GIS Programming (3)

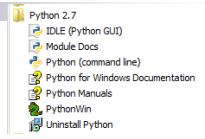
GIS 400/500

Yanli Zhang · Arthur Temple College of Forestry and Agriculture
Stephen F. Austin State University

Outline

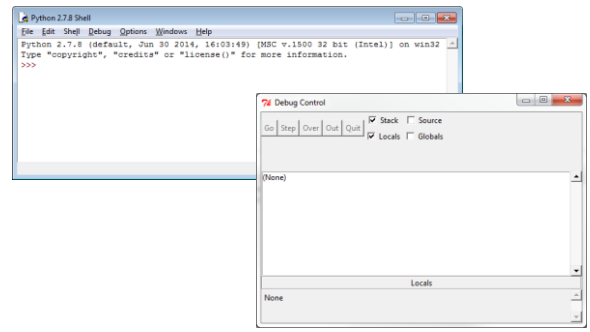
- Python tools
- Python basics

Python tool

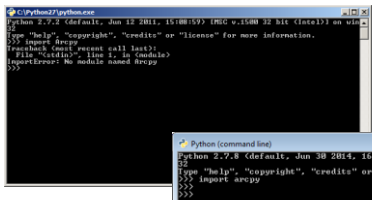


- Python IDLE
  - IDLE stands for Integrated DeveLopment Environment.
  - IDLE is the Python IDE with the tkinter GUI toolkit (wrapper to Tk GUI toolkit)
  - coded in 100% pure Python, using the tkinter GUI toolkit
  - cross-platform: works on Windows and Unix
  - multi-window text editor with multiple undo, Python colorizing and many other features, e.g. smart indent and call tips
  - Python shell window (a.k.a., interactive interpreter)
  - debugger (not complete, but you can set breakpoints, view and step)

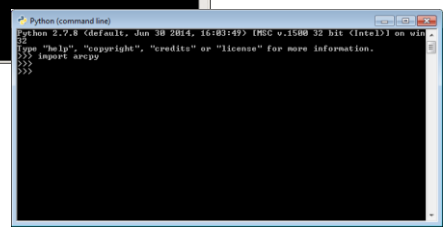
Python IDLE



Python command line



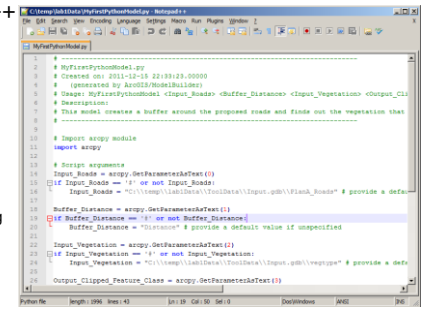
ArcGIS 10.1



ArcGIS 10.3

Python tool

- Text editor
  - Notepad ++
  - jEdit
  - ….

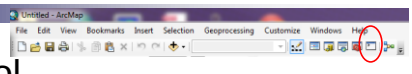


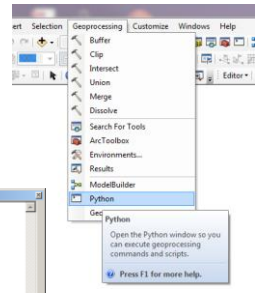
Syntax highlighting

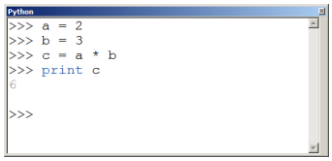
Varies

Python tool

- ArcGIS
  - Python button
  - Geoprocessing
    - Python

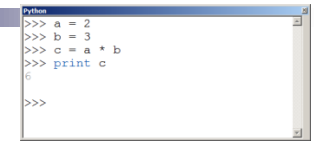






7

Python tool

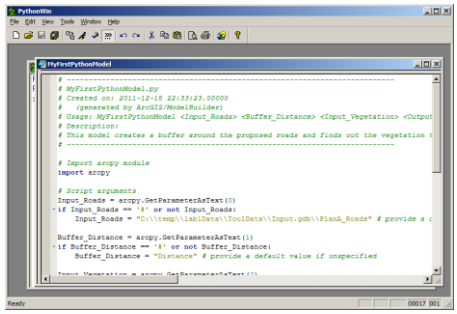


- ArcGIS
  - Embedded interactive Python window in ArcGIS
  - Great for:
    - Testing small blocks of Python code
    - Learning Python
    - Building quick and easy workflows in Python
    - Execution of tools

8

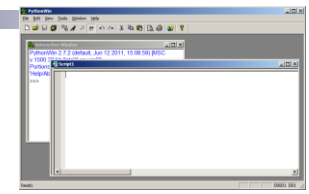
Python tool

- PythonWin



9

Python tool



- PythonWin
  - Visual IDE for creating Python
  - Specifically for Windows environment
  - Script window, interactive window, menus, toolbars, context menus
    - Script window is used to write and save Python code (.py file)
    - Code can be saved, executed, and debugged.

