Praat スクリプト基礎文法 最速マスター

この文章は音声分析ソフト Praat のスクリプト機能の基礎文法を紹介したものである。既に何らかのプログラミング言語を知っている人にとっては、かなりのスピードでマスターできるように書いたつもりである。プログラミングがよく分からないという人にとっては難しいかもしれないが、何かの参考にしていただければ幸いである。

西原 史暁

2013 年 1 月 26 日

1　基礎

1.1　Praat とは

音声学に関する様々な分析が行えるフリーソフト。単純な音声分析だけでなく、音声合成、音声を提示する実験、簡単な統計分析なども可能で、言語音の研究には必要不可欠なソフトである。ダウンロードは以下のページから可能。Windows, Macintosh, Linux などさまざまな OS で使用できる。

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

1.2　Praat のスクリプト機能とは

Praat は基本的に GUI[*1]で使うソフトで、マウスでカチカチすることで分析を行う。しかし、これでは繰り返し作業などが面倒なので、いろんなことを一気に処理できる**スクリプト機能**が用意されている。

Praat のスクリプトは、単純な手続き型言語で、制御構造やサブルーチンも問題なく書ける。プログラミングの経験がある人にとって、Praat のスクリプトの仕組みを理解することは難しくないだろう。

Praat のスクリプトを作るのには以下のようにすれば良い。まず、Praat を立ち上げる。すると、`Praat Objects` というウィンドウが出てくる。このウィンドウのメニューから `Praat` → `New Praat script` を選ぶと、Praat のスクリプトエディタが出てくる。そこにスクリプトを色々書いた上で、スクリプトエディタのメニューから `Run` → `Run` を選ぶとスクリプトが実行される。

[*1] Graphical User Interface の略。アイコンやボタンなど視覚的要素をマウスで操作するのが基本。

1.3 echo 文

echo の後に何か文字列を入れると、それが表示される。

普通のプログラミング言語と違って、表示したい文字列をクォーテーションマークで囲む必要は無い。

```
echo Hello world
```

上記の echo 文を実行すると、Praat Info という出力ウィンドウに、以下のような文字列が表示される。

```
Hello world
```

厳密に言うと、echo は、それまでの出力ウィンドウを消した上で、新しい出力ウィンドウを出して、そこに文字列を表示する命令である。このため、echo を使うと、それまで出力ウィンドウに表示されていたものは消えてしまう。以下のスクリプトを見てみよう。

```
echo Hello world
echo Goodbye world
```

このスクリプトでは、最初の echo 文で Hello world が出力されるが、2 行目の echo 文で Hello world が出力されたウィンドウが消され、新しい出力ウィンドウに Goodbye world が出力される。結果として、Goodbye world しか出力されていないように見えてしまう。

既になされた出力を消さないようにするには、echo の代わりに printline を使う。

```
echo Hello world
printline Goodbye world
```

上記のスクリプトの出力結果は、以下のようになる。なお、上記のスクリ

プトで、Hello world の前の echo を printline にするとうまく動かない。printline は既存のウィンドウに出力する命令であって、出力ウィンドウの作成は行わない。このため、まず echo で出力ウィンドウを作成する必要があるのだ。

```
Hello world
Goodbye world
```

1.4 コメント

行頭に # か ; を入れるとコメントになる。

```
# this is a comment
; this is another comment
```

どっちを使っても良いのだが、恒久的なコメントは # で、臨時的なコメントは ; にすると分かりやすいだろう。

1.5 変数

変数は宣言せずにいきなり代入ができる。データ型の指定も必要ない。

ただし、数値を格納する変数はドル記号を末尾につけず、文字列を格納する変数はドル記号を末尾につけるという決まりがある。例えば、数値を格納する場合、以下のようにする。

```
a = 7.8
```

以下のようにドル記号を末尾につけるとエラーが出てしまう。

```
a$ = 7.8
```

これに対して、文字列を格納する場合は以下のようにする。

```
w$ = "hoge"
```

以下のようにドル記号を末尾につけないとエラーが出てしまう。

```
w = "hoge"
```

変数の中身を出力したい場合、以下のようにすると、単に w$と出力されてしまう。

```
w$ = "hoge"
echo w$
```

w$の中身を出力するには、以下のように変数をシングルクォーテーションで囲む必要がある[*2]。以下の例は文字列変数の場合であるが、数値変数の場合も同様である。

```
w$ = "hoge"
echo 'w$'
```

