

# Praat scripting tutorial Basics

Florian Jaeger,  
Stanford University, Linguistics Department

04/21/2004

## What is Praat scripting?

- Praat is a program for doing “phonetics by computer” ([www.praat.org](http://www.praat.org))
- Praat script is a scripting language that comes with that software

## What can I do with Praat scripting?

- Well, obviously you can ‘automate things’ – i.e. while you go and have a coffee you let your computer analyze the length of 500 vowels ... and, if you want, it even does the stats right on the way.
- *Everything* that can be done in Praat can be automated with Praat scripting:
  - Phonetic measures, reading/writing files, printing to the screen, interacting with the user, manipulating sounds, comparing sounds, creating new files (e.g. TextGrids), adding annotation, reformatting sound files, normalizing, ...

Part I

Intro

## How does Praat scripting think?

- Praat script is a scripting, not a programming language
  - A Praat script rather intuitively describes the steps that as Praat user would have to go through to achieve the same result.
    - Say, you want to select and remove the sound object “HelloWorld” from the object list:

```
select Sound HelloWorld  
Remove
```
  - In addition to those elements, the Praat scripting language (PSL) contains elements that allow you to “automate things”
    - Do ... while; for i from 1 to 10; if X Y elsif Z else W; etc.
    - Variables
    - Calling other Praat scripts
  - Finally, PSL provides you with a way to interact with the user of the script (i.e. probably you).
- Version used for this tutorial: 4.1.28

## The way PSL sees the world

- PSL is object oriented
  - **Objects are**
    - Everything in the left of the Praat window
    - Numbers, strings, booleans
    - The editor
    - Files
    - ...
  - Most of the scripting is about **manipulating objects**

## Ok, let's start...

- How to create a brand new script/to open an already existing script
- How to run a script
- How to edit a script
- How to navigate inside the Praat script editor
  - A bug with older versions of Praat.
- How to save a script
- How to get help

## Part II

First steps – sit back and relax

## The beginning...

- How to start writing a script?
  - [Input: Reading in information from a form](#)
  - [Input: Waiting for a user action](#)
  - Input: Reading information from a file
  - [Output: Print a line to the screen](#)
  - Output: writing into a file
- A first more complicated example
  - [Opening all files that match a certain description](#) (adopted from K. Crosswhite)

## Time for some more syntax...

- At any given point, in the processing of a script, something is implicitly assumed to be the **'selected' object**.
  - `select object_type object_name`
  - `x = selected(object_type object_name)`
  - `x$ = selected$(object_type object_name)`
- So, when a **function** is called it is applied to that 'selected object'.

## Functions...

- You can define functions yourself (a.k.a. *procedures*), but Praat comes with tons of them:
  - What is a function?
    - Remove
    - Rename...
    - Read from file...
  - Some functions are only available for specific object types:
    - Get root-mean-square...
    - To TextTier
  - Some are functions are only available in certain environments (e.g. editor).
- Functions can take arguments
  - Viewport... 0 6 0 6

## Part III

The atoms of PSL  
– it's getting more interactive

## A little quiz - functions

- Did you notice something about functions?
  - The beginning of function names
  - The end of function names
  - Comparison of function names in PSL and menu options in Praat
  - Spaces

## Variables

- Variables are temporary place holders
  - Examples
    - `current_file$`
    - `number_of_open_files`
  - Assigning values to variables
    - `mySound$= selected$("Sound")`
  - “De referencing” of variables
    - Sometimes one wants to **refer to the variable** (e.g. when assigning a values to it), sometimes one wants to refer to **the variable's value**.
      - `printline The file I opened is called: 'file$'`

## A little quiz - variables

- Did you notice something about variables?
  - The beginning of variable names
  - The end of variable names
  - `_`