10

IDE

- Different definitions:
  - An integrated development environment (IDE) or interactive development environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs offer Intelligent code completion features.
  - It is a programming environment meshed into a software program. It provides a text or code editor, a GUI builder, a compiler, interpreter and debugger. Programs such as Visual Studio is an example of IDEs.

11

Other Python IDEs

- http://wiki.python.org/moin/IntegratedDevelopmentEnvironments

12

IDEs with introspection-based code completion and integrated debugger

Name	Platform	Updated	Notes
Thonny	Windows, Linux, Mac OS X, more	2017	For teaching/learning programming. Focused on program runtime visualization. Provides stepping both in statements and expressions, no-hassle variables view, separate mode for explaining references etc.
Komodo	Windows/Linux/Mac OS X	2012	Multi-language IDE with support for Python 2.x and Python 3. Available as Komodo IDE (commercial).
LiClipse	Linux/Mac OS X/Windows	2015	Commercial Eclipse-based IDE which provides a standalone bundling PyDev, Workspace Mechanic, Eclipse Color Theme, StartExplorer and AnyEdit, along with lightweight support for other languages, and other usability enhancements (such as multi-caret-edition).
NetBeans	Linux, Mac, Solaris, Windows	2009	Python/Jython support in NetBeans – Open source, allows Python and Jython Editing, code-completion, debugger, refactoring, templates, syntax analysis, etc., see also http://wiki.netbeans.org/Python. **UPDATE**: NetBeans 7.0 released without Python support. Check http://wiki.netbeans.org/Python72Roadmap for upcoming Python support.
PyCharm	Linux/Mac OS X/Windows	2014	Free open-source IDE with a smart Python editor providing quick code navigation, code completion, refactoring, unit testing and debugger. Has a commercial Professional edition that fully supports Web development with Django, Flask, Mako and Web2Py, and allows to develop remotely. Free PyCharm professional licenses for open-source projects.
Python for VS Code	Linux/Mac OS X/Windows	2017	Free open-source extension for Visual Studio Code (now maintained by Microsoft). Supports syntax highlighting, debugging, code completion, code navigation, unit testing, refactoring, with support for Django, multi threaded, local and remote debugging.
KDevelop	Linux/Mac OS X/(Windows)	2014	Free open-source IDE with a focus on static analysis-based code completion, navigation and highlighting. Also features a VI emulation mode.
PyDev	Eclipse	2015	Free, open-source plugin for Eclipse – Allows Python, Jython, and IronPython editing, code-completion, debugger, refactoring, quick navigation, templates, code analysis, unittest integration, Django integration, etc.
Wing IDE	Windows, Linux, Mac OS X	2017	Commercial Python IDE with advanced debugger, editor with vi, emacs, visual studio and other key bindings, auto-completion, auto-editing, multi-selection, snippets, goto-definition, find uses, refactoring, unit testing, remote development, source browser, and much more. There are several product levels, including free and paid versions with a fully functional trial and free licenses for educational use and unpaid open source developers. See product features and pricing for details.
PyScripter	Windows	2012	MIT licensed IDE written in Delphi with debugger, integrated unit testing, source browser, code navigation and syntax coloring/auto-completing editor.
Spyder	Windows/Linux/Mac OS X	2017	Free open-source scientific Python development environment providing MATLAB-like features: console with variable browser, sys.path browser, environment variables browser, integrated plotting features, autocompletion and tooltips - editor with syntax highlighting, class/function browser, pyflakes/pylint code analysis, inline find/replace and search in files features, code completion and tooltips. 100% pure Python, part of Pythonxy (-) distribution (Windows/Linux).
IDLE	Windows/Linux/Mac OS X/All Tk Platforms	2009	Multi-window colorized source browser, autoindent, autocompletion, tool tips, code context panel, search in files, class and path browsers, debugger, executes code in clean separate subprocess with one keystroke. 100% pure Python, part of Python 2.x and 3.x distributions.
IdleX	Windows/Linux/Mac OS X/All Tk Platforms	2016	IdleX is a collection of over twenty extensions and plugins that provide additional functionality to IDLE, a Python IDE provided in the standard library. It transforms IDLE into a more useful tool for academic research and development as well as exploratory programming.

