Programming with Python

Introduction

- Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.
- It was created by Guido van Rossum during 1985- 1990.
- Python is derived from many other languages

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it.

- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python Features

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax.

- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Databases:** Python provides interfaces to all major commercial databases.

- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

- **Scalable:** Python provides a better structure and support for large programs

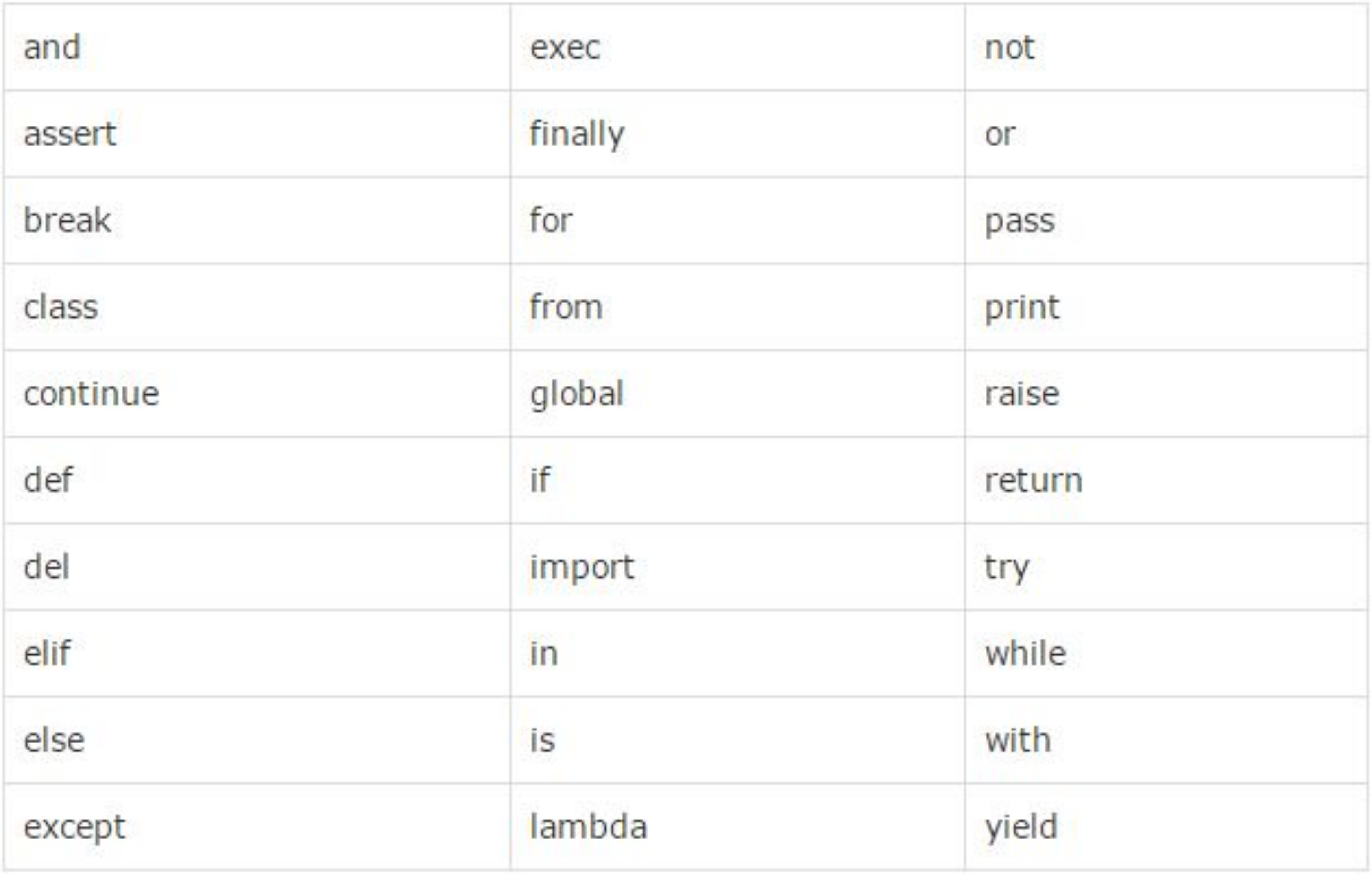
Python Basic Syntax

- print "Hello, Python!" ⍰ **Hello, Python!**

Python Identifiers

- 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 does not allow punctuation characters such as @, $, and % within identifiers.
- Python is a case sensitive programming language.

