SI 506: Programming I
Fall 2019
Lecture 02

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

Override requests

UMSI registrar contacted

no email from me,

see me

Approval contingent upon open seat(s)
in a lab section that fits your schedule.

SI 506 is scheduled to be offered during Winter Semester 2020.
Winter enrollment ~ 40 students.

UMSI

Infrastructure

Q: account creation/activation priorities

Do now

1. Create/activate Python Anywhere account
2. Create/activate Piazza account
3. Create/activate O'Reilly Learning Platform account

Do post-midterm (or now)

4. Create/activate Github account

UMSI

Assignments

Q: are multiple submissions allowed?

Yes

lab exercise

UMSI

Lab Submissions

notifications filling my inbox

Huzzah!

Merriam-Webster: *an expression or shout of acclaim—often used interjectionally to express joy or approbation*. First known use: 1573. A cheer associated with eighteenth-century sailors, not aspiring twenty-first-century programmers.

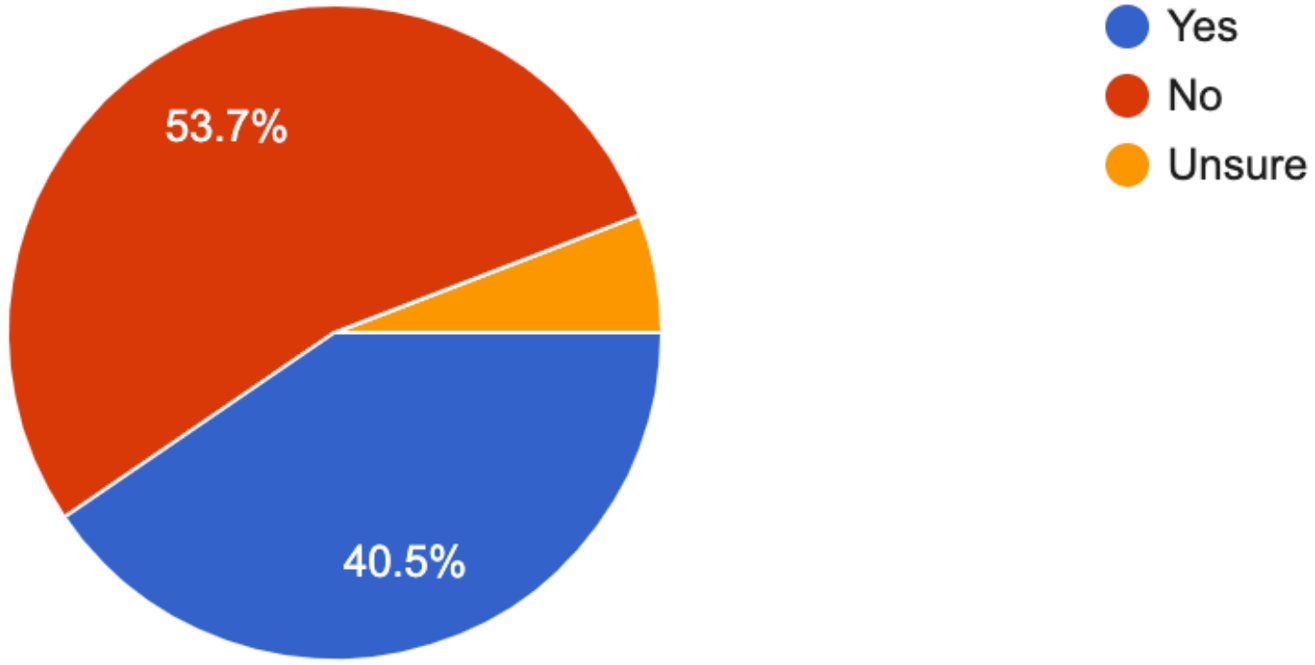
UMSI

On Tuesday

Unix shell usage

Have you ever used a command line interface (CLI) such as Terminal.app (MacOS) or cmd.exe (Windows) to navigate a file system, interact with applications (e.g., start up/shutdown), or run shell scripts?

