

M269 Overview

M269 Overview A

Contents

1	M269 Overview A Tutorial Agenda	1
2	Adobe Connect	3
2.1	Student View	3
2.2	Settings	4
2.3	Student & Tutor Views	6
2.4	Sharing Screen & Applications	8
2.5	Ending a Meeting	8
2.6	Invite Attendees	8
2.7	Layouts	9
2.8	Chat Pods	9
3	M269 Overview	9
4	Basic Computational Components	10
4.1	Computation, Programming, Programming Languages	11
4.2	Programming Languages	13
5	Python	15
5.1	Learning Python	15
5.2	Setting up Python with Komodo	15
5.3	Basic Python	21
5.4	Python Workflows	22
6	Learning Software Packages	24
6.1	Installing Komodo & Python	24
6.2	Learning Komodo	24
7	What Next ?	28
8	Web Links & References	29
	References	29

1 M269 Overview A Tutorial Agenda

- Introductions
- M269 Overview
- Unit 1 — Komodo and Python
- How to survive learning software packages
- *Adobe Connect* — if you or I get cut off, wait till we reconnect (or send you an email)

- Time: about 1 hour
- Do ask questions or raise points.
- Slides [M269Prsntn2019JTutorialOverviewA.beamer.pdf](#)
- Notes [M269Prsntn2019JTutorialOverviewA.article.pdf](#)
- Overview A — Basic Python
- Algorithm design examples

Introductions — Me

- *Name* Phil Molyneux
- *Background*
 - Undergraduate: Physics and Maths (Sussex)
 - Postgraduate: Physics (Sussex), Operational Research (Brunel), Computer Science (University College, London)
 - Worked in Operational Research, Business IT, Web technologies, Functional Programming
- *First programming languages* [Fortran](#), [BASIC](#), [Pascal](#)
- *Favourite Software*
 - [Haskell](#) — pure functional programming language
 - Text editors [TextMate](#), [Sublime Text](#) — previously [Emacs](#)
 - Word processing in [L^AT_EX](#) — all these slides and notes
 - [Mac OS X](#)
- *Learning style* — I read the manual before using the software

Introductions — You

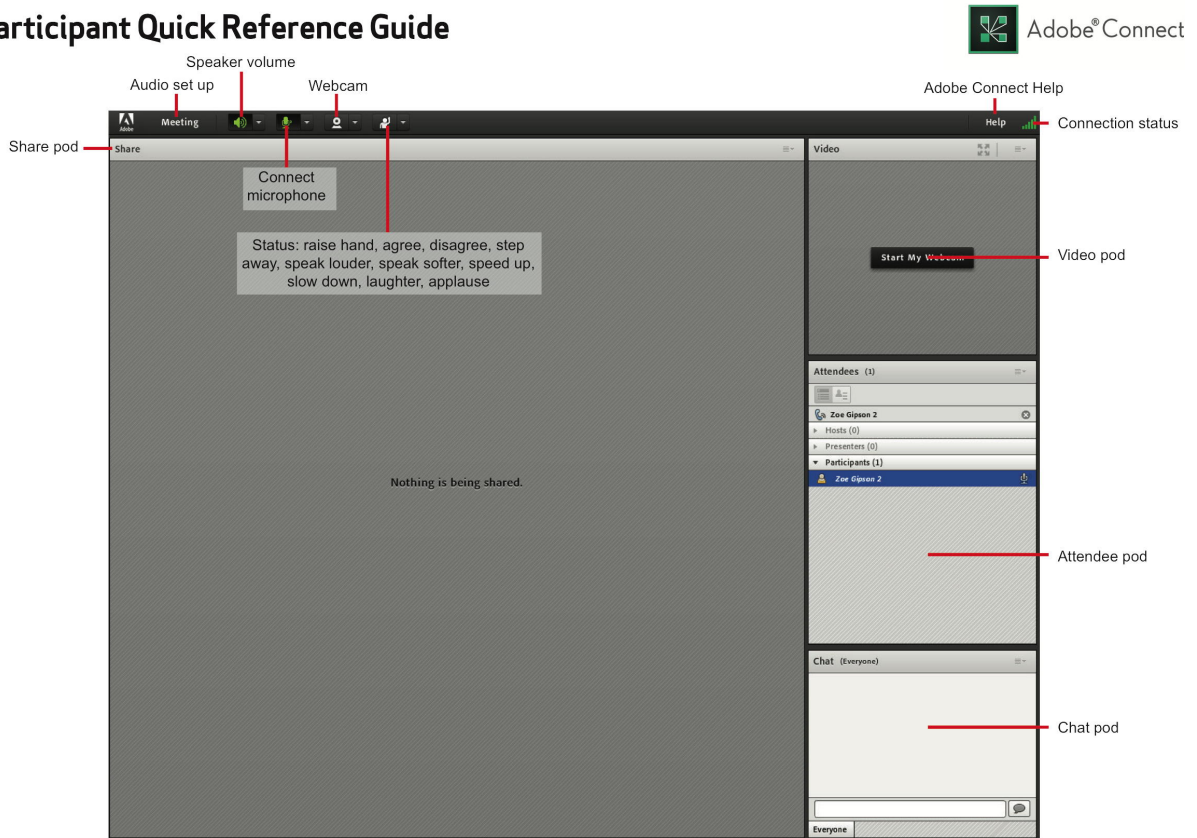
- *Name?*
- *Favourite software/Programming language?*
- *Other OU courses?*
- *Anything else?*

2 Adobe Connect Interface and Settings

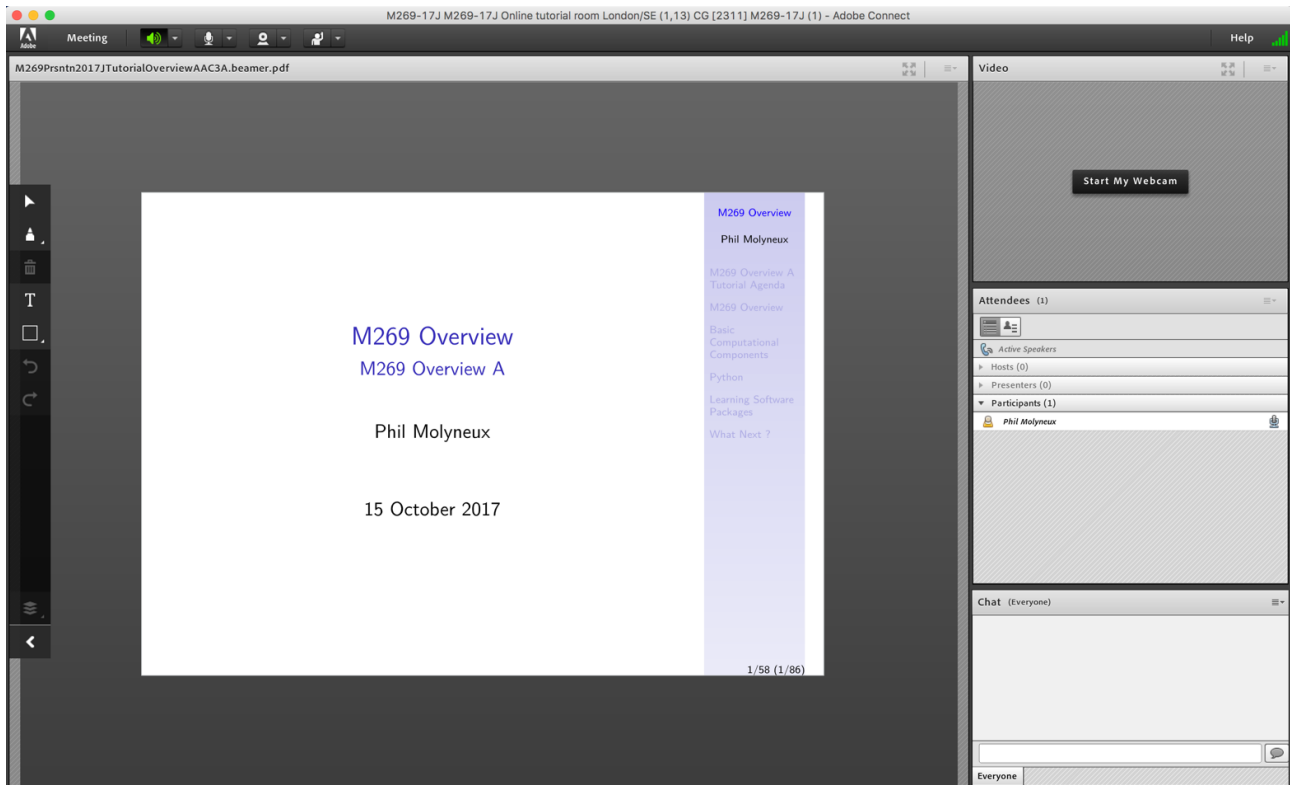
2.1 Adobe Connect Interface — Student View

