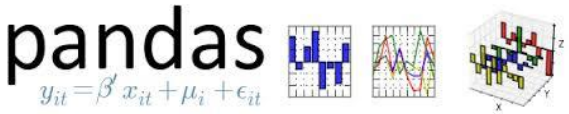
Getting Started with Python

CISA 4358, **Senior Project and Seminar,**

Instructor: Dr. Mohammad Abdel-Rahman

What is Python?

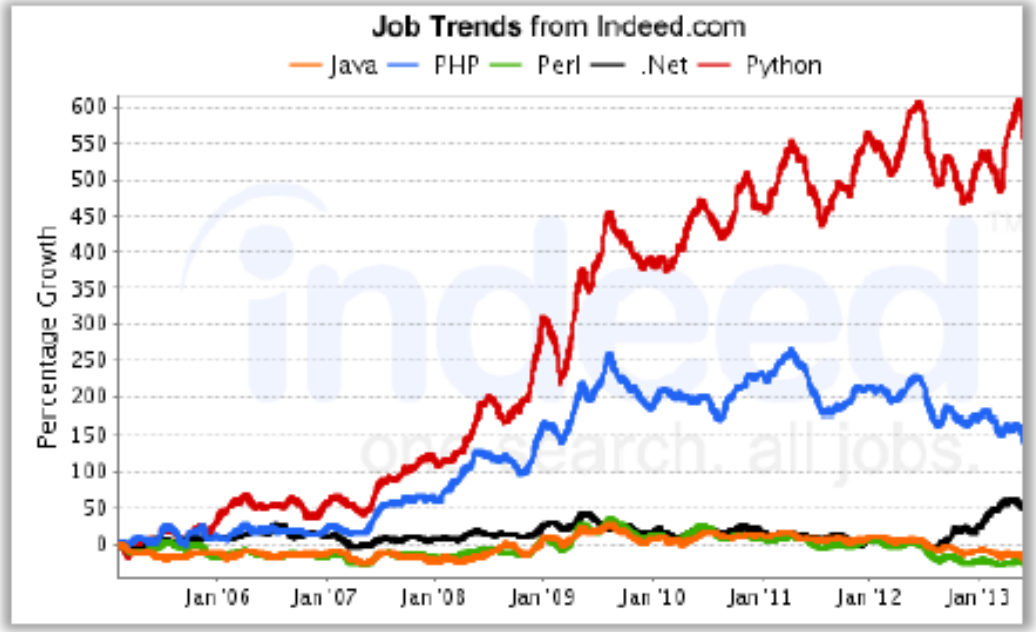
- Python is a general-purpose high level programming language.
- For data analytics:



What is Python

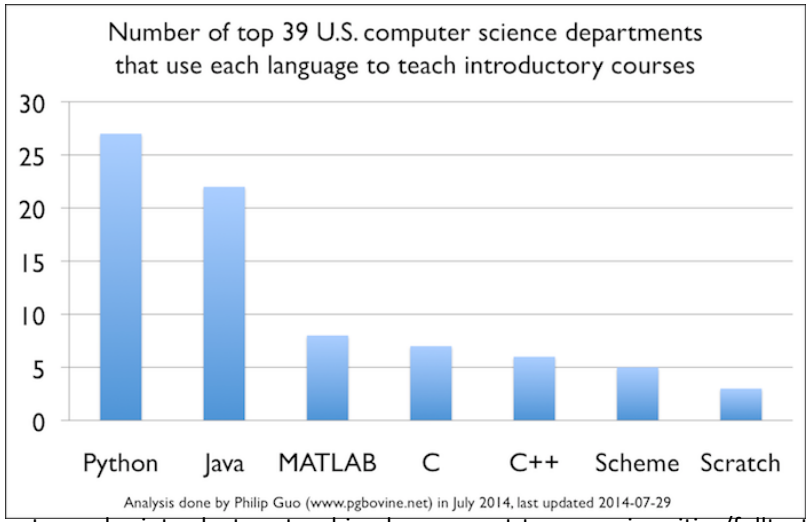
- Invented in the Netherlands in early 90s by Guido van Rossum.
- Named after “Monty Python”, a comedy group, as Python is fun to use.
- Open source and interpreted language.
- Considered a scripting language but is much more than that.
- object-oriented, functional or procedural
- Used by Google, increasingly popular

