Open Source Software:

Programming in Python

https://docs.python.org/3/tutorial/index.html

https://docs.python.org/3/whatsnew/index.html

http://opensourcebridge.org/wiki/2014/A_Few_Python_Tips

Who uses Python?

- What functionality is available?

http://www.pythonforbeginners.com/api/list-of-python-apis

https://developers.google.com/api-client-library/python

About

- What's a scripting language?
  - why is python useful? / who uses it?
- nice interactive interpreter
- Rich standard library & PyPI (package index)
- Data Structures
  - lists / dictionaries / sets / iterators
- object oriented
  - yield/generator/iterator
- uses garbage collection
- can treat a function as an object
- duck typing (dynamic typing)
- pip/ dev tools: pydoc/docstring/debugger/unittest

Guido van Rossum

https://www.python.org/~guido/

Scripting Language

- What is a scripting language?

- Why would you use one?

- Do you really not compile the code?
  - interpreter vs compiler vs byte code & Virtual Machine

Install – Python 3.X

- Windows or Mac
  - https://www.python.org/downloads/
  - Mac: Homebrew http://brew.sh
- Linux
  - via package manager
  - yum, apt-get, zypper/yast ….
- ipython ⇦
  - better Python shell
- IDLE
  - GUI version of the Python shell
- Source
  - the source code is also available

Linux:

```
bart$ idle3

OR

bart$ python3

OR

bart$ ipython3
```

Python Software

- pip
  - install and manage Python packages

    pip-3.3 install virtualenvwrapper

    https://pypi.python.org/pypi ← list packages

- virtual environments

  virtualenv-3.3 CS360_A1
  source CS360_A1/bin/activate
  pip-3.3 install simplejson
  pip-3.3 install "ipython[notebook]"
  pip-3.3 freeze
  deactivate

  https://pypi.python.org/pypi/pip

  http://docs.python-guide.org/en/latest/dev/virtualenvs/

Python on OpenSUSE

```
zypper in python3 python3-virtualenv

virtualenv-3.3 cs360f14_python_env
source cs360f14_python_env/bin/activate

pip-3.3 install "ipython[notebook]"

ipython3
x = 42
print(x)
exit()

python3
import tkinter
tkinter._test()   # this will pop up a small dialog box.
                  #Press the button to quit the dialog box.
exit()

deactivate
```

You should install this on your cs360-# server.

http://richardt.name/blog/setting-up-ipython-notebook-on-windows/

ipython

```
chadd@coffee:~> ipython3

Python 3.3.5 (default, Mar 27 2014, 17:16:46) [GCC]
Type "copyright", "credits" or "license" for more information.

IPython 2.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:
```

- Let's try some commands

print (“HI”)

3 + 4

answer = 3 + 4

print(“The answer is: “, answer)

help()

if

Other interactive options:
- python
- bpython
- IDLE

BNF

- Backus-Naur Form

The "if" statement is used for conditional execution:

```
if_stmt ::= "if" expression ":" suite
            ( "elif" expression ":" suite )*
            ["else" ":" suite]
```

http://en.wikipedia.org/wiki/Backus-Naur_Form

https://docs.python.org/3/reference/grammar.html

https://docs.python.org/3/reference/index.html

Let's use an If statement

- Print hello if answer is greater than 7
- Print bye if answer is less than 7
- Print winner if answer is exactly 7

- It is not evident in the BNF, but indentation is very important
- No curly braces like in C/C++
- Indentation instead

Read the red warnings.
Don't copy and paste already indented code with autoindent turned on!

- http://ipython.org/ipython-doc/dev/interactive/reference.html#autoindent

data

- All data are objects
  - identity id()
  - type type()
  - value
    - mutable (dictionaries, lists, ...)
    - immutable (numbers, strings, …)
- Garbage collection
  - implementation dependent
- None
- NotImplemented

is vs ==

Built in

```
array: mArr = array.array('i')
list: mList = []
dictionary
```

https://docs.python.org/3/reference/datamodel.html

Interrogate

- dir(*type*)
  - what names are available for *type*?
- What methods are available for int ?

```
value = 5
value.method()
```

- dir(__builtin__)

What if you type dir( ) ?

strings - str

- https://docs.python.org/3/tutorial/introduction.html
- single ' or double quotes “   \x to escape x.
- Triple quotes: span lines

Building Strings

- Concatenate: +
- Repeat: *

Strings like Arrays/Lists

- data = “CS360”
- data[0] # 'C' data[1:3] # “S3” data[-1] #

Check out the while statement

- print all the integers from 1 to 10 using a while

  yourName = input(“Name? “)
  yourAge = int(input(“Age? “))

- print all the integers from 1 to yourAge.

int(x, base)

- int(x, base)
  - convert *x*, a string written in base *base* into an int (in base 10)
- bin(x)
  - convert x, an int in base 10, to base 2

int (input("Age ?"))

int (input("Age in binary ?"), base = 2)

int(bin(42), base = 2)

- *keyword arguments*

https://docs.python.org/3/tutorial/controlflow.html#keyword-arguments

Setup

```
cd ~/Documents
virtualenv-3.3 CS360_python
source CS360_python/bin/activate
pip-3.3 install "ipython[notebook]"
ipython3
…
exit()
deactivate
```

Do not put CS360_Python oin GitHub!

