EPROM
ENTREPRENEURIAL PROGRAMMING AND RESEARCH ON MOBILES

http://mit.edu/eprom

Mobile Application Prototyping with Python

A 3-Day Crash Course for the University of Nairobi

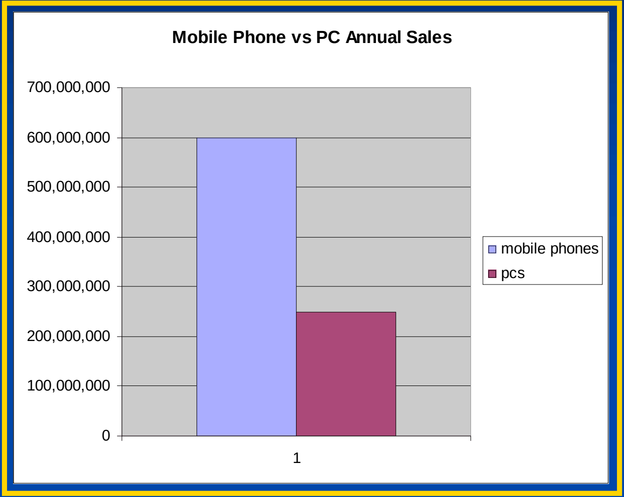
DAY 1

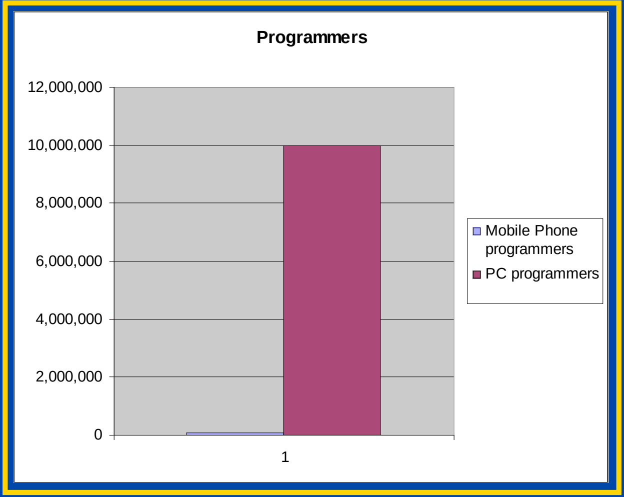
Nathan Eagle, PhD
Research Scientist
MIT Design Laboratory
Massachusetts Institute of Technology

November 16-18 2006
SCI, University of Nairobi

Why Program Phones?

- Mobile phones are computers that are carried by over 1,000,000,000 people around the world.
- They are no longer single use devices but rather can be harnessed to provide a variety of functionalities.
- You'll be part of the first mobile phone application developers in the world.





Stats for bottom graph made up, but probably pretty accurate

EPROM ENTREPRENEURIAL PROGRAMMING AND RESEARCH ON MOBILES

http://mit.edu/eprom

Programmable Phones: Hardware

Data from: "Python for Series 60", Jukka Laurila, EuroPython 2005.

- 100-220+ MHz ARM processor (and a separate processor for telephony functions)
- Typically 4 – 8 MB of free RAM
- FAT formatted Flash as mass storage
- GSM, GPRS, UMTS, Bluetooth, IrDA (although getting phased out), WLAN
- Display: 176x208, LCD panel or sometimes touch-screen
- Integrated camera
- 20 million shipped by Nokia alone
- In other words: A pretty capable computer
  - ...with an always-on Internet connection
  - ... carried with millions of people, all the time
  - ...that you can write own software for
  - ...IF you spend enough effort

- And the numbers are just going up: faster processors, more memory, and lots more available devices.

Programmable Phones: Operating Systems

- Symbian
  - Many platform dependent flavors (UIQ, Series 40, 60, 80, …)
  - 'Independent' company – but partially owned by Nokia
- Microsoft PocketPC / Smartphone
  - Aggressive Marketing, Vendor Connections, and $$
- Linux
  - rare and 'invisible'
  - Motorola, maybe Nokia in the future?

Programmable Phones: Languages

- Java (MIDP)
  - Major Sandboxing
  - Functionality growing with MIDP2, but still not quite there
    - ie: bluetooth (!), but no phone control, no cell tower information, etc.
- C++ (Symbian)
  - Very steep learning curve
  - Frustrating features
  - Designed for 'serious' developers

and now…

- **PYTHON!**

So What's So Special About Python?