Adobe Connect Interface — Student Quick Reference

Participant Quick Reference Guide



Adobe Connect Interface — Student View



2.2 Adobe Connect Settings

Adobe Connect Settings

- **Everybody: Audio Settings** Meeting > Audio Setup Wizard. . .
- **Audio** Menu bar > Audio > Microphone rights for Participants ✓
- Do not *Enable single speaker mode*
- **Drawing Tools** Share pod menu bar > Draw (1 slide/screen)
- Share pod menu bar > Menu icon > Enable Participants to draw ✓ gray
- Meeting > Preferences > Whiteboard > Enable Participants to draw ✓
- Cancel hand tool
- Do not enable green pointer. . .
- Meeting > Preferences > Attendees Pod > Disable *Raise Hand notification*
- **Cursor** Meeting > Preferences > General tab > Host Cursors > Show to all attendees ✓ (default *Off*)
- Meeting > Preferences > Screen Share > Cursor > Show Application Cursor
- **Webcam** Menu bar > Webcam > Enable Webcam for Participants ✓
- **Recording** Meeting > Record Meeting. . . ✓

Adobe Connect — Access

- **Tutor Access**

- [TutorHome](#) > [M269 Website](#) > [Tutorials](#)
- [Cluster Tutorials](#) > [M269 Online tutorial room](#)
- [Tutor Groups](#) > [M269 Online tutor group room](#)
- [Module-wide Tutorials](#) > [M269 Online module-wide room](#)

- **Attendance**

[TutorHome](#) > [Students](#) > [View your tutorial timetables](#)

- **Beamer Slide Scaling** 440% (422 x 563 mm)





- **Clear Everyone's Status**

[Attendee Pod](#) > [Menu](#) > [Clear Everyone's Status](#)

- **Grant Access**

[Meeting](#) > [Manage Access & Entry](#) > [Invite Participants...](#) and send link via email

Adobe Connect — Keystroke Shortcuts

- [Keyboard shortcuts in Adobe Connect](#)
- **Toggle Mic**  + **M** (Mac), **Ctrl** + **M** (Win) (On/Disconnect)
- **Toggle Raise-Hand status**  + **E**
- **Close dialog box**  (Mac), **Esc** (Win)
- **End meeting**  + ****

2.3 Adobe Connect Interface — Student & Tutor Views

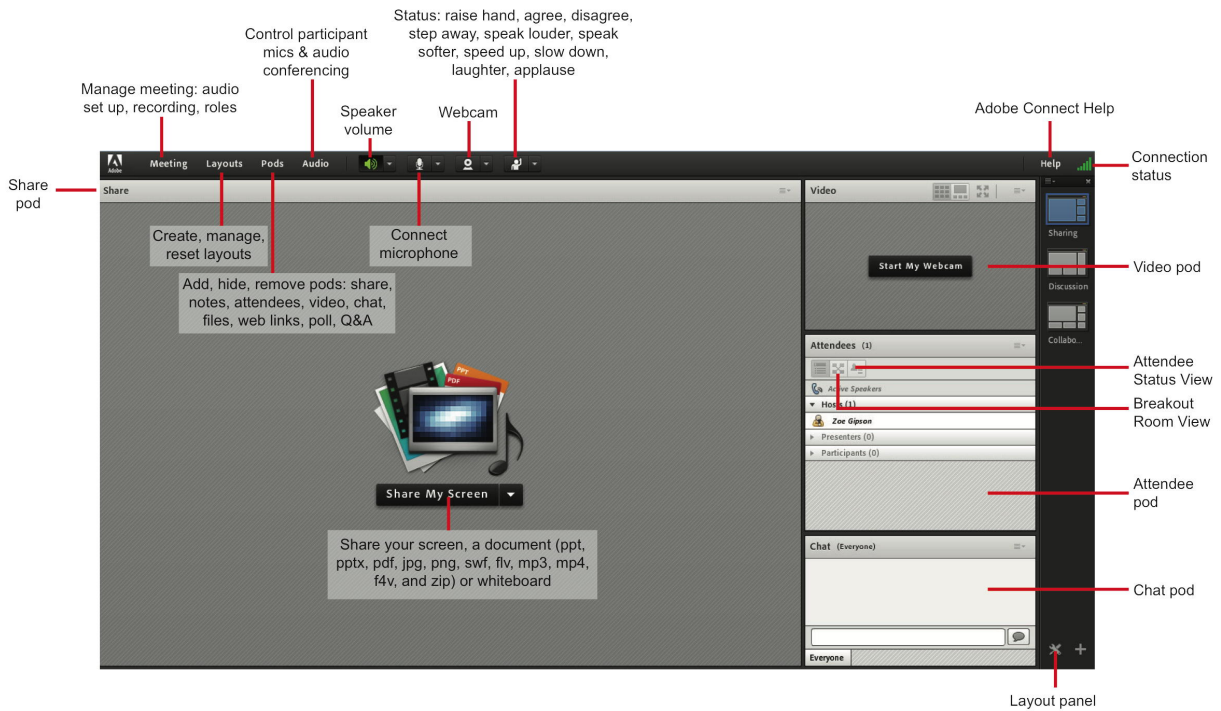
Adobe Connect Interface — Student View (default)

The screenshot shows the Adobe Connect interface in Student View. The main content area displays a Beamer presentation slide titled "M269 Overview" by Phil Molyneux, dated 15 October 2017. The slide content includes "M269 Overview", "M269 Overview A", "Phil Molyneux", and "15 October 2017". A table of contents is visible on the right side of the slide, listing items such as "M269 Overview", "Basic", "Computational Components", "Python", "Learning Software Packages", and "What Next?".

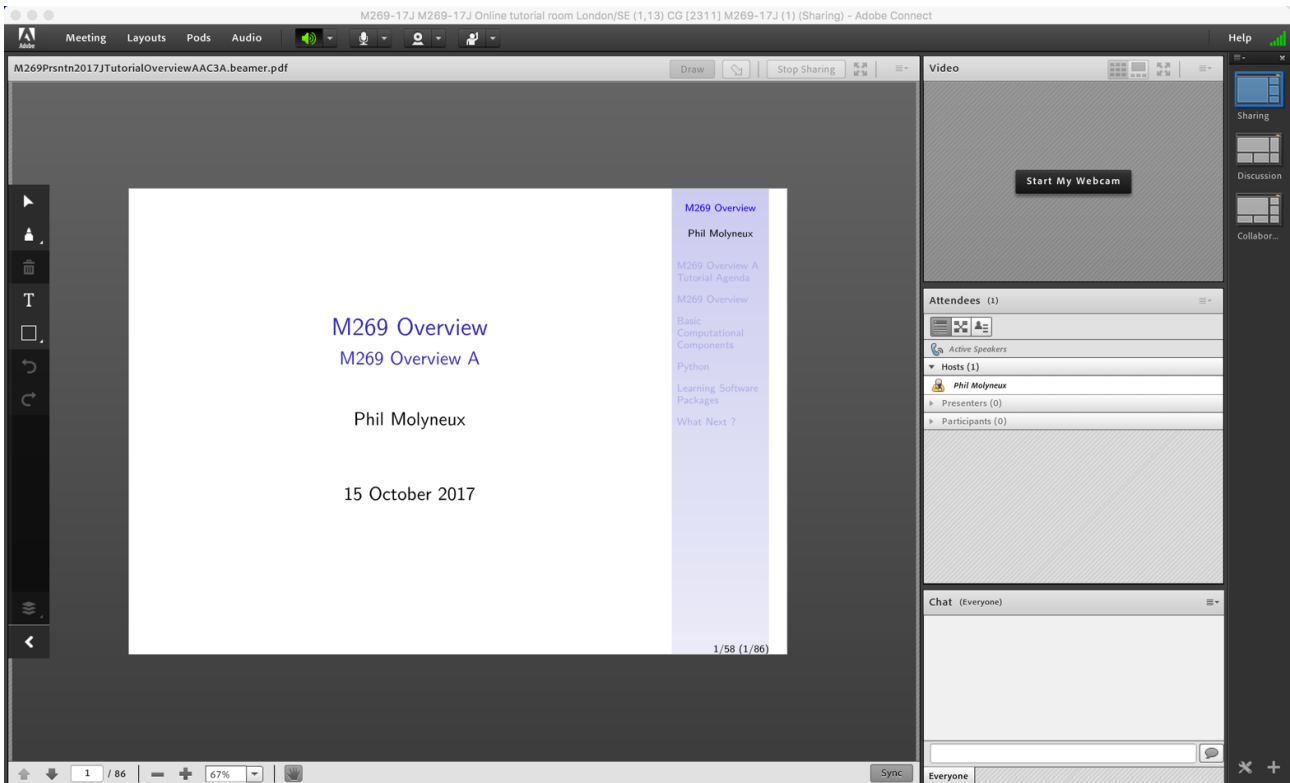
The interface includes a top toolbar with "Meeting" and "Help" buttons. On the right side, there is a "Video" panel with a "Start My Webcam" button, an "Attendees (1)" panel showing "Phil Molyneux", and a "Chat (Everyone)" panel at the bottom right.