- Go to GitHub
- Fork cs360f14/PythonExamples_Lectures

```
cd ~/Documents

git clone …
cd PythonExamples_Lectures

source ../CS360_python/bin/activate
```

- You can commit your ipython logs to GitHub for later!

For loop

Data Structures

- Sequences
  - immutable: String, Tuple, Bytes
  - mutable: Lists, Byte Arrays
- Sets
  - immutable: frozenset
  - mutable: set
- Mappings
  - dictionaries

List [a type of sequence, duplicates allowed]

- vowels = ['a', 'e', 'i', 'o', 'u']
- print (vowels)

  ['a', 'e', 'i', 'o', 'u']
- print(vowels[0])
- print(vowels[-1])
- print(vowels[2:])
- print(vowels+ ['y'])
- vowels[0] = 'A'
- vowels[1:3] = ['E', 'I']
- vowels[1:3] = []
- vowels[:] = []
- functions:
  - len(vowels)
  - vowels.append('y')
- numbers = ['zero', 1, '`II`']

More on Lists

- append()/pop()
- popleft()
- List **Comprehensions**
  - make a list
  - squares = [x**2 for x in range(10) if x % 2 == 0]

expression for clause for/if

  - squaresAndOne = [(x**2, x**2+1) for x in range(10)]
- del

```
for pos, value in enumerate(squares): # position, value
    print (pos, value)

for value in squares:
    print (value)
```

https://docs.python.org/3/tutorial/datastructures.html#more-on-lists

tuple (a type of sequence)

- course = 'cs360', 'fall', 2014

  ('cs360', 'fall', 2014)

- grade = course, 'A'

  (('cs360', 'fall', 2014), 'A')

- unknownGrade = course,

  (('cs360', 'fall', 2014) ,)

- classname, semester, year = course

Set (unordered, no duplicates)

- depts = {'CS', 'Math', 'Bio'}

- 'CS' in depts
  True

- longstring = 'anmfnkjv.....23kljfn,...'
  letters = { x for x in *longstring* if x in vowels }

```
for name in depts:
    print(name)

for name in sorted(depts):
    print(name)
```

Dictionary (mapping)

- office = {'chadd':202, 'shereen':203, 'doug':201}
- office['chadd']
- office['chadd'] = 'supply closet'
- office['boardman'] = 'Price 209'
- office.keys() list(office.keys())
- 'chadd' in office 203 in office

Dictionary

- cs = dict([(202, 'chadd') , (203, 'shereen'), (201, 'doug')])

- squared = { x : x**2 for x in range(10) }

- cs = dict(chadd= 202 , shereen=203, doug=201)

```
for k, v in cs.items() # key, value
    print(k, v)
```

Execution

- Names refer to objects
  - names are bound to objects

    x = MyObj()
- block is a piece of code executed as a unit
- execution frame ~ stack frame
- scope

https://docs.python.org/3/reference/executionmodel.html

Get Started!

- Start your virtual environment
- Start ipython3
- Start your logf le (optional)

Let's put this in a f le

- Open a second terminal!

