python- Examples

/home/cs2041//public_html/16s2/code/python/pythagoras.0.py (pythagoras.0.py)

compute Pythagoras' Theorem

Works with any Python version but prints a newwline after the 2 prompts

```
import math, sys

print("Enter x:")
x = float(sys.stdin.readline())
print("Enter y:")
y = float(sys.stdin.readline())
pythagoras = math.sqrt(x * x + y * y)
print("The square root of %f squared + %f squared is %f" % (x, y, pythagoras))
```

/home/cs2041//public_html/16s2/code/python/pythagoras.1.py (pythagoras.1.py)

compute Pythagoras' Theorem

Works with Python 3.3+

```
import math, sys

print("Enter x: ", end='', flush=True)
x = float(sys.stdin.readline())
print("Enter y: ", end='', flush=True)
y = float(sys.stdin.readline())
pythagoras = math.sqrt(x * x + y * y)
print("The square root of %f squared + %f squared is %f" % (x, y, pythagoras))
```

/home/cs2041//public_html/16s2/code/python/pythagoras.2.py (pythagoras.2.py)

compute Pythagoras' Theorem

Works with any Python version

```
import math, sys

sys.stdout.write("Enter x: ")
sys.stdout.flush()
x = float(sys.stdin.readline())
sys.stdout.write("Enter x: ")
sys.stdout.flush()
y = float(sys.stdin.readline())
pythagoras = math.sqrt(x * x + y * y)
print("The square root of %f squared + %f squared is %f" % (x, y, pythagoras))
```

/home/cs2041//public_html/16s2/code/python/sum_stdin.py (sum_stdin.py)

count how many people enrolled in each course

Read numbers until end of input (or a non-number) is reached then print the sum of the numbers

```
import re, sys

sum = 0

while 1:
    line = sys.stdin.readline()
    line = line.strip() # remove leading & trailing white space
    # Test if string looks like an integer or real (scientific notation not handled!)
    if not re.search(r'^\d[.\d]*$', line):
        break
    sum += float(line)

print("Sum of the numbers is %s" % sum)
```

/home/cs2041//public_html/16s2/code/python/line_chars.py (line_chars.py)

Simple example reading a line of input and examining characters

```
import sys

sys.stdout.write("Enter some input: ")
line = sys.stdin.readline()
if not line:
    sys.stdout.write("%s: could not read any characters\n" % sys.argv[0])
line = line.rstrip('\n')
n_chars = len(line)
print("That line contained %s characters" % n_chars)
if n_chars > 0:
    first_char =line[0]
    last_char = line[-1]
    print("The first character was '%s'" % first_char)
    print("The last character was '%s'" % last_char)
```

/home/cs2041//public_html/16s2/code/python/exponential_concatenation.py (exponential_concatenation.py)

create a string of size 2^n by concatenation

```
import sys
if len(sys.argv) != 2:
    sys.stderr.write("Usage: %s <n>\n" % sys.argv[0])
    sys.exit(1)
n = 0
string = '@'
while  n  < int(sys.argv[1]):
    string =  string + string
    n += 1
print("String of 2^%d = %d characters created\n" % (n, len(string)));
```

/home/cs2041//public_html/16s2/code/python/echo.0.py (echo.0.py)

Python implementation of /bin/echo

Note this prints an extra space on the end of the line

```
from __future__ import print_function  # Python 2.6+ compatibility

import sys

for arg in sys.argv[1:]:
    print(arg, end=' ')
print()
```

/home/cs2041//public_html/16s2/code/python/echo.2.py (echo.2.py)

Python implementation of /bin/echo

```
from __future__ import print_function # Python 2.6+ compatibility

import sys

if len(sys.argv) > 1:
    print(argv[1], end='')
for arg in sys.argv[2:]:
    print(' ' + arg, end='')
print()
```

/home/cs2041//public_html/16s2/code/python/sum_arguments.py (sum_arguments.py)

sum integers supplied as command line arguments no check that arguments are integers

```
import sys

sum = 0
for arg in sys.argv[1:]:
    sum += int(arg)
print("Sum of the numbers is %s" % sum)
```

/home/cs2041//public_html/16s2/code/python/line_count.0.py (line_count.0.py)

Count the number of lines on standard input.

```
import sys

line_count = 0
for line in sys.stdin:
    line_count += 1
print("%d lines" % line_count)
```

/home/cs2041//public_html/16s2/code/python/line_count.1.py (line_count.1.py)

Count the number of lines on standard input.

```
import sys

lines = sys.stdin.readlines()
line_count = len(lines)
print("%d lines" % line_count)
```