Adobe Connect Interface — Tutor Quick Reference

Host Quick Reference Guide



Adobe Connect Interface — Tutor View



2.4 Adobe Connect — Sharing Screen & Applications

- **Share My Screen** > **Application tab** > **Terminal** for **Terminal**
- **Share menu** > **Change View** > **Zoom in** for mismatch of screen size/resolution (Participants)
- (Presenter) Change to 75% and back to 100% (solves participants with smaller screen image overlap)
- Leave the application on the original display
- Beware blue hatched rectangles — from other (hidden) windows or contextual menus
- Presenter screen pointer affects viewer display — beware of moving the pointer away from the application
- First time: **System Preferences** > **Security & Privacy** > **Privacy** > **Accessibility**

2.5 Adobe Connect — Ending a Meeting

- *Notes for the tutor only*
- **Student:** **Meeting** > **Exit Adobe Connect**
- **Tutor:**
- **Recording** **Meeting** > **Stop Recording** ✓
- **Remove Participants** **Meeting** > **End Meeting...** ✓
 - Dialog box allows for message with default message:
 - *The host has ended this meeting. Thank you for attending.*
- **Recording availability** *In course Web site for joining the room, click on the eye icon in the list of recordings under your recording* — edit description and name
- **Meeting Information** **Meeting** > **Manage Meeting Information** — can access a range of information in Web page.
- **Attendance Report** see course Web site for joining room

2.6 Adobe Connect — Invite Attendees

- **Provide Meeting URL** **Menu** > **Meeting** > **Manage Access & Entry** > **Invite Participants...**
- **Allow Access without Dialog** **Menu** > **Meeting** > **Manage Meeting Information** provides new browser window with *Meeting Information* **Tab bar** > **Edit Information**
- Check *Anyone who has the URL for the meeting can enter the room*
- Default *Only registered users and accepted guests may enter the room*
- **Reverts to default next session but URL is fixed**
- Guests have blue icon top, registered participants have yellow icon top — same icon if URL is open

- See [Start, attend, and manage Adobe Connect meetings and sessions](#)

2.7 Layouts

- **Creating new layouts** example *Sharing* layout
- [Menu](#) > [Layouts](#) > [Create New Layout...](#) > [Create a New Layout dialog](#) > [Create a new blank layout](#) and name it *PMolyMain*
- New layout has no Pods but does have Layouts Bar open (see Layouts menu)
- **Pods**
- [Menu](#) > [Pods](#) > [Share](#) > [Add New Share](#) and resize/position — initial name is *Share n*
- **Rename Pod** [Menu](#) > [Pods](#) > [Manage Pods...](#) > [Manage Pods](#) > [Select](#) > [Rename](#) or [Double-click & rename](#)
- Add Video pod and resize/reposition
- Add Attendance pod and resize/reposition
- Add Chat pod — name it *PMolyChat* — and resize/reposition
- Dimensions of **Sharing** layout (on 27-inch iMac)
 - Width of Video, Attendees, Chat column 14 cm
 - Height of Video pod 9 cm
 - Height of Attendees pod 12 cm
 - Height of Chat pod 8 cm
- **Duplicating Layouts** does *not* give new instances of the Pods and is probably not a good idea (apart from local use to avoid delay in reloading Pods)

2.8 Chat Pods

- **Format Chat text**
- [Chat Pod](#) > [menu icon](#) > [My Chat Color](#)
- Choices: Red, Orange, Green, Brown, Purple, Pink, Blue, Black
- Note: Color reverts to Black if you switch layouts
- [Chat Pod](#) > [menu icon](#) > [Show Timestamps](#)

[Go to Table of Contents](#)

3 M269 Overview

M269 Algorithms, data structures and computability

Aims

- Ideas of *computational thinking*

- Introduction to algorithms and data structures (using *Python*)
- Logic and the limits of computation
- Computability
- Complexity

M269 Algorithms, data structures and computability

Units

- *Unit 1 Introduction* — ideas of computation and introduction to *Python*
- *Unit 2 From problems to programs* — algorithms, logic and abstract data types
- *Unit 3 Sorting* — how do we derive and classify sorting algorithms ?
- *Unit 4 Searching* — patterns, strings; calculating positions: hashes; tree data structures for storing and searching.
- *Unit 5 Optimisation* — graph algorithms, dynamic programming.
- *Unit 6 Sets, logic and databases* — truth tables
- *Unit 7 The limits of computation* — computability, Turing machines, proofs, computational complexity

4 Basic Computational Components

Computational Components — Imperative

Imperative or procedural programming has statements which can manipulate global memory, have explicit **control flow** and can be **organised** into procedures (or functions)

- **Sequence** of statements

```
stmt ; stmt
```

- **Iteration** to repeat statements

```
while expr :
    suite

for targetList in exprList :
    suite
```

- **Selection** choosing between statements

```
if expr : suite
elif expr : suite
else : suite
```

Functional programming treats computation as the evaluation of expressions and the definition of functions (in the mathematical sense)

- **Function composition** to combine the application of two or more functions — like sequence but from right to left (notation accident of history)

```
(f . g) x = f (g x)
```

- **Recursion** — function definition defined in terms of calls to itself (with *smaller* arguments) and base case(s) which do not call itself.
- **Conditional expressions** choosing between alternatives expressions

```
if expr then expr else expr
```

4.1 Computation, Programming, Programming Languages

- M269 is not a programming course but . . .
- The course uses Python to illustrate various algorithms and data structures
- The final unit addresses the question:
- *What is an algorithm?* What is programming? What is a programming language?
- So it *is* a programming course (sort of)

Computation, Syntax and Semantics

- **Syntax and Semantics (1)**
- What is each of the following — *first reaction!*
- $4 + 6$
- $4 + 6 \times 3$
- 4
- $19370721 \times 761838257287$
- The above are *expressions in arithmetic*
 - Most students read *what is* as *evaluate*
 - Not easy for the last one
 - *But* you can say:
 - *They are expressions which when evaluated, evaluate to some number*
 - $19370721 \times 761838257287$
 - $= 147573952589676412927 = 2^{67} - 1$
 - demonstrated in a famous meeting of the New York AMS in October 1903 by [F.N.Cole \(Cole, 1903\)](#)

Computation — Cartesian Close Comic Cartoon



Sad fact: many math teachers do not know the difference between equality and reduction.

- **Syntax and Semantics (2)**
- **Evaluate**
- $6 + 4 \times 3$
- $6 - 4 - 1$
- False or True (in Python)
- $5 // 3$ (integer division in Python)
- $1 // 0$ (in Python)
- False or True or $1 // 0$ (in Python)

Syntax and Semantics — Elementary Concepts

- An *expression* can be thought of as a program (and vice versa)
- A set of instructions to find a value.
- Operator *precedence* and *associativity* are there to get rid of some brackets
- (to make the code more *user friendly*!)
- **Precedence** — which operator to use first. This is also called *binding power* or operator *fixity*

- **Associativity** — for the same operator, whether to evaluate from left to right or right to left (or it doesn't matter)
- **Lazy Evaluation** — don't do today what you can put off til tomorrow, because you might never have to do it (useful in computation — not useful for doing TMAs)
- **Sharp edges**
 - Evaluate (in Maths) 2^2 and 2^{2^2} and $2^{2^{2^2}}$
 - In Python `2**2**2**2`
 - Alternate in Python `pow(2, pow(2, pow(2, 2)))`
 - Microsoft Excel `=2^2^2^2`
 - or use LibreOffice, Numbers, ...
- **Sharp edges**
 - Evaluate (in Maths) 2^2 and 2^{2^2} and $2^{2^{2^2}}$
 - $2^{2^2} = 16$ and $2^{2^{2^2}} = 2^{16} = 65536$ (or 64K in computing)
 - Python `2**2**2**2 == 65536`
 - Python `pow(2, pow(2, pow(2, 2))) == 65536`
 - Casio fx-85GT Plus `2^2^2^2` shows 65536
 - Haskell `2^2^2^2 == 65536`
 - Microsoft Excel `=2^2^2^2 == 256`
 - *Beware language semantics*
 - Microsoft Excel `=2^2^2^2^2 == 65536`
 - Haskell `length (show (2^2^2^2^2)) == 19729`
 - $2^{2^{2^2}}$ has 19729 digits
 - What is Excel doing differently ?

4.2 Programming Languages

- Add a tick on the slide next to languages used
- FORTRAN
- BASIC
- Pascal
- SASL
- C
- Miranda
- Prolog

- [JavaScript](#)
- [Java](#)
- [Haskell](#)
- Add names of other languages used
- Are the following programming languages ?
- Excel
- HTML
- Word
- \LaTeX
- SQL
- **Excel**
- Excel has conditional expressions and indirections (so can have loops)
- An [Excel Turing Machine](#) is described in [Feliene's blog](#)
- Excel see [Improving the world's most popular functional language: user-defined functions in Excel](#)
- **HTML**
- [HyperText Markup Language](#) is the standard markup language for Web pages — it describes the structure of the content.
- It can contain CSS (for describing appearance) and
- JavaScript (for describing behaviour)
- HTML is not a programming language
- JavaScript is a Turing complete programming language but embedded in a host environment.
- CSS could be extended to be Turing complete — see [Is CSS Turing complete](#)
- **Word**
- [Microsoft Word](#) interface to text formatting
- Serialised with the markup language [Office Open XML](#)
- [Visual Basic for Applications](#) is embedded and is a programming language
- \LaTeX
- [LaTeX](#) is a format of [TeX](#)
- Markup technology for typesetting documents — oriented towards mathematics and technical documents.
- Is also a Turing complete programming language (Unit 7)
- Used in MST125 Essential Mathematics 2 Unit 2 *Mathematical typesetting*
- **SQL**

- [Structured Query Language](#) based on relational algebra and tuple relational calculus
- Syntactic sugar for first order logic (Unit 6)
- Originally not a [Turing complete programming language](#) (Unit 7)
- but extensions are Turing complete
- [Turing completeness](#) is not everything
- Data languages such as [XML](#), [HTML](#), [JSON](#)
- [Regular languages](#) for [regular expressions](#) in your favourite text editor (and some programming languages)
- [Pushdown automata](#) and [Context-free grammars](#) used in program compiling.
- [Total Functional Programming](#) requires all programs to be provably terminating.