- Cross Platform
- Open Source
- Successful (Google, NASA, etc)
- Scripting Language
- Extending and embedding abilities
- Good standard library
- Reasonable memory footprint
- Access to full phone functionality…

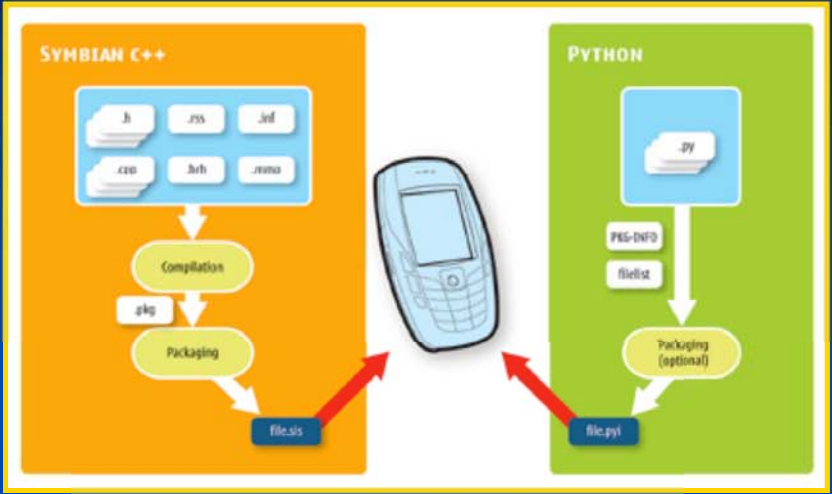
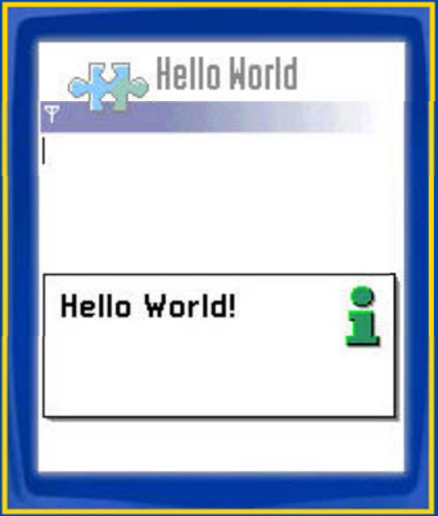


Image from: Michele Marchetti, "Python brings application ideas to life" NRC 2004

```
import appuifw
appuifw.note(u'Hello World!')
```