Python Popularity



Python job trends from indeed.com shows remarkable increase

Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities



http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext

Users of Python

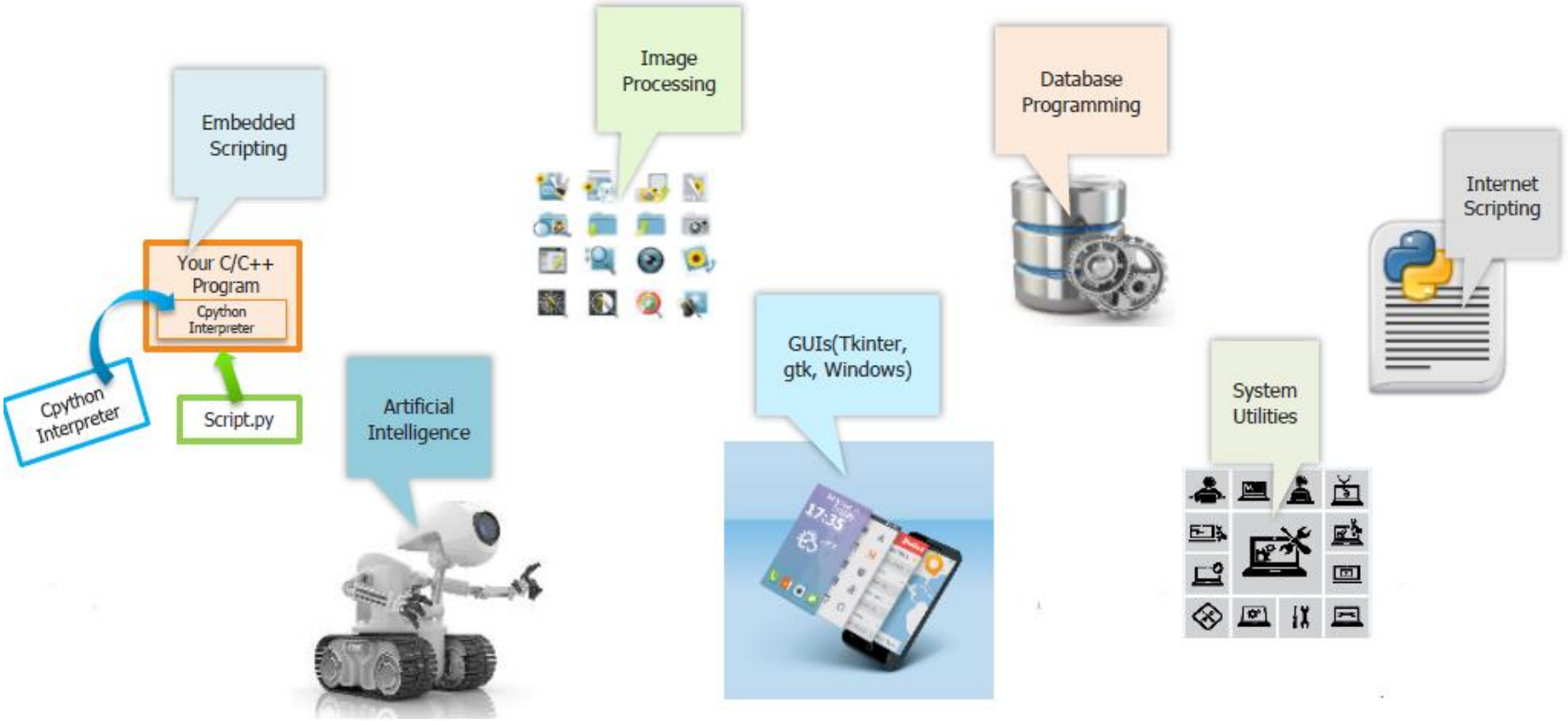
» Youtube is originally written in Python and mysql.

» Dropbox web-based file hosting service is implemented using Python
» Both the Dropbox server (running in the cloud) and desktop client software are primarily written in Python.

» Yahoo acquired Four11, whose address and mapping lookup services were implemented in Python.
» Yahoo Maps uses Python.

» Many components of the Google spider and search engine are written in Python.

Traditional Uses of Python



Uses of Python in Data Analytics

- New use cases of python are the major growth driver for the increasing demand for python skills.
- These Use cases are emerging because of various reasons. One of which is the new packages that were added into the standard library such as Pydoop,Pandas,Scipy etc.
- We will see how to use them in further sessions.