13

IDE	Developer	Latest stable release version	Latest stable release date	Platform	Written in	Widget toolkit	License	Python3x support	Python2x support	Debugger	GUI builder	Integrated toolchain	Profiler	Code coverage	Autocomplete	Static program analysis	GUI based design	Class browser
Anjuta	Naba Kumar	3.22	2016-09-18	Unix-like	C	GTK+	GPL	Unknown	Unknown	Yes	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
Boa Constructor	Riaan Booysen	0.7.2	2014	Unix-like, Windows	Python	wxPython	GPLv2	Yes	No information	Yes	Yes	Unknown	Yes	Unknown	Yes	Yes	Yes	Yes
Visual Studio Code	Microsoft	1.10	2017-11-8	Linux, Microsoft Windows, macOS	JavaScript	HTML	MIT License	Yes	Yes	Yes	No	Unknown	Yes	Unknown	Yes	Unknown	Unknown	Unknown
eric	Detlev Offenbach	17.05	2017-05-06	Linux, Microsoft Windows, macOS	Python	PyQt	GPLv3 "or later"	Yes, until version 4.5.25 and since version 5.5.0[28]	Yes, since version 5.5.0[28]	Yes, for Python 2 & 3	Yes; Qt Creator	Unknown	Yes	Yes	Yes	Multiple integrated checkers and Pylint via plug-in	Yes	Yes
Geany	Team	1.30	2017-03-30	Windows, Linux, macOS, FreeBSD, AIX, OpenBSD, Solaris, other Unix	C	GTK+	GPL	Unknown	Unknown	No	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
IDLE	Guido van Rossum et al.	3.6	2016-12-23	Cross-platform	Python	Tkinter	PSFL	Yes	Yes	Yes	Unknown	Unknown	No	No	Yes	No	Unknown	Yes
Komodo IDE	ActiveState	10.2	2017-02-21	Cross-platform	Unknown	Mozilla platform	Proprietary	Yes	Yes	Yes	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
KDevelop	KDE KDevelop Team	5.0.4	2017-03-04	Cross-platform	C, C++	Qt	GPL	Unknown	Yes	Yes	Unknown	Unknown	Unknown	Unknown	Yes	Unknown	Unknown	Yes
Monkey Studio[29]	Filipe Azevedo	1.9.0.4	2012	Linux	C	Qt4	GPL2	Yes	Unknown	Yes	Yes[29]	Unknown	Unknown	Unknown	Yes	Unknown	Yes	Yes
MonoDevelop	Novell and the Mono community	6.1.2.44	2016-11-11	Windows, Linux, macOS, FreeBSD, OpenBSD, Solaris	C#	GNUI	LGPL	Unknown	Unknown	Yes	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown

http://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments#Python

14

Outline

- Python tool
- Python basics

15

Pet name

Variable

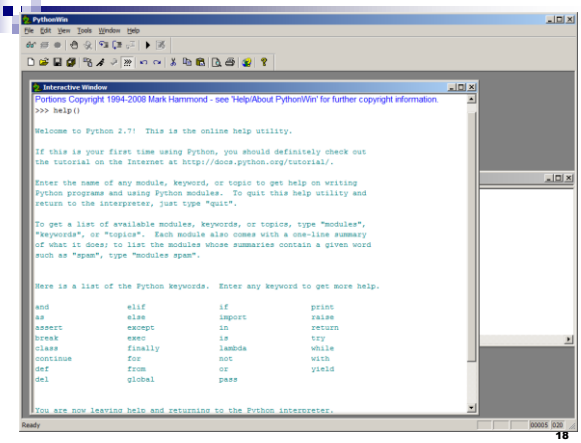
- For label and access values/information
- Every variable has a **name**
  - The name of a variable can contain only numbers, letters, and underscores.
  - The name of a variable cannot start with a number.
    - Start with a lowercase letter (Python tradition)
    - Avoid using an underscore as the first character
    - Try to keep names under 15 characters.
  - No space in the name of a variable
  - The name of a variable must be unique in the area where it is used
  - The name should be meaningful.
    - Such as *finalGrade*