5 Python

5.1 Learning Python

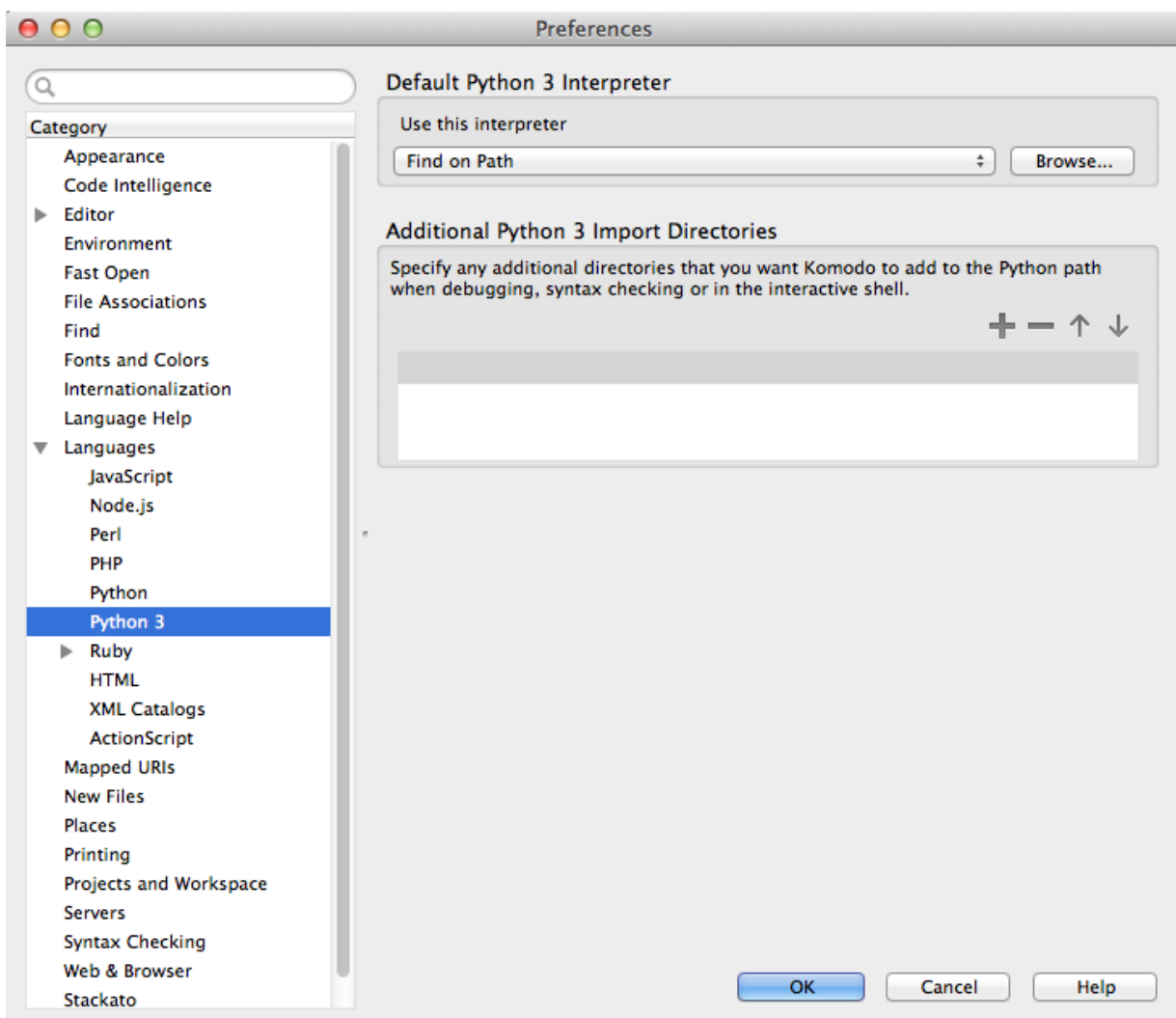
- [Miller & Ranum Problem Solving with Algorithms and Data Structures using Python](#)
- [Python 3 Documentation](#)
- [Python Tutorial](#)
- [Python Language Reference](#)
- [Python Library Reference](#)
- [Hitchhiker's Guide to Python](#)
- [Stackoverflow on Python](#)
- [Dive into Python 3](#)

5.2 Setting up Python with Komodo

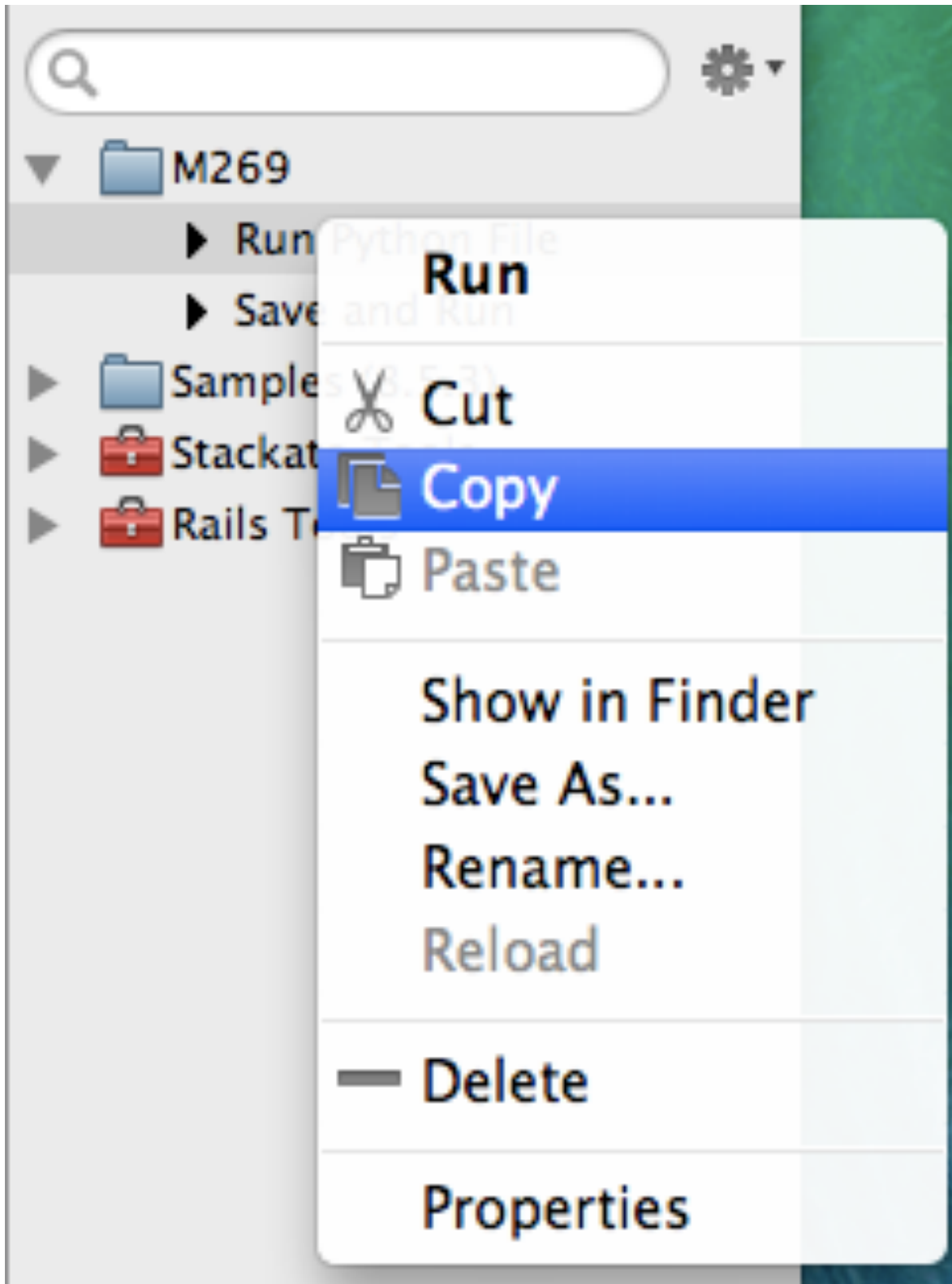
- Install *ActivePython* version 3.x from <http://www.activestate.com/activepython/downloads>
- *Mac OS X* Python 3 is at `/usr/local/bin/python3.3` which is a *symbolic link* to `/Library/Frameworks/Python.framework/Versions/3.3/bin/python3.3`
- *Mac OS X* idle 3 is at `/usr/local/bin/idle3.3` (exact versions will depend on install date)
- *Windows* install location `%SystemDrive%\Python33` and in *Start* menu (if Windows 7)
- Documentation at docs.activestate.com
- *Mac OS X* may need to install correct version of *Tcl/tk* for *IDLE* — <https://www.python.org/download/mac/tcltk>
- Install the *M269 Komodo macros*

