SI 506: Programming I
Fall 2019
Lecture 04

Anthony Whyte <arwhyte@umich.edu>
Lecturer III, School of Information
715 N. University Ave, Ann Arbor, MI 48109
Roumanis Square, 2nd floor ("the loft")

UMSI

preliminaries

UMSI

Office Hours

arwhyte

Friday, 11:30 am - 1:00 PM
NQ 3330

Starts 20 Sept 2019
(next week)

UMSI

Submissions (as of 1:30 PM)

Scores pushed to Canvas next Tuesday

gradescope

SI 506 2019 Fall

Programming I

- Dashboard
- Assignments
- Roster
- Course Settings

INSTRUCTOR

Anthony Whyte

SI 506 2019 Fall | Fall 2019

DESCRIPTION

Introduction to programming with a focus on applications informatics. Covers the fundamental elements of a modern programming language and how to access data on the internet.

THINGS TO DO

- Review and publish grades for Problem Set 01 now that you're all done grading.
- Review and publish grades for Lab Exercise 02 now that you're all done grading.

ACTIVE ASSIGNMENTS	RELEASED	DUE (EDT)	SUBMISSIONS	% GRADED	PUBLISHED	REGRADES
Problem Set 01	SEP 10	SEP 16 AT 11:59PM	153	100%		
Lab Exercise 02	SEP 10	SEP 16 AT 11:59PM	252	100%		

UMSI

Assignment due dates

weekly problem sets and lab exercises

Available

Tuesday, 4:00 PM Eastern

Submission due

following Monday by11:59 PM Eastern

UMSI

Get ready to code

Start your Python console

Send feedback Forums Help Blog Account Log out

pythonanywhere

Dashboard Consoles Files Web Tasks Databases

Dashboard

Welcome, nantin

CPU Usage: 1% used – 1.13s of 100s. Resets in 22 hours, 29 minutes **More Info**

File storage: 0% full – 148.0 KB of your 512.0 MB quota

Upgrade Account

Recent
Consoles + 5 –

You have no recent consoles.

New console:

$ Bash >>> Python

More...

Recent
Files + 5 –

You have no recently edited files.

+ Open another file

Browse files

Recent
Notebooks + 5 –

Your account does not support Jupyter Notebooks. Upgrade your account to get access!

All
Web apps

You don't have any web apps.

Open Web tab

UMSI

Python console

write/execute Python code (only)

Python3.7 console 13351686

Share with others

```
Python 3.7.0 (default, Aug 22 2018, 20:50:05)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import json
>>> console = 'command line interpreter'
>>> purpose = 'accept user input in the form of Python code and attempt to execute it.'
>>> use = 'typically used for quick prototyping and exploration of the language (i.e., teaching).'
>>> data = {}
>>> data['console'] = console
>>> data['purpose'] = purpose
>>> data['use'] = use
>>> json_data = json.dumps(data)
>>> print(json_data)
{"console": "command line interpreter", "purpose": "accept user input in the form of Python code and attempt to execute i
t.", "use": "typically used for quick prototyping and exploration of the language (i.e., teaching)."}
>>>
```


UMSI

Unix shell (Bash)

interact with operating system, issue commands, run scripts

Bash console 13351749

Share with others

```
01:43 ~ $ pwd
/home/arwhyte
01:43 ~ $ ls
README.txt  SI506
01:43 ~ $ cd SI506
01:44 ~/SI506 $ ls -la
total 16
drwxrwxr-x 4 arwhyte registered_users 4096 Sep  5 04:14 .
drwxrwxr-x 5 arwhyte registered_users 4096 Sep  5 22:01 ..
drwxrwxr-x 2 arwhyte registered_users 4096 Sep  5 02:28 lab_exercises
drwxrwxr-x 2 arwhyte registered_users 4096 Sep  2 00:43 problem_sets
01:44 ~/SI506 $ cd lab_exercises
01:44 ~/SI506/lab_exercises $ ls -la
total 12
drwxrwxr-x 2 arwhyte registered_users 4096 Sep  5 02:28 .
drwxrwxr-x 4 arwhyte registered_users 4096 Sep  5 04:14 ..
-rw-rw-r-- 1 arwhyte registered_users 1483 Sep  5 02:28 si506_lab_01.py
01:44 ~/SI506/lab_exercises $ python3 si506_lab_01.py arwhyte
Huzzah! arwhyte writes first Python program at 2019-09-11T21:44:51.572295-04:00
01:44 ~/SI506/lab_exercises $
```


UMSI

Lab attendance

Piazza Q&A

"If we completed the lab set do we have to attend the lab?"