16

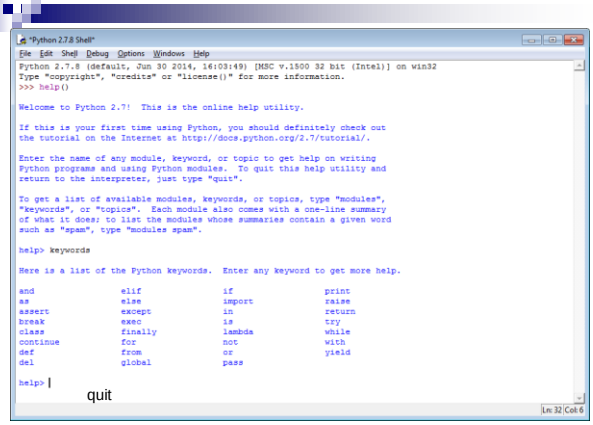
Python keywords (reserved for internal use)

and	as	assert	break
class	continue	def	del
elif	else	except	exec
finally	for	from	global
if	import	in	is
lambda	not	or	pass
print	raise	return	try
while	with	yield	

17



18



Variable declaration




- Direct declaration
- Common data type: int, float, string, and boolean.
  - a = 3
  - a = b = c = 3.0
  - d = a + b
  - e = 'hello'
  - f = True

Variable type

- Variables in Python are dynamically typed
  - Dynamically typed simply means that we do not have to declare the type of the variable.
  - Main difference than Java, VBA or .NET
    - Int intA = 18 (VB.NET)
    - String strName = "Daniel" (VB.NET)

```
>>> name = "GIS"
>>> type(name)
<type 'str'>
>>> name = 15
>>> type(name)
<type 'int'>
```


Math

Numbers

- Plain integers
  - 32 bits of precision
    - Sys.maxint is the maximum plain integer for the current platform
    - -Sysm.maxint - 1
- Long integers
  - Unlimited precision
- Floating point numbers
  - Double in C programming language
  - Sys.float_info
- Complex numbers

Mathematical operation

+	Addition	9 + 2	11
-	Subtraction	9 - 2	7
*	Multiplication	9 * 2	18
/	Division	9 / 2	4
%	Modulus	9 % 2	1
**	exponentiation	9**2	81

```
>>> 9/2
4
>>> 9%2
1
```

Mathematical operation

+	Addition	9.0 + 2	11.0
-	Subtraction	9.0 - 2	7.0
*	Multiplication	9.0 * 2	18.0
/	Division	9.0 / 2	4.5
%	Modulus	9.0 % 2	1.0
**	exponentiation	9.0**2	81.0

```
>>> 9.0+2
11.0
>>> 9.0 -2
7.0
>>> 9.0 * 2
18.0
>>> 9.0/2
4.5
>>> 9.0%2
1.0
```

Note: Starting in Python 3.0, all division will be true division.

25

Mathematical operation

+=	Addition	x += 2	x = x + 2
-=	Subtraction	x -= 2	x = x - 2
*=	Multiplication	x *= 2	x = x * 2
/=	Division	x /= 2	x = x / 2
%=	Modulus	x %= 2	x = x % 2
**=	Exponentiation	x **= 2	x = x ** 2

26

Order of operations

- Parentheses
- Exponentiation
  - 2**2 + 1 = 5
- Multiplication and division are at the same level
- Addition and subtraction are at the same level

27

Demo.

- Assign value
- **Comment**
  - # single line comment
  - " " " Multiline comment.
- print

```
""" this
is
for
gis programming
"""
```

```
>>> width = 20
>>> height = 25
>>> area = width * height
>>> print area   # display the area
500
>>> # I have finished my computing
```

28

Other math operations

abs()	Absolute value
int()	Convert input to integer
long()	Convert input to long integer
float()	Convert input to floating point
math.trunc()	Truncate input to integral
round(x, n)	Round x to n digits, if n is omitted, it defaults to 0)

How many math operations are there?
How do I know all of them?
How are they organized?