- **`cd Documents/PythonExamples_Lectures`**

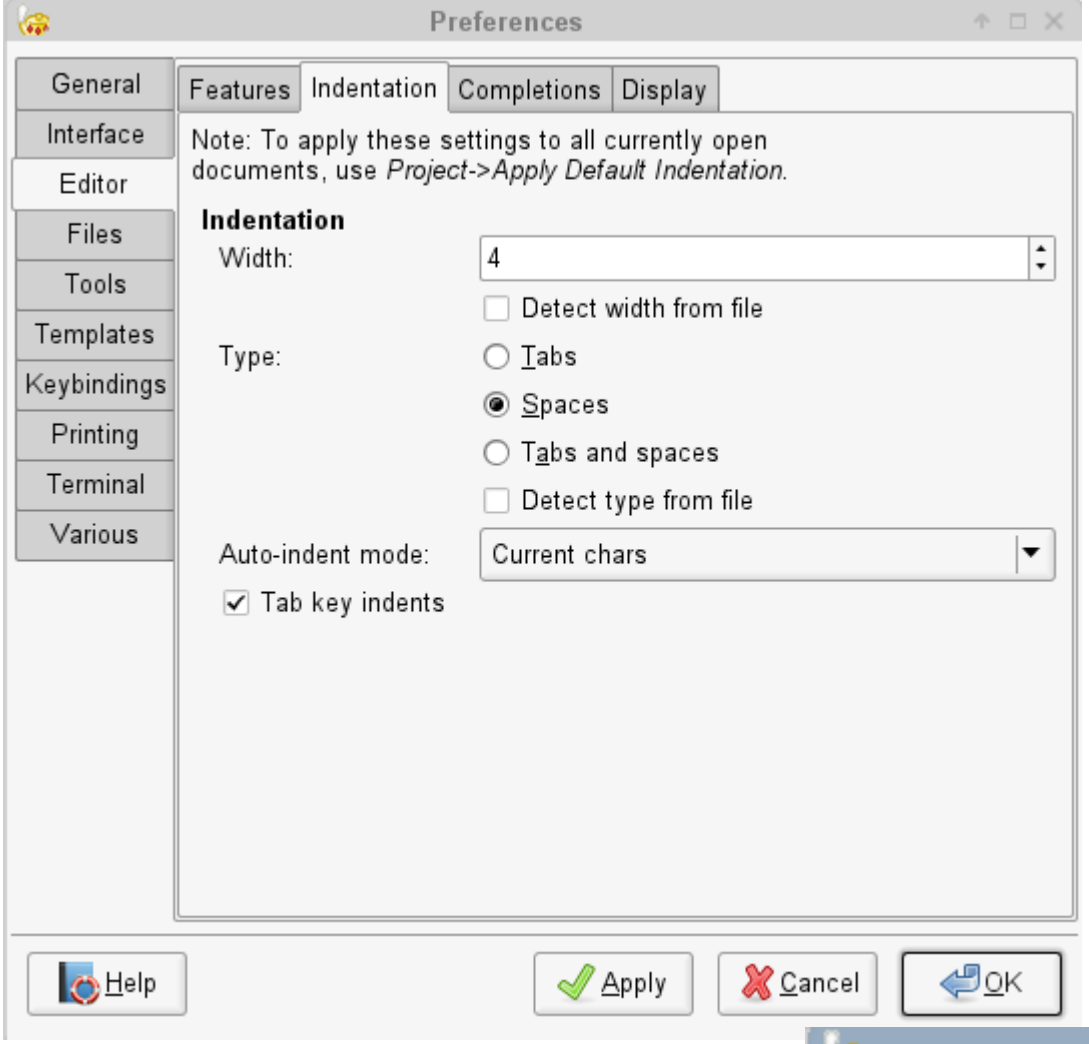
- open f irst.py (geany, nano, *your choice*...)

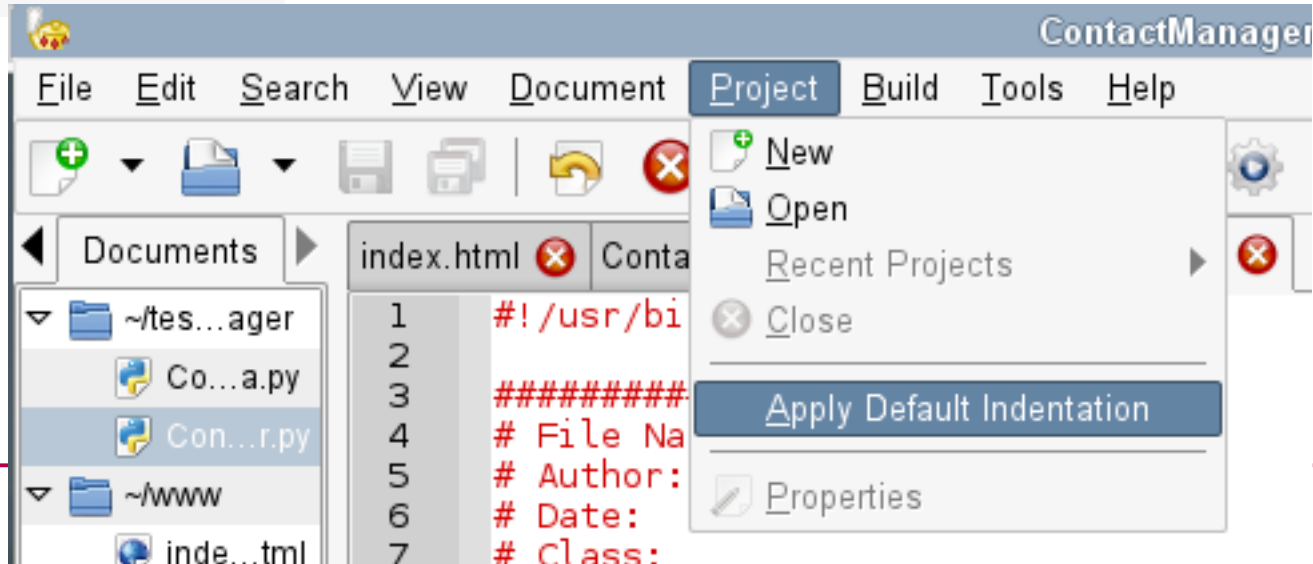
```
#!/usr/bin/python3
print ("Hi")
```

- **`chmod u+x first.py`**
  `./first.py`
  OR
  `python first.py`

- *git add/commit/push*

Set tabs to 4 spaces





Coding Standards

- style guide
  - http://legacy.python.org/dev/peps/pep-0008/
- Zen of Python
  - http://legacy.python.org/dev/peps/pep-0020/

```
####################################
# File Name:
# Author:
# Date:
# Class:
# Assignment Title
# Purpose:
####################################
```

