.3　さまざまな周波数の音を一度に作ろう

今までのことを踏まえて、さまざまな周波数の音を一度に作るスクリプトを書きましょう。具体的に言えば、100 Hz, 200 Hz, 300 Hz と周波数を 100

ずつ増やしていき、500 Hz までの正弦波の音をつくります。合成される音は合わせて 5 個です。なお、話を簡単にするために、frequency に 50 未満の値、あるいは 20000 を超えた値が代入された場合の処理は割愛しました。

```
# Input the main frequency
multiplier = 100
writeInfoLine("Making Sounds")
soundType$ = "sineWave"

for i from 1 to 5
  frequency = multiplier * i
  # Make the name of the sound
  soundName$ = soundType$ + string$(frequency)
  do("Create Sound from formula...", soundName$, 1,
      0.0, 1.0, 44100, "sin(2*pi*frequency*x)")
  appendInfoLine("Created ", soundName$)
  do("Play")
  appendInfoLine("Played ", soundName$)
  do("Remove")
endfor

appendInfoLine("Finished")
```

復習

1. なぜプログラミングでコピーは禁物なのでしょうか。
2. この課の最後のスクリプトを書き換えて、200 Hz, 400 Hz, 600 Hz と周波数を 200 ずつ増やしていき、1000 Hz までの正弦波の音が合成されるスクリプトにしてください。
3. この課の最後のスクリプトを書き換えて、1000 Hz, 1100 Hz, 1200 Hz と周波数を 100 ずつ増やしていき、1400 Hz までの正弦波の音が合成されるスクリプトにしてください。
4. この課の最後のスクリプトを書き換えて、110 Hz, 220 Hz, 440 Hz, 880 Hz, 1760 Hz, 3520 Hz の正弦波の音が合成されるスクリプトにしてください。
5. この課の最後のスクリプトを書き換えて、正弦波の音の代わりにのこ

ぎり波の音を出すスクリプトにしてください。なお、1000 Hz ののこぎり波の数式は、`1000*x-floor(1000*x)` でした。

6. この課の最後のスクリプトに、`frequency` に 50 未満の値、あるいは 20000 を超えた値が代入された場合の処理を付け加えてください。

第 5 課　入力用ウィンドウを作ろう

この課では、今までに書いたスクリプトをもっと使いやすい形にします。スクリプトのしくみを知らない人でも色々な設定ができるように、入力用ウィンドウを作成します。

この課を終えると、次のようなことができるようになります。

- Praat のスクリプトを使って、フォームと呼ばれる入力用ウィンドウを表示する
- Praat のスクリプトのフォームで、選択欄を作る

5.1　簡単に周波数を入力できるようにしよう

前の課の最後のスクリプトはかなり複雑なものになっていました。一旦、単純なものに戻しましょう。次のスクリプトは、frequency に周波数を代入して、その周波数の正弦波の音声を合成し、再生した上で、できあがった音声を削除するものです。

参考　今までのスクリプトでは、1000 Hz の正弦波なら sineWave1000 と、440 Hz の正弦波なら sineWave440 と、合成する音によってその音の名前を変えていましたが、ここでは話を単純にするために、単に sineWave と名付けることにしました。次のスクリプトの 2 行目を確認してください。

```
1 frequency␣=␣1000
2 do("Create␣Sound␣from␣formula...",␣"sineWave",␣1,␣
      0.0,␣1.0,␣44100,␣"sin(2*pi*frequency*x)")
3 do("Play")
4 do("Remove")
```

このスクリプトでは、1 行目で frequency に 1000 を代入しています。別の周波数の正弦波が必要ならば、ここの 1000 を別の数値に書き換えます。ここまで Praat のスクリプトを勉強してきたみなさんならば、スクリプトでの数値を変えるのは簡単でしょう。しかし、スクリプトのしくみが分からな

い人は、どこを変えてよいか分からないかもしれません。

参考 スクリプトが分かっている人でも、間違えて別の所を書き換えてしまってスクリプトが動かなくなるということがあるかもしれません。スクリプトはできればあまりいじりたくないのです。

ですから、もっと数値を変更しやすくする必要があります。スクリプトを用いずに、Praat のメニューから音を合成したときは、入力欄がある設定用ウィンドウが出てきました。このようなウィンドウを提示できれば、スクリプトのしくみが分からない人でも、簡単に設定できるようになるはずです。

Praat のスクリプトには**フォーム**というものがあり、これを使うと設定を行うためのウィンドウを出すことができます。Praat でフォームを作るのはとても簡単です。スクリプトの中で `form` と記述するだけで、フォームを作ることができます。

先ほどのスクリプトに周波数を指定するためのフォームを加えたものが、次のスクリプトです。このスクリプトを入力して、実行してみましょう。

```
1 form␣Input␣frequency
2 ␣␣positive␣frequency␣1000
3 endform
4
5 do("Create␣Sound␣from␣formula...",␣"sineWave",␣1,␣
      0.0,␣1.0,␣44100,␣"sin(2*pi*frequency*x)")
6 do("Play")
7 do("Remove")
```

このスクリプトを実行すると、*Input frequency* というタイトルのウィンドウが出てくるはずです。出てきたウィンドウには、*frequency:*というラベルと、`1000` と表示された入力欄があるはずです。また、入力欄の下には、いくつかのボタンが並んでいるはずです。とりあえず、入力欄にある `1000` という数値は変えずに、`OK` というボタンを押しましょう。すると、今までと同じように、1000 Hz の正弦波の音が流れるはずです。

もう一回このスクリプトを実行してみましょう。先ほどと同じように *Input frequency* というウィンドウが出てくるので、`1000` という数値を `440` に変えて、`OK` ボタンを押してください。すると、440 Hz の正弦波が流れます。このスクリプトを何回か実行してみて、入力欄の数値を様々なものに変

え、周波数が変わることを確認してみてください。

先ほどのスクリプトで、どうやってフォームを作ったのか解説します。

まず、1 行目で、form Input frequency と書きました。フォームについての記述は、form と書くことから始めます。form の直後に書かれている Input frequency が、フォームのタイトルになります。フォームの記述を終えるには endform と書きます。先ほどのスクリプトでは 3 行目に書いてあります。どんな入力欄を設定するかについては form と endform の間に書きます。繰り返しを記述したときに for と endfor の間に繰り返し内容を書いたのと似ています。

> **重要** フォームを作るときは、form と endform を用意する。
> フォームの入力欄の設定は form と endform の間に書く。

フォームについての記述をスクリプトに書くときは、form と endform に挟まれたところにインデントをつけると見やすくなります。先ほどのスクリプトでは、2 行目にインデントをつけています。こうすることで、どこからどこまでがフォームの記述の範囲なのか分かりやすくなります。以前の課で if 構造や for 構造にインデントをつけたのと同じような理由です。

さて、フォームの入力欄はどのように設定するのでしょうか。Praat のスクリプトでは、1 つの行に「入力欄の種類」、「入力欄の名前」、「入力欄のデフォルト値」の 3 つの要素を順番に書くことで入力欄を作ります。これらの 3 つの要素の詳細については後ほど説明します。

先ほどのスクリプトを見てみましょう。このスクリプトでは、2 行目に positive frequency 1000 と書いてありました。これはどういう意味なのでしょうか。順を追ってみてみます。まず、positive が「入力欄の種類」です。次の frequency が「入力欄の名前」です。最後の 1000 は、「入力欄のデフォルト値」、すなわち最初に表示される値を示しています。

さて、「**入力欄の種類**」について説明しましょう。先ほどのスクリプトでは positive となっていました。これは、欄に入力できるのが正の数に限られていることを示しています。先ほどのスクリプトを実行し、*Input frequency* というウィンドウで、周波数を入れる欄に、負の数、例えば、-620 と入れ

てみてください。次のようなエラーメッセージを表示したボックスが表示されるはずです。

> “frequency” must be greater than 0.0.
> Please correct command window “Input frequency” or cancel.