/home/cs2041//public_html/16s2/code/python/line_count.2.py (line_count.2.py)

Count the number of lines on standard input.

```
import sys

print("%d lines" % len(list(sys.stdin)))
```

/home/cs2041//public_html/16s2/code/python/cp.0.py (cp.0.py)

Simple cp implementation using line by line I/O

```
import sys,os
if len(sys.argv) != 3:
    sys.stderr.write("Usage: %s <infile> <outfile>\n" % sys.argv[0])
    # or (Python3 only):
    # print("Usage:",  sys.argv[0], "<infile> <outfile>", file=sys.stderr)
    sys.exit(1)
outfile = open(sys.argv[2], 'w')
for line in open(sys.argv[1]):
    outfile.write(line)
```

/home/cs2041//public_html/16s2/code/python/cp.1.py (cp.1.py)

Simple cp implementation using line by line I/O and with statement

```
import sys,os
if len(sys.argv) != 3:
    sys.stderr.write("Usage: %s <enrollment_file> <course_codes>\n" % sys.argv[0])
    sys.exit(1)
with open(sys.argv[1]) as infile:
    with open(sys.argv[2], 'w') as outfile:
        for line in infile:
            outfile.write(line)
```

/home/cs2041//public_html/16s2/code/python/cp.2.py (cp.2.py)

Simple cp implementation reading file into a list

```
import sys
if len(sys.argv) != 3:
    sys.stderr.write("Usage: %s <enrollment_file> <course_codes>\n" % sys.argv[0])
    sys.exit(1)
lines = open(sys.argv[1]).readlines()
open(sys.argv[2], 'w').writelines(lines)
```

/home/cs2041//public_html/16s2/code/python/cp.3.py (cp.3.py)

Simple cp implementation using line by line I/O

```
import sys,os
if len(sys.argv) != 3:
    sys.stdout.write("Usage: %s <infile> <outfile>\n" % sys.argv[0])
    sys.exit(1)
open(sys.argv[2], 'w').writelines(open(sys.argv[1]))
```

/home/cs2041//public_html/16s2/code/python/cp.4.py (cp.4.py)

Simple cp implementation using shutil.copyfile

```
import sys,shutil
if len(sys.argv) != 3:
    sys.stderr.write("Usage: %s <enrollment_file> <course_codes>\n" % sys.argv[0])
    sys.exit(1)
shutil.copyfile(sys.argv[1], sys.argv[2])
```

/home/cs2041//public_html/16s2/code/python/wget.0.py (wget.0.py)

fetch a web page remove HTML tags, constants, text between script blank lines and print non-empty lines.

There are python libraries which provide a better way to fetch web pages

subprocess.check_output was introduced in Python 2.7.

In older Pythons version you might use: webpage = subprocess.Popen(["wget","-q","-O-",url], stdout=subprocess.PIPE).communicate()[0]

The regex code below doesn't handle a number of cases. It is often better to use a library to properly parse HTML before processing it.

But beware illegal HTML is common & often causes problems for parsers.

```
import sys, re, subprocess
for url in sys.argv[1:]:
    webpage = subprocess.check_output(["wget", "-q", "-O-", url], universal_newlines=True)
    webpage = re.sub(r'(?i)<script>.*?</script>', '', webpage)
    webpage = re.sub(r'(?i)<style>.*?</style>', '', webpage)
    webpage = re.sub(r'&\w+;', ' ', webpage)
    webpage = re.sub(r'<[^>]*>', '', webpage)
    webpage = re.sub(r'\n\s*\n', '\n', webpage)
    sys.stdout.write(webpage)
```

/home/cs2041//public_html/16s2/code/python/wget.1.py (wget.1.py)

fetch a web page remove HTML tags and constants and print non-empty lines

The regex code below doesn't handle a number of cases. It is often better to use a library to properly parse HTML before processing it.

But beware illegal HTML is common & often causes problems for parsers.

```
import re, sys

# urllib package names changed in Python 3
try:
    from urllib import urlopen # Python 2
except ImportError:
    from urllib.request import urlopen # Python 3

for url in sys.argv[1:]:
    response = urlopen(url)
    webpage = response.read().decode()
    webpage = re.sub(r'(?i)<script>.*?</script>', '', webpage)
    webpage = re.sub(r'(?i)<style>.*?</style>', '', webpage)
    webpage = re.sub(r'&\w+;', ' ', webpage)
    webpage = re.sub(r'<[^>]*>', '', webpage)
    webpage = re.sub(r'\n\s*\n', '\n', webpage)
    sys.stdout.write(webpage)
```