- PyDoc
  - https://docs.python.org/3/library/pydoc.html
  - https://docs.python.org/3/library/doctest.html

Add a skeleton to GitHub

- open skeleton.py

```
#!/usr/bin/python3

#####################################
# File Name:
# Author:
# Date:
# Class:
# Assignment:
# Purpose:
#####################################
```

- git add/commit/push

Functions

- Take parameters, return a single value

```
def funcname ( paramlist ) :
    statements
```

Arguments

- Default

```
def funcname ( value, error = 0.1, unit = 'Miles') :
    print(value, error, unit, sep="+")
```

- Keyword

```
funcname(2, unit='km')
funcname(unit='km', error=0.9, value = 9)
```

Keyword, continued

```
def cheeseshop(kind, *arguments, **keywords):
    print("-- Do you have any", kind, "?")
    print("-- I'm sorry, we're all out of", kind)

    for arg in arguments:
        print(arg)

    print("-" * 40)
    keys = sorted(keywords.keys())
    for kw in keys:
        print(kw, ":", keywords[kw])
```

```
cheeseshop("Limburger", "It's very runny, sir.",
           "It's really very, VERY runny, sir.",
           shopkeeper="Michael Palin",
           client="John Cleese",
           sketch="Cheese Shop Sketch")
```

https://docs.python.org/3/tutorial/controlflow.html#documentation-strings

Variable Number (variadic)

```
def funcname(*args)
   .....
```

Unpacking arguments

- I already have my arguments in a list!

```
>>> def parrot(voltage, state='a stiff', action='voom'):
...     print("-- This parrot wouldn't", action, end=' ')
...     print("if you put", voltage, "volts through it.", end=' ')
...     print("E's", state, "!")
…

>>> d = {"voltage": "four million",
         "state": "bleedin' demised",
         "action": "VOOM"}

>>> parrot(**d)
-- This parrot wouldn't VOOM if you put four million volts through it.
 E's bleedin' demised !
```

Doc Strings

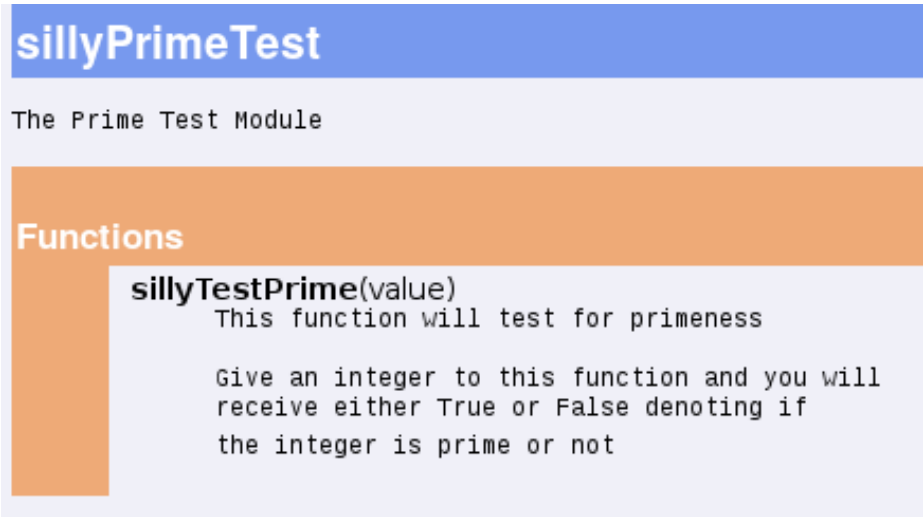
- Doc Strings

```
def funcname () :
    """This is a one line  comment

    This is the longer comment that
     describes the function behavior in detail
    """
    statements......


print(funcname.__doc__)
```

```
This is a one line  comment

    This is the longer comment that
     describes the function behavior in detail
```

PyDoc

```
#!/usr/bin/python

"""
The Prime Test Module
"""

def sillyTestPrime (value) :
    """ This function will test for primeness

    Give an integer to this function and you will
    receive either True or False denoting if
    the integer is prime or not
    """

    counter = 2
    prime = True
    while counter <= value / 2 and prime:
        prime = (value % counter != 0)
        counter += 1

    return prime
```



DocTest

```
"""
doctest Example

>>> sumTwo(2,2)
4
"""

def sumTwo(left, right) :
    """ return the sum of both values

    >>> sumTwo(1,2)
    3

    >>> sumTwo(1.1, 3)
    4.1
    """
    return left + right


if __name__ == "__main__":
     import doctest
     doctest.testmod()
```

python3 test_doctest.py -v

python3 -m doctest -v example.py

Function Annotations (python 3 only)

```
def funcname (param : “first param”,
  value : int = 42) -> “no return stmt” :
    print (funcname.__annotations__)
    print (param, value)
```

```
>>> funcname(2)
{'return': 'no return stmt', 'param': 'first
param', 'value': <class 'int'>}
2 42
```

Get Started!