これは要するに、frequency は 0.0 より大きくなくてはならないから、入力値を修正するかキャンセルしてくれという意味になります。

Praat のフォームで数値を入れることができる入力欄の種類としては、次のようなものがあります。

natural　入力する数値は、正の数でなくてはならない。またその数値は、整数でなくてはならない。つまり、正の整数しか入力できない。

positive　入力する数値は、正の数でなくてはならない。ただしその数値は、整数でも小数でもかまわない。

integer　入力する数値は、正の数でも負の数でも 0 でもかまわない。ただしその数値は、整数でなくてはならない。

real　入力する数値は、正の数でも負の数でも 0 でもかまわない。またその数値は、整数でも小数でもかまわない。

試しに、次のスクリプトを入力してフォームを作ってみてください。これは先ほどのスクリプトの positive となっていたところを、real に変えたものです。positive は正の数しか許しませんでしたが、real は負の数でも問題ありません。次のスクリプトを実行し、負の数、例えば、-620 と入れてみてください。エラーは出ないはずです。

```
1 form␣Input␣frequency
2 ␣␣real␣frequency␣1000
3 endform
```

次に 2 番目の要素である「**入力欄の名前**」について見てみましょう。入力欄の名前は、フォーム上では入力欄のラベルとして働き、フォームの外では欄に入力された値を呼び出す際の変数名として働きます。

先ほどのスクリプトを再掲します。

```
1 form␣Input␣frequency
2 ␣␣positive␣frequency␣1000
3 endform
4
5 do("Create␣Sound␣from␣formula...",␣"sineWave",␣1,␣
      0.0,␣1.0,␣44100,␣"sin(2*pi*frequency*x)")
6 do("Play")
7 do("Remove")
```

このスクリプトの 2 行目で、入力欄の名前として frequency を指定しました。これが、フォーム上では frequency: のように出力されます。また、5 行目の数式で sin(2*pi*frequency*x) とありますが、ここでの frequency は、フォームの入力欄で入れられた数値が格納されている変数です。例えば、フォームで、440 と入力すれば、frequency に 440 という数値が格納されます。

つまり、「入力欄の名前」には 2 つの役割があります。1 つは、フォーム上に出現して、入力欄がどういうものか説明するという役割です。もう 1 つは、入力欄で入力されたものをスクリプトの中から呼び出すための変数という役割です。前者はユーザのために、後者はスクリプトでの処理のためにあります。

最後の要素である「**入力欄のデフォルト値**」について説明しましょう。これは簡単で、フォームを提示するときに最初に入力欄に入っている値のことです。ユーザがフォームで値を変更しなければこの値が用いられます。もちろん、ユーザがフォームで値を変更すれば、その変更した値が用いられます。なお、フォームが表示されているときに、Standards というボタンを押すと、すべての入力欄の値がこのデフォルト値に戻ります。

5.2 フォームに説明を加えよう

先ほど作ったフォームは非常にシンプルなものでした。しかし、実際にはもっと複雑なフォームを作る場合もあります。複雑になると、慣れないユーザは何が何だか分からなくなります。このため、ユーザにとって分かりやすくするために適宜説明を加える必要が出てきます。ここでは、ユーザにもっ

と分かりやすく使ってもらうために、フォームに説明を加える方法を紹介します。

フォームに説明を加えるには comment を使うのが一番簡単です。次のスクリプトのように、comment のあとに空白を置いて、その後に表示したい説明を書きます。もちろん、これは form と endform の間に書く必要があります。フォームに関することは何でも form と endform の間に記述するのです。

参考 positive などは「入力欄の種類」、「入力欄の名前」、「入力欄のデフォルト値」の 3 つの要素からなり、各要素の間を空白で区切りました。comment はこれと異なり、2 つの要素からなっています。次のスクリプトの 2 行目の comment Please input the frequency of the wave. で言うと、comment が「入力欄の種類」に相当する部分で、残りの Please input frequency of wave. が「表示する内容」に相当する部分です。ここで、comment と Please の間の空白は要素を区切る空白、つまり「入力欄の種類」に相当する部分と「表示する内容」に相当する部分を区切る空白になっています。これに対して、Please と input の間の空白などは要素を区切る空白ではありません。これらは単語と単語を区切るためだけに用いられている空白です。

```
1 form␣Input␣frequency
2 ␣␣comment␣Please␣input␣the␣frequency␣of␣the␣wave.
3 ␣␣positive␣frequency␣1000
4 endform
5
6 do("Create␣Sound␣from␣formula...",␣"sineWave",␣1,␣
      0.0,␣1.0,␣44100,␣"sin(2*pi*frequency*x)")
7 do("Play")
8 do("Remove")
```

とりあえず、このスクリプトを入力して、実行してみましょう。そうすると、前に作ったスクリプトと似た感じで、*Input frequency* というタイトルのウィンドウが出てきます。しかし、今回は前と違って、入力欄の上に、*Please input the frequency of the wave.* と表示されているはずです。

問 5–1　確認しよう

Ⓐ 先ほどのスクリプトの 2 行目と 3 行目を入れ替えてください。どうなりますか。

Ⓑ 今扱った comment は、3.3 節で扱った#から行を始めることによって作るコメントとは働きが異なります。両者の役割を説明してください。誰のために書かれているのかと考えてみるとよいでしょう。

先ほどのフォームでは、単に frequency（周波数）と表示されるだけでした。このため、何の周波数か分からないという苦情があったとします。そこで、音の周波数だということをはっきりさせるために、ラベルを *frequency* から、*sound frequency* に改めることになったとしましょう。ここで、positive frequency 1000 を positive sound frequency 1000 にしてはなりません。このように改めると問題が発生します。

なぜ問題が発生するのかフォームの記述方法の面から見てみましょう。フォームの記述に当たっては、「入力欄の種類」、「入力欄の名前」、「入力欄のデフォルト値」という 3 つの要素をを順に書き、それぞれの要素の間には空白を入れました。このことを踏まえて、positive sound frequency 1000 を見てみます。まず positive が入力欄の種類として存在します。次に、そこから空白で区切られた先にある sound が入力欄の名前と見なされます。そして次の空白の先にある残りの frequency 1000 が、入力欄のデフォルト値と見なされてしまいます。これではおかしいというわけです。

参考　frequency と 1000 の間の空白は、要素と要素を区切る空白ではありません。フォームの入力欄は 3 つの要素からなるわけですから、要素と要素を区切る空白は 2 つあれば足りるのです。残りの空白は最後の要素、すなわち「入力欄のデフォルト値」の一部と見なされます。