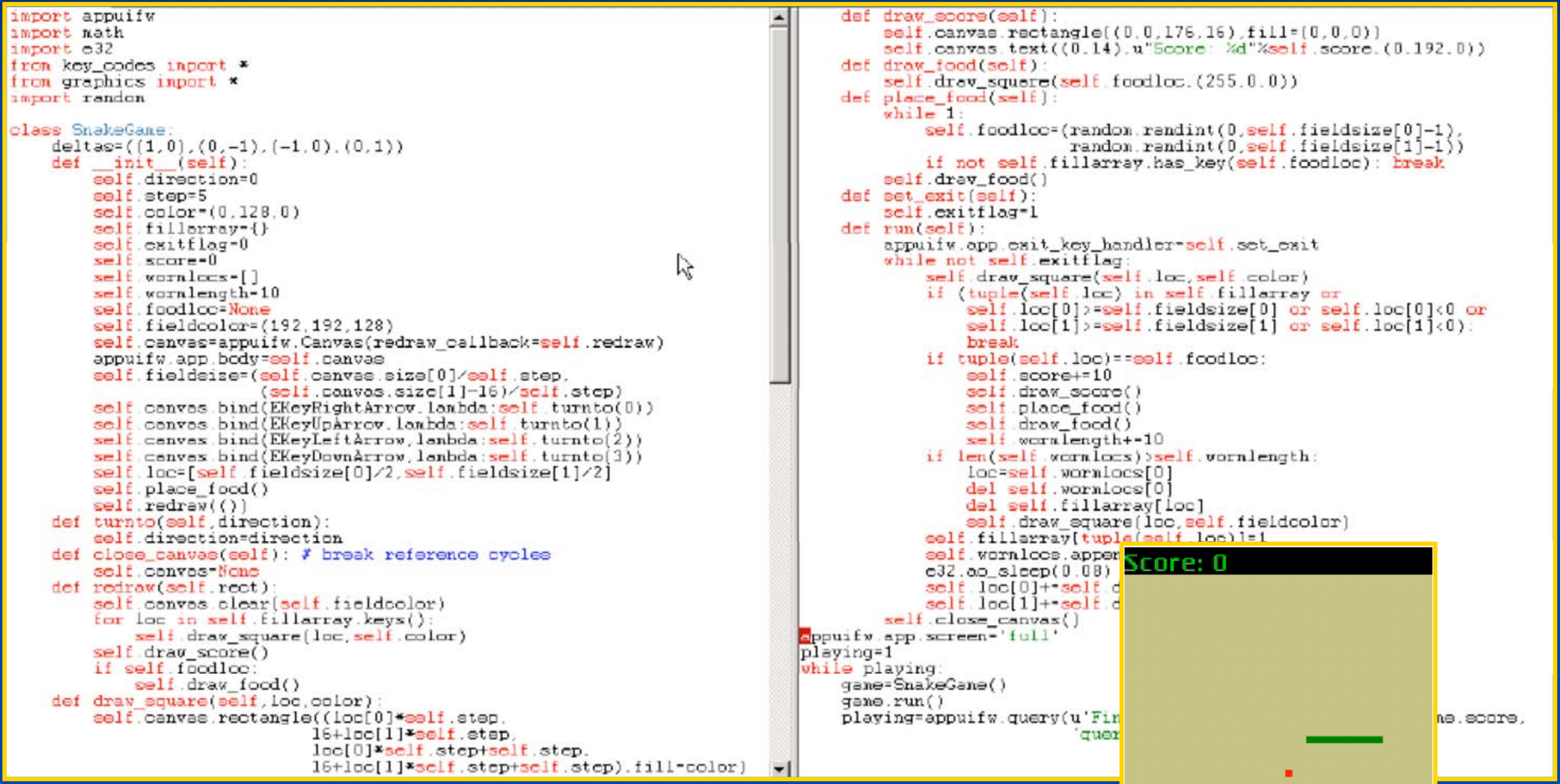


Hello World!

- Answer: It's SO Easy!

Snake – in 93 lines!

Images from: Kari Pulli, "Rapid Development on PyS60" 2006



```
import appuifw
import math
import e32
from key_codes import *
from graphics import *
import random

class SnakeGame:
    deltas=((1,0),(0,-1),(-1,0),(0,1))
    def __init__(self):
        self.direction=0
        self.step=5
        self.color=(0,128,0)
        self.fillarray={}
        self.exitflag=0
        self.score=0
        self.wormlocs=[]
        self.wormlength=10
        self.foodloc=None
        self.fieldcolor=(192,192,128)
        self.canvas=appuifw.Canvas(redraw_callback=self.redraw)
        appuifw.app.body=self.canvas
        self.fieldsize=(self.canvas.size[0]/self.step,
                        (self.canvas.size[1]-16)/self.step)
        self.canvas.bind(EKeyRightArrow,lambda:self.turnto(0))
        self.canvas.bind(EKeyUpArrow,lambda:self.turnto(1))
        self.canvas.bind(EKeyLeftArrow,lambda:self.turnto(2))
        self.canvas.bind(EKeyDownArrow,lambda:self.turnto(3))
        self.loc=[self.fieldsize[0]/2,self.fieldsize[1]/2]
        self.place_food()
        self.redraw(())
    def turnto(self,direction):
        self.direction=direction
    def close_canvas(self): # break reference cycles
        self.canvas=None
    def redraw(self,rect):
        self.canvas.clear(self.fieldcolor)
        for loc in self.fillarray.keys():
            self.draw_square(loc,self.color)
        self.draw_score()
        if self.foodloc:
            self.draw_food()
    def draw_square(self,loc,color):
        self.canvas.rectangle((loc[0]*self.step,
                               16+loc[1]*self.step,
                               loc[0]*self.step+self.step,
                               16+loc[1]*self.step+self.step),fill=color)
```

```
    def draw_score(self):
        self.canvas.rectangle((0,0,176,16),fill=(0,0,0))
        self.canvas.text((0,14),u"Score: %d"%self.score,(0,192,0))
    def draw_food(self):
        self.draw_square(self.foodloc,(255,0,0))
    def place_food(self):
        while 1:
            self.foodloc=(random.randint(0,self.fieldsize[0]-1),
                          random.randint(0,self.fieldsize[1]-1))
            if not self.fillarray.has_key(self.foodloc): break
        self.draw_food()
    def set_exit(self):
        self.exitflag=1
    def run(self):
        appuifw.app.exit_key_handler=self.set_exit
        while not self.exitflag:
            self.draw_square(self.loc,self.color)
            if (tuple(self.loc) in self.fillarray or
                self.loc[0]>=self.fieldsize[0] or self.loc[0]<0 or
                self.loc[1]>=self.fieldsize[1] or self.loc[1]<0):
                break
            if tuple(self.loc)==self.foodloc:
                self.score+=10
                self.draw_score()
                self.place_food()
                self.draw_food()
                self.wormlength+=10
            if len(self.wormlocs)>self.wormlength:
                loc=self.wormlocs[0]
                del self.wormlocs[0]
                del self.fillarray[loc]
                self.draw_square(loc,self.fieldcolor)
            self.fillarray[tuple(self.loc)]=1
            self.wormlocs.appen
            e32.ao_sleep(0.08)
            self.loc[0]+=self.d
            self.loc[1]+=self.d
        self.close_canvas()
appuifw.app.screen='full'
playing=1
while playing:
    game=SnakeGame()
    game.run()
    playing=appuifw.query(u'Fin                                   me.score,
                          'quer
```