- See *M269 Software Installation*
 - Make sure the *Toolbox* and *Command output* tabs are visible — **View** > **Tabs & Sidebars**
 - Right-Click in *Toolbox* and select **Add** > **New Folder...** to create *M269* folder in *Toolbox*
 - Select *M269* folder, right-click and select **Import/Export** > **Import Files from File System** and select both files from the M269 macro download.
- Ensure Komodo is using Python 3
 - **Preferences...** > **Languages Category** > **Python 3** and select your Python 3
 - In the *Toolbox* right-click *Run Python File* and select *Properties*

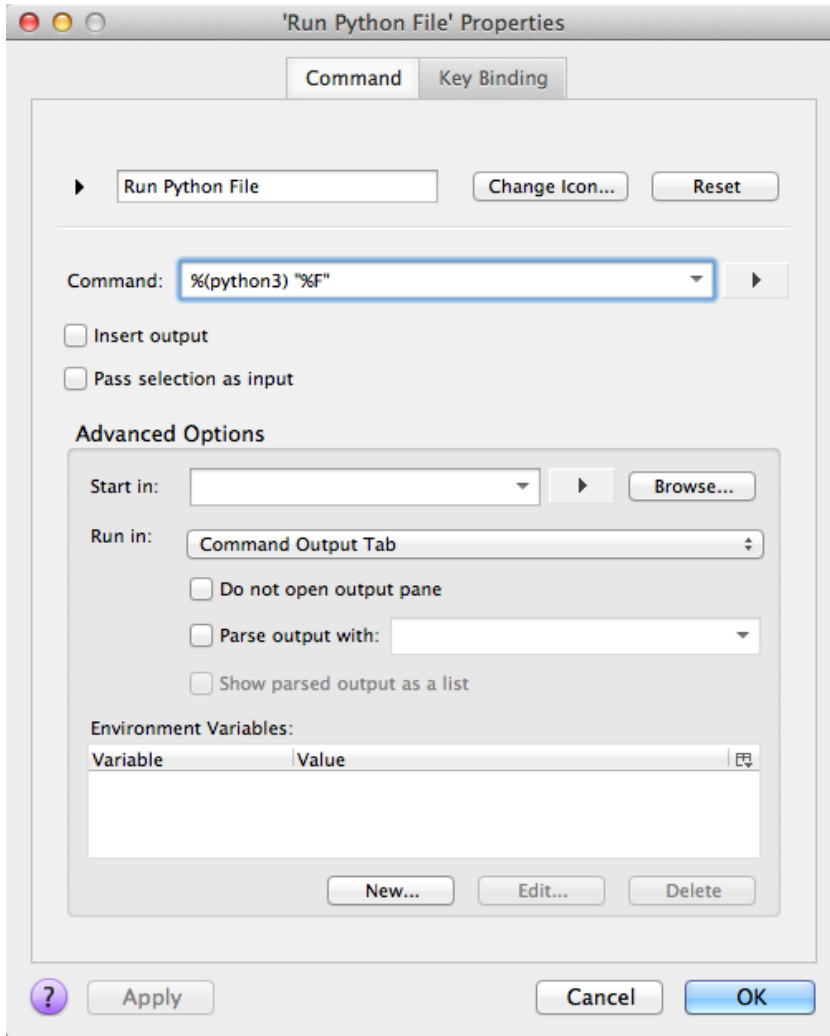
Komodo Preferences: Languages Python 3



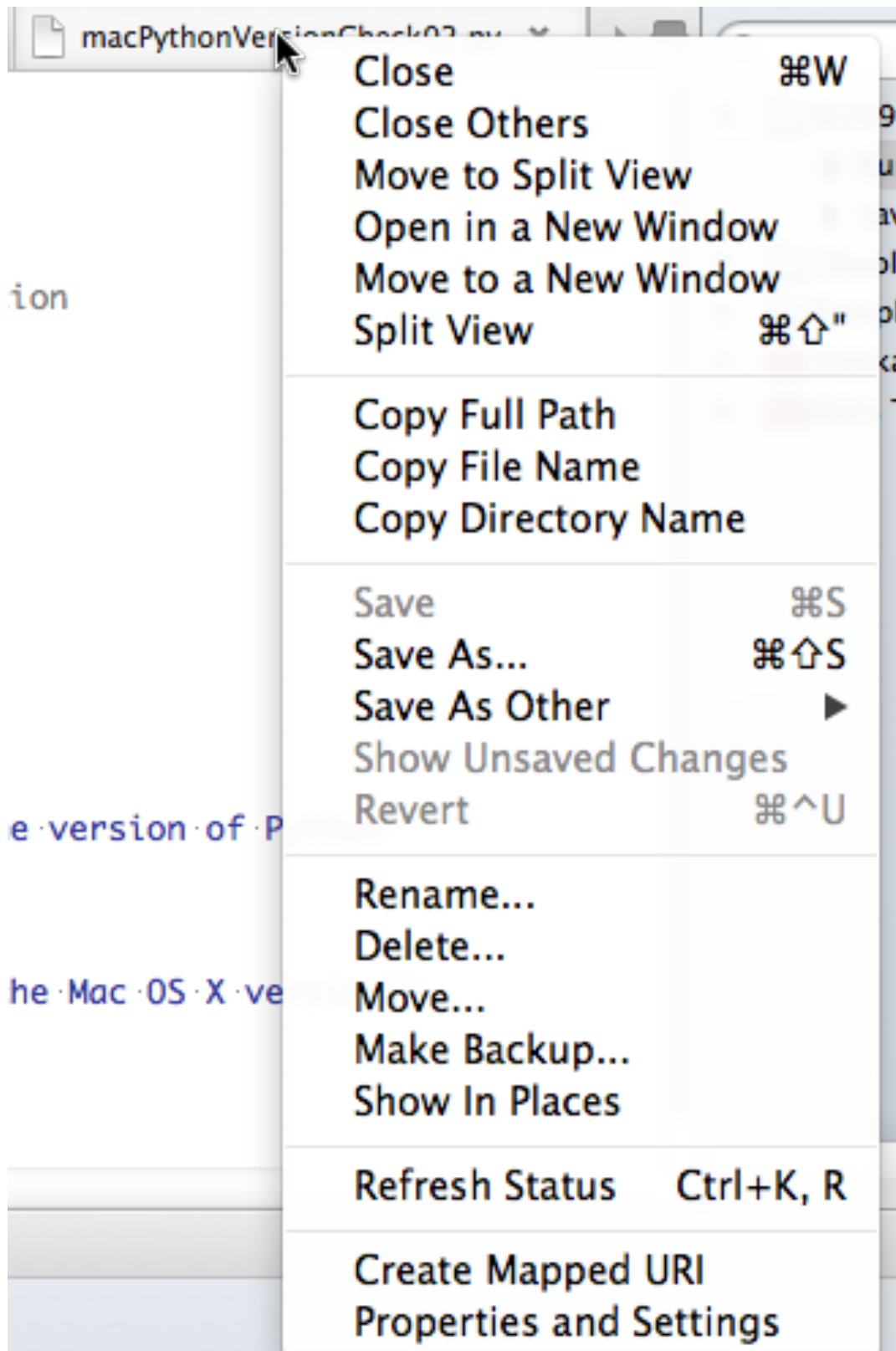
Komodo Run Command Context Menu



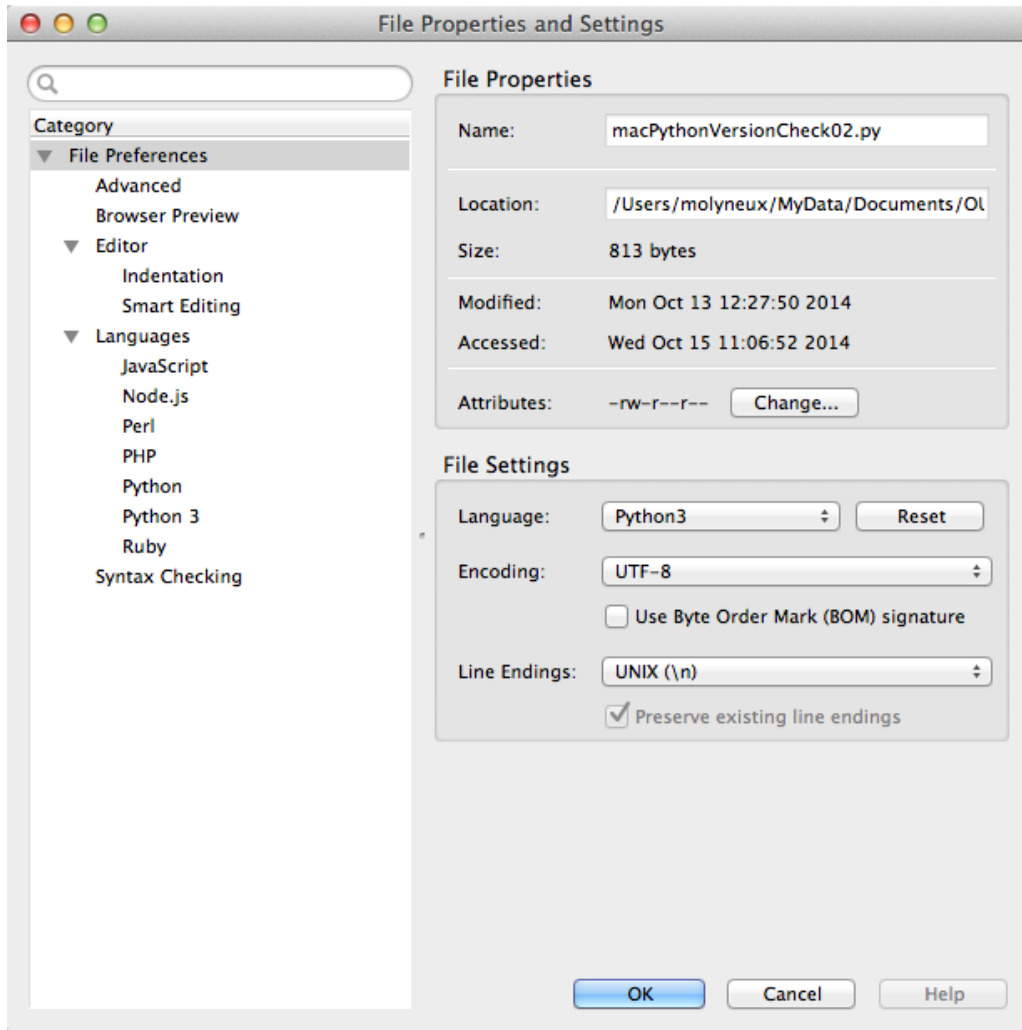
Komodo *Run Python File* Properties



Komodo File Tab Context Menu



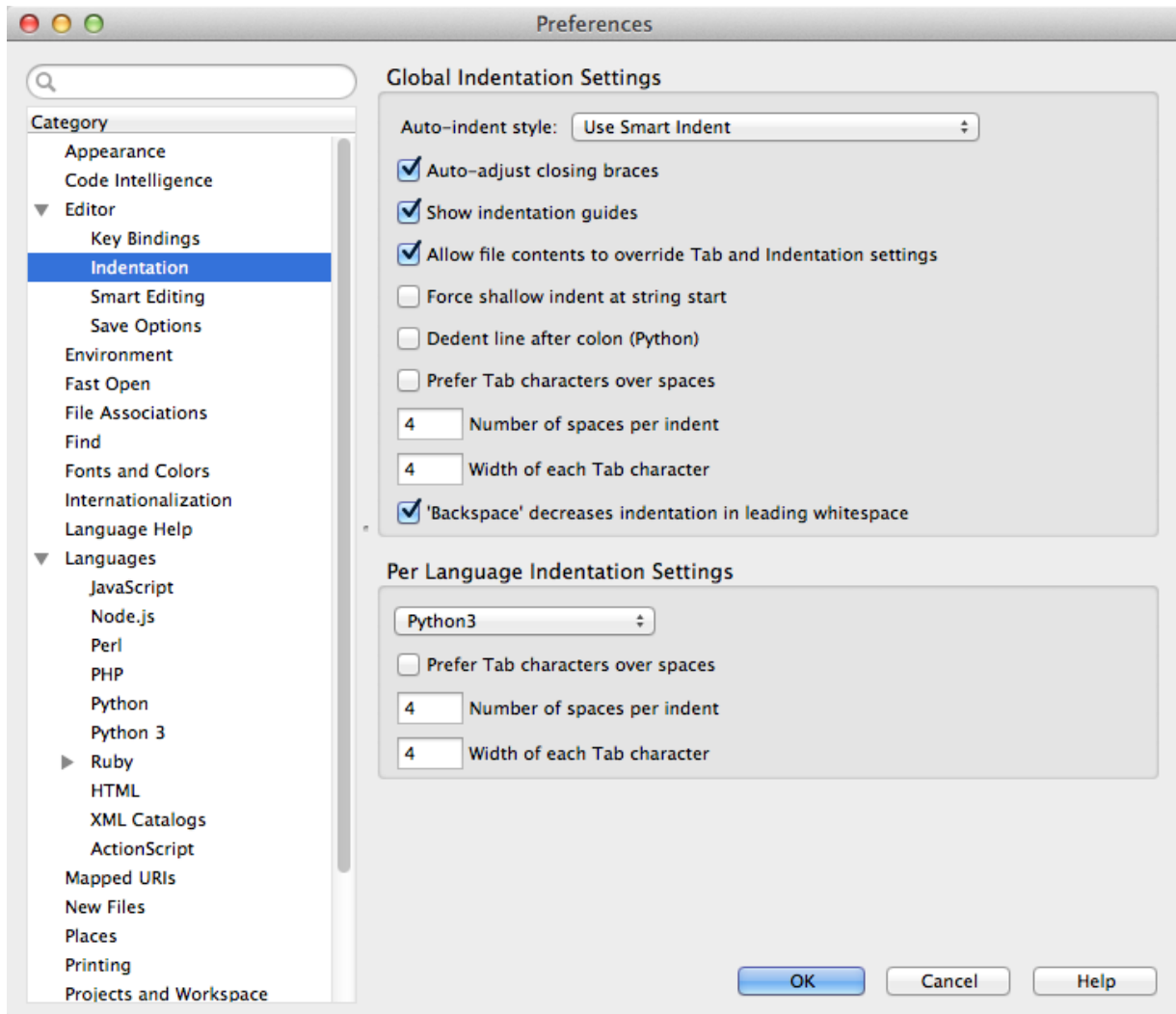
Komodo File Properties and Settings



Indentation and Tabs

- How do you set spaces per indent to 2 or 4 ?
- How do you make the *Tab* key issue spaces ?
- Why is the *Tab* character *evil* ?
- How do you set spaces per indent to 2 or 4 ?
 Preferences... > Editor > Global Indentation Settings (default 8)
- How do you make the *Tab* key issue spaces ?
 Preferences... > Editor > Global Indentation Settings and uncheck *Prefer Tab characters over spaces*
- Why is the *Tab* character *evil* ?
 See [Tabs vs Spaces](#), [Tab key](#)
- See [Python Enhancement Proposals \(PEP 8\)](#) — Style Guide for Python Code
- Mixing tabs and spaces can lead to inconsistent layout when copying from one editor to another or *MS Word*
- Tab character is [Unicode U+0009](#) or `^I` or `HT` or `\t` see [C0 and C1 control codes](#)

Komodo Preferences: Editor Indentation



5.3 Basic Python

Python Usage

- How do you enter an interactive Python shell ?
- How do you exit Python in *Terminal* (Mac) or *Command prompt* (Windows) ?
- How do you get help in a shell ?
- How do you exit the *interactive help utility* ?
- How do you enter an interactive Python shell ?
Windows python3 in Command Prompt; Mac python3 in Terminal; or idle3 in either
- How do you exit Python in *Terminal* (Mac) or *Command prompt* (Windows) ?
`quit()`
- How do you get help in a shell ?
`help()`
- How do you exit the *interactive help utility* ?

quit

Sequences Indexing, Slices

- `xs[i:j:k]` is defined to be the sequence of items from index `i` to `(j-1)` with step `k`.