変数名では小文字と大文字が区別される。つまり、a と A は別々の変数ということになる[*3]。

[*2] Praat の echo 文におけるクォーテーションの有無のふるまいは普通のプログラミング言語と逆になっていることに注意する必要がある。普通の言語では、echo x とすると変数 x の中身を出力し、echo 'x' で 'x' という文字列そのものを出力する。逆に Praat では、普通の言語では、echo x で 'x' という文字列を出力し、echo 'x' で変数 x の中身を出力するのである。

[*3] 例外的に、後述する GUI ダイアログで取得する変数については小文字と大文字が区別されない。

2　主な命令

2.1　四則演算など

四則演算などの書き方は、普通のプログラミング言語と同じ。

- 足し算：`4+5`
- 引き算：`12-7`
- 掛け算：`5*6`
- 割り算：`24/8`
- 累乗：`7^2`
- 平方根：`sqrt(3)`
- 三角関数（ラジアンで）：`sin(1/4*pi)`

`x++` という形のインクリメント[*4]はできない。なお、`x = x+1` の意味[*5]で、`x += 1` と書くことはできる。

なお、Praat は音声分析に関するソフトということもあって、音声に関する命令や統計に関する命令が色々ある。例えば、標準正規分布の累積分布を出すには `gaussP(0.5)` のようにすれば良いし、ヘルツをバーク尺度に変えるには `hertzToBark(1000)`[*6]のようにすれば良い。

2.1　文字列演算

`+` で文字列をつなげることができる。以下のスクリプトを実行すると、`hogepiyo` が出力される。

[*4] 変数に格納されている数値を 1 だけ大きくすること。

[*5] この操作によって、変数 `x` に格納されている値を 1 だけ大きくしている。

[*6] 普通のプログラミング言語と同じく、命令の後の括弧の中に入っているものが引数と見なされる。例えば、`hertzToBark(1000)` なら、`1000` が `hertzToBark` の引数となっている。

```
w1$ = "hoge"
w2$ = "piyo"
result$ = w1$ + w2$
echo 'result$'
```

2.3 スクリプトから GUI メニューにアクセスする

Praat には、GUI メニューの名前がそのままスクリプトのコマンドになるという特徴がある。例えば、`Praat Objects` ウィンドウを表示しているとき、GUI メニューならば、メニューから `Help` の `Praat Intro` を選ぶと、Praat のヘルプが表示される。スクリプトから同じことをしたければ、スクリプトにこのメニューの名前をそのまま書けば良い。つまり、以下のようにスクリプトを書くだけ[*7]で、Praat のヘルプが表示される。

```
Praat Intro
```

このように GUI メニューの名前を並べることで、GUI でできることがスクリプトでもできるようになる。メニューの名前を書くとき、小文字と大文字は区別されるので要注意。

より具体的な例を見てみよう。Praat で、1 kHz の正弦波[*8]を作成し、それを再生し、作成された音声を削除する操作をしたいとする。GUI メニューを使う場合、以下のような手順で行う。

1. Praat を起動して、`Praat Objects` というウィンドウを選ぶ。
2. このウィンドウのメニューから `New` → `Sound` → `Create Sound`

[*7] GUI メニューで上位にあるものをスクリプトに書く必要はない。この例の場合、GUI メニューで `Praat Intro` は `Help` メニューの下にあるが、スクリプトを書くときには `Help` を書く必要はない。

[*8] いわゆる「ピー音」のこと。

`from formula...` を選ぶ。

3. すると、細かい設定をするためのダイアログが出てくるので、入力欄に以下の要領で入力し、`OK` ボタンを押す。
   - Name: 作成する音声の名前。ここでは `sineWave` と入れておく。
   - Number of channels: 作成する音声のチャンネル数。モノラル音声で良いので、ここでは `1` と入れておく。
   - Start time: 音声の開始時間。ここでは `0.0` と入れておく。
   - End time: 音声の終了時間。ここでは `1.0` と入れておく。
   - Sampling frequency: 作成する音声サンプリング周波数。ここでは `44100` と入れておく。
   - Formula: 作成する音声がどのようなものなのかを数式で示す。ここでは 1 kHz、すなわち 1000 Hz の正弦波を作りたいので `sin(2*pi*1000*x)` と入れておく。
4. これで音声が作成されたので、`Praat Objects` の `Play` というボタンを押す。これで音声が再生される。
5. その後、`Praat Objects` の `Remove` というボタンを押せば、先ほど作成した音声が消える。

