Praat Scripting

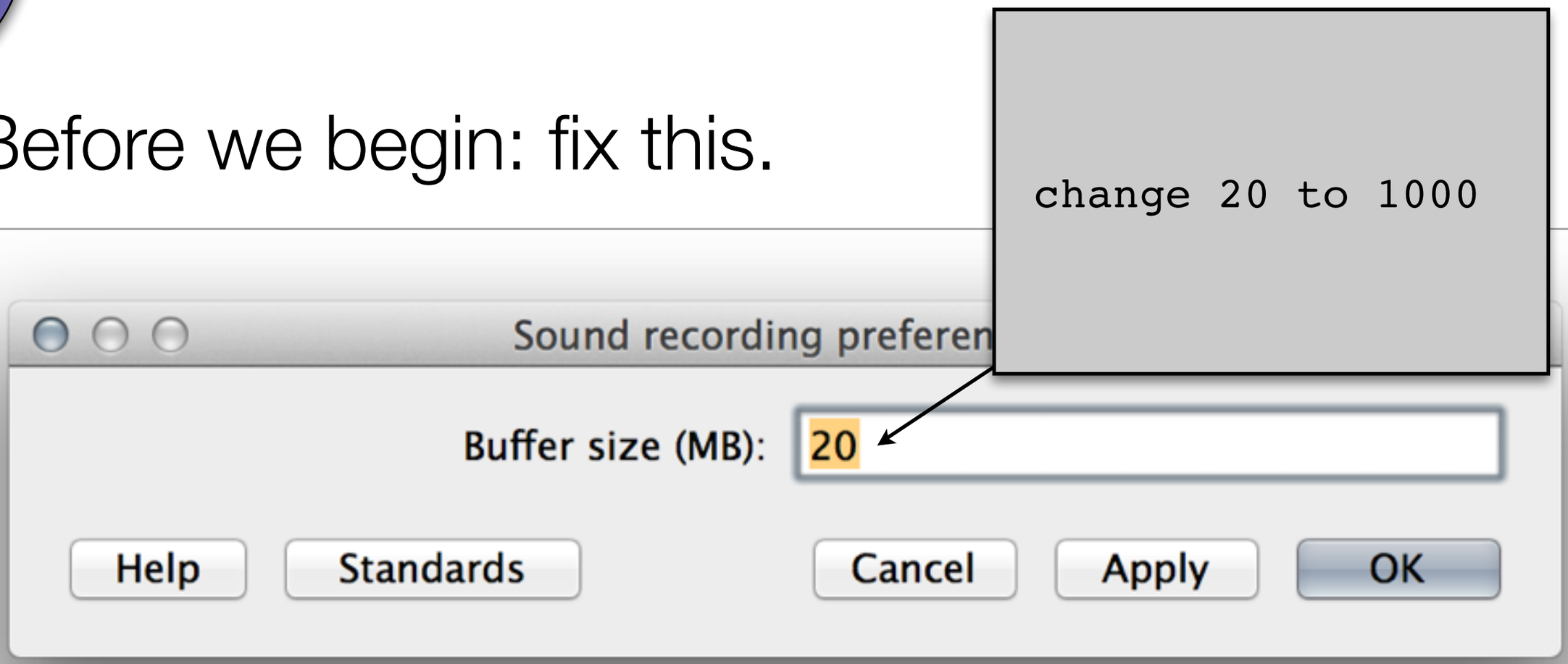
Part 1: Praatapalooza

Before we begin...

- Please launch Praat and check your version number (Praat → About Praat...)

- Then check the Praat home page for the current latest version and see what's new

Before we begin: fix this.



- 20 mb buffer gives you < 4 minutes of recording time (assuming 16 bit/44.1 kHz)
- 1000 mb gives you 3.3 hours of recording time.
- Notice that, in Praat, “Standards” means “Restore Defaults”

Citing Praat

- It is standard practice to include version number and download date when citing Praat in published work.

- Be sure to make a note of these when upgrading.

Boersma, Paul & Weenink, David (2014). Praat: doing phonetics by computer [Computer program]. Version 5.3.79, retrieved 21 June 2014 from http://www.praat.org/

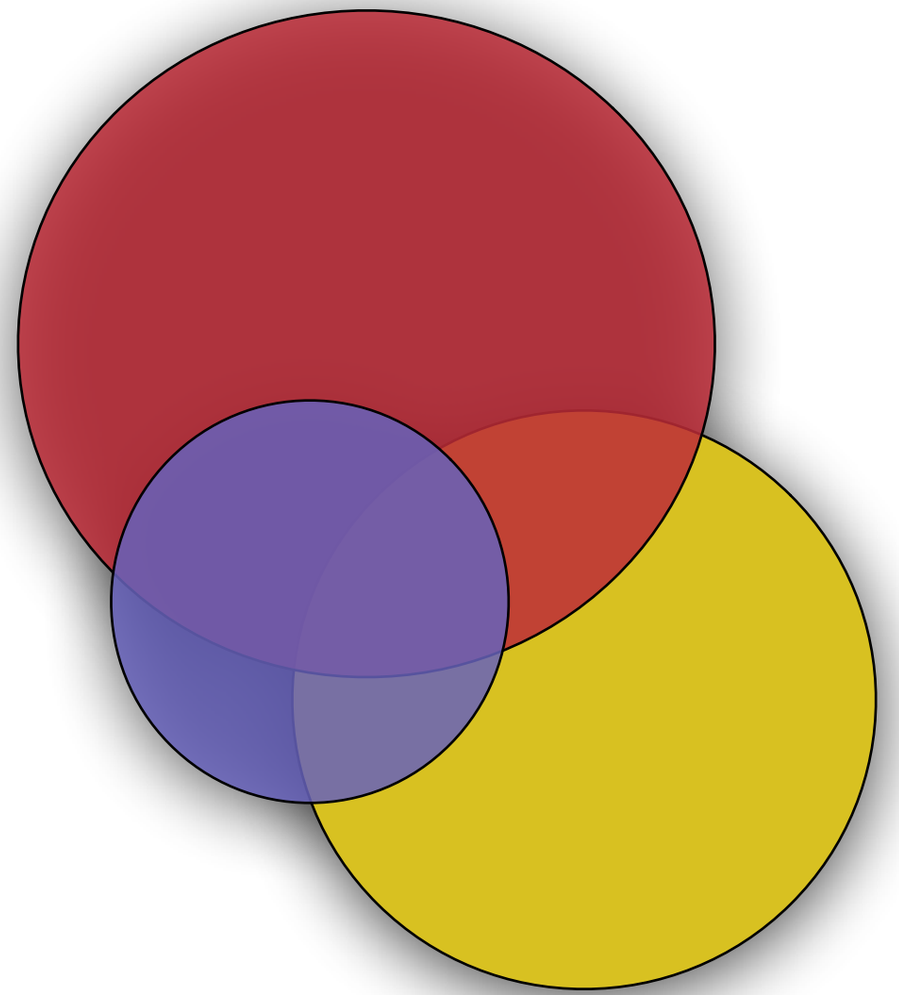
Mailing List

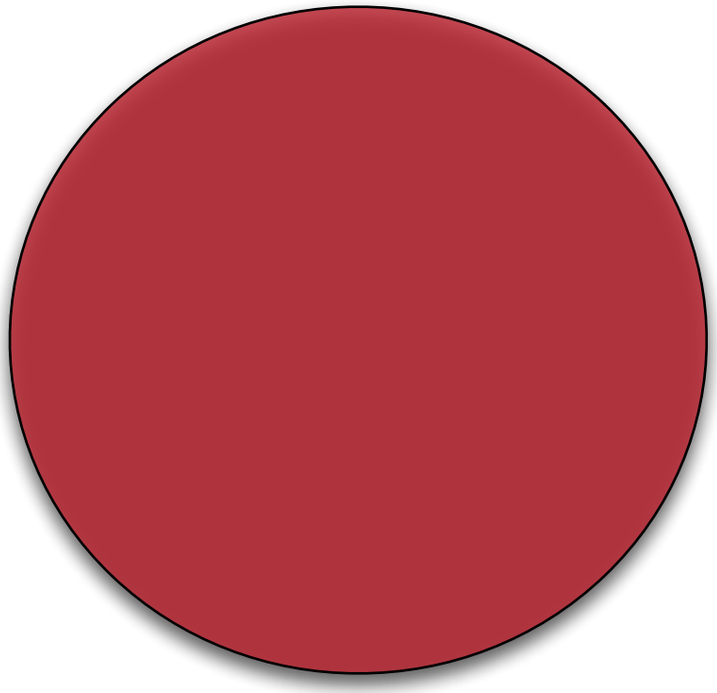
http://uk.groups.yahoo.com/group/praat-users/

praat-users@yahoogroups.co.uk

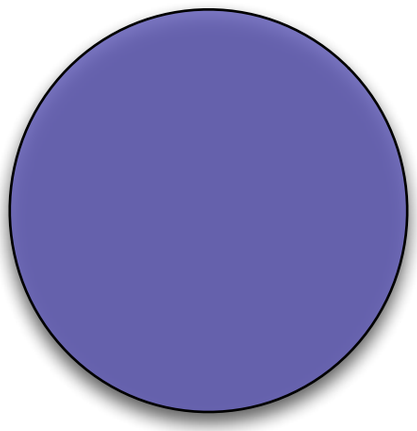
What exactly is our task?

- Effective Praat scripting depends on mastery of three quite separate domains of knowledge.