/home/cs2041//public_html/16s2/code/python/wget.2.py (wget.2.py)

fetch a web page remove HTML tags and constants using HTML parser BeautifulSoup and print non-empty lines

```
import re, sys
from urllib import urlopen
import BeautifulSoup

# on Python 3 instead do
# from urllib.request import urlopen # Python 3
# import bs4 as BeautifulSoup
# and change BeautifulSoup(webpage) to BeautifulSoup(webpage, "lxml")

def traverse(html):
    if isinstance(html, BeautifulSoup.Tag):
        if html.name in ['style', 'script']:
            return ""
        else:
            return traverse(html.contents)
    elif isinstance(html, list):
        return "".join([traverse(h) for h in html])
    else:
        return html

for url in sys.argv[1:]:
    webpage = urlopen(url).read().decode()
    soup = BeautifulSoup.BeautifulSoup(webpage)
    text = traverse(soup)
    text = re.sub(r'\n\s*\n', '\n', text)
    sys.stdout.write(text)
```

/home/cs2041//public_html/16s2/code/python/snap_memory.py (snap_memory.py)

Reads lines of input until end-of-input

Print snap! if a line has been seen previously

```
import sys
seen = {}
while 1:
    sys.stdout.write("Enter line: ")
    sys.stdout.flush()
    line = sys.stdin.readline()
    if not line:
        break
    if line in seen:
        print("Snap!")
    seen[line] = 1
```

/home/cs2041//public_html/16s2/code/python/count_enrollments.py (count_enrollments.py)

count how many people enrolled in each course

```
import fileinput, re

course_names = {}
for line in open("course_codes"):
    m = re.match(r'(\S+)\s+(.*\S)', line)
    if m:
        course_names[m.group(1)] = m.group(2);

count = {}
for line in fileinput.input():
    course = re.sub(r'\|.*\n', '', line)
    if course in count:
        count[course] += 1
    else:
        count[course] = 1

for course in sorted(count.keys()):
    print("%s has %s students enrolled"%(course_names[course], count[course]))
```

/home/cs2041//public_html/16s2/code/python/count_first_names.py (count_first_names.py)

run as count_first_names.py enrollments count how many people enrolled have each first name

```
import fileinput, re

already_counted = {}
fn = {}
for line in fileinput.input():
    fields = line.split('|')
    student_number = fields[1]
    if student_number in already_counted:
        continue
    already_counted[student_number] = 1
    full_name = fields[2]
    m = re.match(r'.*,\s+(\S+)', full_name)
    if m:
        first_name = m.group(1)
        if first_name in fn:
            fn[first_name] += 1
        else:
            fn[first_name] = 1

for first_name in sorted(fn.keys()):
    print("There are %2d people with the first name %s"%(fn[first_name], first_name))
```

/home/cs2041//public_html/16s2/code/python/course_statistics.py (course_statistics.py)

for each courses specified as arguments print a summary of the other courses taken by students in this course

```
import sys,re

if len(sys.argv) < 3:
    sys.stderr.write("Usage: %s <enrollment_file> <course_codes>\n" % sys.argv[0])
    # or (Python3 only):
    # print("Usage:",  sys.argv[0], "<enrollment_file> <course_codes>", file=sys.stderr)
    sys.exit(1)
enrollment_file = sys.argv.pop(1)

course_names = {}
for line in open("course_codes"):
    line = line.rstrip()
    code = line[0:8]
    name = line[9:]
    course_names[code] = name

courses = {}
names = {}
for line in open(enrollment_file):
    (course,upi,name) = line.split("|")[0:3]
    m = re.match(r'(.*),\s+(.*\S)', name)
    if m:
        name = m.group(2) + " " + m.group(1)
    if upi in courses:
        courses[upi].append(course)
    else:
        courses[upi] = [course]
    names[upi] = name.rstrip()


for course in sys.argv[1:]:
    n_taking = {}
    n_students = 0
    for upi in list(courses.keys()):
        if course not in courses[upi]:
            continue
        n_students += 1
        for c in courses[upi]:
            if c in n_taking:
                n_taking[c] += 1
            else:
                n_taking[c] = 1
    for c in sorted(list(n_taking.keys()), key=lambda x: n_taking[x]):
        print("%5.1f%% of %s students take %s %s"%(100*n_taking[c]/n_students, course, c, course_names[c]))
```

/home/cs2041//public_html/16s2/code/python/duplicate_first_names.py (duplicate_first_names.py)

run as duplicate_first_names.py /home/cs2041/public_html/11s2/lec/perl/examples/enrollments