## Commands

- Commands are the manipulative mechanisms that are inherent to PSL (rather than being Part of Praat such as functions)
  - Examples
    - `print, printline`
    - `clearinfo`
    - `select, plus`
    - `sentence, boolean, integer, positive`
    - `form, endform`
    - `if, elsif, else, endif, ...`
    - `for/endor, while/endwhile, do/until`
    - `call procedure_name (and maybe arguments)`
    - `exit`
  - A special kind of command is the **comment** (and you should make extensive use of it if you want to understand your own scripts next time you look at them)
    - `#`

## Summary of naming conventions

- Functions
  - Start with capital letter
  - End with ... if they take arguments
  - May contain spaces
- Commands
  - Lowercase
  - One word, no special characters
  - May have arguments
- Variables
  - Lowercase
  - Cannot contains spaces (underscore instead)
  - End with \$ if strings

Part IV

Reading PSL

## A note on style

```
form Draw a vowel chart
comment Give the path of the directory:
text directory ../sounds/
comment Which tier should be used for
analysis?
integer Tier 1
comment Which segments should be
analysed?
sentence Segment_label a
comment Where would you like to save
the results?
text resultfile ../results.txt
comment Formant analysis options
positive Time_step 0.01
integer Max_number_of_formants 5
endform
```

```
form Draw a vowel chart
comment Give the path of the
directory:
text directory ../sounds/
comment Which tier should be used
for analysis?
integer Tier 1
comment Which segments should be
analysed?
sentence Segment_label a
comment Where would you like to
save the results?
text resultfile ../results.txt
comment Formant analysis options
positive Time_step 0.01
integer Max_number_of_formants 5
endform
```

## Everyone:

```
# Current version of script by Florian Jaeger.
form Convert lab-to-TextGrid (c) 2004. tflfo@stanford.edu
comment This script will open all xwaves annotation files in the
comment specified directory and convert them into TextGrid files.
comment The lab file objects are automatically closed.
comment
sentence Source_directory d:\test\lab\
sentence Lab_file_extension .lab
sentence Goal_directory d:\test\lab\
boolean Save_new_files 1
sentence Save_only_files_with_label s2
endform

clearinfo

Create Strings as file list... list 'source_directory'*lab_file_extensions'
numberOfFiles = Get number of strings
for file to numberOfFiles
select Strings list
files = Get string... #file
call convert
endfor
select Strings list
Remove
```

## continued:

```
procedure convert
#
Read TextTier from Xwaves... 'source_directory'files'
names = leftfiles, (insex (file$, lab_sis_extensions) -
1)
select TextTier 'names'
into TextGrid
select TextTier 'names'
Remove
select TextGrid grid
Rename... 'names'

c= Count labels... 'save_only_files_with_label'
if c < 1
Insert interval tier... 2 focus
Extract tier... 1
Get points... #off
11= Get time from index... 1
12= Get time from index... 2
Remove
select TextGrid 'names'
Insert boundary... 2 11
Insert boundary... 2 12
select TextTier 'names'
Get points... #off1
11= Get time from index... 1
12= Get time from index... 2
Remove
select TextGrid 'names'
Insert boundary... 2 11
Insert boundary... 2 12
select TextTier 'names'
Get points... #off1
11= Get time from index... 1
12= Get time from index... 2
Remove
select TextGrid 'names'
Insert boundary... 2 11
end
else
Remove
end
endproc
```

## Final examples

- **Labelhelp:** Opening sounds & their TextGrids - creating TextGrids if not present
- **Phonetic analyzer v5.13:**
  - Read in sound files
  - Check consistency of files (e.g. sample rate)
  - Read in matching annotation
  - Read in tables with information on speakers, items, etc.
  - Get measurements' sentence means (e.g. duration, max f0, mean f0, f0 range, intensity, energy, ..)
  - Get measurements for all annotated targets
    - Choose best-suited algorithm for pitch measures depending on length of target
    - Smooth f0 curve before measuring
  - Normalize data
  - Check consistency of data (e.g. matching conditions)
  - Write data into file, create log-files (for debugging)

## Part V

### Writing PSL - your examples

(I'd be really surprise if we get here)