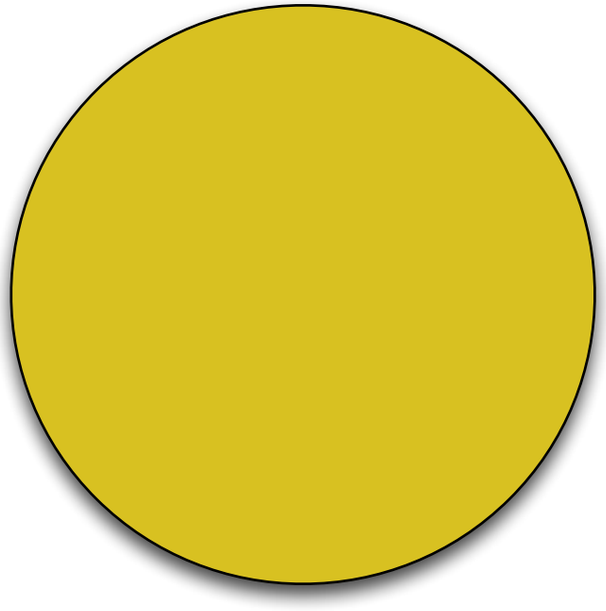




Linguistics domain knowledge



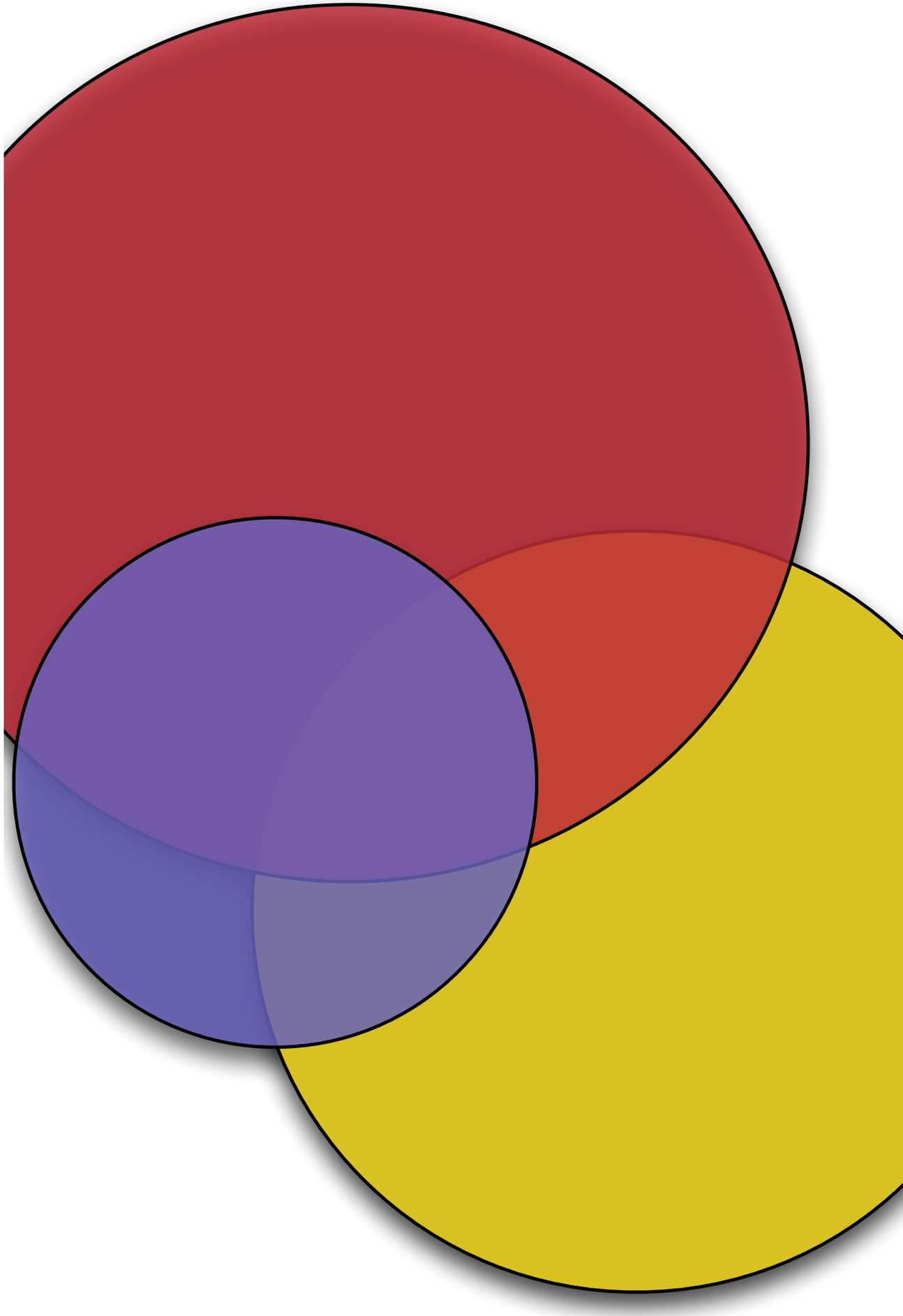
Knowledge of Praat



Programming & Debugging

Our task...

- Linguistic knowledge is outside the scope of this workshop: this is not a phonetics refresher
- Our focus will be on Praat itself
- And on learning to think just a little bit like software engineers
- Problems in Praat are generally easier to solve if you can identify which of these domains the problem comes from.



What is debugging?

Problem solving.

- Understand problem
- think creatively about solutions, and
- express a solution clearly and accurately
- repeat (maybe a lot)

Three kinds of errors

- **Syntax errors**

  Praat verifies the well-formedness of your script *before* executing it.
  `2 + 2 = 4` is grammatical, but `2 + = 2 4` will throw an error.

- **Runtime errors**

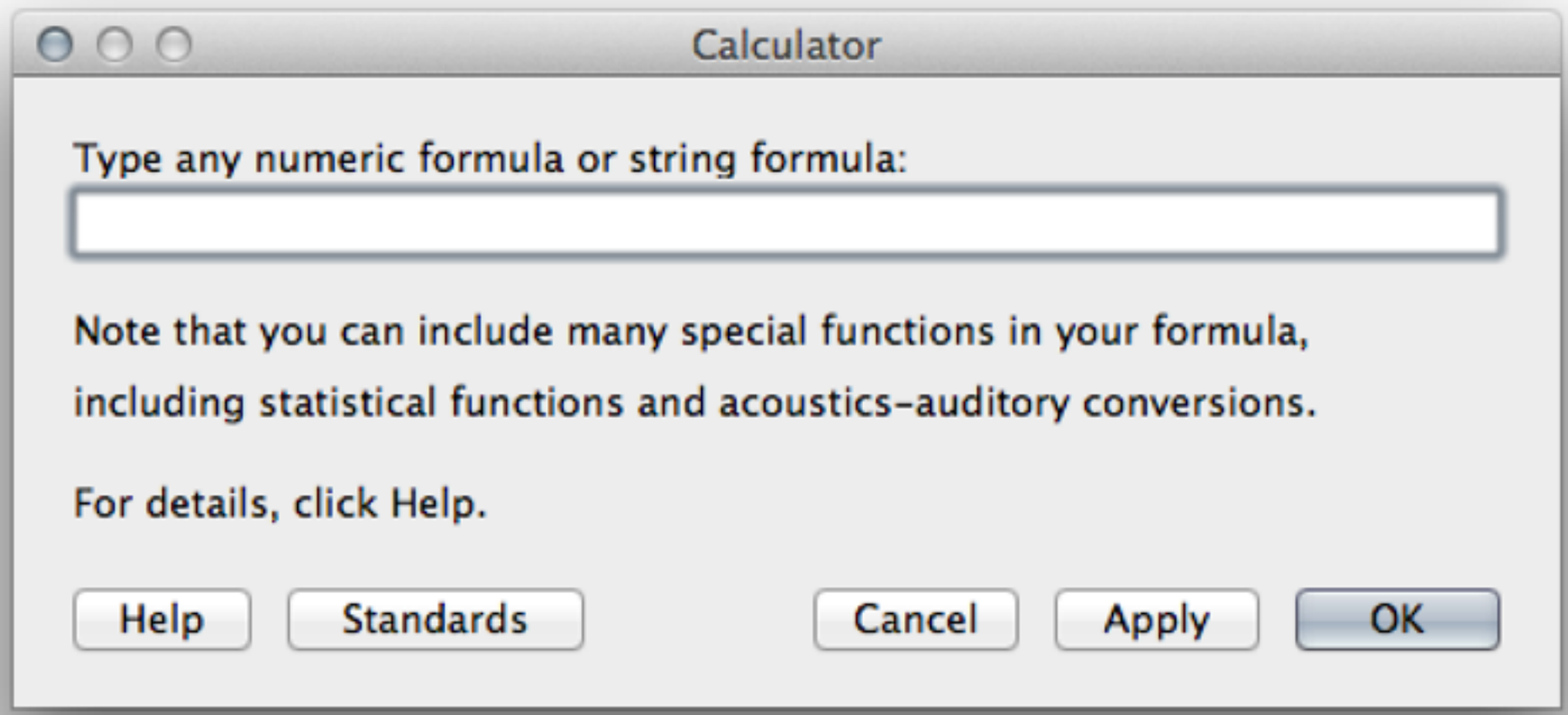
  If your script is syntactically correct, Praat will attempt to do precisely what it says. You will get a runtime error if you tell Praat to open a file that does not exist or try to perform a Textgrid action on a Sound object.

- **Semantic errors**

  Semantic errors occur when the syntax of your script is correct, Praat is able to do *something* for every command you gave it, but the result is not what you wanted or expected. An example of a semantic error is measuring F1/F2 of a diphthong at the temporal midpoint of the vowel.

Let’s generate some syntax errors!