- Start your virtual environment

- fetch upstream PythonExamples_Lectures

```
(CS360_python)you@machine:~> python3 file.py
```

lambda - lambdaExample.py

- anonymous function
  - function not bound to an identifier
  - used to:
    - pass as a parameter to another function
    - returned from a function
  - restricted to single expression

https://docs.python.org/3/tutorial/controlflow.html#lambda-expressions

pass lambda function as parameter

```
>>> pairs = [(1, 'one'), (2, 'two'), (3, 'three'), (4, 'four')]

>>> pairs.sort(key=lambda pair: pair[1])

>>> pairs
[(4, 'four'), (1, 'one'), (3, 'three'), (2, 'two')]

>>> type(pairs)
list
```

https://docs.python.org/3/library/stdtypes.html#list.sort

yield/generate/iterator generatorExample.py

- iterator
  - idiom to access each single item one at a time

```
for value in squares:
    print (value)
```

- generator
  - a way to create iterators

```
def squared(data):
    for value in data:
        yield value**2
```

- yield
  - generation of a single item

```
numbers =[0,1,2,3,4,5]
for square in squared(numbers):
    print (square)
```

- generator expressions

```
sum(i*i for i in range(3))
```

Classes - classExample.py

- class members are public
  - no private except by convention!
- member functions are virtual

```
class CSCourse :
    """Represent a single CS Course"""
   kind = 'CS' # class variable shared by all CSCourses

   def __init__(self, name, number) :
      self.name = name         # instance variable
      self.number = number

   def display(self):
      print("CS Course: " , self.name, self.number, sep=" ")

   def __str__(self):
      return kind + self.name + str(self.number)

cs360=CSCourse("Special Topics", 360)
cs360.display()
print(str(cs360))
```

Inheritance

inheritanceExample.py

```
class Course :
    """Represent a single Course"""
    kind = 'Gen Ed'

    def __init__(self, name, number) :
        self._name = name # 'private' instance variable
        self._number = number
        self.__display()

    def display(self):
        print(self.kind,"Course:" , self._name, self._number, sep=" ")
    __display = display # private copy

class CSCourse(Course) :
    """Represent a single CS Course"""
    kind = 'CS' # class variable shared by all CSCourses

    def __init__(self, name, number, language, numberOfPrograms) :
        Course.__init__(self, name,  number)
        self._language = language
        self._numberOfPrograms = numberOfPrograms

    def display(self):
        Course.display(self)
        print('Language', self._language,
          'Number Of programs:', self._numberOfPrograms, sep = ' ')
```

On the Fly - dynamicClassExample.py

```
class Numbers:
    pass


def print(value):
  print(value.integer)


data = Numbers()
```

Exceptions - exceptionsExample.py

- Produce an error that can be handled programmatically

```
try:
    statements
except ExceptionType as err :
    ExceptionType_occurred
except DifferentExceptionType :
    DifferentExceptionType_occurred
else :
    no_exception_occurred
finally :
    always_run_statements
```

https://docs.python.org/3/library/exceptions.html

```
raise NameError('unknown name!')
```

Debugger - debug_example.py

- pdb
- python -i example.py
  - dump you into an interactive session when the code f nishes or crashes
  - use dir()
- python -m pdb example.py
  - break f lename:lineno
  - list
  - step
  - print *var*

https://docs.python.org/3/library/pdb.html

unittest - unittestExample.py

cards.py
unittestCards.py

- Unit Test: Test a small unit of code
- Python module unittest
- subclass unittest.TestCase
- setUp(self)
- tearDown(self)
- test_XXXX(self)
  - self.assertEqual() / self.assertNotEqual()
  - self.assertRaises()
  - self.assert*???????*()

https://docs.python.org/3/tutorial/

https://docs.python.org/3/library/unittest.html

Standard Library

threadExample.py
functionThreadExample.py

- Text Processing
- DataTypes
- Math
- Decimal Floats
- Files / OS
- Threads
- Networking
- Multimedia

```
import os
dir(os)

from x import y
```

https://docs.python.org/3/library/index.html

https://docs.python.org/3/tutorial/stdlib.html

https://docs.python.org/3/tutorial/stdlib2.html

Outside the Standard Library

`pip-3.3 install requests`

requestsExample.py

- Allow you to handle HTTP (web) fetches easily
- Why?

```
(CS360_python) you@there:~> python3 requestsExample.py
```