Report cases where there are multiple people of the same same first name enrolled in a course

```
import fileinput, re

cfn = {}
for line in fileinput.input():
    fields = line.split('|')
    course = fields[0]
    full_name = fields[2]
    m = re.match(r'.*,\s+(\S+)', full_name)
    if not m:
        continue
    first_name = m.group(1)
    if course not in cfn:
        cfn[course] = {}
    if first_name in cfn[course]:
        cfn[course][first_name] += 1
    else:
        cfn[course][first_name] = 1

for course in sorted(cfn.keys()):
    for first_name in sorted(cfn[course].keys()):
        n = cfn[course][first_name]
        if n > 1:
            print("In %s there are %d people with the first name %s"%(course, n, first_name))
```

/home/cs2041//public_html/16s2/code/python/cp.5.py (cp.5.py)

Simple cp implementation by running /bin/cp

```
import sys,subprocess
if len(sys.argv) != 3:
    sys.stderr.write("Usage: %s <enrollment_file> <course_codes>\n" % sys.argv[0])
    sys.exit(1)
subprocess.call(['cp', sys.argv[1], sys.argv[2]])
```

/home/cs2041//public_html/16s2/code/python/echo.1.py (echo.1.py)

Python implementation of /bin/echo

Clumsy but works with any Python version

```
import sys

if len(sys.argv) > 1:
    sys.stdout.write(sys.argv[1])
for arg in sys.argv[2:]:
    sys.stdout.write(' ' + arg)
print()
```

/home/cs2041//public_html/16s2/code/python/echo.3.py (echo.3.py)

Python implementation of /bin/echo

```
import sys

print(' '.join(sys.argv[1:]))
```

/home/cs2041//public_html/16s2/code/python/gender_reversal.0.py (gender_reversal.0.py)

For each file given as argument replace occurrences of Hermione allowing for some misspellings with Harry and vice-versa.

Relies on Zaphod not occurring in the text.

```
import re, sys,os
for filename in sys.argv[1:]:
    tmp_filename = filename + '.new'
    if os.path.exists(tmp_filename):
        sys.err.write("%s: %s already exists\n" % (sys.argv[0], tmp_filename))
        sys.exit(1)
    with open(filename) as f:
        with open(tmp_filename, 'w') as g:
            for line in f:
                line = re.sub(r'Herm[io]+ne', 'Zaphod', line)
                line = line.replace('Harry', 'Hermione')
                line = line.replace('Zaphod', 'Harry')
                g.write(line)
    os.rename(tmp_filename, filename)
```

/home/cs2041//public_html/16s2/code/python/list_student_courses.py (list_student_courses.py)

print the courses being taken by each student enrolled in the specified courses

```
import sys,re
if len(sys.argv) < 3:
    sys.stdout.write("Usage: %s <enrollment_file> <course_codes>\n" % sys.argv[0])
    sys.exit(1)
enrollment_file = sys.argv.pop(1)

course_names = {}
for line in open("course_codes"):
    line = line.rstrip()
    code = line[0:8]
    name = line[9:]
    course_names[code] = name

courses = {}
names = {}
for line in open(enrollment_file):
    (course,upi,name) = line.split("|")[0:3]
    m = re.match(r'(.*),\s+(.*\S)', name)
    if m:
        name = m.group(2) + " " + m.group(1)
    if upi in courses:
        courses[upi] += " " + course
    else:
        courses[upi] = course
    names[upi] = name.rstrip()


for course in sys.argv[1:]:
    for upi in list(courses.keys()):
        student_courses = courses[upi].split()
        if course not in student_courses:
            continue
        print("%s is taking"%(names[upi]))
        for course in student_courses:
            print("%s %s"%(course, course_names[course]))
```

/home/cs2041//public_html/16s2/code/python/nth_word.py (nth_word.py)

Print the nth word on every line of input files/stdin output is piped through fmt to make reading easy

```
from __future__ import print_function # Python 2.6+ compatibility

import fileinput, sys, subprocess

if len(sys.argv) < 2:
    sys.stdout.write("Usage: %s <n>\n" % sys.argv[0])
    sys.exit(1)

nth_word = int(sys.argv.pop(1))
p = subprocess.Popen(["fmt","-w","40"], stdin=subprocess.PIPE, universal_newlines=True)
for line in fileinput.input():
    words = line.rstrip().split()
    if len(words) > nth_word:
        print(words[nth_word], file=p.stdin)
p.stdin.close()
```

/home/cs2041//public_html/16s2/code/python/print_odd.py (print_odd.py)