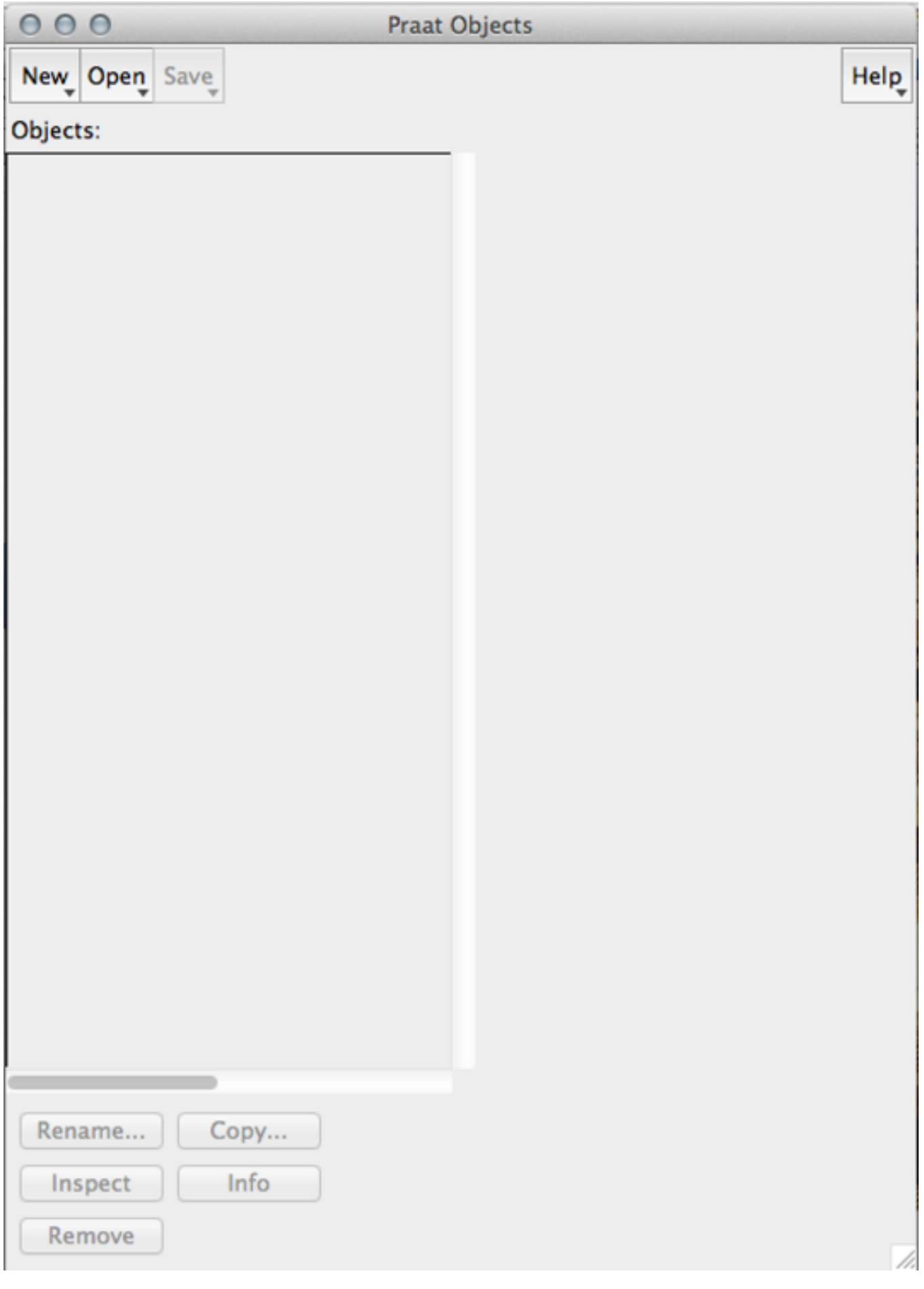
- Praat → Goodies → Calculator
- Keyboard: press ⌘U (ctrl-U on Windows or Linux)



Let’s take a quick tour...

Objects window

- The Objects window is the center of the Praat universe.
- Any data that you load from a file, record with a microphone, extract from another file, or generate with a script will appear as an **object** in this window.
- Each object has a **type** (see available types)
- Each type has a set of associated **data** and **behaviors**



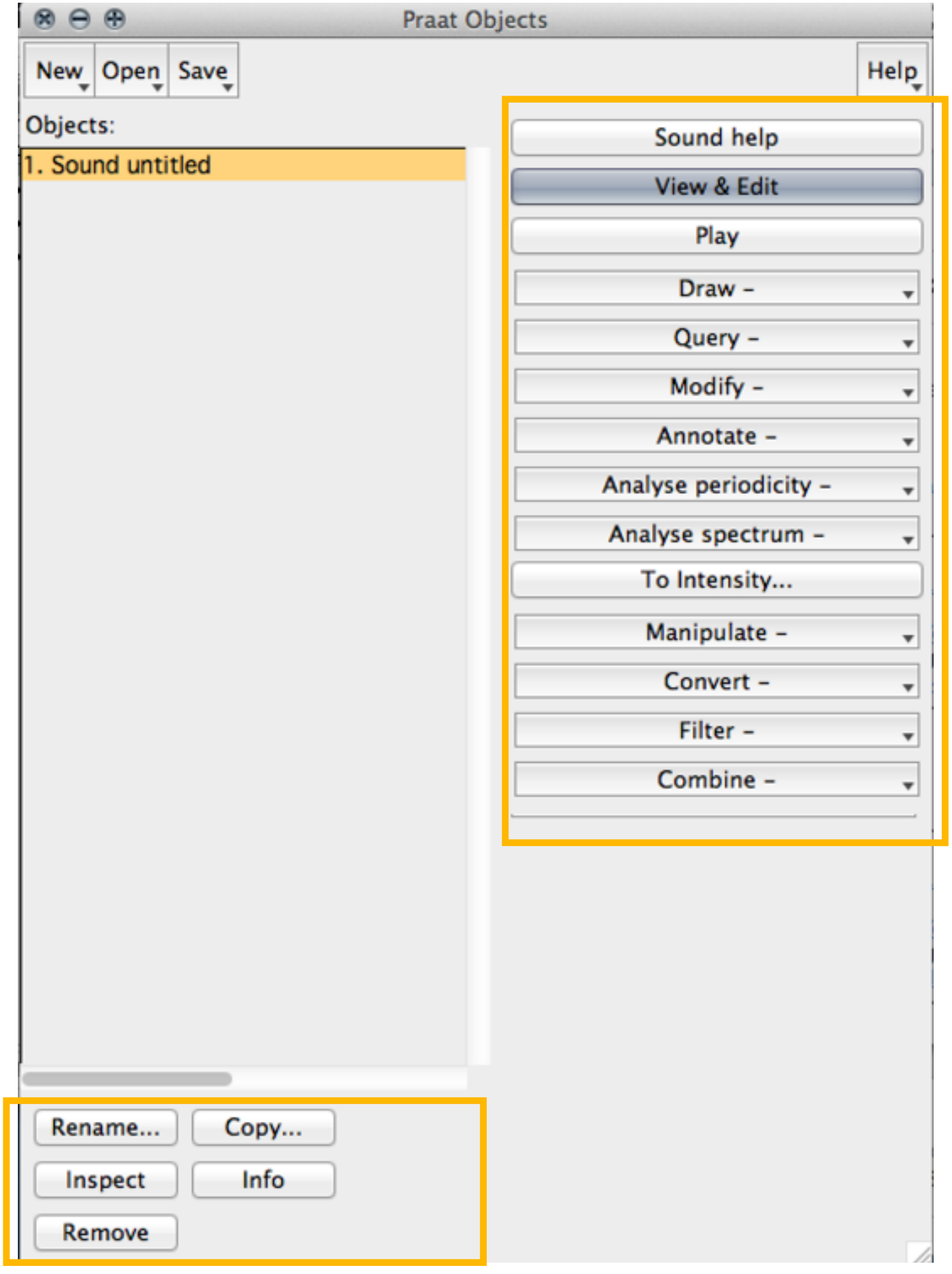
Sound object

- One example of an object is the **sound object**.

- Notice how selecting a sound object populates the right side of the Objects window with **dynamic buttons** and **menus**

- These are specific to the Sound object type.

- At the bottom of the screen are the **fixed buttons**. These never disappear and can't be deleted (more on this later)