257 responses



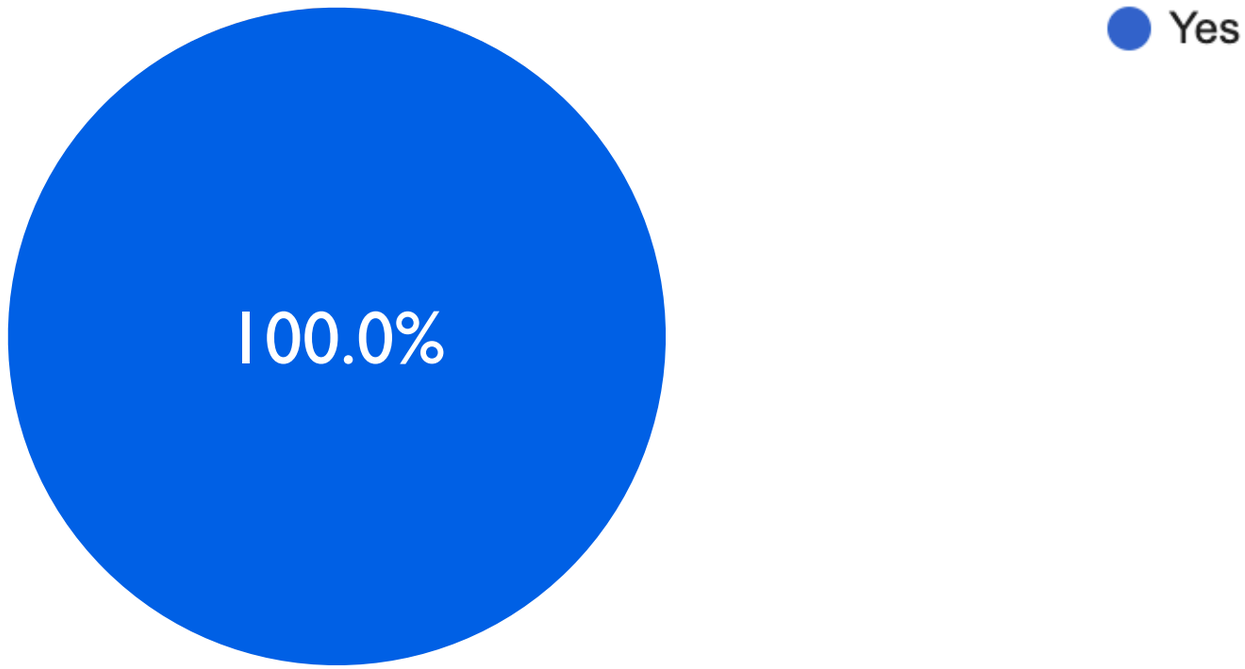
UMSI

On Thursday

A mere 48 hours later . . .

Have you ever used a command line interface (CLI) such as Terminal.app (MacOS) or cmd.exe (Windows) to navigate a file system, interact with applications (e.g., start up/shutdown), or run shell scripts?

257 responses



UMSI

vocabulary & concepts

UMSI

Python Anywhere

Log in please

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

program

UMSI

Program
formal definition

"a sequence of instructions that specifies a computation"

Charles Severance, *Python for Everybody*

UMSI

Program

informal "baking bread" analogy

Implementation of a "recipe" designed to solve a task

Ana Bell, *Get Programming: Learn to Code with Python*

UMSI

script

UMSI

Python script
definition

Text file* containing Python code that is designed to be executed directly by a user.

* designated with a .py extension

UMSI

module

Python module
definition

Text file containing Python code that is designed to be imported and used by another Python file.