それでは、フォームでの名前として空白を含むものは使えないのでしょうか。そんなことはありません。フォーム上で *sound frequency* と表示したければ、スクリプトでは空白をアンダーバーに置き換えて sound_frequency のように表記します。次がその例です。

```
form␣Input␣frequency
␣␣comment␣Please␣input␣the␣frequency␣of␣the␣wave.
␣␣positive␣sound_frequency␣1000
endform

do("Create␣Sound␣from␣formula...",␣"sineWave",␣1,␣
    0.0,␣1.0,␣44100,␣"sin(2*pi*frequency*x)")
do("Play")
do("Remove")
```

なお、sound_frequency という変数に、*sound frequency* という欄に入力した数値が入ります。要するに、ユーザに見えるところでは *sound frequency* と空白でつながった形になっているのですが、スクリプトの中ではすべて sound_frequency とアンダーバーでつながった形になっているのです。

アンダーバーでつなぐテクニックを紹介したついでに、もう 1 つテクニックを紹介したいと思います。次のスクリプトを実行して見てください。実行して出てきたフォームでは、*sound frequency (Hz)* と表示されます。この欄に入力した数値を格納する変数は、今までのパターンで行けば、sound_frequency_(Hz) となりそうですが、そうはなりません。実際、writeInfoLine(sound_frequency_(Hz)) とするとエラーが出ます。Praat では、フォームでの入力欄の名前に括弧を使用した場合、括弧の中身が無視されるという特殊なふるまいをします。つまり、*sound frequency (Hz)* と表示されている欄に入力した値は、sound_frequency という変数に格納されます。

```
form␣Input␣frequency
␣␣comment␣Please␣input␣the␣frequency␣of␣the␣wave.
␣␣positive␣sound_frequency_(Hz)␣1000
endform

writeInfoLine(sound_frequency)
```

なお、括弧の中身が無視されるというのは、入力欄の名前に限った話ではありません。次のスクリプトを見てください。

```
1 form␣Input␣frequency
2 ␣␣comment␣Please␣input␣the␣frequency␣of␣the␣wave.
3 ␣␣positive␣sound_frequency_(Hz)␣1000␣(=␣1␣kHz)
4 endform
5
6 writeInfoLine(sound_frequency)
```

デフォルトの値が 1000 (= 1 kHz) とありますが、括弧の中身は無視され、そのまま変更しなければ sound_frequency には 1000 が格納されます。

フォームを作る際のテクニックをもう 1 つ紹介しましょう。次のスクリプトを見てください。3 行目で Frequency のように大文字から始まる名前の入力欄を作っています。こうすると、フォーム上は *Frequency* と大文字で表示されます。しかし、スクリプトの中の変数としては、Frequency ではなく、frequency と見なされます。と言うのも、Praat では変数は必ず小文字から始まらないといけないという決まりがあり、大文字から始まる変数があると困るからです。

```
1 form␣Input␣frequency
2 ␣␣comment␣Please␣input␣the␣frequency␣of␣the␣wave.
3 ␣␣positive␣Frequency␣1000
4 endform
5
6 do("Create␣Sound␣from␣formula...",␣"sineWave",␣1,␣
      0.0,␣1.0,␣44100,␣"sin(2*pi*frequency*x)")
7 do("Play")
8 do("Remove")
```

> 問 5–2　確かめてみよう
>
> A このスクリプトの 6 行目の frequency を Frequency にするとどうなりますか。実際に試してみてください。

なお、Praat のフォームの入力欄には、文字列を入れるための入力欄もあります。数値を入力する際に使った positive などの代わりに、word などを用いれば簡単に文字列用の入力欄を作ることができます。文字列用の入力

欄に関しても、1 つの行に「入力欄の種類」、「入力欄の名前」、「入力欄のデフォルト値」の 3 つの要素を順番に書いていきます。

文字列に関する「入力欄の種類」には次のようなものがあります。

word　文字列で空白を含まないものが入力できる。英語で言うと 1 単語しか入れられない。

sentence　文字列が入力できる。この文字列には空白を含んでもよい。

text　文字列が入力できる。この文字列には空白を含んでもよい。基本的に sentence と同じだが、こちらの方が入力欄が長めである。その代わり、入力欄の名前が表示されなくなる。

文字の入力欄があるスクリプトの例を 1 つ挙げてみましょう。これは名前 (Given name) と名字 (Family name) を入力させて、それを表示するという単純なものです。

```
1 form␣Input␣your␣name
2 ␣␣word␣Given_name␣Taro
3 ␣␣word␣Family_name␣Yamada
4 endform
5
6 writeInfoLine("Your␣name␣is␣",␣given_name$,␣"␣",␣
      family_name$,␣".")
```

フォームの作り方は、先に周波数の数値の入力欄を作ったときと全く同じです。ただし、文字列の場合、入力した値を呼び出すには、「入力欄の名前」にドル記号をつけなくてはなりません。実際、先ほどのスクリプトでは、writeInfoLine を使うときに、given_name$のようにしています。これは、文字列変数であるため、末尾にドル記号をつける必要があるのです。

5.3　波形を選べるようにしよう

先ほどのスクリプトでは、合成される音の波形は正弦波でした。今度は、正弦波以外の波形の音も合成できるようにスクリプトを書き直したいと思います。

これから作るスクリプトの流れは次の通りになります。

1. 合成する音の詳細を決めるためのフォームを提示する
   (a) 基本周波数を入力する欄
   (b) 音声の波形（正弦波、のこぎり波、三角波）を選ぶ欄
2. フォームで指定した波形によって、数式を決定する
3. 決定した数式に基づき、音声を合成する
4. 音を流す
5. 合成した音を消す

1 番の (a)、3 番、4 番、5 番は、すでに作ったことがあります。残った 1 番の (b) と 2 番も、今までの知識を応用すればできます。

さて、まず「音声の波形（正弦波、のこぎり波、三角波）を選ぶ欄」を作るところから取りかかりましょう。フォームを作る場合、form を使えばよいのでした。このフォームに載せる「基本周波数を入力する欄」も今まで作ってきたものが流用できます。とりあえず、今まで作ってきたものを以下に掲げます。

```
1 form␣Input␣frequency
2 ␣␣comment␣Please␣input␣the␣frequency␣of␣the␣wave.
3 ␣␣positive␣Frequency_(Hz)␣1000
4 endform
```

あとはこれに波形を選ぶ欄を加えるだけです。Praat のフォームで選択欄を作るには、choice と button を使います。次のスクリプトの 4 行目から 7 行目を見てください。

```
1 form␣Input␣frequency␣and␣shape
2 ␣␣comment␣Please␣input␣the␣frequency␣of␣the␣wave␣and␣
      select␣the␣wave␣shape.
3 ␣␣positive␣Frequency_(Hz)␣1000
4 ␣␣choice␣Wave_shape␣1
5 ␣␣␣␣button␣Sine␣wave
6 ␣␣␣␣button␣Sawtooth␣wave
7 ␣␣␣␣button␣Triangle␣wave
8 endform
```

まず、choice を用いて選択欄の大枠を作っています。choice の後に続

く Wave_shape がこの選択欄の名前になります。その次の 1 はデフォルト値を表しています。選択欄のデフォルト値については後で詳しく説明します。5 行目から 7 行目では、button を使って、この選択欄の選択肢としてどういうものを設定するかを記述しています。ここでは Sine wave（正弦波）、Sawtooth wave（のこぎり波）、Triangle wave（三角波）の 3 種類の選択肢を設定しています。