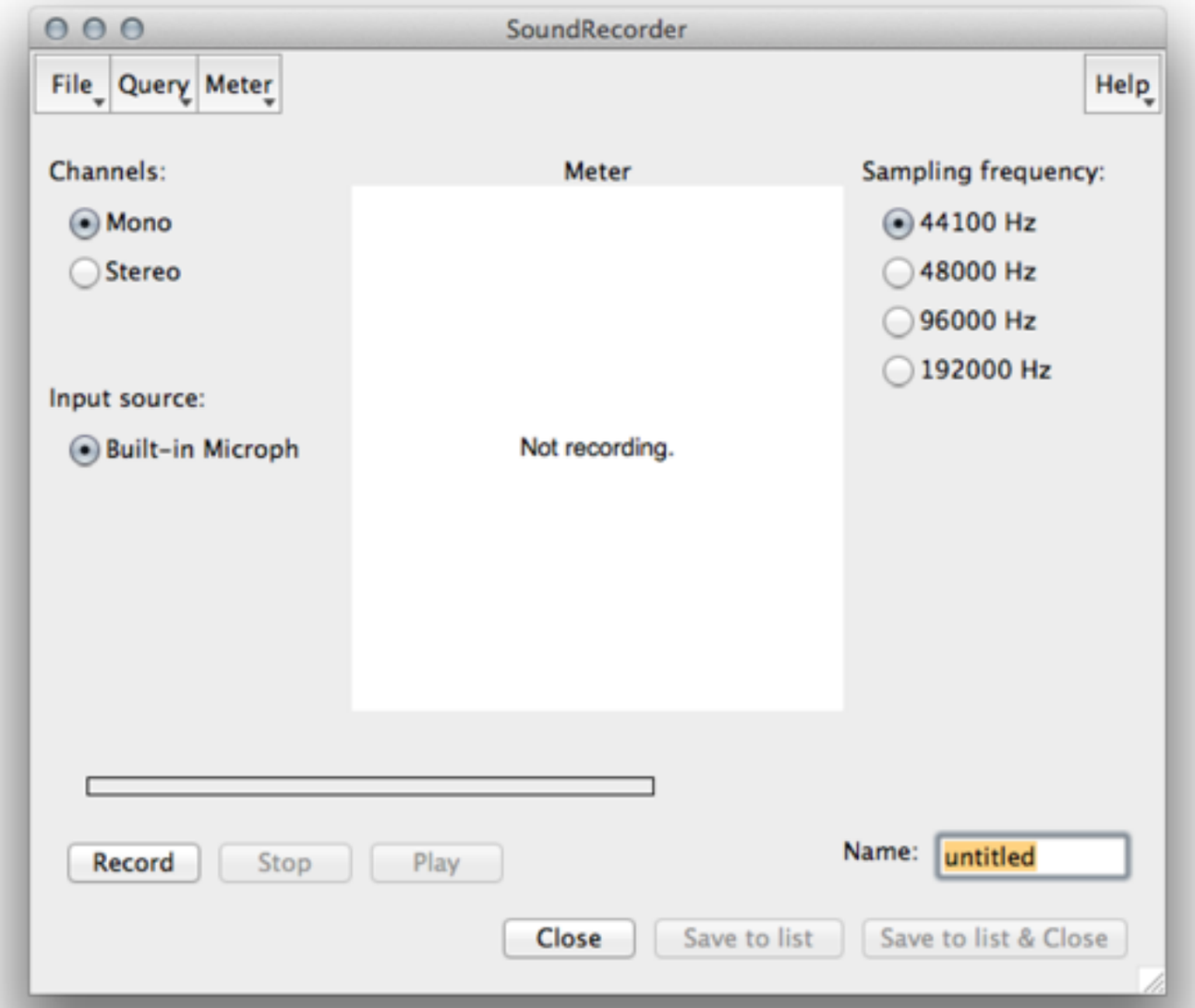
Exercise

- **1.** Open the SoundRecorder window (Objects window → New → Record mono sound (or ⌘R)

- **2.** Record yourself saying something very short (e.g. “doh!”)

- **3.** Save directly to a file (bypassing the Objects window).

- **4.** Now press the button “Save to list & Close”



Q: what buttons are available?

SciptEditor (manual)

- One of the many features Praat offers is a built-in **text editor** that you can use to write and edit your Praat scripts.

- This editor lacks many of the features that are kind of essential for efficient programming (auto-indent, syntax highlighting, brace matching, etc.) but it *does have* three killer features that mean you will end up using it at least occasionally during the development and testing phase of every script you write.

    1. The History Mechanism

    2. Run (⌘R) and Run Selection (⌘T) commands

    3. The ability to add scripts to existing Praat fixed and dynamic menus.

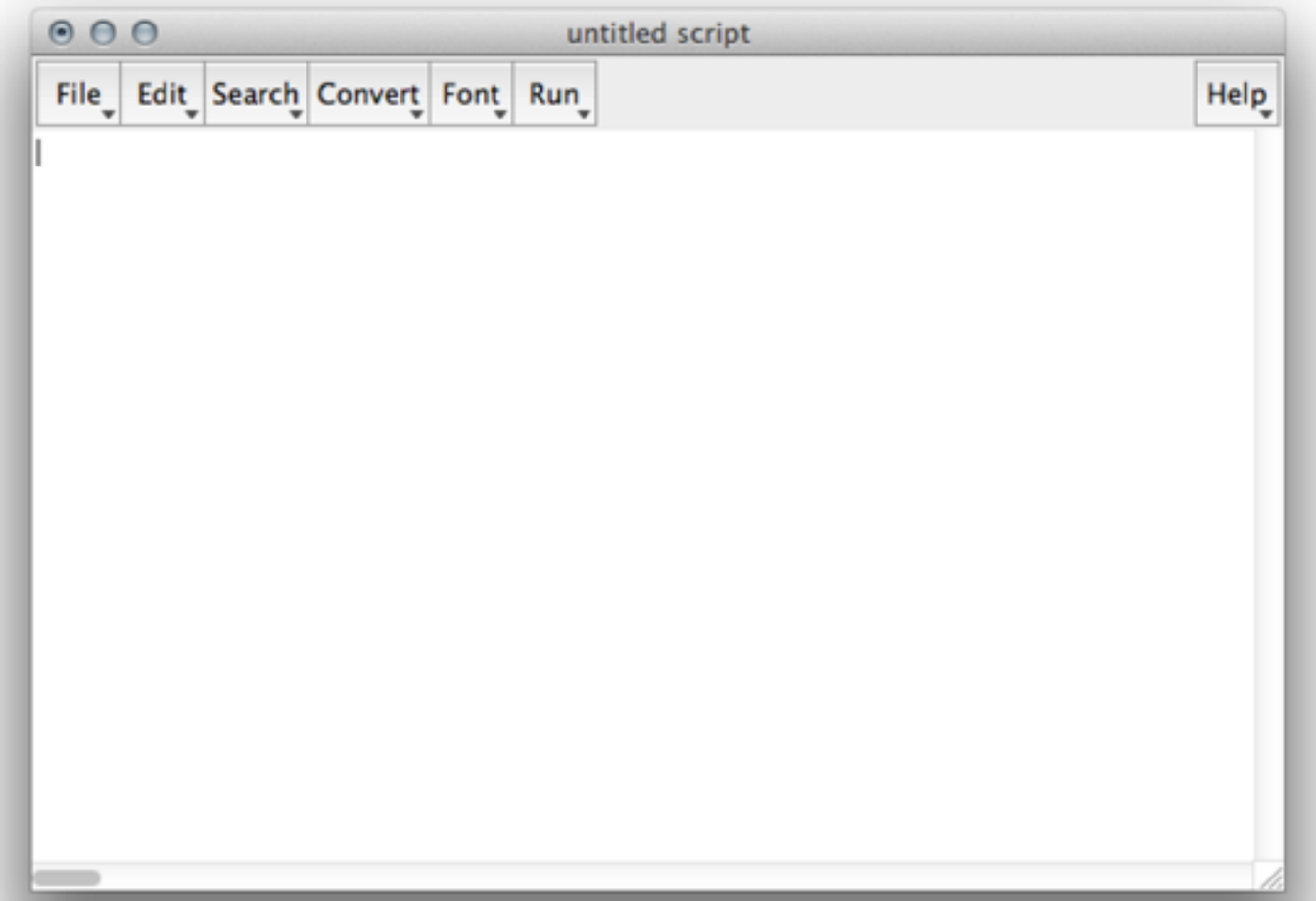
Exercise

1. Open the ScriptEditor window
   (Praat → New Praat script)

2. Press escape [ESC] several times

3. Paste your command history into the ScriptEditor window (Edit → Paste history) or (⌘H)

4. Inspect your new Praat script!



Discussion:

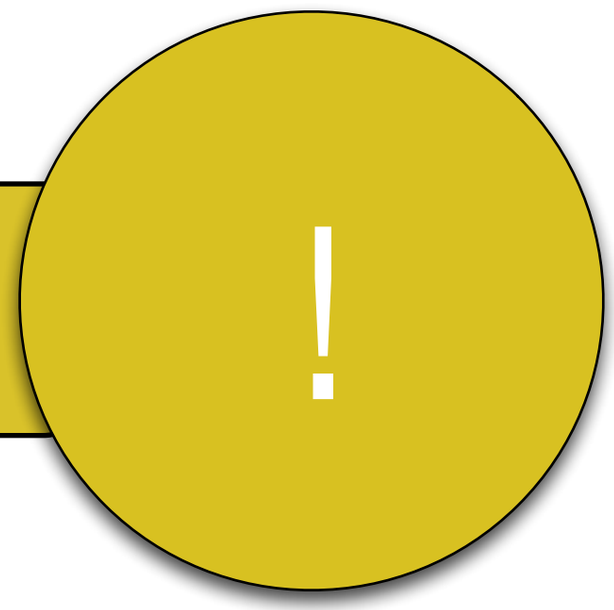
1. What observations can you make about the text that appears?
2. What sorts of commands *do not* appear?

Question

- What would happen if we were to run the Praat script we just created?
- **Answer:**

What happened when you tried it?

Don’t be afraid to experiment!



Let’s write (and break) our first real Praat script

1. Erase any text in the ScriptEditor window (select all and then Edit → Erase)

2. Select the Sound object in the Objects window

3. Clear your command history in the ScriptEditor (Edit → Clear history)

4. Press the dynamic button ‘Play’

5. Paste your command history into the ScriptEditor window (Edit → Paste history) or (⌘H)

6. Run your new script (Run → Run) or (⌘R)

7. Create a new TextGrid object (Objects window → Annotate → To TextGrid...) & accept defaults by clicking [ok]

8. Now what happens when you run your script?

How could we reorder these steps to fix the problem in 8?

Read the error message!

- Praat's error messages are pretty informative and often tell you precisely what is wrong with your script.