Python Users for Data Analytics

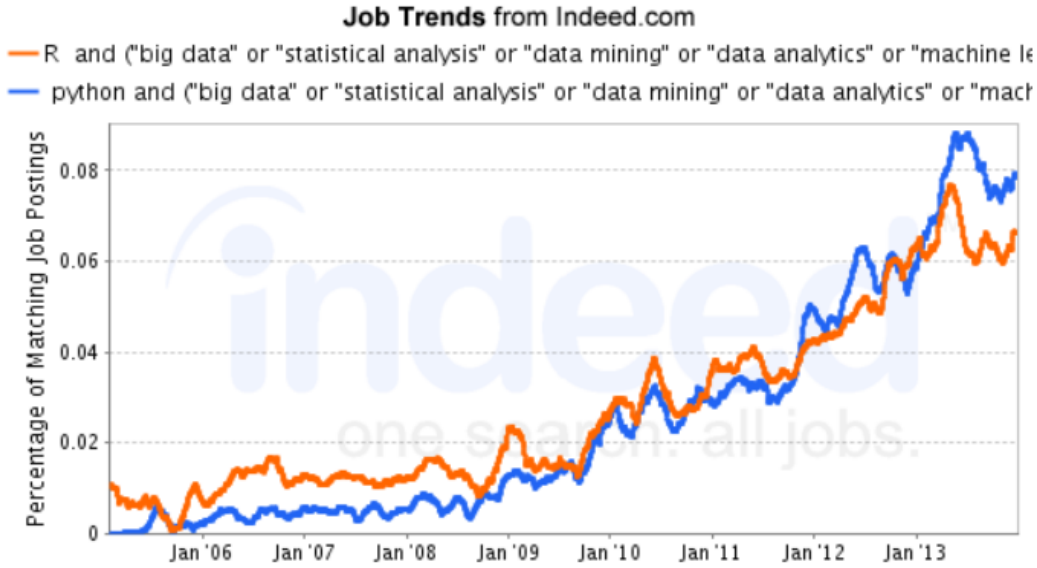
appnexus

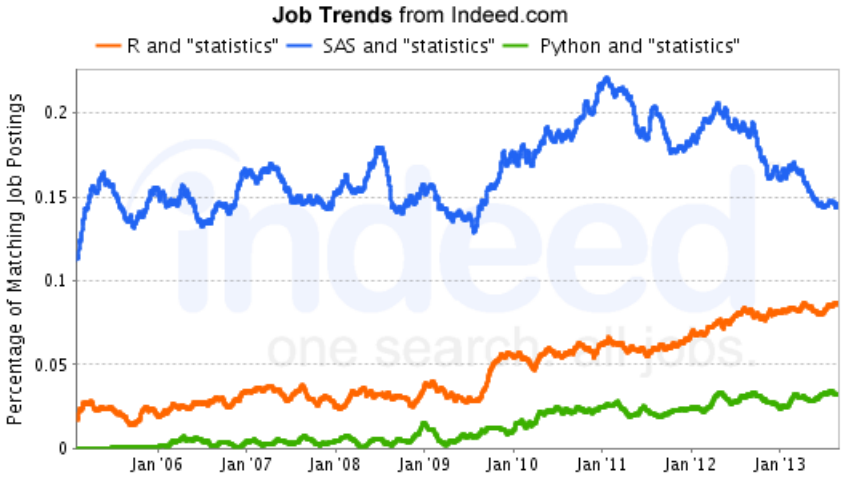
- Used for weather fore casting by http://www.forecastwatch.com/
- AppNexus a real-time online-advertising-platform company uses the Python programming language to help conduct heavy-duty data analysis.

(http://www.bloomberg.com/video/appnexus-at-forefront-of-online-programmatic-advertising-xWT3KqBSTkC5aEkg59MFXA.html)