参考 choice を用いて選択欄を作る場合、その選択肢となっている button はインデントしておくと後で見やすくなります。

参考 先ほどのスクリプトを見て、button Sine wave は button Sine_wave にすべきだと思った人がいるかもしれません。結論から先に言うと、空白をアンダーバーに置き換える必要はありません。positive のような普通の入力欄は、3 つの要素からなりたちます。これに対して、button は 2 つの要素のみからなりたちます。1 つは「入力欄の種類」に相当する部分で、もう 1 つは「入力欄の名前」に相当する部分です。button Sine wave で言うと、button が「入力欄の種類」に相当する部分で、Sine wave が「入力欄の名前」に相当する部分です。この 2 つの要素を区切っているのは、button と Sine の間の空白です。これ以外の空白、すなわち Sine と wave の間の空白は単なる空白で、要素を区切る役割は果たさないのです。前に空白をアンダーバーに置き換えたのは、空白ならば要素を区切る空白であると誤解される場所にあったからです。ここでは誤解されないので空白のままでよいのです。

先ほどのスクリプトを実行してみましょう。Input frequency and shape というタイトルのウィンドウが出るはずです。このウィンドウには、上から順に、(1) *Please input the frequency of the wave and select the wave shape.* という説明、(2) 周波数を入力する欄、(3) 波形を選択する欄が表示されていると思います。波形を選択する欄には、その欄のラベルとして、*Wave shape:*という文言があります。この文言はスクリプトの 4 行目の choice の後の Wave_shape に対応します。Praat のフォームでは、「入力欄の種類」、「入力欄の名前」、「入力欄のデフォルト値」の順に記述するのでした。「入力欄の種類」が choice の場合でも同じです。positive の後に書

いた Frequency_(Hz) がラベルとして登場したということを思い出してください。直後に書いたものがラベルになるというのは、choice も同じです。

さて、この *Wave shape:*という選択欄には選択肢が 3 つあります。上から順に、*Sine wave*, *Sawtooth wave*, *Triangle wave* という選択肢があります。スクリプトで記述した順番で並ぶのです。このウィンドウが表示された時は、*Sine wave* があらかじめ選択された状態になっています。あらかじめ何を選択するかは、choice のデフォルト値によって決まります。今のスクリプトでは、choice Wave_shape 1 と書いてあります。ここの 1 がデフォルト値になるわけです。これは、何番目の選択肢かを示しています。1 番目の選択肢は *Sine wave* ですから、これがあらかじめ選択された状態で表示されるのです。もし *Triangle wave* をあらかじめ選択しておきたければ、*Triangle wave* は 3 番目の選択肢ですので、choice Wave_shape 3 とします。あらかじめ選択された状態にしたくなければ、choice Wave_shape 0 とするか、デフォルト値を書かずに単に choice Wave_shape とします。

ところで、フォームでユーザが選んだ選択肢の情報は、どこにどのように格納されるのでしょうか。この情報は、choice の直後に書いた選択欄の名前と同じ変数に、選択肢の番号が格納されます。先ほどのスクリプトの例で言えば、wave_shape という変数に選択肢の番号が格納されます。もし、*Triangle wave* を選んだとすれば、これは 3 番目の選択肢なので、wave_shape に 3 という数値が格納されます。次のスクリプトは、先ほどのスクリプトに、フォームで入力・選択したものを出力する命令を付け加えたものです。実行して、どう出力されるか確かめてみましょう。

```
1  form␣Input␣frequency␣and␣shape
2  ␣␣comment␣Please␣input␣the␣frequency␣of␣the␣wave␣and␣
       select␣the␣wave␣shape.
3  ␣␣positive␣Frequency_(Hz)␣1000
4  ␣␣choice␣Wave_shape␣1
5  ␣␣␣␣button␣Sine␣wave
6  ␣␣␣␣button␣Sawtooth␣wave
7  ␣␣␣␣button␣Triangle␣wave
8  endform
9
10 writeInfoLine("Frequency:␣",␣frequency)
11 appendInfoLine("Wave␣shape:␣",␣wave_shape)
```

このスクリプトを実行しましょう。フォームで値を変更せずにそのまま OK ボタンを押したとします。つまり、周波数の欄には *1000* が入力された状態で、波形は *Sine wave* が選択された状態で OK ボタンを押したわけです。すると、Praat Info ウィンドウに次のように表示されるはずです。

Frequency: 1000
Wave shape: 1

周波数の欄に入力した値が frequency という数値変数に、波形の欄で選んだ選択肢の番号が wave_shape という数値変数に格納されているのが分かります。

このままでも大きな問題はないのですが、単に *1* と書いてあるだけでは、後から人間が見直したときにどの選択肢を選んだのか分かりにくくなります。今はフォームでの選択肢の順番を覚えているので、1 番目が *Sine wave* だということが分かります。しかし、後から見直すときは、フォームでの選択肢の順番を忘れてしまっているかもしれません。ですから、*1* のように単に選択肢の番号を書くのではなく、*Sine wave* のように選択肢の名前を表示することにします。こうすれば、人間にとって分かりやすくなります。

選択肢の番号の代わりに選択肢の名前を表示するように書き換えたのが次のスクリプトです。先ほどのスクリプトと違うのは、最後の行です。先ほどのスクリプトで wave_shape と書いたのをこのスクリプトでは wave_shape$としました。変数名の最後にドル記号を加えただけです。

```
1  form␣Input␣frequency␣and␣shape
2  ␣␣comment␣Please␣input␣the␣frequency␣of␣the␣wave␣and␣
       select␣the␣wave␣shape.
3  ␣␣positive␣Frequency_(Hz)␣1000
4  ␣␣choice␣Wave_shape␣1
5  ␣␣␣␣button␣Sine␣wave
6  ␣␣␣␣button␣Sawtooth␣wave
7  ␣␣␣␣button␣Triangle␣wave
8  endform
9
10 writeInfoLine("Frequency:␣",␣frequency)
11 appendInfoLine("Wave␣shape:␣",␣wave_shape$)
```

このスクリプトを実行し、先ほどと同じように、フォームで値を変更せず

にそのまま OK ボタンを押したとします。すると、Praat Info ウィンドウに次のように表示されるはずです。

Frequency: 1000
Wave shape: Sine wave

つまり、choice Wave_shape 1 と選択欄を記述した場合、数値変数 wave_shape には選んだ選択肢の番号が、文字列変数 wave_shape$には選んだ選択肢の名前が格納されます。

参考 Praat のスクリプトでは、通常、名前が同じであっても、数値変数と文字列変数は全くの別物として扱われます。例えば、数値変数 hoge と文字列変数 hoge$は全く無関係です。choice を使ったときのみ、このような特別なふるまいを見せます。

どの選択肢を選んだかという情報を扱う際は、数値を使った方がよいのでしょうか。それとも文字列を使った方がよいのでしょうか。それは場合によります。数値を使った方がよいときもありますし、文字列を使った方がよい場合もあります。とはいえ、何も基準がないとどちらを使えばよいか悩むでしょう。慣れないうちは次のような基準に従ってください。まず、基本的には数値を使って処理してください。ただし、writeInfoLine などを使って人間に見せる場合には、文字列を使ってください。そうすれば、大概うまくいきます。

