Distributed Computing

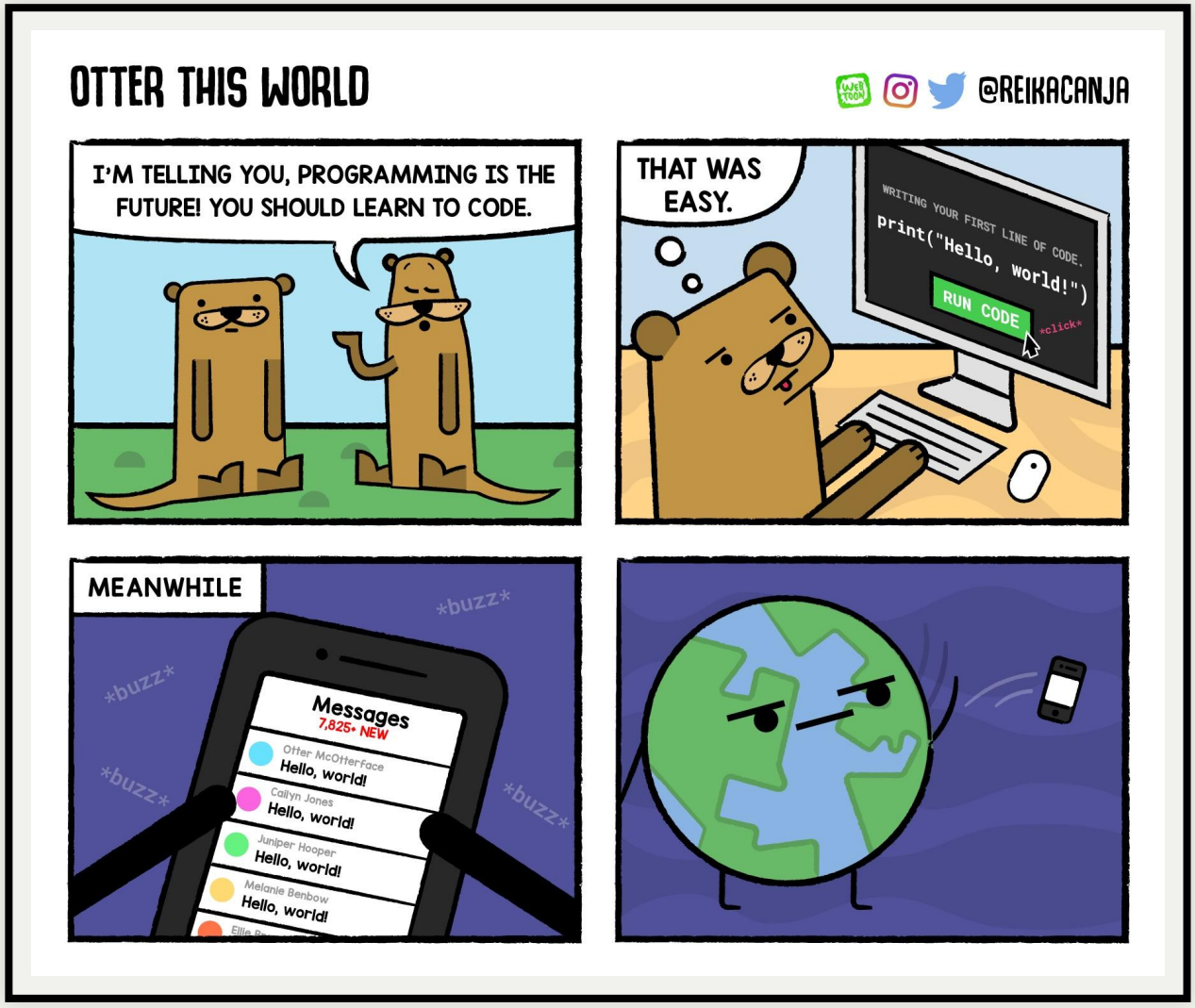
COMP 3010

Robert Guderian

Python for people who have programmed before

Syntax, usage, debugging

I mean, what else?



Octothorpe!

```
#!/usr/bin/python
```

aka `shebang`

Blocks

```
if not aThing and 10 > x:
  # ends when we stop the indent
  print("In it")
  print('also single quotes work')
print('done the if')
```

if/elif/else

```
if (x < 9):
  print("this one")
elif (x < 42):
  print("other one")
else:
  print("The last one")
```

Data structures

No arrays, everything is a list, has functions

```
list1 = []
list2 = [1, 2, "pants"]
list1.append(42)
print(list1)
print(list2)
```

Tables for free

Totally nuts key / value pair table (hash)

```
dict1 = {}
dict1['key'] = "whatever"
dict2 = {"key matter": 42, "other": "mixed data"}
```

Loops

`while` is what you expect

```
something = 0
while something > 10:
  # oh yeah... ++ doesn't work
  something += 1
```

`for` is a mess

`for` is an iterator

```
aList = [1,2,3,100,1000]
for anItem in aList:
    print(anItem)
```

Works for `dicts`, `sets`, and a lot of other things.

`for`-like

To recreate C/Java `for`...

```
for i in range(20):
  print(i)
```

`file`-like objects

Sockets, files, and other devices that are read / write are file-like objects

We get `read`, `write`, and sometimes `seek`.

Open a text file

```
# uses relative paths to where python was invoked
aFile = open('best_file.txt', 'r')
theText = aFile.read()
```

line-by-line

With some string format

```
# uses relative paths to where python was invoked
aFile = open('best_file.txt', 'r')
lineNumber = 0
for line in aFile:
  print("line %d: %s".format(lineNumber, line))
  lineNumber += 1
```

Standard in is a file-like object

```
import sys
# with closes resources for us at the end of the block
# sys.stdin is a file-like object
with sys.stdin as inStream:
  # wait for a line
  line = inStream.readline()
print(line)
```

Functions

```
def aFunctionName(parameters, goHere):
  print("first one " + parameter)
  print("second one " + goHere)
  return 1

theRetValue = aFunctionName("in", "order")
```

Objects

```
class GreatClass:
  def __init__(self): # the constructor
    name = "Samwise"
    age = 31

  def addAge(self, amount):
    self.age += amount
```

pdb

```
import pdb; pdb.set_trace()
```

You drop into a `gdb`-like session.


;
PYTHON