Lab attendance

small group learning

lab section != lab exercise

- Ask Questions
- Discuss lecture topics
- GSI demos
- Practice coding
- Do lab exercise (extra credit)
- Start problem set
- Help classmates (learn by teaching)

UMSI

Gentle Reminder

premature disclosure is unhelpful

Don’t post solutions

UMSI

Piazza: programming questions

encourage self-discovery when replying

[Dr Chuck, P4E] Chapter 2, Exercise 3
Exercise 3: Write a program to prompt the user for hours and rate per hour to compute gross pay.
Enter Hours: 35
Enter Rate: 2.75
Pay: 96.25

MY CODE:

```
hours = input('Enter hours:')
rate = input('Enter rate:')

print(hours * rate)
```

Why am I receiving an error? I know it has something to do with the word "rate" in print. Thanks in advance!

Lost opportunity 1: go read the docs

returns string

input([prompt])

If the prompt argument is present, it is written to standard output without a trailing newline. The function then reads a line from input, converts it to a string (stripping a trailing newline), and returns that. When EOF is read, EOFError is raised.

Source: https://docs.python.org/3/library/functions.html#input

Lost opportunity II: go read the docs

recast to type float

float([x])

Return a floating point number constructed from a number or string x.

If the argument is a string, it should contain a decimal number, optionally preceded by a sign, and optionally embedded in whitespace. . . .

Source: https://docs.python.org/3/library/functions.html#float

UMSI

new stuff continued

Problem set: instruction text

consider the object type to which the instructions refer

```
# Use the appropriate operator to append (i.e., concatenate) Guido's current position
# at the Python Software Foundation (see https://www.python.org/psf/members/#officers)
# to the value assigned to the variable you chose you in Problem 01A. Assign
# the concatenated value to the variable python_foundation_officer.
```

“prepend” / “append”

y before x

y after x

For Problem Set 01, the object in question is a string *not* a list so attempt to use .append() with a string will trigger a runtime error.

UMSI

Python string concatenation

add, append, concatenate uniqname to nick_name

```
$ python3
>>> nick_name = 'Dr Chuck'
>>> uniqname = 'csev'
>>> name = nick_name + ', ' + uniqname
>>> print(name)
>>> Dr Chuck, csev
```

See: https://realpython.com/python-string-formatting/

Problem set instructions: <…>

placeholder convention

```
# Adopt the following format for
the new string:

# "<name>, President"
```

positional requirement with hint

(replace <…> with value)

UMSI

Python placeholders (string formatting)

old school %s, %d, etc., placeholders

```
$ python3
>>> x = "Hello %s %d" % ('SI', 506)
>>> print(x)
Hello SI 506
```

See: https://realpython.com/python-string-formatting/
https://docs.python.org/3/library/stdtypes.html#str.format
https://python-reference.readthedocs.io/en/latest/docs/str/formatting.html

UMSI

Python str.format() placeholders

more placeholder conventions

```
$ python3
>>> nick_name = 'Dr Chuck'
>>> uniqname = 'csev'
>>> name = "{0}, {1}".format(nick_name,
                             uniqname)
>>> print(name)
>>> Dr Chuck, csev
```

See: https://realpython.com/python-string-formatting/

UMSI

Built-in function: str.join()

concatenate strings with an iterable (say what?)

str.join(iterable)

Return a string which is the concatenation of the strings in iterable. A TypeError will be raised if there are any non-string values in iterable, including bytes objects. The separator between elements is the string providing this method.

Source: https://docs.python.org/3/library/stdtypes.html#str.replace

Iterable?: better check stackoverflow

wisdom of the crowd (with limits)

http://bit.ly/2kCbgXL

UMSI

Spam, Spam, Spam

little blue can of protein …

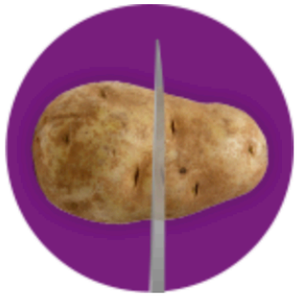
Six Simple Ingredients

Ah, the age-old question; what is the meat in that special can of SPAM® Classic? Many myths abound, but the answer is actually quite simple.