- In this case, you should have gotten something like:

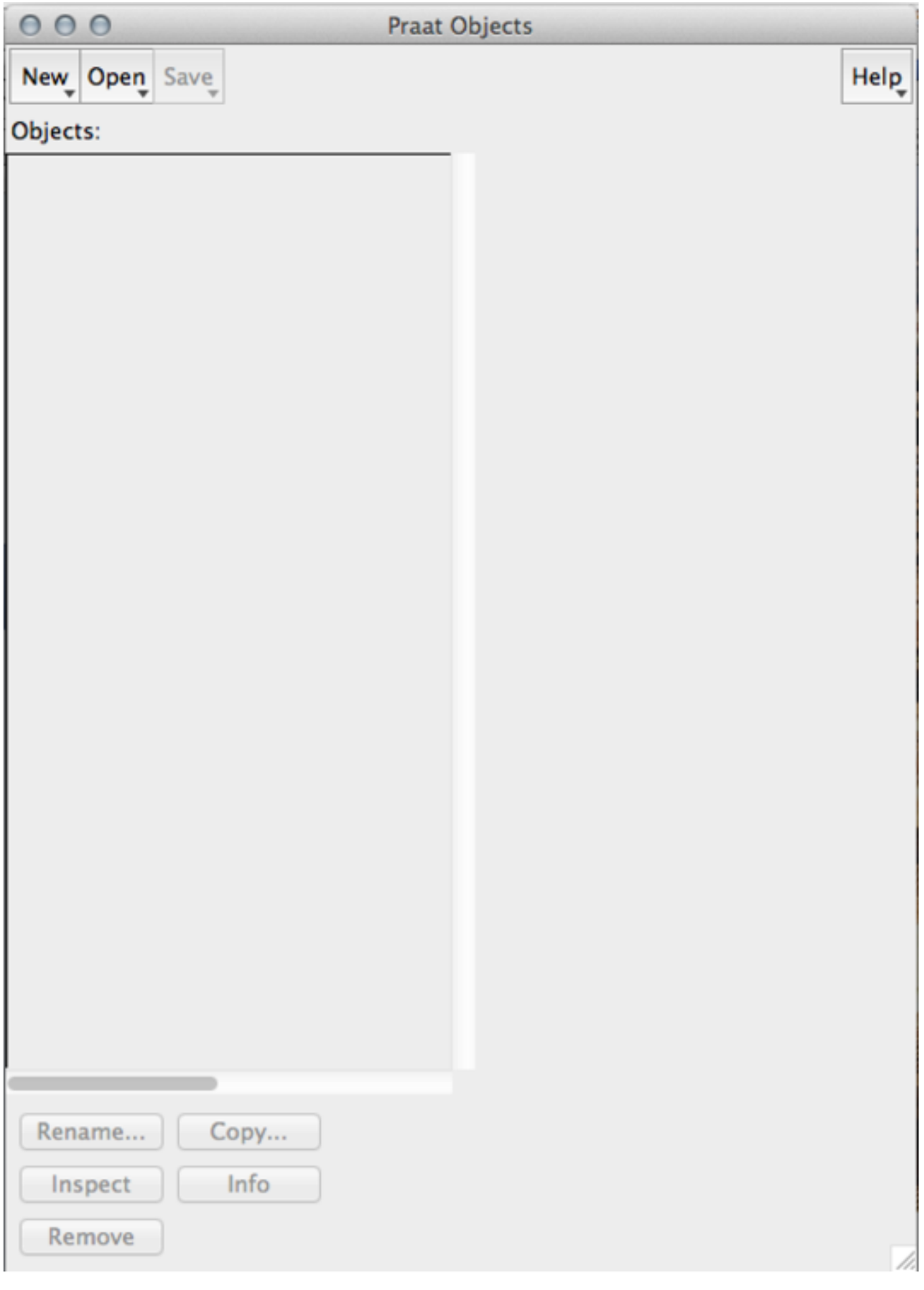
Objects window

- Remember: the Objects window is the center of the Praat universe.

- **Before you can use an object, you have to select it!**

- In the GUI (graphical user interface), you do this with a mouse click.

- In a script you have two choices...



Selecting an object

- Selecting an object makes it active for Praat.

- You might think of it as focusing Praat's attention on the object or making it the focus of any commands or requests for data.

- This happens implicitly when you create an object (this is why the new TextGrid broke the play() command –the TextGrid was implicitly selected)

- *Or* you can do it explicitly with selectObject() (manual)

```
selectObject: "Sound untitled"
selectObject: "TextGrid untitled"
selectObject: 1
```

Editor windows

- One of the behaviors available to some object types is to launch an editor window

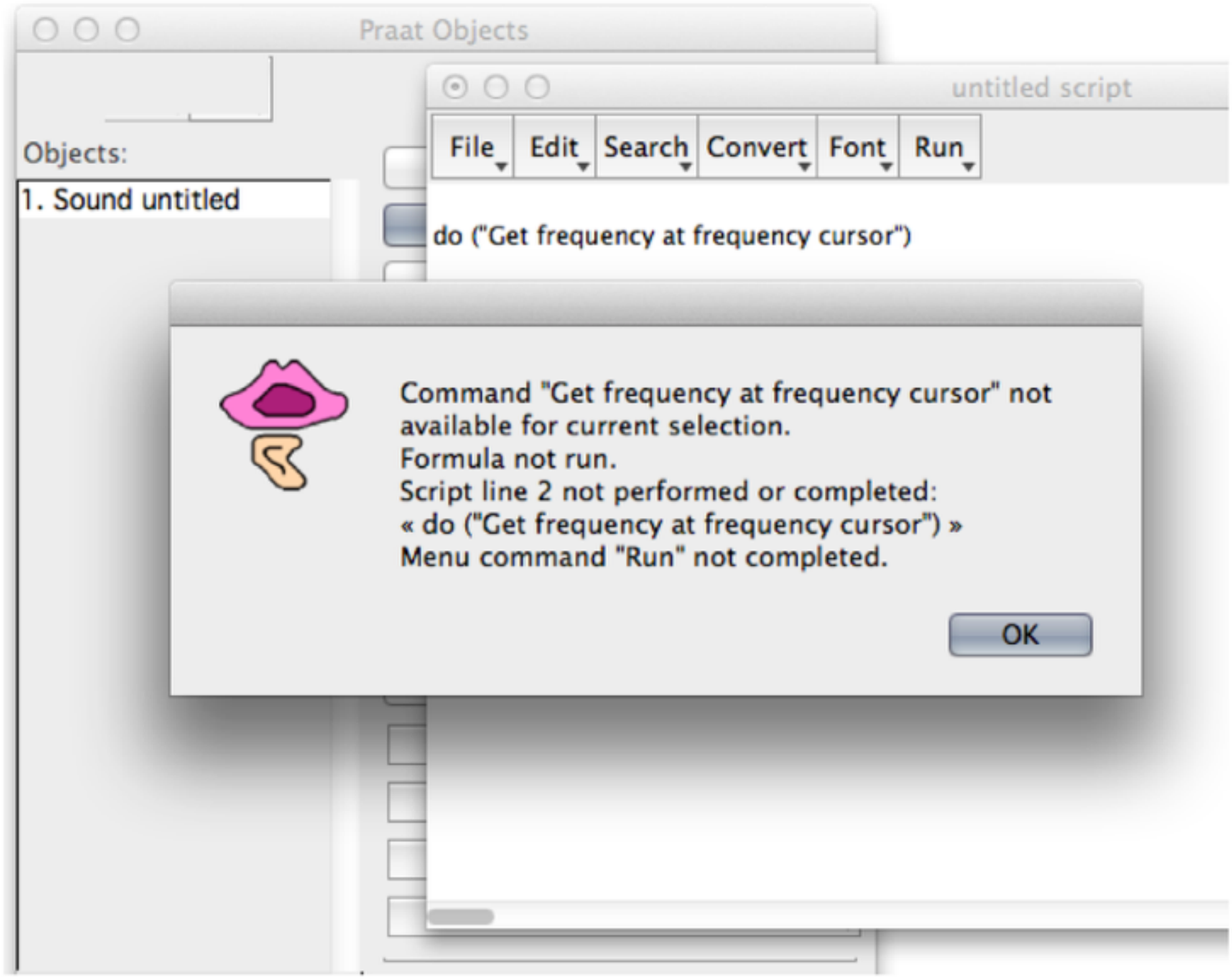
- The ones you will probably encounter most often are the SoundEditor and the TextGridEditor, but there are 10 others (help: Editors)

- You can control most editor window functions with a script (but normally you don't want to!)

- A very common source of runtime errors in Praat is attempting to issue an Editor window command from the Objects window (or vice versa).

- The error looks like this…

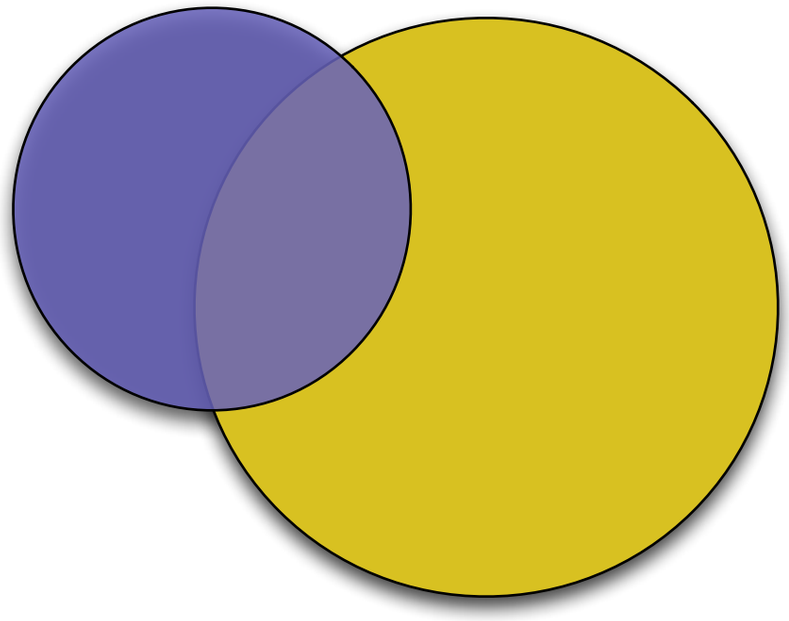
SoundEditor command from Objects Window



Praat Scripting

Part 2: Hello, World!

Putting scripts in files...



Files

- Rarely you might write Praat scripts that you only need or want for the current session

- But *typically* you will want to save your script to reuse later

- By convention, these files end with the extension `.praat`

- I create a 'scripts/' directory associated with each project I am working on and keep all associated scripts there

- Be sure to use comments!