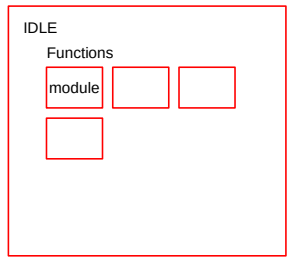
29

Python organization

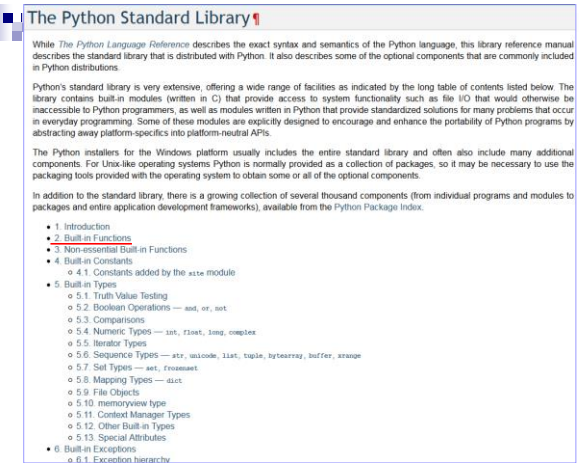
30

Python organization

- IDLE
  - Common functions
  - Modules
    - functions



31


The Python Standard Library ¶

While *The Python Language Reference* describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually includes the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

In addition to the standard library, there is a growing collection of several thousand components (from individual programs and modules to packages and entire application development frameworks), available from the Python Package Index.

- 1. Introduction
- 2. Built-in Functions
- 3. Non-essential Built-in Functions
- 4. Built-in Constants
  - 4.1. Constants added by the site module
- 5. Built-in Types
  - 5.1. Truth Value Testing
  - 5.2. Boolean Operations — and, or, not
  - 5.3. Comparisons
  - 5.4. Numeric Types — int, float, long, complex
  - 5.5. Iterator Types
  - 5.6. Sequence Types — str, unicode, list, tuple, bytearray, buffer, xrange
  - 5.7. Set Types — set, frozenset
  - 5.8. Mapping Types — dict
  - 5.9. File Objects
  - 5.10. memoryview type
  - 5.11. Context Manager Types
  - 5.12. Other Built-in Types
  - 5.13. Special Attributes
- 6. Built-in Exceptions
  - 6.1. Exception hierarchy

Module

- A module is a file containing Python definitions and statements.
  - If think Python as one factory, a module can be thought as one assembly line.
  - If think Python as Lego parts, a module can be thought as one box of Lego parts for a specific toy.
- To use a module, we need to *import* it.

33

List modules

```
>>> help('modules')

Please wait a moment while I gather a list of all available modules...

AddGeometryAttributes Tix                 configHelpSourceEdit platform
AutoComplete          Tkconstants         configSectionNameDialog plistlib
AutoCompleteWindow    Tkdnd               contextlib          popen2
AutoExpand            Tkinter             cookielib           poplib
BaseHTTPServer        ToolTip             copy                posixfile
Bastion               Transposee          copy_reg            posixpath
BatchDefineSR         TreeWidget          csv                 pprint
BatchProject          UndoDelegator       ctypes              profile
BatchPyramids         UserDict            curses              pstats
BatchStats            UserList            datetime            pty
Bindings              UserString          dateutil            py_compile
CGIHTTPServer         Weights             dbhash              pyclbr
CalculateAreas        WeightsUtilities    decimal             pydoc
CalculateDistanceBand WidgetRedirector    difflib               pydoc_data
CalculateEndDate      WindowList          dircache            pyexpat
CallTipWindow         WriteFeaturesToTextFile dis                 pylab
CallTips              ZoomHeight          distutils           pyparsing
Canvas                _LWPCookieJar       doctest             pythonaddins
CentralFeature        _MozillaCookieJar   dumbdbm             quopri
ClassBrowser          __builtin__         dummy_thread        random
CodeContext           __future__          dummy_threading     re
CollectEvents         _abcoll             dynOptionMenuWidget repr
ColorDelegator        _ast                email               rexec
CompressFileGeodatabaseData _bisect               encodings           rfc822
ConfigParser          _bsddb              errno               rlcompleter
ConversionUtils       _codecs             exceptions          robotparser
Cookie                _codecs_cn          filecmp             rpc
```

34

- Math module demo.

```
>>> round(4.5)
5.0
>>> trunc(4.5)

Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    trunc(4.5)
NameError: name 'trunc' is not defined
>>> import math
>>> math.trunc(4.5)
4
>>> round(4.4)
4.0
>>> round(4.565, 2)
4.57
>>> Round(4.56)

Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    Round(4.56)
NameError: name 'Round' is not defined
>>>
```