http://docs.python-requests.org/en/latest/

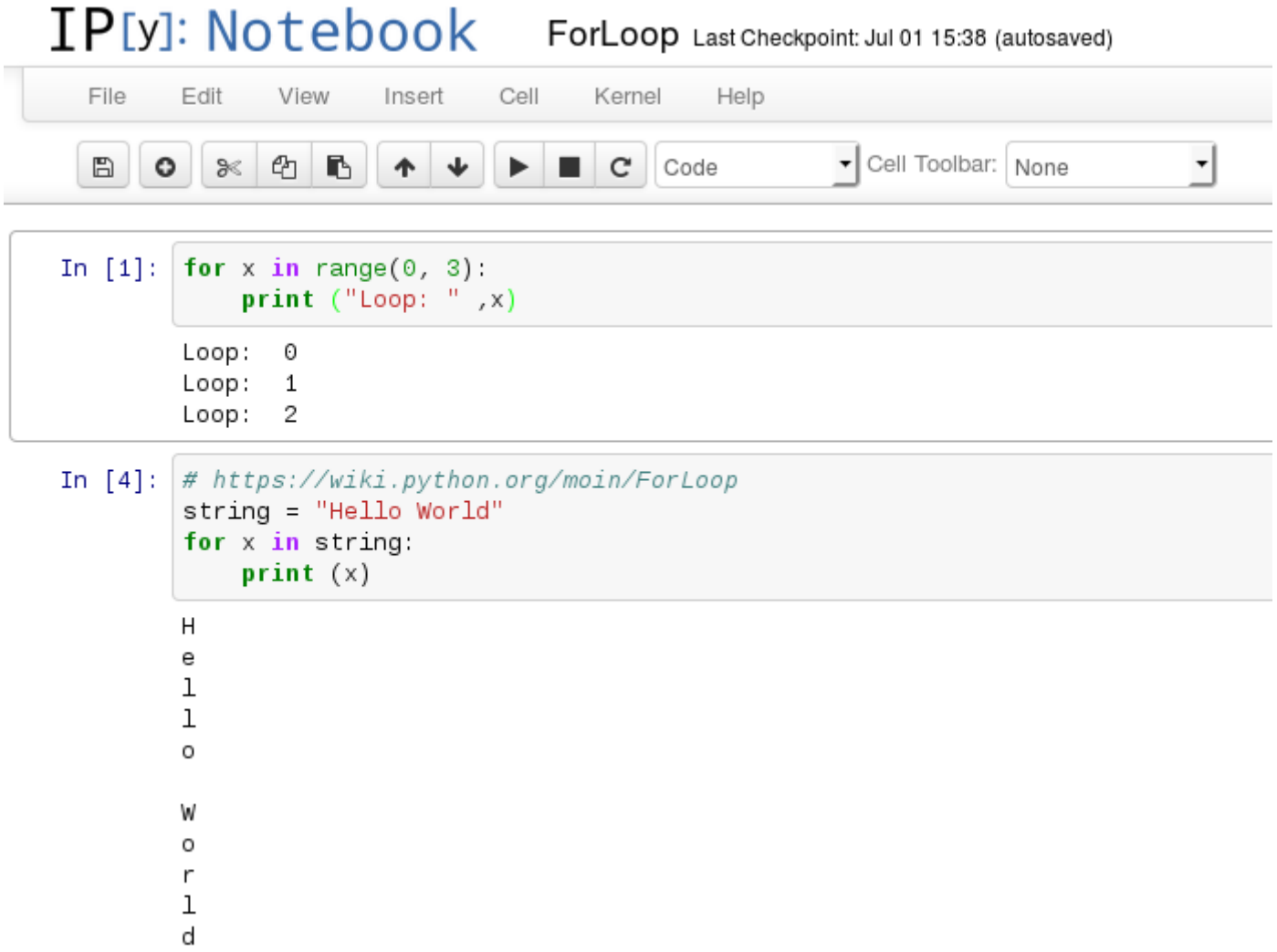
ipython notebook

Python in your browser!

Save the input and output to a nice format

JSON

Can be output as HTML




```
$ ipython3 notebook
```

http://ipython.org/ipython-doc/stable/notebook/index.html
http://richardt.name/blog/setting-up-ipython-notebook-on-windows/
http://www.lfd.uci.edu/~gohlke/pythonlibs/

Get Started!

- Start your virtual environment

- fetch upstream PythonExamples_Lectures

```
(CS360_python)you@machine:~> python3 file.py
```