次の目標は、フォームで選んだ波形によって、数式を決定することです。ここまでの内容が理解できれば、さほど難しいことではありません。次のスクリプトを見てください。

```
1 form␣Input␣frequency␣and␣shape
2 ␣␣comment␣Please␣input␣the␣frequency␣of␣the␣wave␣and␣
      select␣the␣wave␣shape.
3 ␣␣positive␣Frequency_(Hz)␣1000
4 ␣␣choice␣Wave_shape␣1
5 ␣␣␣␣button␣Sine␣wave
6 ␣␣␣␣button␣Sawtooth␣wave
7 ␣␣␣␣button␣Triangle␣wave
8 endform
9
```

```
#␣Set␣formula
#␣Sine␣wave
if␣wave_shape␣=␣1
␣␣formula$␣=␣"sin(2*pi*"␣+␣string$(frequency)␣+␣"*x)"
#␣Sawtooth␣wave
elsif␣wave_shape␣=␣2
␣␣formula$␣=␣string$(frequency)␣+␣"*x-floor("␣+␣
    string$(frequency)␣+␣"*x)"
#␣Triangle␣wave
elsif␣wave_shape␣=␣3
␣␣formula$␣=␣"2*(abs("␣+␣string$(frequency)␣+␣"*x-
    floor("␣+␣string$(frequency)␣+␣"*x+0.5)))-1"
endif

writeInfoLine(formula$)
```

このスクリプトを実行すると、フォームで指定した周波数と波形に基づく音がどんな数式となるのかが表示されます。このスクリプトの 1 行目から 8 行目は、フォームに関する記述です。

また、10 行目から 20 行目で、波形に合った数式を指定しています。フォームで何番目の波形を選んだかについては、数値変数 wave_shape に格納されています。10 行目から 20 行目の記述では、3 つの選択肢それぞれについて数式を個別に指定しています。1 番目の選択肢、つまり *Sine wave* を選んだ場合、wave_shape の値は 1 となりますから、12 行目の if の条件に合致し、13 行目で formula$に正弦波の数式が代入されます。同様に、2 番目の選択肢、つまり *Sawtooth wave* を選んだ場合、wave_shape の値が 2 となって、15 行目の elsif の条件に合致し、16 行目で formula$にのこぎり波の数式が代入されます。3 番目の選択肢、つまり *Triangle wave* を選んだ場合も同様で、wave_shape の値が 3 となって、18 行目の elsif の条件に合致し、19 行目で formula$にのこぎり波の数式が代入されるのです。

あとは、こうして得られた数式をもとに音を合成するだけです。次のスクリプトが完成版となります。

```
form␣Input␣frequency␣and␣shape
␣␣comment␣Please␣input␣the␣frequency␣of␣the␣wave␣and␣
    select␣the␣wave␣shape.
␣␣positive␣Frequency_(Hz)␣1000
```

```
    choice Wave_shape 1
        button Sine wave
        button Sawtooth wave
        button Triangle wave
endform

# Set formula
# Sine wave
if wave_shape = 1
    formula$ = "sin(2*pi*" + string$(frequency) + "*x)"
# Sawtooth wave
elsif wave_shape = 2
    formula$ = string$(frequency) + "*x-floor(" +
        string$(frequency) + "*x)"
# Triangle wave
elsif wave_shape = 3
    formula$ = "2*(abs(" + string$(frequency) + "*x-
        floor(" + string$(frequency) + "*x+0.5)))-1"
endif

do("Create Sound from formula...", "sound", 1, 0.0,
    1.0, 44100, formula$)
do("Play")
do("Remove")
```

復習

1. Praat のスクリプトで、フォームの個々の入力欄を描写するとき、通常は 3 つの要素が必要とされます。それらは何ですか。どんな順番で書きますか。
2. 次の条件にあったフォームを作るスクリプトを書いてください。
    - *Please tell me about you.*（あなたについて教えてください）という説明を表示する。
    - 名前 (Name) を入れる欄を作る。
    - 性別 (Sex) を男 (male) と女 (female) から選択する欄を作る。
    - 年齢 (Age) を入れる欄を作る。

第 6 課　実験画面を作ろう

この課では、Praat を使って知覚実験を行えるようにします。
この課を終えると、次のようなことができるようになります。

- ポーズ機能を使ってユーザからの入力を受け付ける
- ファイルに出力する

6.1　被験者に聞こえたかどうかを質問する

このドリルで行うべき実験の手順を確認しましょう。手順は大まかに言えば、次の通りになります。

1. ある周波数の音を合成する（周波数の値はだんだん高くする）
2. 合成した音を流す
3. その音が聞こえたか被験者に質問する
   (a) 被験者が「聞こえた」と答えた場合、最初に戻る
   (b) 被験者が「聞こえなかった」と答えた場合、そこで中断する

1 番目の「ある周波数の音を合成する」は問題なくできるはずです。適切な設定を行った上で、今までと同様に `do` と `Create Sound from formula...` を組み合わせて使えばよいのです。

2 番目の「合成した音を流す」も簡単です。例によって、`do` と `Play` を使います。

問題は 3 番目の「その音が聞こえたか被験者に質問する」です。

前の課で扱ったフォームを使えばよいだろうと考えた人がいるかもしれません。しかし、フォームはこの状況には適していません。実は 1 つのスクリプトにつき、フォームは 1 回しか使えません。この実験では、音が聞こえたかどうかを何度も質問しなくてはなりませんから、1 回しか使えないのでは困るのです。

参考　しかも、フォームを出せるのはスクリプトが動き出した直後に限られます。

こういう時に便利なのが、Praat スクリプトのポーズ (pause) 機能です。**ポーズ機能**は、基本的に言えば、ユーザからの返事を待つための機能です。フォームと違って、1 つのスクリプトの中で何度も使うことができます。

> **重要** フォームはスクリプトの中で 1 回しか使えない。ポーズは複数回使うことができる。

ポーズ機能を働かせるための最も簡単な方法は、pause と打ち込んで、その後に表示したいメッセージを入力することです。次のスクリプトを入力して、実行してみてください。

```
1 writeInfoLine("Start")
2 pause␣Are␣you␣ready?
3 writeInfoLine("Done")
```

このスクリプトを実行すると、まず 1 行目の記述により Praat Info ウィンドウに *Start* というメッセージが表示されます。その後、2 行目の記述により、即座に Pause: stop or continue というタイトルのウィンドウが出てきます。このウィンドウには *Are you ready?*というメッセージが表示されているはずです。先ほど、スクリプトの 2 行目に pause Are you ready?と書きましたが、pause の後に書いた Are you ready?というメッセージが、ウィンドウにそのまま表示されるわけです。また、ウィンドウの下部には、Stop と Continue というボタンが配置されています。このウィンドウが表示されている間は、Praat はユーザの返事を待っている状態になります。待っている間は、Praat は何もしない状態になります。ユーザが返事をするには、ウィンドウ下部のボタンのいずれか、ここでは Stop か Continue のいずれかを押します。

とりあえず、Continue というボタンを押してください。*Continue* とは英語で「続ける」という意味です。すると、スクリプトの 3 行目の記述により writeInfoLine が実行され、Praat Info ウィンドウに Done という文字列が表示されます。ここでの Praat の動きを確認します。まず、2 行目の pause で、ユーザからの返事を受け取るための Pause: stop or continue というウィンドウを提示し、ユーザからの返事が来るまで Praat

は待機している状態になります。2 行目で待機しているので、3 行目の命令は実行されなかったわけです。ここで、ユーザが Continue というボタンを押します。すると、Praat は「続けてもよい」と認識し、スクリプトの続き、ここでは 3 行目の writeInfoLine を実行します。つまり、Continue を押すことで、文字通り、スクリプトの「続き」を実行したわけです。

さて、先ほどの Pause: stop or continue というタイトルのウィンドウで、Continue というボタンを押す代わりに、Stop というボタンを押すとどうなるのでしょうか。Stop とは英語で「止める」という意味です。このボタンを押すと、そこでスクリプトが中断されます。実際に試してみましょう。先ほどのスクリプトをもう 1 度実行してみてください。すると、前と同じようにウィンドウが出てくるので、Stop ボタンを押してください。

Stop ボタンを押すと、次のようなエラーメッセージが表示されます。

> You interruputed the script.
> Script line 2 not performed or completed:
> « pause Are you ready? »
> Menu command “Run” not completed.