PORK WITH HAM

SALT

WATER



POTATO STARCH

SUGAR

SODIUM NITRITE

UMSI

Built-in function: str.join()

interactive shell (type this)

```
$ python3
>>> s = 'spam spam spam'
>>> print(s)
>>> t = s + ' ' + s # operator concatenation
>>> print(t)
>>> t = ''.join([s, ' ', s]) # join to empty string
>>> print(t)
>>> u = ''.join([t, ' baked beans ', s, ' + spam'])
>>> print(u) # also on the menu (Spam sketch)
```

See: https://en.wikipedia.org/wiki/Spam_(Monty_Python)

UMSI

Monty Python sketch: Spam

proof: cafe menu

See: https://en.wikipedia.org/wiki/Spam_(Monty_Python)

UMSI

Built-in function: str.splitlines()

splits multi-line string at line boundaries

str.splitlines([keepends])

Return a list of the lines in the string, breaking at line boundaries. Line breaks are not included in the resulting list unless keepends is given and true.

splits on: \n (line feed), \r (carriage return), \r\n, etc.

Source: https://docs.python.org/3/library/stdtypes.html?highlight=splitlines#str.splitlines

UMSI

Multi-line strings: victors.txt

Stored in Canvas

http://bit.ly/2meFQqV

UMSI

Multi-line strings

use double quotations x 3

```
$ python3
>>> victors = """Hail! to the victors valiant
... Hail! to the conqu'ring heroes
... Hail! Hail! to Michigan
... the leaders and best"""
>>> print(victors)
```


UMSI

Built-in function: str.splitlines()

interactive shell (type this)

```
$ python3
>>> lines = victors.splitlines()
>>> print(lines)
['Hail! to the victors valiant', "Hail! to the
conqu'ring heroes", 'Hail! Hail! to
Michigan', 'the leaders and best’]
>>> len(lines)
4
```

See: https://en.wikipedia.org/wiki/Spam_(Monty_Python)

UMSI

for loop

interactive shell (type this)

```
$ python3
>>> for line in lines:
...     print(line) # indent 4 spaces / tab
...
>>> for line in lines:
...     print(line.replace('Hail!', 'Huzzah!'))
...
```

See: https://en.wikipedia.org/wiki/Spam_(Monty_Python)

UMSI

Next week

topics

- string slicing
- lists
- accumulator patterns
- conditional expressions

finis

directors cut

UMSI

Built-in functions

interactive shell (type this)

```
$ python3
>>> y = input("Type 'spam' 3 times: ")
>>> print(y)
>>> type(y)
>>> y.__class__ # just to confirm
>>> z = y
>>> print(y, z, sep=' wonderful ', end='\n\n\n')
>>> len(z)
>>> type(len(z)) # function as argument
```

Built-in function: str.split()

interactive shell (type this)

```
$ python3
>>> s = 'spam,spam,spam'
>>> s.split()
>>> s.split(',')
>>> s.split(sep=',', maxsplit=1)
>>> s = 'spam spam spam'
>>> p = s.split()
>>> type(p)
>>> len(p)
```

Built-in functions

interactive shell (type this)

```
$ python3
>>> e = s.replace('spam', 'egg')
>>> print(e)
>>> print(s)
>>> b = s.replace('spam', 'egg bacon +', 1)
>>> print(b)
>>> m = s[:9].replace('spam', 'egg bacon +', 1)
>>> print(m) # actually on the menu (Spam skit)
```

See: https://en.wikipedia.org/wiki/Spam_(Monty_Python)

UMSI

Variable

naming do's and don'ts

Good

```
course_short_name = 'SI506'
course_list = ['SI506', 'SI507', 'SI508']
```

Bad

```
c = 'SI506' # opaque (unfriendly)
courseList = ['SI506', 'SI507', 'SI508'] # camelcase
```

Ugly (interpreter will complain)

```
506_umsi = 'SI506'
$number = 506
course-list = ['SI506', 'SI507', 'SI508']
course name = 'SI506'
```