- If `k` is omitted or `None`, it is treated as `1`.
- If `i` or `j` are negative then they are relative to the end.
- If `i` is omitted or `None` use `0`.
- If `j` is omitted or `None` use `len(xs)`

Python Quiz — Lists

Given the following definitions

```
xs = [10.9,25,"Phil",3.14,42,1985]
ys = [[5]] * 3
```

Evaluate

```
xs[1]
xs[0]
xs[5]
ys
xs[1:3]
xs[::2]
xs[1:-1]
xs[-3]
xs[:]
ys[0].append(4)
```

Python Quiz — Lists — Answers

Given the following definitions

```
xs = [10.9,25,"Phil",3.14,42,1985]
ys = [[5]] * 3
```

Evaluate

```
xs[1]          == 25
xs[0]          == 10.9
xs[5]          == 1985
ys             == [[5],[5],[5]]
xs[1:3]        == [25, 'Phil']
xs[::2]        == [10.9, 'Phil', 42]
xs[1:-1]       == [25, 'Phil', 3.14, 42]
xs[-3]         == 3.14
xs[:]          == [10.9, 25, 'Phil', 3.14, 42, 1985]
ys[0].append(4) == [[5, 4], [5, 4], [5, 4]]
```

5.4 Python Workflows

Komodo Python Workflow

1. Create `someProgram.py` with assignment statements defining variables and other data along with function definitions.

2. There may be auxiliary files with other definitions (for example, *Python Activity 2.2* has *Stack.py* with the *Stack* class definition) — this uses the *import* statement in *someProgram.py*

```
from someOtherDefinitions import someIdentifier
```

3. Load *someProgram.py* into *Komodo Edit* and use the *Run Python File* macro from the *Toolbox*
4. For further results, edit the file in *Komodo Edit* and use the *Save and Run* macro from the *Toolbox*

Standalone Python Workflow

1. Create *someDefinitions.py* with assignment statements defining variables and function definitions.
2. In *Terminal* (Mac) or *Command Prompt* (Windows), navigate to *someDefinitions.py* and invoke the *Python 3* interpreter
3. Load *someDefinitions.py* into the *Python 3* with the command

```
import someDefinitions as sdf
```

The `as sdf` gives a shorter qualifier for the namespace — names in the file are now `sdf.x`

Note that the commands are executed — any print statement will execute, for example

4. At the *Python 3* interpreter prompt, evaluate expressions (remember that they may have side effects and alter the current definitions)

Standalone Python Workflow 2

1. For further results, edit the file in *Your Favourite Editor* and use one of the following commands:

```
reload(sdf)

import imp
imp.reload(sdf)
```

Note the use of the name `sdf` as opposed to the original name.