上記の GUI メニューでの手順をスクリプトにすると以下のようになる[*9]。

```
Create Sound from formula... sineWave 1 0.0 1.0 44100
... sin(2*pi*1000*x)
Play
Remove
```

[*9] 2 行目、すなわち`...` から始まっている行は、1 行目のコマンドの続きである。だから、1 行目の内容と 2 行目の内容をくっつけて、`Create Sound from formula... sineWave 1 0.0 1.0 44100 sin(2*pi*1000*x)` と 1 行にまとめて書いても良い。ここでは、まとめて書くと長くなりすぎてしまうので、コマンドの途中で改行した。そして、途中で改行したということを示すために、2 行目の冒頭に`...` を付している。

GUI で操作した際は `Create Sound from formula...` で、細かい設定をするためのダイアログに数値などを入力した。しかし、スクリプトの場合はダイアログが出てこないので、GUI だとしたらダイアログに入れるべき内容を `Create Sound from formula...` の後に並べる[*10]必要がある。

なお、Praat の仕組みとして、`Create Sound from formula...` のように、末尾にピリオドが 3 つあるものは、GUI では設定用のダイアログが出現するので、スクリプトではそのダイアログに対応する設定内容をコマンドの後に書かなくてはならない。

3 制御構造

3.1 if 構造

以下の例では、a に格納されている数値が正なら b は 5、負なら b は 8、さもなければ、b は 0 となる。

```
if a > 0
  b = 5
elsif a < 0
  b = 8
else
  b = 0
endif
```

[*10] GUI ダイアログで上にあるものから並べるようにする。

3.2 while 構造

`while` と `endwhile` とで囲む。以下では、`i` が 10 未満という条件が満たされている限り、`i` に 1 ずつ加算している。

```
i = 0
while i < 10
  i = i + 1
endwhile
```

3.3 for 構造

`for` と `endfor` で囲む。以下では、1 から 5 までの整数の和がいくつになるかを求めている。

```
sum = 0
for i from 1 to 5
  sum = sum + i
endfor
echo 'sum'
```

Praat の制御構造には他に `repeat-until` 文があるが、その説明は割愛する。

3.4 比較演算子

3.4.1 数値比較

- `a = b`（同じか？）
- `a <> b`（異なるか？）
- `a < b`（a は b より小さいか？）

- a > b（a は b より大きいか？）
- a <= b（a は b 以下か？）
- a >= b（a は b 以上か？）

3.4.2 文字列比較

- a$ = b$（同じか？）
- a$ <> b$（異なるか？）
- a$ < b$（ASCII 順[*11]で a は b より先か？）
- a$ > b$（ASCII 順で a は b より後か？）
- a$ <= b$（ASCII 順で a は b より先もしくは同じか？）
- a$ >= b$（ASCII 順で a は b より後もしくは同じか？）

4 サブルーチン

サブルーチンを定義するには、命令を procedure と endproc で囲む。procedure の直後に書かれたものがそのサブルーチンの名前になる（以下の例では、hogePiyo）。引数を持たせたい場合は、サブルーチンの名前の次に、変数名を書けば良い（以下の例では、variable）。

```
procedure hogePiyo variable
  #some commands
endproc
```

メインルーチンなどから、サブルーチンを呼び出すには、call を使う。

```
call hogePiyo variable
```

[*11] ASCII とは、文字コードの仕組みの 1 つ。極めて単純化して言えば、英字に関しては辞書順に並ぶ。

4.1 スコープ

Praat の変数は、常にグローバルで解釈されるのが基本である。メインルーチンでの変数がサブルーチンからそのままアクセスできるし、その逆も可能。スコープを限定したい場合、`.a` のように変数の前にピリオドをつける。

```
.a = 42
 b = 56

call subRoutine
echo '.a'
printline 'b'

procedure subRoutine
  .a = -17
   b = -29
endproc
```