Comments

```
# lines beginning with a # are comments
; so are lines starting with a semicolon
```

- You can leave a helpful comment like this:

```
    # user needs to hear, and approve, the sound
    Play
```

- And while you can't use # for a short **inline comment:**

```
*Play # user needs to hear, and approve, the sound
```

- You *can* do this:

```
    Play ; user needs to hear, and approve, the sound
```

Best practices for writing comments

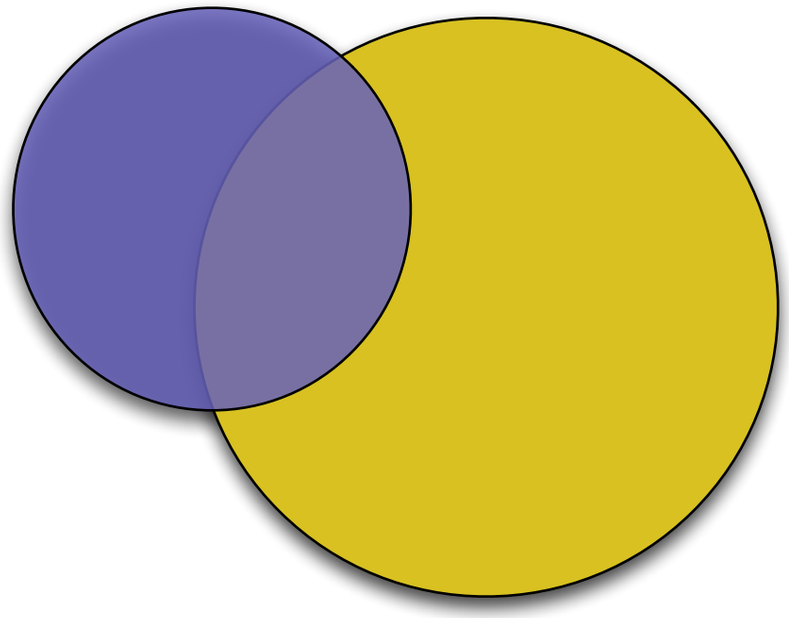
- Don't just describe what the code is doing –assume your reader knows the language or can look it up. **Your code should tell readers how you are doing a thing, your comments should tell them why!**

- Avoid comments like:

```
# has various idiosyncrasies
```

- Write comments *while you are coding* to explain why you are doing things (because you will forget)

Exercise: Hello, world!



Version 1: The Info window

- Please write the following Praat script and save it as 'hello.praat'

```
# hello, world! in the Info window

writeInfoLine: "hello, world!"
```

```
# another good (short term) use of comments is to leave
# yourself notes while developing or debugging...

# NOTES:
# experiment with changing the text and running this
# script multiple times.
# what EXACTLY does ‘writeInfoLine’ do?
```

Version 1: The Info window

```
# hello, world! in the Info window

writeInfoLine: "hello, world!"

# experiment with changing the text and running this
# script multiple times.
# what EXACTLY does writeInfoLine do?

# hint: compare it with the behavior of:

appendInfoLine: "it's nice to meet you."



# see http://www.fon.hum.uva.nl/praat/manual/
Scripting_3_1__Hello_world.html
```

Version 2: The Picture window

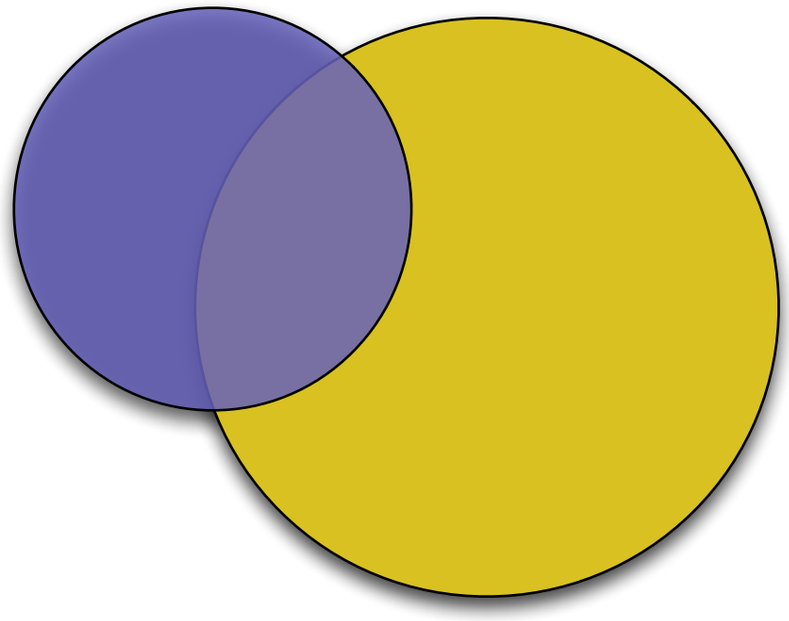
```
# hello, world! in the Picture window

Erase all
Text top: "yes", "hello, world!"

# question:
# what are the ellipses in the second line for?
# what does the ‘yes’ in the second line mean?
```

```
# see http://www.fon.hum.uva.nl/praat/manual/
Scripting_3_1__Hello_world.html
```

Version 3: The Demo window

```
# hello, world! in the demo window

demo Erase all
demo Axes: 0 10 0 10
demo Text: 5 centre 5 half Hello

# experiment with changing the parameters!
# what happens if you play with the numbers?
# can you identify x and y positions?
# what if you don’t Erase all first?
# can you change the font, size and color?



# see: http://www.fon.hum.uva.nl/praat/manual/Demo_window.html
```

Variables



A more flexible greeting

```
# store the greeting in a string variable
# (more on these later)

greeting$ = "hello, world!"
writeInfoLine: greeting$
```

A more flexible greeting

```
# or get it at run time in a form

form Greet the nice user
    comment Please name the entity we should greet:
    text greeting
endform

writeInfoLine: "hello, ", greeting$, "!"
appendInfoLine: "hello, " + greeting$ + "!"
appendInfoLine: "hello, 'greeting$'!"

# question: what do the plus signs do in the second line?
# question: what are the single quotes for in the third line?
```

Variables

- As we've just seen, a **variable** works like a named bucket that can store an arbitrary value and return it when you call for the bucket by name again

- A variable is an address in your computer's memory. This address points at a **value**.

- The variable is on the left side of the equals operator (=) and the value of that variable is on the right side

```
# in Praat, a variable assignment looks like this:
greeting$ = "world!"
```

- We can read this as: the string variable 'greeting' points at the string constant "world!"

Variables (cont)

- Variables have three attributes:

1. **name:** a unique identifier for the memory address
2. **type:** defines the affordances and behaviors of the variable
3. **value:** the information stored in the variable –can be a **literal** or **constant** value typed-into the script, read from the world, or calculated at runtime

Variable names

- Variable names in Praat must begin with a lowercase ascii letter (a through z)
- The following variable names will generate syntax errors:
  - 1variable (but variable1 or var1abl3 are fine)
  - Variable (but variable is fine)
  - üvariable (but värïäblë is fine)
  - åvariable (but vårïable is fine)

Uppercase & lowercase in Praat

- Not only are all *variables* lowercase-initial in Praat, but all script-specific commands are also lowercase-inititial.

    - Examples include: `demo, do, form, writeInfoLine, if, while,` selectObject, etc.

- All of Praat's Button commands (i.e. menu commands, Editor commands, dynamic and fixed commands for objects) begin with an uppercase letter

    - Examples include: `Play, Paint..., About Praat..., Select..., To Spectrogram...`, etc.

Variable names: available characters

- In general, it is safe to use letters, numbers, and underscore (_) in variable names.

- Praat variable names cannot contain punctuation

- Non-ascii characters can be used as data or constants in Praat scripts, but not as variable names

```
# legal praat code
caution$ = "حذر"
writeInfoLine( caution$ )

# syntax error
慎重$ = "caution"
writeInfoLine( 慎重$ )
```

Basic variable types

- Praat has two basic variable types:
    - **string variables**, like we have used so far, end in a $ character and can hold anything from the empty string to (at least) four copies of War and Peace.
    - **numeric variables** can be integers or real (decimal) numbers
- As with object types, variable types dictate what actions and behaviors are available for a variable.

Numeric variables

- Praat's numeric variables can hold values between -1,000,000,000,000,000 and +1,000,000,000,000,000 or real numbers between -10^{308} and 10^{308}

- Numeric constants may not contain commas (try it, see what happens)

- Decimal numbers must have at least a zero to the left of the decimal point.

- Numeric assignments look like these:

```
c = 35000       ; cm/s speed of sound in air
length = 17.5   ; cm vocal tract
f1_ə = (2 * 1 - 1) * c /  ( 4 * length )
f2_ə = (2 * 2 - 1) * c / ( 4 * length )

writeInfoLine( "According to the tube theory, F1 is ", f1_ə, " and F2 is
", f2_ə )
```

Exercise

1. Create a string variable whose value is a number.
   (this may take a few tries)

2. Try multiplying this string variable by 2

- You can convert a string to a number or a number to a string.

```
a$ = string$: a

a = number: a$
```

Extended Exercise 1

- Record yourself saying “Hello, world!” in any language that you like and save to a wav file.

- Write a Praat script to read this wav file and paint a spectrogram of this sound in the Picture window (please show 0 to 8000Hz). Let Praat “garnish” this spectrogram.

- Your script should also add a title to the figure of the form: “Dia dhaoibh, a dhomhain!” Irish spoken by Caoimhín Mac Gobhann

- Begin the script with a comment that gives your name and a brief description of what the script is meant to do.

- E-mail me the script file (YourName.praat), your wav file, and a PDF (or EPS on Windows/Linux) of the spectrogram if you want me to see it!