Read the following references about the dangers of reloading as compared to recycling *Python 3*

- <http://stackoverflow.com/questions/684171/how-to-re-import-an-updated-package-while-in-python-interpreter>
- <http://pyunit.sourceforge.net/notes/reloading.html>
- <http://stackoverflow.com/questions/12400467/how-to-import-and-reimport-file-when-it-needed>

6 Learning Software Packages

Key questions

1. Where is the package source ?
2. What version are you using ?
3. What documentation is available ?
4. What are the *names* for the parts of the interface ?
5. How do you leave the package ? How do you enter the package ?
6. Is there any on-line help and, if so, how is it used ?
7. Are there any initialisation files, configuration or preferences and how are they used ?
8. How do you import and export data from the package ?
9. *When all else fails*, how can you obtain advice ?

6.1 Installing Komodo & Python

M269 Notes

- See *M269 Software Installation Guide* under *Study Resources*
- MS Windows has *PythonWin Shell*
- Mac OS X uses *idle3.3* from *Terminal* — [/usr/local/bin/python3.3](#) and [/usr/local/bin/idle3.3](#) are *symbolic links* to [/Library/Frameworks/Python.framework/Versions/3.3/bin/](#)
- Mac OS X *idle3.3* may require new version of *Tcl/tk* from <http://www.activestate.com/activetcl/downloads> — see <https://www.python.org/download/mac/tcltk> for version required.

6.2 Learning Komodo

Key Questions — Exercise

1. Where do you get *Komodo Edit Help* ?
2. Where does *Help* describe the *Komodo interface* ?
3. How do you globally and permanently disable the *Minimap* ?
4. How do you show *whitespace* and *EOL* characters ? And why would you want to ?
5. How do you show line numbers ?
6. How do you get *syntax colouring* ?
7. How does Komodo detect what language a file has ?
8. How do you comment out a block of code ?
9. How do you set user environment variables ? (and why would you ?)

10. How do you export code with syntax highlighting into *MS Word* ?

11. How do you stop a runaway program ?

Komodo Interface

The screenshot shows the Komodo IDE interface. The main editor window displays a Python script named `macPythonVersionCheck02.py`. The script checks the Python version and the Mac OS X version. The Command Output window at the bottom shows the execution results.

```

1  #!/usr/bin/env python3
2
3  # Python script to check Python version and OS version
4
5  # See near http://docs.python.org/3/index.html for Python 3 documentation
6  # See near http://docs.python.org/2.7/index.html for Python 2.7 documentation
7  # platform is a library in the Generic Operating System Services section
8  # (15 in Python 2.7, 16 in Python 3.3)
9
10 import platform
11
12 # sys is a library in the Python Runtime System Services section
13 # (27 in Python 2.7, 28 in Python 3.3)
14
15 import sys
16
17 newLineChar = "\n"
18
19 sysVersionOutMessage = "The next line is the output of 'sys.version' - the version of Python"
20 print(sysVersionOutMessage)
21 print(sys.version)
22
23 print(newLineChar, "The next line is the output of 'platform.mac_ver()' - the Mac OS X version")
24

```

Command Output

```

The next line is the output of 'sys.version' - the version of Python
3.3.4 (default, Feb 25 2014, 14:59:00)
[GCC 4.2.1 (Apple Inc. build 5664)]

The next line is the output of 'platform.mac_ver()' - the Mac OS X version
('10.9.5', ('', '', ''), 'x86_64')

End of script

```

... M269PythonForumNotes > PythonForumMacVersionCheck20131027 > macPythonVersionCheck02.py UTF-8 Python3 Ln: 9 Col: 1

Key Questions — Answers 1

1. Where do you get *Komodo Edit Help* ?

Help > Help

2. Where does *Help* describe the *Komodo interface* ?

Komodo Edit Help > Contents > Welcome to Komodo > The Komodo Workspace

3. How do you globally and permanently disable the *Minimap* ?

Preferences > Category > Editor > Scrolling > uncheck Use the Minimap Scrollbar

Key Questions — Answers 2

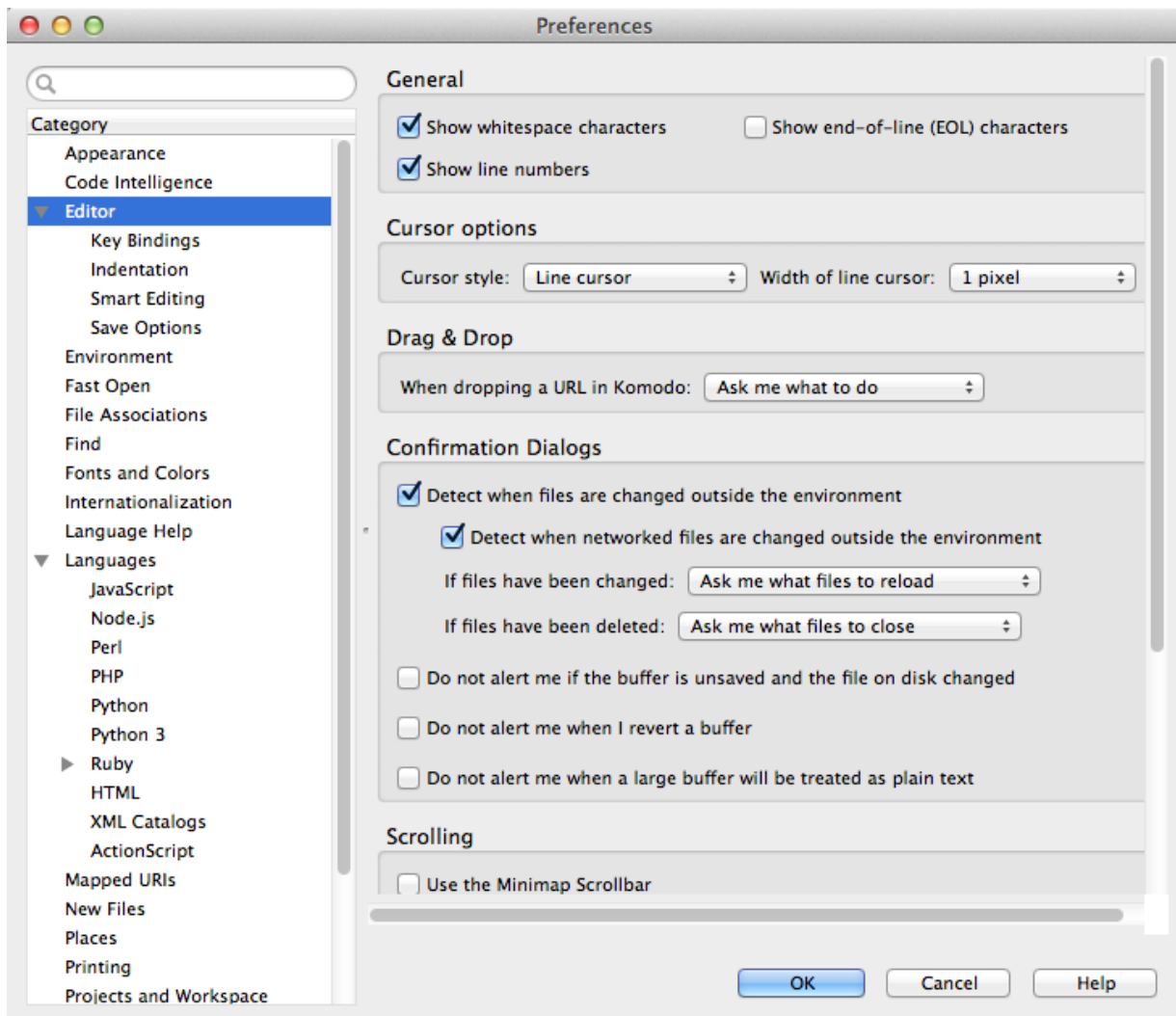
4. How do you show *whitespace* and *EOL* characters ? And why would you want to ?