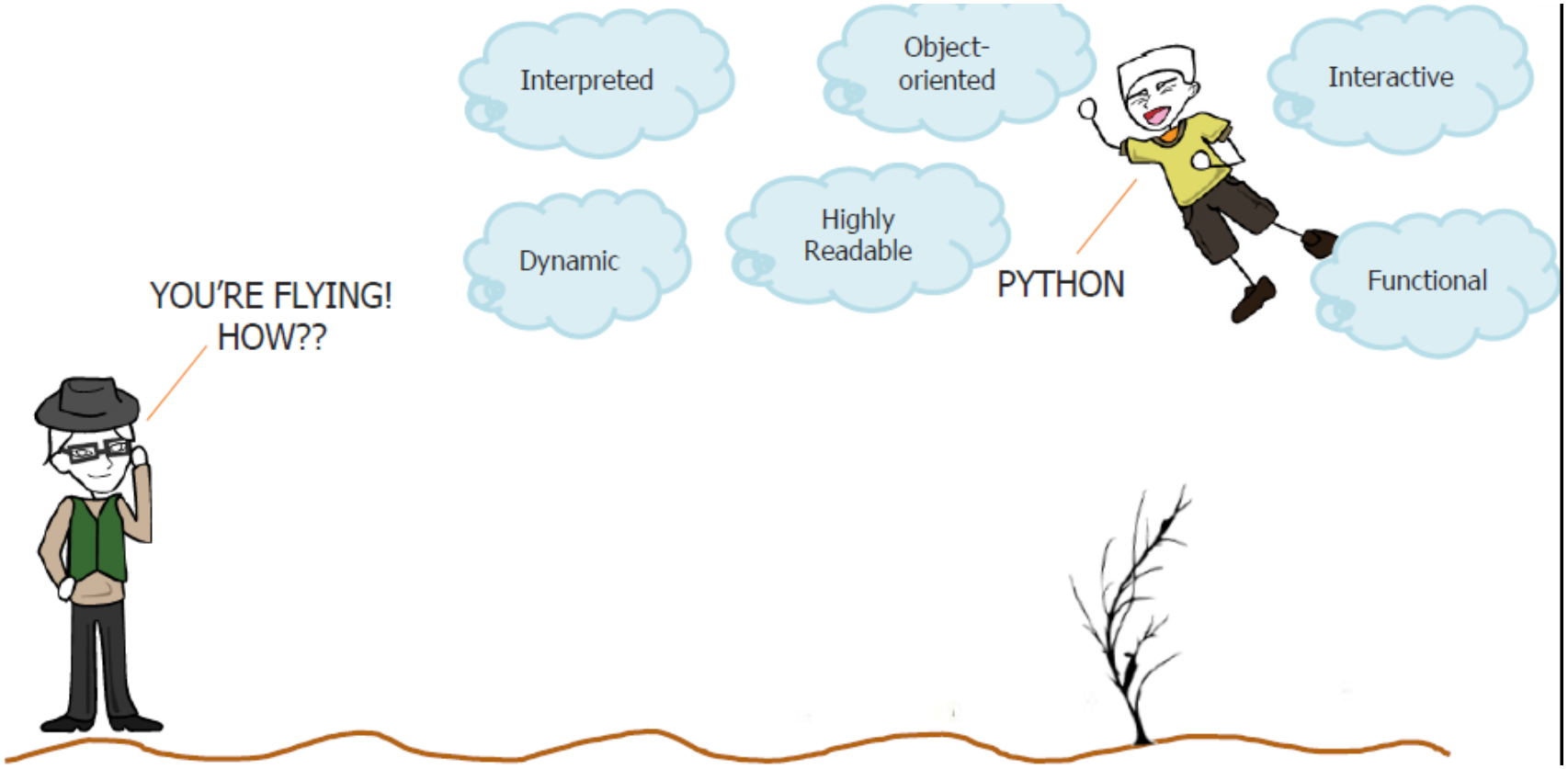
- The company is a leading provider of multi-manager/multi-asset risk management analytics. A complete Risk Management System is written in Python.
- Scientists in the Theoretical Physics Division at Los Alamos National Laboratory are using Python to control large-scale physics codes on massively parallel supercomputers, high-end servers, and clusters. Python plays a central role in controlling these simulations, performing data analysis, and visualization.

Popularity of Python in data analytics





Language features



Interpretive language

- In addition to being a programming language, Python is also an interpreter. The interpreter reads Python programs and commands and executes them.
- Note that Python programs are compiled automatically before being scanned by the interpreter. The scanning process is hidden which makes Python faster than a pure interpreter.

Comment

- Comments: Any text to the right of the # symbol is mainly useful as notes for readers.
- Bulk Comments: Enclose the code in triple quoted strings (“””).

Indentation

- Indentation is a MUST in Python
- There are no braces to indicate blocks of code for class and function definitions or flow control.
- Blocks of code are denoted by line indentations, which is rigidly enforced.
- The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount.
- Standard is to use 4 whitespaces as per official recommendation.
- Some editors automatically takes care of the indentation.

Identifier

A Python Identifier is a name used to identify a variable, function, class, module or other object.