このエラーメッセージは、「あなたはスクリプトを中断しました。スクリプトの 2 行目が終了しませんでした。」ということを意味しています。Stop ボタンを押すと、その場でスクリプトの実行が中断されます。ここでは、2 行目に pause を書いたので、2 行目で中断します。中断されたのですから、それ以降の記述は実行されません。ここでは、3 行目の writeInfoLine が実行されず、Praat Info ウィンドウに Done という文字列が表示されることはありません。このため、Praat Info ウィンドウには 1 行目の writeInfoLine で表示された Start という文字列が残ったままになっているはずです。

このように pause を使うと、続けるか中断するかをユーザに選択させることができます。このしくみを使えば、音が聞こえたかどうかを被験者に質問し、聞こえたら続け、聞こえなかったら中断するということができます。

```
1 pause␣Did␣you␣hear␣any␣sounds?
```

例えば、このようにスクリプトを書けば、被験者に *Did you hear any sounds?*（何か音が聞こえましたか？）というメッセージを表示して、

Continue か Stop かをユーザに選ばせることができます。

しかし、このままでは被験者にとって使いづらい面があります。「音が聞こえたか」と質問されたのに、答えとしては Continue（続ける）と Stop（中断する）の 2 つしかないというのは変です。ここはもっと分かりやすく、Yes（はい）と No（いいえ）の 2 つの選択肢を挙げたいところです。

実は、うまく設定することで、ポーズ機能をこういった場合でも使いやすくすることができます。単にポーズ機能を動かしたければ、pause で済みますが、ポーズ機能を細かく設定したければ、pause の代わりに beginPause と endPause を使う必要があります。

参考 beginPause と endPause を使ってポーズ機能を記述するのは、form と endform を使ってフォームを記述するのとよく似ています。しかし、後で述べるようにポーズとフォームでは記述方法にいくつかの違いがあります。

簡単な例を見てみましょう。次のスクリプトを入力して実行してください。beginPause と endPause は、P が大文字になっていることに気をつけてください。

```
1 writeInfoLine("Start")
2 beginPause␣("Question")
3 ␣␣comment␣("Did␣you␣hear␣any␣sounds?")
4 endPause␣("Yes",␣"No",␣1)
5 writeInfoLine("Done")
```

このスクリプトを実行すると、Praat Info ウィンドウに *Start* と表示された後、Pause: Question というタイトルのウィンドウが出てきます。そして、*Did you hear any sounds?*というメッセージと、Stop, Yes, No という 3 つのボタンが表示されるはずです。Stop ボタンを押せばそこで中断され、5 行目の writeInfoLine("Done") は実行されません。また、Yes あるいは No を押した場合は、スクリプトの続きが実行され、5 行目の記述に基づき、Praat Info ウィンドウに *Done* と表示されます。

このスクリプトでのポーズの記述について簡単に説明しましょう。まず、2 行目の beginPause の直後に、("Question") と書いてありますが、これがウィンドウのタイトルになります。これは、フォームを記述する際に、form の後にフォームのタイトルを記述したことと似ています。ただし、form で

は form Question のようにタイトルとして表示したいものをそのまま書けばよかったのですが、beginPause では表示したいものをダブルクォーテーションで囲った後、さらに丸括弧で囲む必要があります。書き方がまぎらわしいので注意しましょう。

参考 ダブルクォーテーションで囲っているのは、ダブルクォーテーションの中身を文字列として認識させるためです。

3 行目は、comment を使って、ウィンドウに表示するメッセージを設定しています。これは、基本的にフォームの中の comment と一緒です。ただし、記述の方法が少し違います。フォームの中では comment Did you hear any sounds?のように表示したいメッセージをそのまま書けばよかったのですが、ポーズ機能の記述の際には comment ("Did you hear any sounds?") のようにダブルクォーテーションと丸括弧で囲む必要があります。

4 行目では、endPause と書いて、ポーズ機能の記述を終えています。また、endPause の後の、("Yes", "No", 1) は、ウィンドウに表示するボタンの記述です。最後に書いてある 1 という数字はとりあえず無視して、残りの部分に着目してください。"Yes"と"No"という記述がありますので、Yes と No というボタンが表示されます。また、特に何もしなければ、Stop ボタンは自動的に表示されます。

参考 この最後の数値を間違ってダブルクォーテーションで囲まないでください。ダブルクォーテーションで囲って endPause ("Yes", "No", "1") とするとちゃんと動きません。ダブルクォーテーションで囲む必要があるのは文字列です。数値は囲む必要がありません。

参考 Stop を表示させない方法は後で扱います。

ボタンはいくつ並べてもかまいません。例えば、次のスクリプトでは、Stop ボタンの他に、Black, Blue, Green, Pink, Purple, Red, White, Yellow というボタンが表示されます。

参考 ボタンの数がいくつになっても、Stop ボタンを押せば中断し、それ以外のボタンを押せば継続します。

```
beginPause␣("Question")
␣␣comment␣("What␣is␣your␣favorite␣color?")
endPause␣("Black",␣"Blue",␣"Green",␣"Pink",␣"Purple",
    ␣"Red",␣"White",␣"Yellow",␣1)
```

ところで、先ほど endPause ("Yes", "No", 1) と書いたときに、最後の 1 についてちゃんと説明していませんでした。この数値は、デフォルトのボタンの番号を指定するものです。ここでは 1 番目のボタン、すなわち Yes ボタンがデフォルトのボタンとして扱われます。ポーズのウィンドウが表示されたときに、キーボードでエンターキーを押すと、このデフォルトのボタンを押したと見なされます。なお、デフォルトのボタンは、そうでないボタンと、影の描き方が異なって表示されます。先ほどのスクリプトを実行したとき、Yes のボタンについている影は、他のボタンの影と違っていたはずです。デフォルトとするボタンを変えたければ、この数値を書き換えます。例えば、先ほどのスクリプトで、デフォルトを No のボタンにしたければ、No は 2 番目なので、endPause ("Yes", "No", 2) と書いてください。

どのボタンもデフォルトのボタンにしたくなければ、endPause ("Yes", "No", 0) のように 0 と書いてください。要するに、0 番目という存在しないボタンをデフォルトにするので、実質的にはデフォルトのボタンが存在しなくなるのです。このドリルで行う実験からすると、デフォルトのボタンを設定しない方が無難なので、これからは endPause ("Yes", "No", 0) としておきます。

参考 被験者にはマウスを使って答えてもらいますが、その際うっかりキーボードに触れてしまうかもしれません。触れたときにエンターキーを押してしまったら、デフォルトのボタンを設定していれば、そのボタンを押したと見なされてしまいます。