UMSI

Keywords

reserved: cannot be used as ordinary identifiers

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Source: *https://docs.python.org/3/reference/lexical_analysis.html?highlight=reserved%20keywords#keywords*

Arithmetic operators

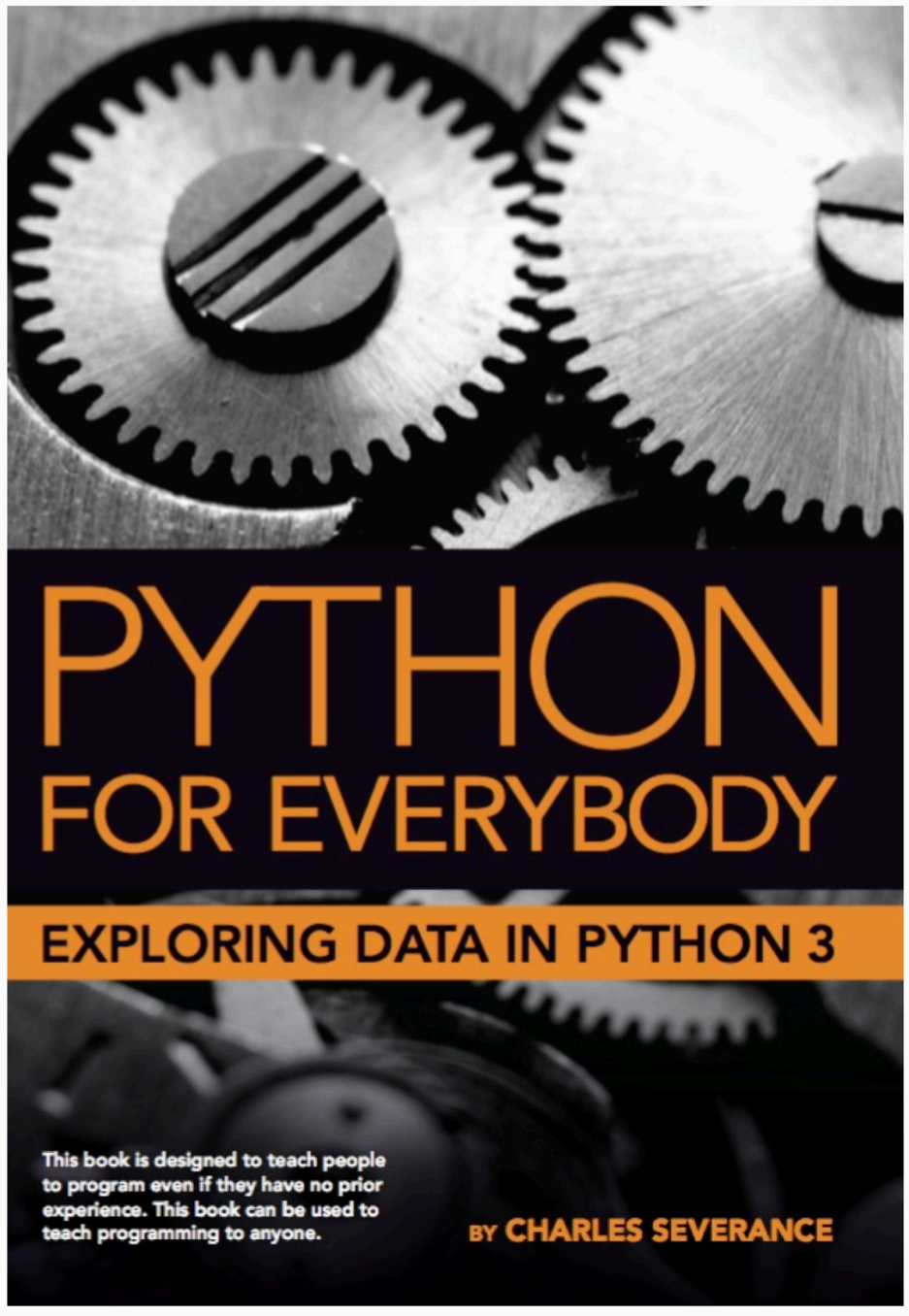
operations on operands x and y

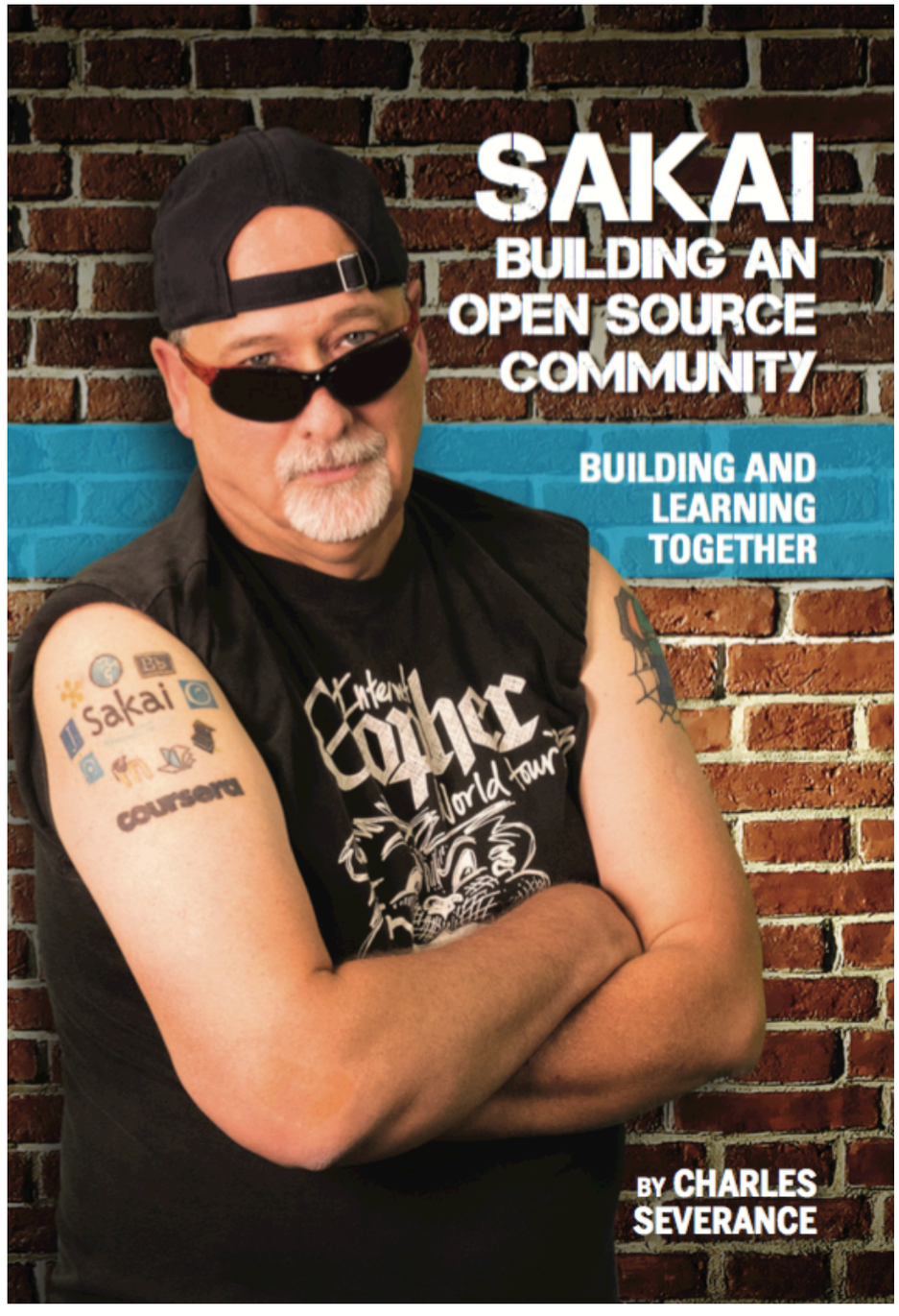
Operator	Description	Example
+	Addition	x + y = 506
-	Subtraction	x - y = 494
*	Multiplication	x * y = 3000
/	Division	x / y = 83.33
%	Modulus (divides x by y and returns remainder)	x % y = 2
**	Exponent (exponential power calculation)	x**y = 15625000000000000
//	Floor division (result rounded to whole number)	x // y = 83

Q: what are the values assigned to x and y?

Readings: Dr Chuck

If new to programming, start with P4E chapters





UMSI