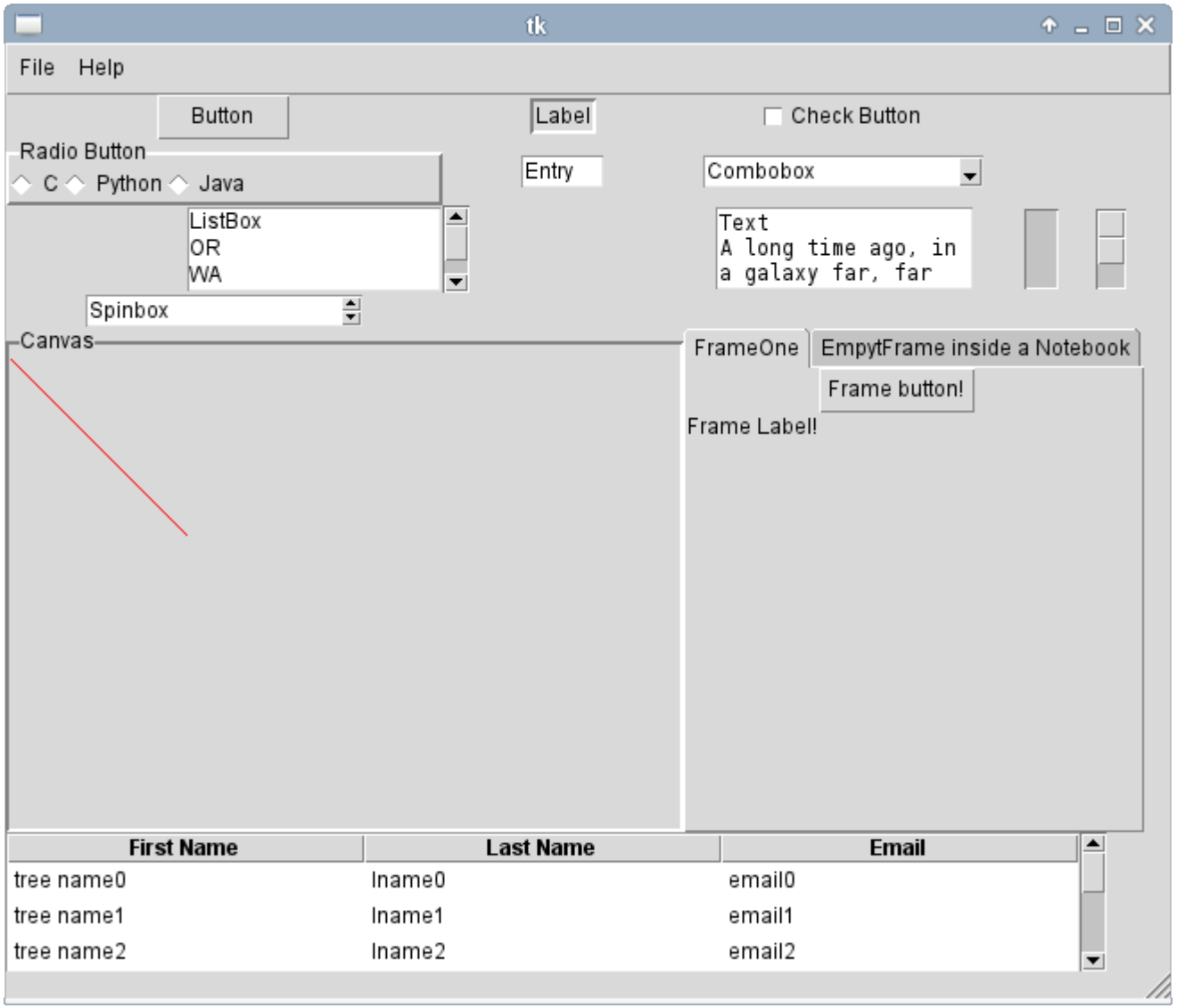
TK GUI - tkinter

- TK: cross platform widget (UI) toolkit
- Mac, Windows, Linux
  - native look and feel
- Many languages
  - Python, Tcl, Perl, Ruby, Ada, C, C++, …
- http://www.tkdocs.com/tutorial/onepage.html
  - gives examples in Tcl, Ruby, Perl, Python
- https://wiki.python.org/moin/TkInter
- http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html
- http://tkinter.unpythonic.net/wiki/
- https://docs.python.org/3/library/tkinter.html