次に、ユーザが押したボタンの情報を得る方法について説明します。

```
beginPause␣("Question")
␣␣comment␣("Did␣you␣hear␣any␣sounds?")
clicked␣=␣endPause␣("Yes",␣"No",␣0)

writeInfoLine("You␣clicked␣Button␣",␣clicked,␣".")
```

このスクリプトの 3 行目では、イコール (=) を使って代入をしています。具体的に言うと、endPause の結果を clicked に代入しています。endPause の結果というのは、どのボタンを押したのかという情報に他なりません。とりあえず、このスクリプトを実行し、Yes というボタンを押してください。すると、*You clicked Button 1.* と表示されるはずです。Yes というボタンは 1 番目のボタンなので、このボタンを押すことで、endPause の結果が 1 になり、その結果が clicked に代入されるわけです。

参考 ここでは clicked という名前の数値変数を使っていますが、他の名前にしてもかまいません。

参考 Stop ボタンを押した場合は、そこで処理が中断するので、代入が行われません。

> 問 6–1　確認しよう
>
> A 先ほどのスクリプトで、No というボタンを押すと、どんなメッセージが表示されますか。

さて、先ほどのスクリプトでは、Stop というボタンが出てきてしまいます。Yes と No さえあれば問題ないので、Stop ボタンは消したいところです。Stop ボタンを消すには次のスクリプトの 3 行目のようにします。

```
1 beginPause␣("Question")
2 ␣␣comment␣("Did␣you␣hear␣any␣sounds?")
3 clicked␣=␣endPause␣("Yes",␣"No",␣0,␣2)
4
5 writeInfoLine("You␣clicked␣Button␣",␣clicked,␣".")
```

このスクリプトを実行して、Stop ボタンが消えていることを確認してください。

参考 endPause ("Yes", "No", 0, 2) の 0 は今までと同じようにデフォルトのボタンが何番目かを示しています。0 番目という存在しないボタンをデフォルトにすることで、実質的にデフォルトのボタンがない状態にしています。

問 6–2　確認しよう

Ⓐ 先ほどのスクリプトを実行し、Yes ボタンも No ボタンも押さずに、Pause: Question ウィンドウを閉じてください。どうなりますか。

Ⓑ 先ほどのスクリプトの 3 行目の数字の 2 を 1 に改めて、clicked = endPause ("Yes", "No", 0, 1) としてください。その後、スクリプトを実行し、Yes ボタンも No ボタンも押さずに、Pause: Question ウィンドウを閉じてください。Windows ならばウィンドウ右上の×印のボタンを押すことでウィンドウを閉じることができます。閉じるとどうなりますか。

Ⓒ endPause の最後の数、すなわち、endPause ("Yes", "No", 0, 2) でいうところの 2 の場所に来る数はどんな働きをしていると考えられますか。

6.2　実験用のスクリプトを完成させる

今までやってきたことを踏まえれば、実験用のスクリプトの完成まであと一息です。

具体的にどれぐらいの周波数の音を聞かせるかは今までちゃんと定めていませんでした。ここでしっかり決めておこうと思います。最初は 10000 Hz の正弦波を聞かせることにします。11000 Hz, 12000 Hz, 13000 Hz と 1000 Hz ずつ高くしていき、20000 Hz の正弦波でやめることにしましょう。

10000 Hz からはじめて、1000 Hz ずつ高くしていき、20000 Hz まで再生するスクリプトは次の通りになります。

```
multiplier = 1000

for i from 10 to 20
  frequency = multiplier * i
  formula$ = "sin(2*pi*" + string$(frequency) + "*x)"
  do("Create Sound from formula...", "sound", 1, 0.0,
```

```
    ␣1.0,␣44100,␣formula$)
␣␣do("Play")
␣␣do("Remove")
endfor
```

問 6–3　確認しよう

Ⓐ 8000 Hz からはじめて、500 Hz ずつ大きくしていき、18000 Hz まで再生したい場合、このスクリプトをどう変更すればよいでしょうか。

Ⓑ 逆に 18000 Hz からはじめて、500 Hz ずつ小さくしていき、8000 Hz まで再生したい場合、このスクリプトをどう変更すればよいでしょうか。

Ⓒ 先ほどのスクリプトでは、1 つの周波数の音が 1 回しか流れません。複数回流したい場合はどうすればよいでしょうか。

先ほどのスクリプトに、被験者への質問をするためのポーズに関する記述を加えたのが次のスクリプトです。音が 1 回再生されるごとに、ポーズのウィンドウが出現し、Yes か No のボタンを押す必要が出てきます。なお、このスクリプトでは、Yes を押した場合と No を押した場合とで違いはありません。

```
multiplier␣=␣1000

for␣i␣from␣10␣to␣20
␣␣frequency␣=␣multiplier␣*␣i
␣␣formula$␣=␣"sin(2*pi*"␣+␣string$(frequency)␣+␣"*x)"
␣␣do("Create␣Sound␣from␣formula...",␣"sound",␣1,␣0.0,
    ␣1.0,␣44100,␣formula$)
␣␣do("Play")
␣␣do("Remove")
␣␣beginPause␣("Question")
␣␣␣␣comment␣("Did␣you␣hear␣any␣sounds?")
␣␣clicked␣=␣endPause␣("Yes",␣"No",␣0,␣2)
endfor
```

さて、Yes を押した場合と No を押した場合とで違いがないと困ります。実験の際には、Yes が押されたらそのまま継続し、for によるループを繰り返することが望まれます。また、No が押されたら中断して for によるループを繰り返さないようにしたいところです。先ほどのスクリプトでは、clicked にどのボタンを押したかという情報が格納されています。これを使って、動作を振り分けましょう。if を使えば簡単に振り分けることができます。

Yes が押されたら clicked に 1 が、No が押されたら clicked に 2 が格納されます。ですから、clicked の中身が 1 なら継続、2 なら中断すればよいのです。次のスクリプトを見てください。13 行目から 15 行目で if を使って「clicked の中身が 2 なら中断する」という記述がなされています。実際に試してうまく動くか見てください。

```
1  multiplier␣=␣1000
2
3  for␣i␣from␣10␣to␣20
4  ␣␣frequency␣=␣multiplier␣*␣i
5  ␣␣formula$␣=␣"sin(2*pi*"␣+␣string$(frequency)␣+␣"*x)"
6  ␣␣do("Create␣Sound␣from␣formula...",␣"sound",␣1,␣0.0,
       ␣1.0,␣44100,␣formula$)
7  ␣␣do("Play")
8  ␣␣do("Remove")
9  ␣␣beginPause␣("Question")
10 ␣␣␣␣comment␣("Did␣you␣hear␣any␣sounds?")
11 ␣␣clicked␣=␣endPause␣("Yes",␣"No",␣0,␣2)
12
13 ␣␣if␣clicked␣=␣2
14 ␣␣␣␣exit␣Clicked␣No␣button.
15 ␣␣endif
16
17 endfor
```

問 6–4　確認しよう

Ⓐ このスクリプトの 8 から 10 行目が何をしているのか説明してください。また、10 行目の endPause の後の括弧の中にあるものについても説明してください。