Preferences > Category > Editor > General > check Show whitespace characters

5. How do you show line numbers ?

Preferences > Category > Editor > General > check Show line numbers

Komodo Preferences: Editor



Key Questions — Answers 3

6. How do you get *syntax colouring* ?

Global: Preferences > Category Fonts and Colors > Language Specific tab

File: File tab context menu > File Properties and Setting dialogue > File Preferences > Advanced > Document-Specific Performance Settings > check Enable Syntax Coloring

7. How does Komodo detect what language a file has ?

Preferences > File Associations > View and Edit Associations

For shebang line see [http://en.wikipedia.org/wiki/Shebang_\(Unix\)](http://en.wikipedia.org/wiki/Shebang_(Unix))

- Note that Komodo can have global or per file settings

Key Questions — Answers 4

8. How do you comment out a block of code ?

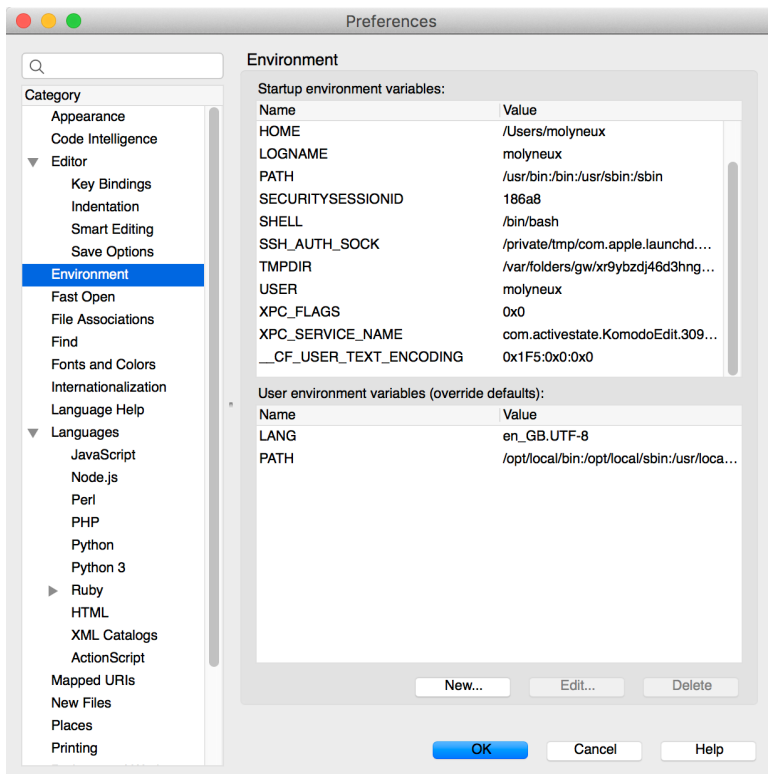
Code > Comment Region (^3)

Code > Un-comment Region (^2)

Key Questions — Answers 5

9. How do you set user environment variables ? (and why would you ?)

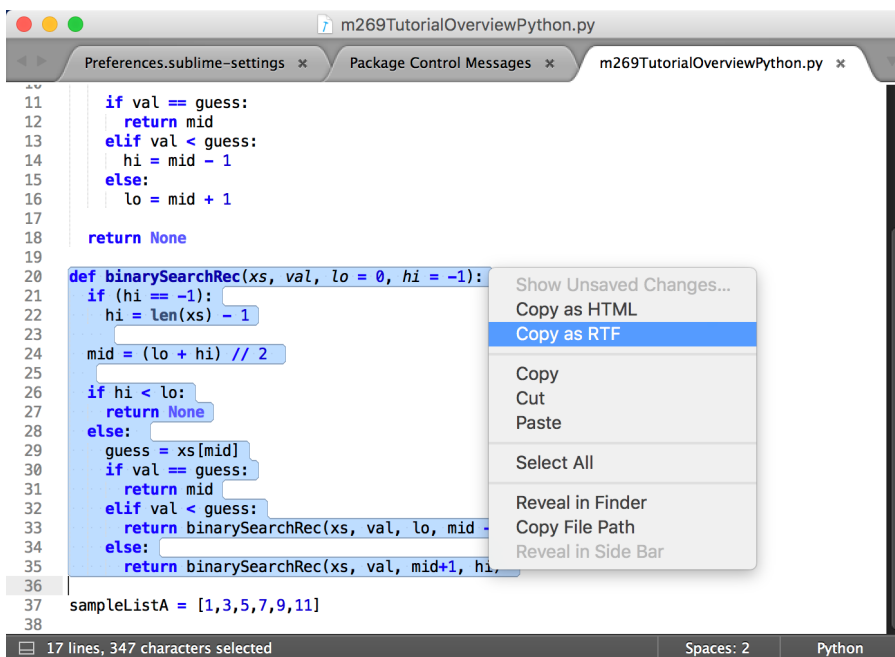
Preferences > Environment



Key Questions — Answers 6

10. How do you export code with syntax highlighting into MS Word ?

Use an editor such as [Sublime Text](#) that has *Copy as RTF*



11. How do you stop a runaway program ?

Bottom pane > Command Output tab > Terminate Process button

Terminate Process button looks like a grey square but its position will vary according to the version of Komodo

7 What Next ?

Programming, Debugging, Psychology

Although programming techniques have improved immensely since the early days, the process of finding and correcting errors in programming — known graphically if inelegantly as *debugging* — still remains a most difficult, confused and unsatisfactory operation. The chief impact of this state of affairs is psychological. Although we are happy to pay lip-service to the adage that to err is human, most of us like to make a small private reservation about our own performance on special occasions when we really try. It is somewhat deflating to be shown publicly and incontrovertibly by a machine that even when we do try, we in fact make just as many mistakes as other people. If your pride cannot recover from this blow, you will never make a programmer.

Christopher Strachey, Scientific American 1966 vol 215 (3) September pp112-124

- To err is human, to really foul things up requires a computer.
- Attributed to [Paul R. Ehrlich](#) in [101 Great Programming Quotes](#)
- Attributed to [Bill Vaughn](#) in [Quote Investigator](#)
- Derived from [Alexander Pope](#) (1711, [An Essay on Criticism](#))
- *To Err is Humane; to Forgive, Divine*
- This also contains
 - A little learning is a dangerous thing;*
 - Drink deep, or taste not the [Pierian Spring](#)*
- In programming, this means you have to *read the fabulous manual (RTFM)*

Overview B and Unit 2

- Overview B (repeat of A) Online 10:00 Sunday 18 October 2020
- Basic Python
- Python Workflows
- Example Algorithm Design
- Writing Programs & Thinking — *The Steps*
- Unit 2 From Problems to Programs
- Some logic
- Preconditions, postconditions
- Abstract Data Types

- Tutorial online (PM) 10:00 Saturday 5 December 2020 — Units 1,2 Abstract Data Types, Logic

8 Web Links & References

- The *offside rule* (using layout to determine the start and end of code blocks) comes originally from Landin (1966) — see https://en.wikipedia.org/wiki/Off-side_rule for other programming languages that use this.
- The *step-by-step* approach to writing programs is described in Glaser et al. (2000)
- The difficulty in learning programming is described in many articles — see, for example, Dehnadi and Bornat (2006)
- UTF-8 (mentioned in the Komodo Environment) is Unicode (or Universal Coded Character Set) Transformation Format — 8-bit — one of the character encodings for the Unicode characters or code points

References

- Cole, Frank N (1903). On the factoring of large numbers. *Bulletin of the American Mathematical Society*, 10(3):134–137.
- Dehnadi, Saeed and Richard Bornat (2006). The camel has two humps. Web (Last checked 22 October 2015). URL <http://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf>.
- Glaser, H; P J Hartel; and P W Garratt (2000). Programming by numbers: a programming method for complete novices. *The Computer Journal*, 43(4):252–265. A functional approach to learning programming.
- Guttag, John V (2016). *Introduction to Computation and Programming Using Python*. MIT Press. ISBN 0262529629. URL <https://mitpress.mit.edu/books/introduction-computation-and-programming-using-python-1>.
- Landin, Peter J. (1966). The next 700 programming languages. *Communications of the Association for Computing Machinery*, 9:157–166.
- Lutz, Mark (2011). *Programming Python*. O'Reilly, fourth edition. ISBN 0596158106. URL <http://learning-python.com/books/about-pp4e.html>.
- Lutz, Mark (2013). *Learning Python*. O'Reilly, fifth edition. ISBN 1449355730. URL <http://learning-python.com/books/about-1p5e.html>.
- Miller, Bradley W. and David L. Ranum (2011). *Problem Solving with Algorithms and Data Structures Using Python*. Franklin, Beedle Associates Inc, second edition. ISBN 1590282574. URL <http://interactivepython.org/courselib/static/pythonds/index.html>.
- Pirnat, Mike (2015). *How to Make Mistakes in Python*. O'Reilly. ISBN 978-1-491-93447-0. URL <http://www.oreilly.com/programming/free/how-to-make-mistakes-in-python.csp>.

- Strachey, Christopher (1966). Systems analysis and programming. *Scientific American*, 215(3):112–124.
- Tollervey, Nicholas H. (2015). *Python in Education*. O'Reilly. ISBN 978-1-491-92462-4. URL <http://www.oreilly.com/programming/free/python-in-education.csp>.
- van Rossum, Guido and Fred Drake (2003a). *An Introduction to Python*. Network Theory Limited. ISBN 0954161769.
- van Rossum, Guido and Fred Drake (2003b). *The Python Language Reference Manual*. Network Theory Limited. ISBN 0954161785.
- van Rossum, Guido and Fred Drake (2011a). *An Introduction to Python*. Network Theory Limited, revised edition. ISBN 1906966133.
- van Rossum, Guido and Fred Drake (2011b). *The Python Language Reference Manual*. Network Theory Limited, revised edition. ISBN 1906966141.
- VanderPlas, Jake (2016). *A Whirlwind Tour of Python*. O'Reilly. ISBN 978-1-491-96465-1. URL <http://www.oreilly.com/programming/free/a-whirlwind-tour-of-python.csp>.
- Wirth, Niklaus (1975). *Algorithms Plus Data Structures Equals Programs*. Prentice Hall. ISBN 0130224189.