Python is case sensitive

35

Module

```
# Intro To Python:  Modules
# book.py

"""
Class:  Book( title, author, keywords )
Each book object takes a title, optional author, and optional keywords.
"""
class Book:
    def __init__(self, title, author="Unknown", keywords=[]):
        """
        Books take three arguments to their constructor
        The first and only required argument is the title,
        followed by the optional arguments, author, and a list
        of keywords that can be used to look up a specific book.
        """
        self.title = title
        self.author = author
        self.keywords = keywords

    def setTitle(self, title):
        """
        Takes one argument, the title of the book object.
        """
        self.title = title

    def setAuthor(self, author):
        """
        Takes one argument, the author of the book object.
        """
        self.author = author
```

36

Module

- Built-in modules
  - Go to the interpreter prompt and enter *help()*
  - At the help prompt, enter *modules*

```
>>>help()
help>modules
help>quit
>>> import math
>>>help(math)
>>>dir(math)
>>>help(math.sqrt)
```

37

import math

- "import math" will load math module into current namespace. We need to use dot to track the imported module's methods/functions.
  - Import math
  - math.trunc()

38

from math import trunc

- Also can use "from math import *"
- No dot needed to track the imported module's methods/functions.

```
from math import trunc

print trunc(4.5)
```

39

from math import *

- If many modules are loaded, very possibly there will be conflict of names in the programming project.
- Use "import" is safe, as you always need to refer to the module name to use its methods
- Use "from math import *" for specific module which you used a lot in your programming (save time for not referring)

40

dir()

- dir () is used to find out which names a module defines. It returns a sorted list of strings

```
>>> dir (math)
['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2',
'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1',
'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isinf', 'isnan',
'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh',
'sqrt', 'tan', 'tanh', 'trunc']
>>> help(math.sqrt)
Help on built-in function sqrt in module math:

sqrt(...)
    sqrt(x)

    Return the square root of x.
```

41

String

42

Strings

```
Python
>>> print "I said: 'Let's do it!'"
I said: 'Let's do it!'
```

- Text
- Use single quote or double quote (pairs)
  - >>>print “I said: ‘Let’s do it!’”
- Concatenate two strings together with the + operator
- Strings can be repeated with *
- Escape (The backslash (\) character is used to escape characters that otherwise have a special meaning, such as newline, backslash itself, or the quote character)
  - \\ backslash
  - \’ single quote
  - \” double quote
  - \n new line, moves cursor to beginning of next line

Demo.

```
>>> 'spam eggs'
'spam eggs'
>>> 'doesn\'t'
"doesn't"
>>> "doesn't"
"doesn't"
>>> '"Yes,", he said.'
'"Yes,", he said.'
>>> hello = "We are have a \n
Traceback (  File "<interactive input>", line 1
    hello = "We are have a \n
                           ^
SyntaxError: EOL while scanning string literal
>>> hello = 'We are having a \n\'
... GIS programming'
>>> print hello
We are having a
GIS programming
>>>
```

```
>>> word = 'Help ' + 'me! '
>>> word
'Help me! '
>>> print word * 5
Help me! Help me! Help me! Help me! Help me!
>>>
```

\n\
1. \n means a new line
2. \ means this line of code will continue to next line (only one commend statement)

Demo.

```
>>> print "a", "b"
a b
>>> print "a" "b"
ab
>>> print "a" "\n" "b"
a
b
>>> print "a \n b"
a
 b
>>> print "a \n \
    b"
a
     b
```

```
>>> print "a \n \
b"
a
 b
>>> print "a \n \ b"
a
 \ b
>>> "a" "b"
'ab'
>>> print "a \n \b"
a
 \b
```


String functions

upper()	Returns the uppercase version of the string.
lower()	Returns the lowercase version of the string.
swapcase()	Returns a new string where the case of each letter is switched.
capitalize()	Returns a new string where the first letter is capitalized and the rest are lowercases.
strip()	Returns a string where all the white space (tabs, spaces, and newlines) at the beginning and end is removed.
title	Returns a new string where the first letter of each word is capitalized and all others are lowercase.
len()	Returns the length of a string
str()	Returns a string value of input

Demo.

- String module

```
>>> string.upper('boy')
'BOY'
>>> string.lower('Boy')
'boy'
>>> len('boy')
3
>>> string.swapcase('Boy')
'bOY'
>>> string.capitalize('boY')
'Boy'
>>> str('101')
'101'
>>>
```


Summary

- Python tools
  - Python Shell
  - PythonWin
- Python basics
  - Variable
  - Math
  - Python structure: module
  - String