- An identifier starts with a letter A to Z or a to z or an underscore(_) followed by zero or more letters, underscores and digits(0 to 9).
- Python is a case sensitive programming language.

  - Python does not allow special characters such as @,$and % within identifiers.
- Variables are used by just assigning them a value. No declaration or data type definition is needed/used.

Identifier naming convention for Python

- Class names start with an upper case letter and all other identifiers with a lower case letter.
- Starting an identifier with a single leading underscore indicates by convention that the identifier is meant to be private. Starting an identifier with two leading underscores indicates a strongly private identifier.
- If the identifier begins with two leading underscore and ends with two trailing underscores, the identifier is a language-defined special name (a way for the Python system to use names that won't conflict with user names). E.g.,**def __init__**(self, …)

Reserved Keywords

- Reserved Keywords are the reserved words in Python which cannot be used as:
  - »variable name
  - »function name, or
  - »any other identifier
- They are used to define the syntax and structure of the Python language
- All the Python keywords contain lower case letters only

Few are listed below

and	try	from	lambda
if	exec	global	for
import	raise	assert	print
while	finally	pass	break

Data types - Numbers

Python supports several different numeric types:

- Integers
- Examples: 0, 1, 1234, -56

Long integers
Example: 99999999999999999999L
Must end in either l or L
Can be arbitrarily long

Floating point numbers
Examples: 0., 1.0, 1e10, 3.14e-2, 6.99E4

Operations on numbers

- Basic algebraic operations
- Four arithmetic operations: a+b, a-b, a*b, a/b
- Modulo : a%b
- Exponentiation: a**b
- Other elementary functions are not part of standard Python, but included in packages like math, NumPy and SciPy
- Comparison operators

Greater than, less than, etc.: a < b, a > b, a <= b, a >= b

Identity tests: a == b (we use for strings, lists, etc.), a != b

Strings

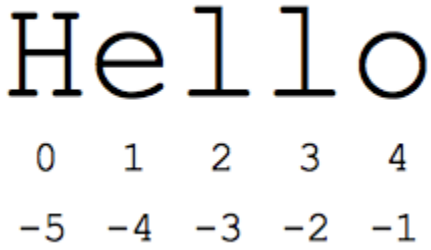
- Python does not support a character type; these are treated as strings of length one, thus also considered a substring.
- Strings are immutable.
- Strings are ordered blocks of text
- Strings are enclosed in single or double quotation marks.
- Examples: 'abc', “ABC”.

String Operations

Concatenation and repetition

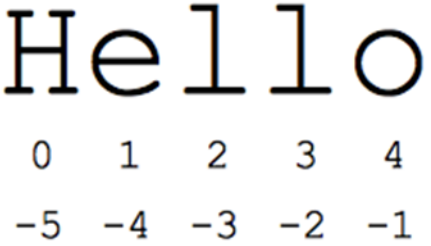
- Strings are concatenated with the + sign:>>> 'abc'+'def‘ 'abcdef'
- Strings are repeated with the * sign:>>> 'abc'*3 'abcabcabc’

Indexing and Slicing Operation



- Python starts indexing at 0.
- A string will have indexes running from 0 to len(s)-1 (where len(s) is the length of s) in integer quantities.
- s[i] fetches the ith element in s.

String slicing



- s[1:4] is 'ell' -- chars starting at index 1 and extending up to but not including index 4
- s[1:] is 'ello' -- omitting either index defaults to the start or end of the string
- s[:] is 'Hello' -- omitting both always gives us a copy of the whole thing (this is the pythonic way to copy a sequence like a string or list)
- s[1:100] is 'ello' -- an index that is too big is truncated down to the string length

String Operations

- Membership Checking

- In -Returns true if a character exists in the given string.

- not in -Returns true if a character does not exist in the given string.

Build-in string methods

- s.capitalize(): Capitalizes first letter of string.
- s.count(str, beg= 0,end=len(string)): Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given.
- s.find(st) -- searches for the given other string (not a regular expression) within s, and returns the first index where it begins or -1 if not found
- s.index(str, beg=0, end=len(string)): Same as find(), but raises an exception if str not found.