5 different ways to print the odd numbers in a list - illustrating various aspects of Python

simple for loop

```
def print_odd0(list):
    for element in list:
        if element % 2 :
            print(element)

# list comprehension
def print_odd1(list):
    odd = [x for x in list if x % 2]
    for element in odd:
        print(element)

def odd(x):
    return x % 2

# filter+helper function
def print_odd2(list):
    for element in filter(odd, list):
        print(element)

# filter+lambda expression
def print_odd3(list):
    for element in [x for x in list if x % 2]:
        print(element)

# join+map+filter+helper function
def print_odd4(list):
    print("\n".join(map(str, filter(odd, list))))

a = list(range(1, 10))
for version in range(0, 5):
    print("print_odd%s"%version)
    eval("print_odd%s(a)"%version)
```

/home/cs2041//public_html/16s2/code/python/snap_consecutive.py (snap_consecutive.py)

Reads lines of input until end-of-input

Print snap! if two consecutive lines are identical

```
import sys
sys.stdout.write("Enter line: ")
sys.stdout.flush()
last_line = sys.stdin.readline()
while 1:
    sys.stdout.write("Enter line: ")
    sys.stdout.flush()
    line = sys.stdin.readline()
    if not line:
        break
    if line == last_line:
        print("Snap!")
    last_line = line
```

/home/cs2041//public_html/16s2/code/python/sort_dates.py (sort_dates.py)

```
import re
from random import randint

def random_date():
    return "%02d/%02d/%04d"%(randint(1,28), randint(1,12), randint(2000,2020))

def parse_date(date1):
    (day, month, year) = re.split(r'\D+', date1)
    return (year, month, day)

random_dates = [random_date() for x in range(0,5)]
print("random_dates: " + ','.join(random_dates))
sorted_dates = sorted(random_dates, key=parse_date)
print("sorted dates: " + ','.join(sorted_dates))
```

/home/cs2041//public_html/16s2/code/python/sort_days0.py (sort_days0.py)

```
import random

days = {'Sunday':0, 'Monday':1, 'Tuesday':2, 'Wednesday':3,
         'Thursday':4, 'Friday':5, 'Saturday':6}

def random_day():
    return str(random.choice(list(days.keys())))

def day_number(day):
    return days[day]

random_days = [random_day() for x in range(0,5)]
print("random_days = " + ','.join(random_days))

sorted_days = sorted(random_days, key=day_number)
print("sorted_days = " + ','.join(sorted_days))
```

/home/cs2041//public_html/16s2/code/python/sort_days1.py (sort_days1.py)

```
import random

day_names = "Sunday Monday Tuesday Wednesday Thursday Friday Saturday"
days = dict(list(zip(day_names.split(), list(range(0,7)))))

random_days = [random.choice(list(days.keys())) for x in range(0,5)]
print("random_days = " + ','.join(random_days))

sorted_days = sorted(random_days, key=lambda x:days[x])
print("sorted_days = " + ','.join(sorted_days))
```

/home/cs2041//public_html/16s2/code/python/split_join.py (split_join.py)

simple implementations of re.split & str.join to exhibit various python features

```
import re
from functools import reduce

def my_join1(separator, list):
    if not list:
        return ""
    string = str(list.pop(0))
    for element in list:
        string += separator + str(element)
    return string

def my_join2(separator, list):
    return reduce(lambda x, y:x+separator+y, map(str, list))

# using (tail) recursion
def my_split1(regexp, string):
    m = re.match(r'(.*?)%s(.*)'%regexp, string)
    if m:
        return [m.group(1)] + my_split1(regexp, m.group(2))
    else:
        return [string]

# more concise but perhaps less reable version using Python' ternary operator
def my_split2(regexp, string):
    m = re.match(r'(.*?)%s(.*)'%regexp, string)
    return [m.group(1)] + my_split1(regexp, m.group(2)) if m else [string]

# iterative form
def my_split3(regexp, string):
    list = []
    pattern = re.compile(r'(.*?)%s(.*)'%regexp)
    while 1:
        m = pattern.match(string)
        if m:
            list.append(m.group(1))
            string = m.group(2)
        else:
            list.append(string)
            return list

a = my_join1("+", list(range(1,11)))
print(eval(a))

b = my_join2("+", list(range(1,11)))
print(eval(b))

print(my_split1(",", "1,2,3,4,5,6,7,8,9,10"));
print(my_split2(",", "1,2,3,4,5,6,7,8,9,10"));
print(my_split3(",", "1,2,3,4,5,6,7,8,9,10"));
```