Introduction to Python

Mubashir Adnan Qureshi

What is Python?

- Designed by Guido van Rossum, in 1990s
- Dynamic, interpreted language
  - does not declare types of variables or parameters
  - short and flexible code
  - no compile-time type checking
- Good for fast prototyping
Python Interpreter

- Installed on every cs machine

```python
import python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.4.5] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> l = [i*i for i in range(15)]
>>> l
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196]
>>> sum(l)
1015
```
- ctrl-D to exit or use exit()

Python Scripts

- Python scripts suffix with .py
- In general,
  - `$python ./helloworld.py`
- Create executable scripts
  - in the first line of the script, add
    ```python
    #!/usr/bin/python
    ```
  - make the file executable, type in the terminal
    ```bash
    chmod a+x helloworld.py
    ```
  - type the name of the script to execute
    ```bash
    ./helloworld.py
    ```
**Syntax**

- Much of Python syntax is similar to C
- Missing operators: `++`, `--`
- Code blocks denoted by line indentation
  - class and function definitions, control flow
  - same amount of indentation within the same block

```python
if True:
    print 'true'
print 'answer'
else:
    print 'false'
```

- Hash sign (`#`) begins a comment

**Built-in Types**

- Numerical types
  - `integer`, `float`, `complex`
  - bitwise operations on integer types are the same as in C
- Sequence types
  - `string`, `list`, `tuple`, `bytearray`, `xrange`, `buffer`, ...
- Mapping type - dictionary
  - `dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'};
  - `dict['Name'] = ?`, `dict['Age'] = ?`
### String Methods

- **Concatenation**
  - “hello” + “world” → “helloworld”

- **Repetition**
  - “hello”*3 → “hellohellohello”

- **Length**
  - len(“hello”) → 5

- **Indexing**
  - “hello”[2], “hello”[-1]

- **Slice**
  - “hello”[1:3], “hello”[-2:]
    - “hello”[:], “hello”[1:10]

---

### String Methods (Cont.)

```python
defs = "hello+world"
```

- `s.find(“world”)` → 6
- `s.split(“+”)` → [“hello”, “world”]
- “+”.join([“hello”, “world”]) → “hello+world”
- “lo” in s (”lo” not in s) → True
- `s.upper()` (s.lower()) → “HELLO+WORLD”
- `str(3.14)` → “3.14”
List Methods

- A compound data type
  - [1,"hello", True, 3.2]

- Same operators as for strings
  - a+b, a*3, a[-1], a[1:], len(a)

```python
items = [1,"hello", 9.2, True]
```

- Append an element
  - items.append("world")

```python
items = [1,"hello", 9.2, True]
items.append("world")
```

- Extend the list
  - items.extend(["world"])

```python
items = [1,"hello", 9.2, True]
items.extend(["world"])""
```

List Methods (Cont.)

```python
items = [1,"hello", 9.2, True]
```

- Insert an element
  - items.insert(2, "world")

```python
items = [1,"hello", 9.2, True]
items.insert(2, "world")""
```

- Remove an element
  - items.remove("hello")
  - items.pop()

```python
items = [1,"hello", 9.2, True]
items.remove("hello")
items.pop()
```

- Reverse the order of the list
  - items.reverse()

```python
items = [1,"hello", 9.2, True]
items.reverse()
```

- Generate a list
  - range(5)
  - [i*i for i in range(5)]

```python
range(5)
[i*i for i in range(5)]
```

```python
[0,1,2,3,4]
[0,1,4,9,16]
```
Control Flow – if/elif/else

if a == 0:
    print "zero!"
elif a < 0:
    print "negative!"
else:
    print "positive!"

- blocks identified by indentation
- colon (:) used at end of lines containing control flow keywords

Control Flow – for loop

for x in list:
    do something...

Example:
a = [3, 1, 4, 1, 5, 9]
for x in a:
    print x
Functions

- Defining functions
  - begins with keyword `def`, function name, and parentheses
  - parameters are placed within parentheses
  - code block starts with colon and indented

```
def inc(x):
    y = x + 1
    return y
```

- Calling functions
  ```
  print inc(3)
  ```

- Functions must be defined before they are called

Modules

- files containing Python definitions and statements
  - code reuse, easier maintenance

```
from socket import *
socket(AF_INET, SOCK_STREAM)
import random
random.randint(1, 10)
```
Packing Datagrams

- **struct module**
  - `from struct import *`
- A message has two fields. Field 1 has 2 bytes, field 2 has 4 bytes
  - **sender side**
    ```python
    >>> s = pack('!HL', 12, 100000)
    >>> s
    '\x00\x0c\x00\x01\x86\xa0'
    ```
  - **receiver side**
    ```python
    >>> unpack('!HL', s)
    (12, 100000)
    ```
- Details in Python documentation

When You Need Help...

- **Google search**
  - 'python list', 'python string uppercase', ...
- **Official Python docs site** - docs.python.org
- Many questions (answers) can be found on StackOverflow
- Use `help()` inside the Python interpreter
  - `help(len), help(list), ...`
Example: TCP client in Python

```python
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET,
                      SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

Example: TCP server in Python

```python
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,
                      SOCK_STREAM)
serverSocket.bind(('','serverPort'))
serverSocket.listen(1)
print 'The server is ready to receive'
while 1:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024)
capitalizedSentence = sentence.upper()
connectionSocket.send(capitalizedSentence)
connectionSocket.close()
```
Example: UDP client in Python

```python
from socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(AF_INET,
    SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message,(serverName,
    serverPort))
modifiedMessage, serverAddress =
    clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()
```

Example: UDP server in Python

```python
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,
    SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "The server is ready to receive"
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```