* designated with a .py extension

UMSI

si506_lab_01.py

script anatomy and workflow

load

```
import ...
```
module imports

instructions

entry point

```
main(argv)
```
process, delegate, print

```
huzzah(name)
```
delegate, build, format

```
umich_now()
```
transform

invoke

```
if __name__ == '__main__':
    main(sys.argv[1:]) # do something
```


UMSI

object

Python object
definition

"Objects are Python's abstraction for data. All data in a Python program is represented by objects or by relations between objects."

Source: https://docs.python.org/2.0/ref/objects.html

Python object

the stuff of an object

Identity

Type

Value

Source: https://docs.python.org/2.0/ref/objects.html

UMSI

Console exercise

interactive shell (type this)

```
$ python3
. . .
>>> 506
>>> type(506)
>>> num = 506
>>> type(num)
>>> num = ‘Five O Six’
>>> type(num)
```


UMSI

Built-in Types

a.k.a integer, float, boolean, string, list, dictionary, tuple, class, etc.

- numerics
- sequences
- mappings
- classes
- instances
- exceptions

Memorize this

Python is a dynamically-typed language

Types are associated with objects *not* variables

Java
statically-typed language

```
public abstract class AbstractEntity implements CaliperEntity, CaliperCoercible {

    protected final String id;

    private final CaliperEntityType type;

    private final String name;

    private final String description;

    private final DateTime dateCreated;

    private final DateTime dateModified;

    private final Map<String, Object> extensions;

    . . .
}
```

Variable

a name that refers to a value

Rules: variable names are composed of letters, numbers, and underscore (_) *only*.

- no hyphens or other special characters are permitted.
- no whitespace between characters is permitted.
- variable names cannot begin with a number.

Style: per *PEP 8* variable names should be *lowercase* with words separated by underscores (_) to improve readability.

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*

Console exercise

interactive shell (type this)

```
$ python3
. . .
>>> class = ‘SI 506’
```

statement

Statement
Definition

A statement is a unit of code that the Python interpreter can evaluate

Statements

interactive shell (type this)

```
$ python3
. . .
>>> uniqname = ‘arwhyte’
>>> print(uniqname)
>>> uniqname = ‘csev’
>>> print(uniqname)
```


UMSI

expression

Expression
Definition

An expression is a combination of values, variables, and operators

UMSI

operators and operands

(a brief intro)

UMSI

Operators

Types

- arithmetic
- comparison
- assignment
- logical (and, or, not)
- membership (in, not in)
- identity (is, is not)
- bitwise

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?