Reserved Words



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

Lines and Indentation

- Python provides no braces to indicate blocks of code for class and function definitions or flow control.
- Blocks of code are denoted by line indentation, which is rigidly enforced.

```
if True:
        print "True"
else:
        print "False“
```

```
if True:
    print "Answer"
    print "True"
else:
    print "Answer"
    print "False“

if expression :
   suite
```

Multi-Line Statements

- Statements in Python typically end with a new line.
- Python does, however, allow the use of the line **continuation character (\)** to denote that the line should continue.

```
total = item_one + \
        item_two + \
        item_three
```

Quotation in Python

- Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

- word = 'word'

- sentence = "This is a sentence."

- paragraph = """This is a paragraph. It is made up of multiple lines and sentences."""

Comments in Python

- A hash sign (#) that is not inside a string literal begins a comment.

- # First comment
- print "Hello, Python!" # second comment

Python Variable Types

- Variables are nothing but reserved memory locations to store values.
- This means that when you create a variable you reserve some space in memory.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.
- **Python variables do not need explicit declaration to reserve memory space.**
- **The declaration happens automatically when you assign a value to a variable.**
- **The equal sign (=) is used to assign values to variables.**

- counter = 100   # An integer assignment
- miles   = 1000.0   # A floating point
- name   = "John"   # A string

- ***print counter***
- ***print miles***
- ***print name***

- **Multiple Assignment a = b = c = 1**

- **a,b,c = 1,2,"john"**

Standard Data Types

- **Python has five standard data types –**

- Numbers
- String
- List
- Tuple
- Dictionary

- **Python Numbers**
- Number data types store numeric values
- Var1=0
- Var2=10

- **Python Strings**

- str = 'Hello World!'

- print str   # Prints complete string
- print str[0]   # Prints first character of the string
- print str[2:5]   # Prints characters starting from 3rd to 5th
- print str[2:]   # Prints string starting from 3rd character
- print str * 2   # Prints string two times
- print str + "TEST" # Prints concatenated string

Python Lists

- list = ['abcd', 786 , 2.23, 'john', 70.2]
- tinylist = [123, 'john']

- print list # Prints complete list
- print list[0] # Prints first element of the list
- print list[1:3] # Prints elements starting from 2nd till 3rd
- print list[2:] # Prints elements starting from 3rd element
- print tinylist * 2 # Prints list two times
- print list + tinylist # Prints concatenated lists

Python Tuples

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas.

- **The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.**

- tuple = ('abcd', 786 , 2.23, 'john', 70.2)
- tinytuple = (123, 'john')

- print tuple   # Prints complete list
- print tuple[0]   # Prints first element of the list
- print tuple[1:3]   # Prints elements starting from 2nd till 3rd
- print tuple[2:]   # Prints elements starting from 3rd element
- print tinytuple * 2   # Prints list two times
- print tuple + tinytuple # Prints concatenated lists

Python Dictionary

- They work like associative arrays and consist of key-value pairs.

- dict = {}
- dict['one'] = "This is one"
- dict[2]   = "This is two"

- tinydict = {'name': 'john','code':6734, 'dept': 'sales'}

- print dict['one']   # Prints value for 'one' key
- print dict[2]   # Prints value for 2 key
- print tinydict   # Prints complete dictionary
- print tinydict.keys()   # Prints all the keys
- print tinydict.values() # Prints all the values

- This is one
- This is two
- {'dept': 'sales', 'code': 6734, 'name': 'john'}
- ['dept', 'code', 'name']
- ['sales', 6734, 'john']