Ⓑ このスクリプトでは、「clicked の中身が 1 なら継続」ということを書いた if はありません。なぜそれでもうまく動くのでしょうか。

先ほどのスクリプトでは、全部の音が聞こえた場合、何も表示されないままに終わってしまいます。また、聞こえない音があって No を押した場合、その音の周波数がいくつだったのかを表示しないまま終了しました。これでは不都合なので、周波数がどれだけだったか出力するようにしましょう。周波数が Praat Info ウィンドウに出力するようにしたのが次のスクリプトです。

```
1  multiplier␣=␣1000
2
3  for␣i␣from␣10␣to␣20
4  ␣␣frequency␣=␣multiplier␣*␣i
5  ␣␣formula$␣=␣"sin(2*pi*"␣+␣string$(frequency)␣+␣"*x)"
6  ␣␣do("Create␣Sound␣from␣formula...",␣"sound",␣1,␣0.0,
       ␣1.0,␣44100,␣formula$)
7  ␣␣do("Play")
8  ␣␣do("Remove")
9  ␣␣beginPause␣("Question")
10 ␣␣␣␣comment␣("Did␣you␣hear␣any␣sounds?")
11 ␣␣clicked␣=␣endPause␣("Yes",␣"No",␣0,␣2)
12
13 ␣␣if␣clicked␣=␣2
14 ␣␣␣␣previousFrequency␣=␣frequency␣-␣multiplier
15 ␣␣␣␣writeInfoLine("Maximum␣audible␣frequency␣is␣in␣
       the␣range␣from␣",␣string$(previousFrequency),␣"␣
       to␣"␣+␣string$(frequency)␣+␣".")
16 ␣␣␣␣exit␣Clicked␣No␣button.
17 ␣␣endif
```

```
18
19  endfor
20
21  writeInfoLine("Maximum␣audible␣frequency␣is␣more␣than
        ␣"␣+␣string$(frequency)␣+␣".")
```

問 6–5　確認しよう

A　このスクリプトの 13 から 17 行目が何をしているのか説明してください。特に previousFrequency が何を表しているのか注意して説明してください。

先ほどのスクリプトでは、No を押すと、16 行目の記述により exit が発動して、エラーメッセージとともに終了します。仕方がないと言えば仕方がないのですが、エラーという訳でもないのに、エラーメッセージが出るのは何だか変です。exit を使わずに終了させる方法はないのでしょうか。

ややトリッキーな手法になりますが、goto と label を使うことで、終了まで持っていくことができます。

参考　goto はこれだけで一単語です。go to のように go と to の間に空白を入れないでください。

次のスクリプトを実行して、No を押しても、エラーメッセージ無しで終了することを確認してください。

```
1   multiplier␣=␣1000
2
3   for␣i␣from␣10␣to␣20
4   ␣␣frequency␣=␣multiplier␣*␣i
5   ␣␣formula$␣=␣"sin(2*pi*"␣+␣string$(frequency)␣+␣"*x)"
6   ␣␣do("Create␣Sound␣from␣formula...",␣"sound",␣1,␣0.0,
        ␣1.0,␣44100,␣formula$)
7   ␣␣do("Play")
8   ␣␣do("Remove")
9   ␣␣beginPause␣("Question")
10  ␣␣␣␣comment␣("Did␣you␣hear␣any␣sounds?")
```

```
11  ␣␣clicked␣=␣endPause␣("Yes",␣"No",␣0,␣2)
12
13  ␣␣if␣clicked␣=␣2
14  ␣␣␣␣previousFrequency␣=␣frequency␣-␣multiplier
15  ␣␣␣␣writeInfoLine("Maximum␣audible␣frequency␣is␣in␣
        the␣range␣from␣",␣string$(previousFrequency),␣"␣
        to␣"␣+␣string$(frequency)␣+␣".")
16  ␣␣␣␣goto␣point
17  ␣␣endif
18
19  endfor
20
21  writeInfoLine("Maximum␣audible␣frequency␣is␣more␣than
        ␣"␣+␣string$(frequency)␣+␣".")
22
23  label␣point
```

このスクリプトの 16 行目では、goto point と書かれています。これは文字通り、point という場所に行けという命令です。それでは、point という場所はどこにあるのでしょうか。Praat では場所を定義するのに label というものを使います。スクリプトの最後の行を見てください。ここに label point と書いてあります。ですから、この最後の行が point という場所になります。この場所より先にスクリプトの記述はないわけですから、ここに来てしまえばスクリプトは終わります。ここでは point という名前をつけましたが、別の名前でも問題ありません。

> 問 6–6　確認しよう
>
> A 先ほどの例では、スクリプトの最後の場所に行きましたが、goto と label を使ってスクリプトの最初に戻るにはどうすればよいでしょうか。

6.3　実際に実験する際に

このドリルで紹介したスクリプトで実験をする際には、次のようなことに注意するとよいでしょう。

- 静かな場所で実験を行う。
- 実験開始前に、イヤホンやヘッドホンからの音量を被験者自身に調整させる。
- 聞くことができる音の高さに影響を与えるかもしれない要因をいくつか考え、それを被験者から聞いておく。例えば、年齢、性別など。これらの要因があまり変わらない人ばかりだとつまらないので、色々な年代層の人に実験を頼むなど、適宜工夫するとよい。
- 正弦波を聞かせるだけでなく、他の波形の音ではどうなるかを比べてみると面白いかもしれない。

また、先ほどのスクリプトでは、writeInfoLine を使って、Praat Info ウィンドウに周波数の情報を出力していました。Praat Info ウィンドウに出力されたものをファイルに保存するには、Praat Info ウィンドウのメニューから File を選んでクリックした上で、Save as... を選びます。このようにファイルに保存すれば、出力結果をあとで利用しやすくなります。

しかし、手作業で毎回ファイルに保存するのは面倒です。スクリプトからもファイルに保存できるので、その方法を紹介しましょう。

次のスクリプトでは、output$という文字列変数の中身（ここでは、*This is an example*）を result.txt というテキストファイルに書き出しています。result.txt というテキストファイルが存在しなければ、Praat が勝手に result.txt というファイルを作ります。また、既に result.txt というファイルが存在していたら、そのファイルを上書きします。つまり、もともと result.txt に書かれていた内容は消されます。注意しましょう。

参考　result.txt は、通常、Praat が置かれているフォルダに作られるはずです。

```
1 output$ = "This is an example."
2 writeFileLine("result.txt", output$)
```

上書きでなく、既に書かれている内容の末尾に追記したい場合は、次のようにします。

参考 writeFileLine による上書きと appendFileLine による追記の関係は、writeInfoLine と appendInfoLine の関係と同じです。writeFileLine や writeInfoLine では今まで書いてあったものがすべて消されます。これに対して、appendFileLine や appendInfoLine は、今まで書いてあったものの下に出力を付け加えます。

```
1 output$␣=␣"This␣is␣an␣example."
2 appendFileLine("result.txt",␣output$)
```

なお、先ほど扱った周波数の実験スクリプトに、ファイル保存の操作を組み込んだスクリプト例はここでは挙げません。各自考えてみてください。

復習

1. pause と form の違いは何ですか。
2. goto と label でどのようなことができますか。
3. 先ほどの実験スクリプトでは、最初の周波数の音が聞こえなかった場合、どうなりますか。実は問題があるのですが、どう改めればよいか考えてください。
4. 先ほどの実験スクリプトでは、聞こえたかどうかの回答はあくまでも自己申告によるものでした。もしかしたら、聞こえていないのに Yes のボタンを押してしまう人もいるかもしれません。こういう人への対策を考えてみてください。