EPROM ENTREPRENEURIAL PROGRAMMING AND RESEARCH ON MOBILES

MIT http://mit.edu/eprom

The Goal for this Course:

- Become a Mobile Phone Hacker!
  - Write applications that send SMS messages
  - Turn the phone's microphone on and record audio files to the memory card.
  - Develop an Address Book application that has more functionality than the built-in Contacts app.
  - Write a program that has a GUI that automatically changes depending on the user's location
  - Be able to write a mobile phone virus (but don't).

A Taste of Python: STRINGS

From Guido van Rossum’s tutorial: http://www.**python**.org/doc/essays/ppt/lwnyc2002/intro22.pp

- Guido van Rossum’s Intro to Python, 2002:

"hello"+"world"	"helloworld"	# concatenation
"hello"*3	"hellohellohello"	# repetition
"hello"[0]	"h"	# indexing
"hello"[-1]	"o"	# (from end)
"hello"[1:4]	"ell"	# slicing
len("hello")	5	# size
"hello" < "jello"	1	# comparison
"e" in "hello"	1	# search

A Taste of Python: LISTS

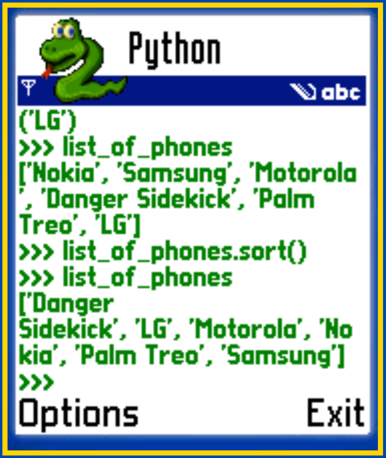
From Guido van Rossum's tutorial: http://www.**python**.org/doc/essays/ppt/lwnyc2002/intro22.pp

- Flexible arrays, not Lisp-like linked lists
  - a = [99, "bottles of beer", ["on", "the", "wall"]]
- Same operators as for strings
  - a+b, a*3, a[0], a[-1], a[1:], len(a)
- Item and slice assignment
  - a[0] = 98
  - a[1:2] = ["bottles", "of", "beer"]
    - [98, "bottles", "of", "beer", ["on", "the", "wall"]]
  - del a[-1]   # -> [98, "bottles", "of", "beer"]

A Taste of Python: LIST METHODS

From Guido van Rossum's tutorial: http://www.python.org/doc/essays/ppt/lwnyc2002/intro22.pp

```
>>> a = range(5)          # [0,1,2,3,4]
>>> a.append(5)           # [0,1,2,3,4,5]
>>> a.pop()               # [0,1,2,3,4]
5
>>> a.insert(0, 42)       # [42,0,1,2,3,4]
>>> a.pop(0)              # [0,1,2,3,4]
42
>>> a.reverse()           # [4,3,2,1,0]
>>> a.sort()              # [0,1,2,3,4]
```