Aside: Text editor for programming

- Praat scripts (just like python, perl, R, matlab, C, etc...) need to be stored in **plain text files**

- Programs like Word, Pages, LibreOffice Writer, etc. don't do this; their files contain hidden formatting commands and **binary** (as in not text) information

- Praat's built-in ScriptEditor will suffice for short scripts and will be necessary at some point in the life of almost every script

- But you should try out a text editor that is specifically designed for programming and see if you like it

- Learning to use one will save you time in the long run

Aside: Suggested editors

- **Windows**: notepad++ http://notepad-plus-plus.org/download
    - free, open source program, possibly with syntax highlighting
- **Macintosh**: TextWrangler http://www.barebones.com/products/textwrangler/
    - free as in beer (not free as in speech) and also possibly with syntax highlighting
- **Unix (including Mac)**: vim (already installed on your computer)
    - free, insanely powerful, open source program with syntax highlighting
    - steep learning curve but with a game to teach you

Praat Scripting

Part 3: Variables and Conditionals

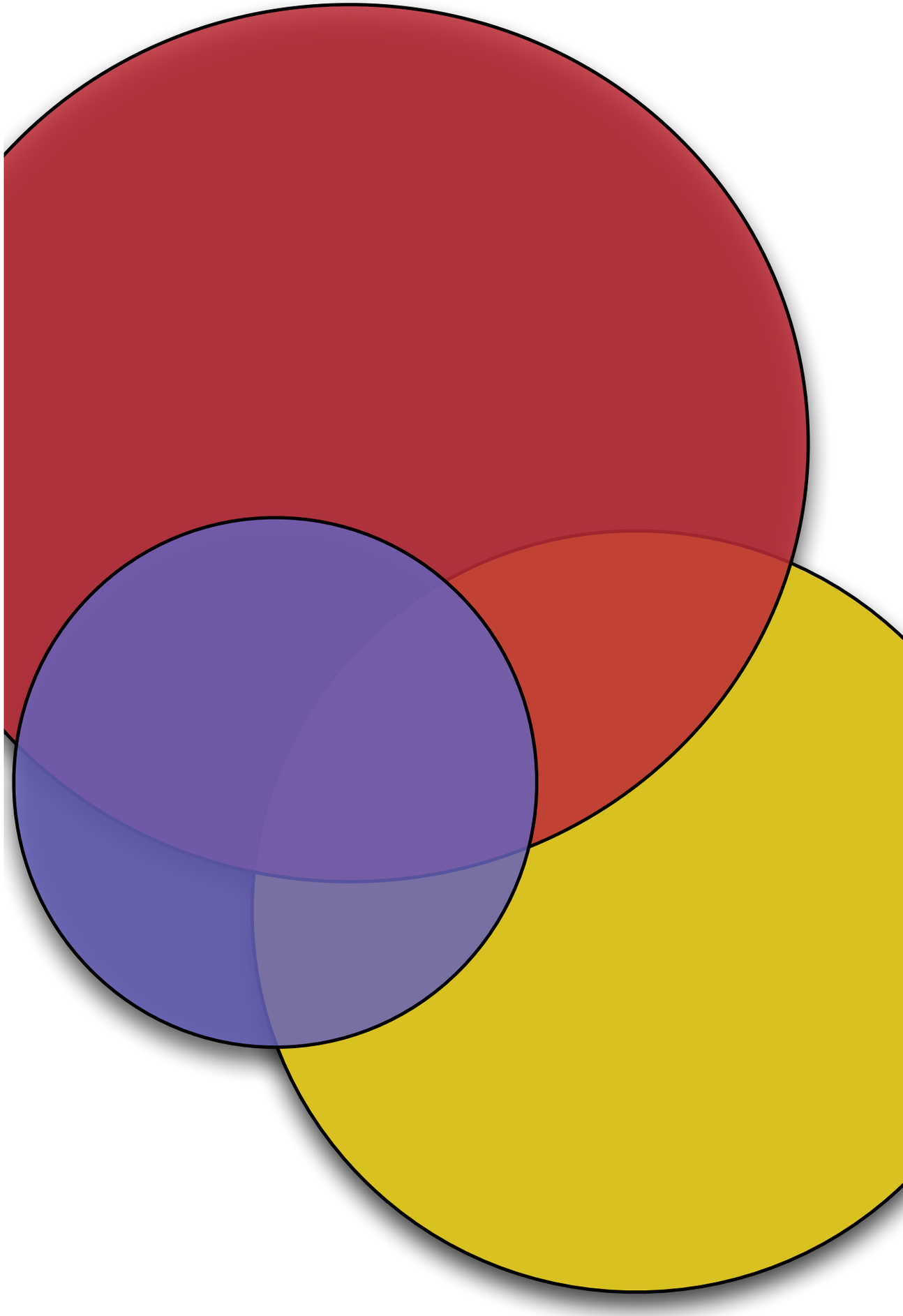
Review

- So far we have investigated the Praat Objects window, the center of Praat's universe;

- We have practiced writing simple scripts;

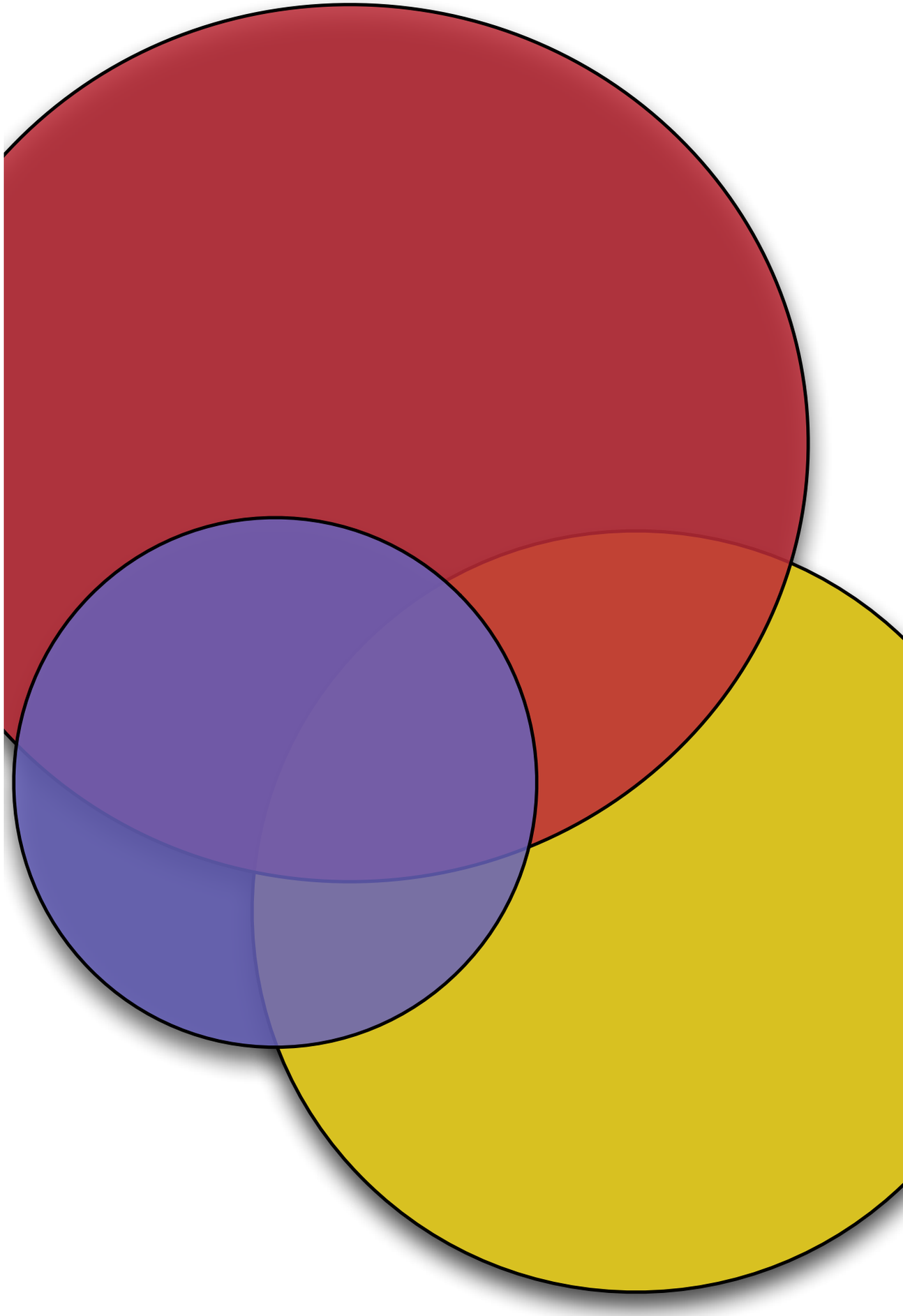
- adding comments;

- and working with numeric and string variables



Overview

- Next we will go over Extended Exercise 1;
- Finish looking at simple variables;
- Introduce conditionals (if/elsif/else/endif)



Extended Exercise 1: Review

EA1: Example solution

sound = Read from file: "/Users/clunis/Desktop/untitled.wav"
sgram = To Spectrogram: 0.005, 8000, 0.002, 20, "Gaussian"

Paint: 0, 0, 0, 8000, 100, "yes", 50, 6, 0, "yes"
Text top: "no", """Hello, world!"" in Irish spoken by Kevin McGowan"

Save as PDF file: "DiaDuitArDomhan.pdf"

EA1: Example solution (better)

 # Paint a publishable spectrogram
#
 # Reads a sound from a wav file, creates and paints a spectrogram of that
 # sound, adds a meaningful title, and saves a PDF file.
#
 # 6-25-2013 Kevin B. McGowan <kbmcgowan@stanford.edu>

sound = Read from file: "/Users/clunis/Desktop/untitled.wav"
sgram = To Spectrogram: 0.005, 8000, 0.002, 20, "Gaussian"

Paint: 0, 0, 0, 8000, 100, "yes", 50, 6, 0, "yes"
Text top: "no", """Hello, world!"" in Irish spoken by Kevin McGowan"