More string methods

- s.max(str): Returns the max alphabetical character from the string str.
- s.min(str): Returns the min alphabetical character from the string str.
- **s.replace**(old, new [, max]):Replaces all occurrences of old in string with new or at most max occurrences if max given.
- s.upper():Converts lowercase letters in string to uppercase.
- s.lower()
- s.isalpha()/s.isdigit()/s.isspace()... -- tests if all the string chars are in the various character classes
- **s.strip()** -- returns a string with whitespace removed from the start and end
- **s.startswith**('other'), s.endswith('other') -- tests if the string starts or ends with the given other string
- **s.split('delim')** -- returns a list of substrings separated by the given delimiter. The delimiter is not a regular expression, it's just text. 'aaa,bbb,ccc'.split(',') -> ['aaa', 'bbb', 'ccc']. As a convenient special case s.split() (with no arguments) splits on all whitespace chars.
- s.join(list) -- opposite of split(), joins the elements in the given list together using the string as the delimiter. e.g. '---'.join(['aaa', 'bbb', 'ccc']) -> aaa---bbb---ccc

Control flow

- The if statement is used to check a condition. If the condition is true, we run a block of statements(called the if-block), else we process another block of statements (called the else-block)

  - The else clause is optional

- If-else Statement
- Syntax:

If condition:

    statements…

else:

    statements…

Example:

```
if speed >= 80:
    print (“You are so busted”)
else:
    print (“Have a nice day”)
```

- **Note: Keep a check on indentation and don’t forget the colon(:)**

If-elif-else Statement

- There is no switch statement in Python. You can use an if..elif..else statement to do the same thing
- The elif statement allows you to check multiple expressions for truth value and execute a block of code as soon as one of the conditions evaluates to be true

Example:

```
if speed >= 80:
    print 'License and registration please'
    if mood == 'terrible' or speed >= 100:
      print 'You have the right to remain silent.'
    elif mood == 'bad' or speed >= 90:
      print "I'm going to have to write you a ticket."
      write_ticket()
    else:
      print "Let's try to keep it under 80 ok?"
```

While loop

- The while statement allows you to repeatedly execute a block of statements as long as a condition is true
- Indentation and Colon should be respected
- Example:

```
n = 5
while n != 0:
        if n % 2 == 0:
                print n,
                print " is en even number "
                n-=1
        else:
                pass
        n=n-1
```

We are going to talk about for loop later

Break and continue

- Break and Continue statements are used to exit from a loop
- The break statement is used to breakout of a loop statement i.e. stop the execution of a looping statement, even if the loop condition has not become False or the sequence of items has not been completely iterated over
- The continue statement is used to tell Python to skip the rest of the statements in the current loop block and to continue to the next iteration of the loop

Sequence

- »As the name suggests, a Sequence is a succession of values bound together by a container that reflects their type

- Types of Sequences:
  - »Lists
  - »Tuples
  - »Xrange (range)
  - »String
  - »Dictionaries
  - »Sets

List

- The list type is a container which holds a number of other objects, in a given order.
- The list type implements the sequence protocol, and it also allows you to add and remove objects from the sequence
- A list is an ordered set of elements enclosed in square brackets

Simple definition of list li=[] li = list()

List indexing

- Accessing elements in a list:
- n = len(li)
- item = li[index] #Indexing
- slice = li[start:stop] #Slicing
- list[i]returns the value at index i, where I is an integer
- A negative index accesses elements from the end of the list counting backwards. The last element of any non-empty list is li[-1]
- Python raises an Index Error exception, if the index is outside the list

List manipulation

- Change a value: simple a[0] = value.
- li.append(value)
  - Add an item to the end of the list. Change the list in place
- li.extend(list of values) or li+li2
  - Extend the list by appending all the items in the given list. . Change the list in place
- li.insert(i, value)
  - Insert an item at a given position
- li.remove(value)
  - Removes an item from the list
- li.pop()
  - Removes and returns the last item of the list
  - li.pop(index) – removes and removes the item at index
- li.sort() -- sorts the list in place (does not return it). (The sorted() function shown below is preferred.)
- li.reverse() -- reverses the list in place (does not return it)
- The del statement

List search

- Index() finds the first occurrence of a value in the list and returns the index
- Python raises an exception, if the value is not found in the list
- Use in to test whether a value is present in the list. It returns True if the value is found else it returns False

List accessing using for loop

- To read elements in a list one by one, we usually use “for *elemen*t in *list*:”

```
squares = [1, 4, 9, 16]
sum = 0
for num in squares:
    sum += num
print(sum)
```

If you want to change the elements one by one use “for i in range(len(a))”

```
a = [1, 4, 9, 16]
for i in range(len(a)):
    a[i] += 1
print a
```