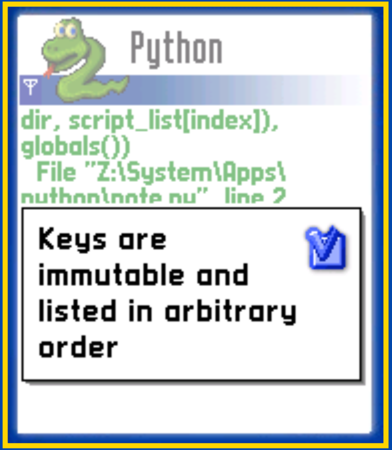


```
('LG')
>>> list_of_phones
['Nokia', 'Samsung', 'Motorola
', 'Danger Sidekick', 'Palm
Treo', 'LG']
>>> list_of_phones.sort()
>>> list_of_phones
['Danger
Sidekick', 'LG', 'Motorola', 'No
kia', 'Palm Treo', 'Samsung']
>>>
```

A Taste of Python: Dictionaries

- Dictionaries
  - Associative Arrays / Hash Tables of Keys and Items
    - d = {"Brand": "Nokia", "Model": "6600"}
    - Methods: keys(), values(), items(), has_key()
  - Look Up
    - d["Model"]

      6600
  - Delete
    - del d["Brand"]



```
dir, script_list[index]),
globals())
  File "Z:\System\Apps\
python\note.py", line 2
```

Keys are immutable and listed in arbitrary order

A Taste of Python: Indenting

From Guido van Rossum's tutorial: http://www.**python**.org/doc/essays/ppt/lwnyc2002/intro22.pp

- Control Structures
  - Indenting instead of braces
  - Easy, but Dangerous

```
#           in Python
for i in range(20):
    if i%3 == 0:
        print i
        if i%5 == 0:
            print "Bingo!"
    print "---"
```

```
/* in C */
for (i = 0; i < 20; i++)
{
    if (i%3 == 0) {
        printf("%d\n", i);
        if (i%5 == 0) {
    printf("Bingo!\n"); }
      }
      printf("---\n");
}
```

```
0
Bingo!
---
---
---
3
---
---
---
6
---
---
---
9
---
---
---
12
---
---
---
15
Bingo!
---
---
---
18
---
---
```

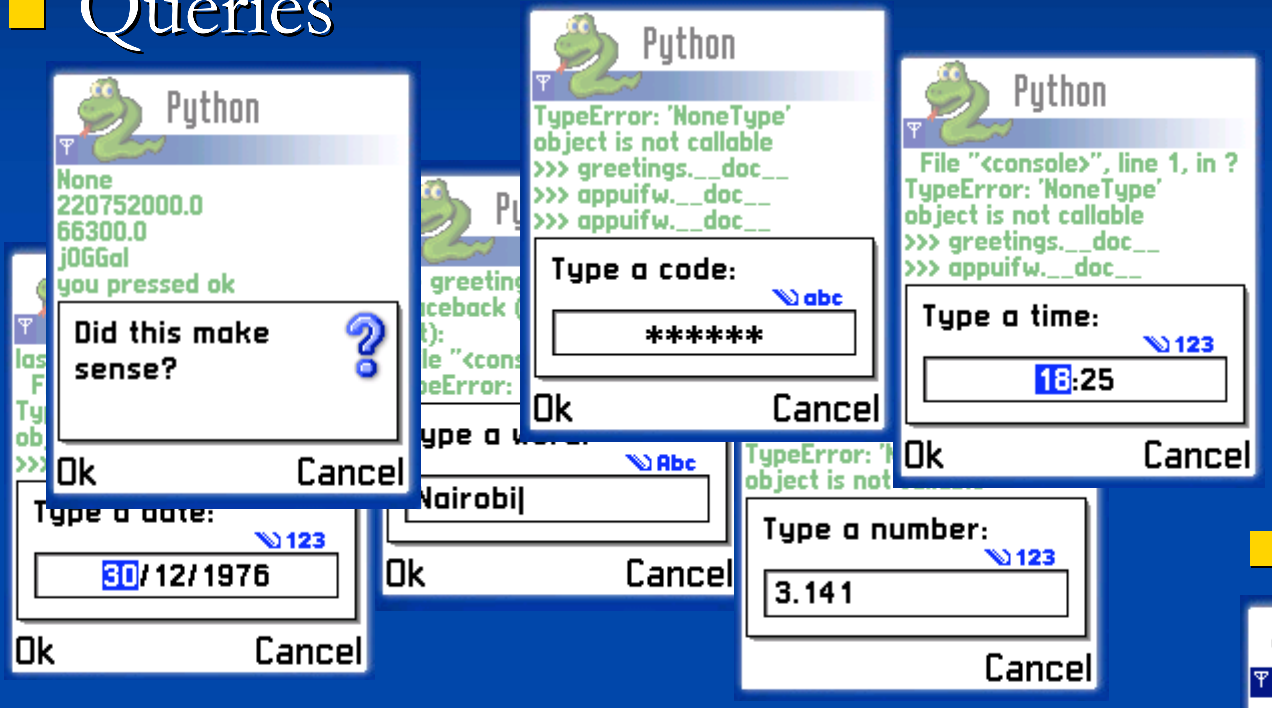
A Taste of Python: Functions and Modules