Save as PDF file: "DiaDuitArDomhan.pdf"

removeObject(sound)
removeObject(sgram)

EA1: Example solution (best)

```
# Paint a publishable spectrogram
#
# Reads a sound from a wav file, creates and paints a spectrogram of that
# sound, adds a meaningful title, and saves a PDF file.
#
# 6-25-2013 Kevin B. McGowan <kbmcgowan@stanford.edu>

sound = Read from file: "/Users/clunis/Desktop/untitled.wav"
sgram = To Spectrogram: 0.005, 8000, 0.002, 20, "Gaussian"

Erase all
Select outer viewport: 0, 6, 0, 4
Paint: 0, 0, 0, 8000, 100, "yes", 50, 6, 0, "yes"
Text top: "no", """Hello, world!"" in Irish spoken by Kevin McGowan"

Save as PDF file: "DiaDuitArDomhan.pdf"

removeObject: sound
```

EA1: Example solution (more bester)

```
#
# 6-25-2013 Kevin B. McGowan <kbmcgowan@stanford.edu>

sound = Read from file: "/Users/clunis/Desktop/untitled.wav"
sgram = To Spectrogram: 0.005, 8000, 0.002, 20, "Gaussian"

Erase all
Select outer viewport: 0, 6, 0, 4
Paint: 0, 0, 0, 8000, 100, "yes", 50, 6, 0, "yes"
Text top: "no", """Hello, world!"" in Irish spoken by Kevin McGowan"

if macintosh
    Save as PDF file: "DiaDuitArDomhan.pdf"
else
    Save as EPS file: "DiaDuitArDomhan.eps"
endif

removeObject: sound
```

Attributes of Objects (manual)

- Often you will use variables to store some fact about an object or the output of a Button command.

- These can be cumbersome in calculations. For example, to find the midpoint of a recording, you might write the following code:

```
# find the midpoint of our sound object
selectObject: "Sound untitled"
duration = Get total duration

midpoint = duration / 2

# print it to 2 decimal places
writeInfoLine: "midpoint is: ", fixed$( midpoint, 2 ), "s"
```

Attributes of Objects

- But if you don't actually *need* the duration variable for anything else (we use it to calculate midpoint and that's all) you can use the Sound object's attribute instead:

```
# find the midpoint of our sound object
midpoint = Sound_untitled.xmax / 2

# print it to 2 decimal places
writeInfoLine: "midpoint is: ", fixed$( midpoint, 2 ), "s"
```

Object attributes

- Object attributes are a relatively closed class in Praat
- They are documented in the Praat manual

Exercise

- Please write a script to:

1. Generate a pure tone (hint: use command history for this)

2. Calculate the sampling frequency of that pure tone (hint: 1 / dx)

3. Write the sampling frequency to the Info window without removing any text that might already be there.

Quotes

- double quotes ("") are used to define a string literal
  e.g. `greeting$ = "world!"`

- Single quotes can be used to force Praat to substitute a string variable for its value. e.g. `appendInfoLine: "hello, 'greeting$'!"`

This use of single quotes for variable substitution used to be (prior to last year) very common in Praat and resulted in ugly, difficult syntax and strange bugs. It should be avoided now. (see Boersma's message on the subject)

More on quotes

- Single quotes cannot be used to define a string constant. The following generates a syntax error:

- To embed double quotes in a double quoted string, double the double quotes:

```
*greeting$ = 'world!'
```

- To embed double quotes in a double quoted string, double the double quotes:

```
writeInfoLine: "And he was all, ""whoa."" and I"
appendInfoLine: "was all, ""right?"""
```

Numeric Operators & Precedence (manual)

precedence -->

negation (-) and exponentiation (^)	2^2^3 = 2^8 = 256 5^-3 = 5^{-3} 0.008	(right to left)
multiplication (*) and division (/)	1 / 4 * 2 = 0.5 1 / (4 * 2) = 0.125	
addition (+) and subtraction (-)	7 * 2 + 2 = 16 7 * -2 -2 = -16	
comparison operators	eq. (=), not eq. (<>), lt (<), gt (>), lt or eq. (<=) and gt or eq. (>=)	
not (!), and (&), or (|)	not 7 + 3 = 10 6 & 6 x > 5 and x < 10	

<-- precedence

String Comparison (manual)

a$ = b$
true (1) if the strings are equal, and false (0) otherwise.

a$ <> b$
1 if the strings are unequal, and 0 otherwise.

a$ < b$
1 if a$ precedes b$ in ASCII sorting order.

a$ > b$
1 if a$ comes after b$ in ASCII sorting order.

a$ <= b$
1 if a$ precedes b$ in ASCII sorting order, or if the strings are equal.

a$ >= b$
1 if a$ comes after b$ or the two are equal.

Making decisions

- Sometimes it is necessary for a program to be capable of doing different things depending on input or calculations

- As in natural language, computer science calls the expressions capable of introducing multiple possible worlds **conditionals**

- The simplest form of a conditional has the following structure:

```
# note, this is not in any particular programming
# language.  This is an example of pseudocode.
if <test>
     what to do if the test is true
endif
```

Making decisions

- Notice that this pseudocode does nothing at all if the test is false.

- Here's an example with real Praat code:

```
if <test>
     what to do if the test is true
endif
```

- Here's an example with real Praat code:

```
# what does this code do when executed?
if 7 > 9
     writeInfoLine: "hello, world!"
endif
```

if, elsif, else, endif

- Praat has four reserved words set aside for conditional expressions:
- **if** – introduces a conditional, must include a test.
- **elsif** – allows for the inclusion of another possible true outcome (note the lack of 'e'! Can also be spelled 'elif'.
- **else** – if none of the preceding tests was true, execute this **block** of code
- **endif** – ends the conditional

Space best practices

- Notice the indentation:

```
if 7 >= 9
     # we have dropped through a wormhole into
     # a parallel (and inconsistent) universe
     writeInfoLine( "hello, not our world!" )
else
     # the physical laws of our universe apply
     writeInfoLine( "hello, world!" )
endif
```

- You should use 4 spaces to indent lines inside a block (some people prefer 2)

- Avoid tabs, there is no standard on the placement and display of tab stops

Conditional example

```
num = -3

if num = 0
     writeInfoLine: "number *is* 0."
elsif num < 0
     writeInfoLine: "number is less than 0."
else
     writeInfoLine: "number is greater than 0."
endif

appendInfoLine: ”wasn't that fun?"
```

Exercise: Random Test

- Praat has a function `randomInteger(min, max)` which is essentially telling Praat to pick a number between min and max (inclusive). e.g. `randomInteger(1, 10)` could be read "pick a number from 1 to 10".

- Please write a script to:

1. Generate a random number between 1 and 10 and store it in a numeric variable.

2. Test whether this number is greater than or equal to 6. If so, write "The number was 6 or higher." to the Info window.

3. If the number is not equal to or greater than 6, write "Your number was 5 or lower." to the Info window.

Exercise: Guess my number

- **[Difficult]** Now that you have a working random number script, modify it to:

1. Prompt the user with a form that says: "I am thinking of a number between 1 and 10. Can you guess it?"

2. Use the form to get an `integer` called `guess` from the user

3. Tell the user if their `guess` was (1) correct (2) too high or (3) too low

String Concatenation and Truncation

a$ + b$
Concatenates the two strings.

a$ - b$
Subtracts the second string from the end of the first.

```
soundFile$ = "bananas.wav"
soundFile$ = soundFile$ - ".wav"
tgFile$ = soundFile$ + ".TextGrid"
# tgFile$ now points to "bananas.TextGrid"
```

But notice!

```
soundFile$ = "bananas.aiff"
soundFile$ = soundFile$ - ".wav"
tgFile$ = soundFile$ + ".TextGrid"
# tgFile$ now points to "bananas.aiff.TextGrid"
# to fix this you want string functions
```

Using functions

- The behaviors available to perform on the different variable types are called **functions**

- We have already seen two examples of these in the `string$()` and `number()` functions

- Praat has a length() function that simply calculates and returns the length of a single string **argument**

```
string$ = "what has it got in its pocketses?"
length = length: string$
writeInfoLine: "The string '", string$, "' is ", length,
..." characters long."
```

- (you can break up a long line by inserting ellipses after newline)

Testing functions

- Remember that, in a conditional, `0` is false and any other number is true.

- You can verify this yourself with something like:

```
n = -4
if n
     writeInfoLine: "beauty!"
endif
```

- We can take advantage of this fact to use functions in conditionals...

Testing functions (cont)

- For example, to test if a file is a wav file or a TextGrid, you could write:

```
if endsWith: filename$, ".wav"
     # file is a wav, do sound file things to it
     # ...
elsif endsWith: filename$, ".TextGrid"
     # file is a TextGrid, do TextGrid file things to it
     # ...
else
     # file is neither (or the case didn't match )
     # ...
endif
```

Available functions

- The Praat documentation lists the available functions:
  - Mathematical functions (manual: Formulas 4)
  - String functions (manual: Formulas 5)

Predefined variables

- Praat comes with a number of pre-defined variables and constants, some of them are:

- **Numeric**: `praatVersion` stores the current version of Praat you are running. `macintosh`, `windows`, and `unix` are true (1) if you are on that platform. `pi`, `e`, and `undefined` (`manual: constants`)

- **String**: `praatVersion$` version of Praat as a string. `newline$`, `tab$` store your platform's take on newline and tab. `defaultDirectory$` is the path containing the script file. `homeDirectory$`, `preferencesDirectory$`, and `temporaryDirectory$` provide useful directory paths.

Exercise: digits of π

- How many digits of π does Praat's constant give you?
- Input: Praat's `pi` constant
- Output: Info window should read:

```
Praat stores pi to N decimal places.
```

- Where 'N' is replaced with the correct number of decimal places.
- **Hint**: Praat will give you a number, but you probably want to work with it as a string.