上記のスクリプトを実行すると、以下のように表示される。

```
42
-29
```

変数 `b` には、最初は 56 が格納されていたものの、`subRoutine` が呼び出されたところで、24 が代入されたのである。

5 その他

- インデントは無視される。

- 基本的に 1 行に 1 コマンドずつ書く。ただし、長いコマンドを分割することも可能。行の冒頭に...（ピリオド 3 つ）を置くと、前の行の続きであると見なされる。

5.1 ファイル入出力

（テキスト）ファイルの中身を変数に入れる。

```
hoge$ < piyo.txt
```

変数の中身を新しい（テキスト）ファイルに書き出す。

```
hoge$ > piyo.txt
```

変数の中身を既存の（テキスト）ファイルに書き加える。

```
hoge$ >> piyo.txt
```

5.2 配列

厳密に言えば、Praat のスクリプトには配列はない。しかし、変数を以下のように書くことができるので、見た目だけは配列があるように見せることができる。

```
a[0] = 5
a[1] = 10
a[2] = 15
a[3] = 20
```

しかし、配列として宣言しているわけではないし、配列に一気に代入することもできないので、あまり意味がない。以下[*12]のように制御構造などで少し使えるかもしれないといった程度である。

```
for i from 1 to 10
  squared[i] = i^2
endfor
```

ちなみに、配列の添数は数値であれば何でも良いので、以下のように小数や負数が添数になってもエラーが出ない。

```
a[4.3] = 38
a[-15] = 53
```

5.3 GUI ダイアログの作成

Praat のスクリプトでは、`form` と `endform` で囲むことで、簡単に GUI ダイアログを作ることができる。`form` の直後に書かれたものが、そのダイアログのタイトルになる。例えば、以下では、“Personal Information” がタイトルとなる。ダイアログに含まれる数値などの入力欄の設定は `form` と `endform` の間に書く。

```
form Personal Information
   word Family_Name Sato
   word First_Name Taro
   natural Age 42
endform
```

[*12] `i` を 1 から 10 まで 1 つずつ増やしている間、`squared[i]` に `i` の平方を代入していくという操作を表している。

ダイアログに表示される入力欄は 1 行ずつ書く。各行は「欄の種類」、「欄の名前」、「その欄のデフォルト値」[*13]の順で記述する。上のスクリプトの `word Family_Name Sato` は、単語を入れる入力欄 (`word`) を作成せよ、その欄の名前は `Family_Name` とせよ、デフォルト値は `Sato` とせよということを示している。また、`natural Age 42` は、正の整数を入れる入力欄 (`natural`) を作成せよ、その欄の名前は `Age` とせよ、デフォルト値は 42 とせよということを示している。

欄の種類には、様々なものがある。以下に代表的なものを挙げる[*14]。

- `real`：実数（正負問わず）
- `positive`：正の実数
- `integer`：整数（正負問わず）
- `natural`：正の整数
- `word`：空白を含まない文字列
- `sentence`：文字列[*15]

上記のように作成した GUI ダイアログに入力された値をスクリプトの中で取得するには、欄の名前を使う。例えば、上述のスクリプトで、`natural Age 42` と定義したが、この欄の値を取得したければ、スクリプト中で `Age` と書けば良い[*16]。また、上述のスクリプトで、`word Family_Name Sato` と定義したが、この欄の値を取得したければ、スクリプト中で `Family_Name$`と書けば良い。`Family_Name` の欄には文字列が入るので、変数として呼び出すときには、ドル記号をつける必要があることに注意。

[*13] 「欄の種類」、「欄の名前」、「その欄のデフォルト値」は空白で区切る。

[*14] これはあくまでも入力欄の種類を表したものであって、データの型を表したものではない。

[*15] 短めの文字列を入力することが想定されている。空白を含んでも良い。

[*16] 実は `age` と書いても OK。Praat の変数名は小文字と大文字を区別するのだが、`form` と `endform` で囲んで作ったダイアログに基づく変数は小文字と大文字を区別しないようだ。

もっと Praat のスクリプトについて知りたい場合

Praat に関しては、公式ウェブサイトに掲載されているマニュアルに非常に詳しい情報が掲載されている。英語で書かれているため読みづらいかもしれないが、それを読むにこしたことはない。

また、以前、「音声分析ソフト Praat の使用に役立つウェブサイト」というタイトルで、Praat に関する説明などが載っているウェブサイトを紹介する文章を書いたのでそちらも参照すると良いだろう。

- `http://id.fnshr.info/2013/01/24/praatweb/`

ライセンス