- Functions
  - Example: greetings.py
- Modules
  - Example: import appuifw
  - Check out what you get with dir
    - ie: dir(appuifw)

```
def greetings():
    "this is an example function"
    name = appuifw.querry(u'What is your name?','text')
    appuifw.note(u'Hail '+name, 'conf')
greetings()
```

The appuifw Module

- Queries



```
Python
None
220752000.0
66300.0
j0GGal
you pressed ok
```

Did this make sense? — Ok / Cancel

Type a date: 123 — 30/12/1976 — Ok / Cancel

```
greeting
ceback (
t):
le "<cons
peError:
```

Type a word: Abc — Nairobi| — Ok / Cancel

```
Python
TypeError: 'NoneType'
object is not callable
>>> greetings.__doc__
>>> appuifw.__doc__
>>> appuifw.__doc__
```

Type a code: abc — ****** — Ok / Cancel

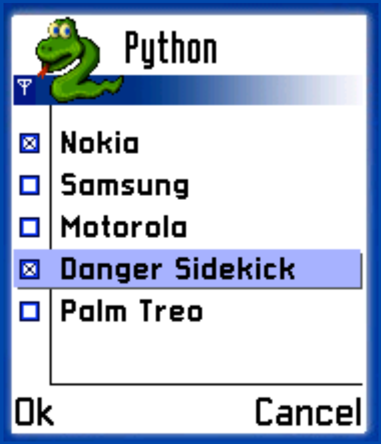
```
TypeError: '
object is not
```

Type a number: 123 — 3.141 — Cancel

```
Python
 File "<console>", line 1, in ?
TypeError: 'NoneType'
object is not callable
>>> greetings.__doc__
>>> appuifw.__doc__
```

Type a time: 123 — 18:25 — Ok / Cancel

- Forms



Python

- [x] Nokia
- [ ] Samsung
- [ ] Motorola
- [x] Danger Sidekick
- [ ] Palm Treo

Ok / Cancel

EPROM ENTREPRENEURIAL PROGRAMMING AND RESEARCH ON MOBILES

MIT http://mit.edu/eprom

The Better Way to Learn: HACK!

- Download the API Reference here:
  - http://reality.media.mit.edu/pdfs/pys60_api.pdf
- Get a computer set up with the phone emulator and Python
  - S60 SDK for 2nd Edition, FP 2 http://www.forum.nokia.com/main/0,,034-483,00.html
  - Python for S60 SDK: http://www.forum.nokia.com/main/0,,034-821,00.html

Running Your Own Scripts

- Transferring Scripts from PC to Phone
  - Bluetooth Send / Sync / Console
  - Infrared Transfer
  - Memory Card Transfer
  - GPRS Transfer
- Using the Symbian Emulator
  - Path for .py scripts (or something similar)
    - C:\Symbian\8.0a\S60_2nd_FP2\epoc32\release\wins\udeb\z\system\APPS\python
  - Run Emulator in debug mode
  - Open Python -> Options ->Run Scripts

Day 1 Exercises: User Interfaces

- Hello World
  - Making Sure Everyone Can Program the Phone
- Hail!
  - Introduction to Queries
- Class Survey
  - Advanced UIs
  - file i/o…

Day 2 : Application Development

- Building Applications
  - Title, Screen Size, Tabs,
- GUI Design
  - Customizing Your Own Graphical User Interfaces
- Graphics and Drawing
- Keyboard Keys
- XML
- Contacts and Calendar Databases

Day 3: Using the Phone as a Sensor

- OS Read and Writes
  - File IO
  - System Information
- Sound Recording and Playing
  - Call Logs
  - .wav processing?
- Location
  - Logging the Cell Towers
  - BT GPS Interface
- Imaging
  - Image Capture using the Camera
  - Image Handling
- Bluetooth Sensing
  - Identifying Who is Around…

Extra Exercises

- Networking
- Creating a Logging Script?
- Mobile Phone Virus?
- Standalone Apps: py2sis