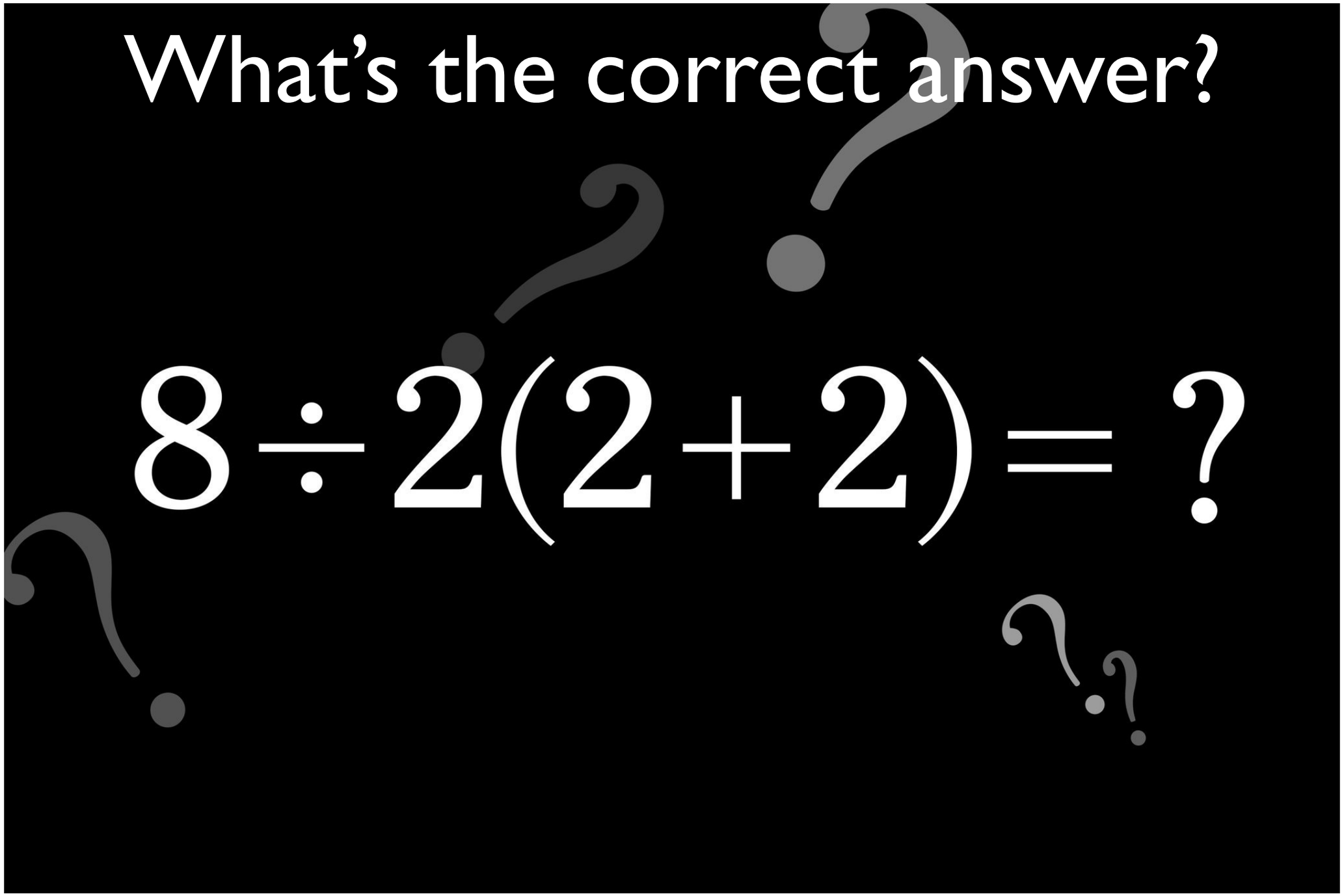
operator precedence

(matters)

UMSI

How will Python handle this equation?

debate generated a recent tweet storm



UMSI

Console exercise

interactive shell (type this)

```
$ python3
. . .
>>> 8 / 2(2 + 2)
```

The answer?

NY Times article 01

Steven Strogatz: "The standard convention holds that multiplication and division have equal priority. To break the tie, we work from left to right. So the division goes first, followed by the multiplication. Thus, the right answer is 16."

```
$ python3
...
>>> 8 / 2 (2+2)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not callable
>>> 8 / 2 * (2 + 2)
16.0
```

Pushback

NYT article 02: implicit multiplication given higher priority

Steven Strogatz: " . . .the implicit multiplication in 2(2 + 2) is given higher priority than the explicit division in 8÷2(2 + 2). In other words, 2(2+2) should be evaluated first. Doing so yields 8÷2(2 + 2) = 8÷8 = 1. By the same rule, many commenters argued that the expression 8÷2(4) was not synonymous with 8÷2×4, because the parentheses demanded immediate resolution, thus giving 8÷8 = 1 again."

```
$ python3
...
>>> 8 / (2 * (2 + 2))
1.0
```

NY Times

three articles on this tweet storm debate

https://www.nytimes.com/2019/08/02/science/math-equation-pedmas-bemdas-bedmas.html

https://www.nytimes.com/2019/08/05/science/math-equation-pemdas-bodmas.html

https://www.nytimes.com/2019/08/06/science/math-equation-pemdas.html

UMSI

and now for something completely different

Zen of Python

type this

```
$ python3
. . .
>>> import this
```