Other options:
PyQt / PySide
wxPython
PyGObject

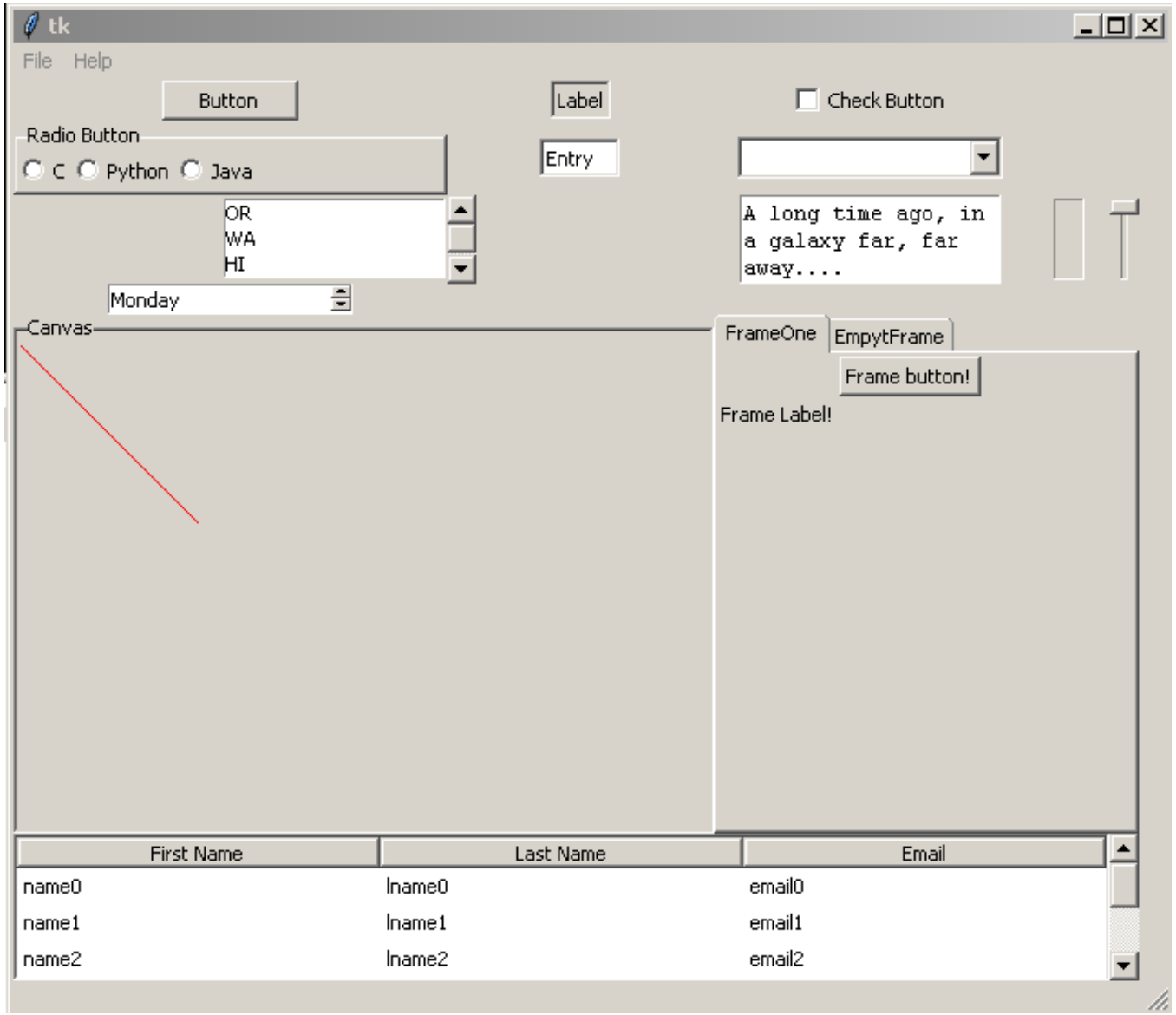
Widgets

widgets.py



Linux

TK - windows


tk
File
Help
Button
Label
Check Button
Radio Button
C
Python
Java
Entry
OR
WA
HI
A long time ago, in a galaxy far, far away....
Monday
Canvas
FrameOne
EmpytFrame
Frame button!
Frame Label!
First Name
Last Name
Email
name0
lname0
email0
name1
lname1
email1
name2
lname2
email2

Does TK work?

```
>>> import tkinter
>>> tkinter._test()
>>> dir(tkinter)
```

Basics

- Widget

simpleButton.py

simpleEntry.py

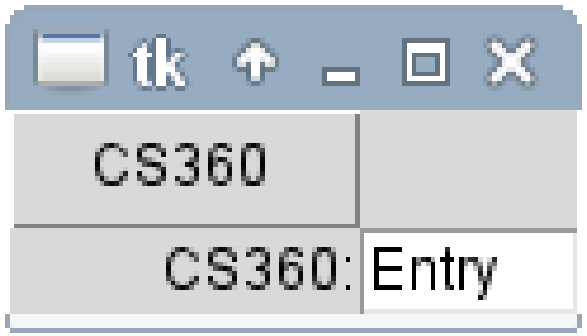
TKExample.py

widgets.py

- Geometry

- Event Handling

Build Me



Pressing the CS360 button should toggle the Entry box between displaying 'CS360' and 'Python'. 'Entry' is displayed in the Entry box only when the application is first launched.

I recommend building a WidgetApp class so the widgets can interact with each other via instance variables, not global variables.

BONUS: Right justify the text in the Entry.

Commit this to your personal **PythonExamples_Lectures/StudentSubmissions/TK** and make a Pull Request back to the main repository.

Name the file: **BuildMe_PUNetID.py**

Get Started!

- Start your virtual environment

- fetch upstream PythonExamples_Lectures

```
(CS360_python)you@machine:~> python3 file.py
```

Standard Library

- re - Regular Expressions
  - reExamples.py
  - https://docs.python.org/3/library/re.html
- csv - Comma Separated Value f le reader
  - csvExample.py
  - https://docs.python.org/3/library/csv.html
- heapq - heap queue (priority queue)
  - heapqExample.py
  - https://docs.python.org/3/library/heapq.html
- datetime - dates and times
  - datetimeExample.py
  - https://docs.python.org/3/library/datetime.html

Exercise

- Read the list of events in the file history.csv into a heap.
- Sort by date
- Print all the events that involve the US in historical order (first to last)

SIP

- (Easily) Allow Python to access C or C++ libraries

Python → Python API → C API → C code