CS521 O2
Information Structures with Python

Lecture 1

Guanglan Zhang

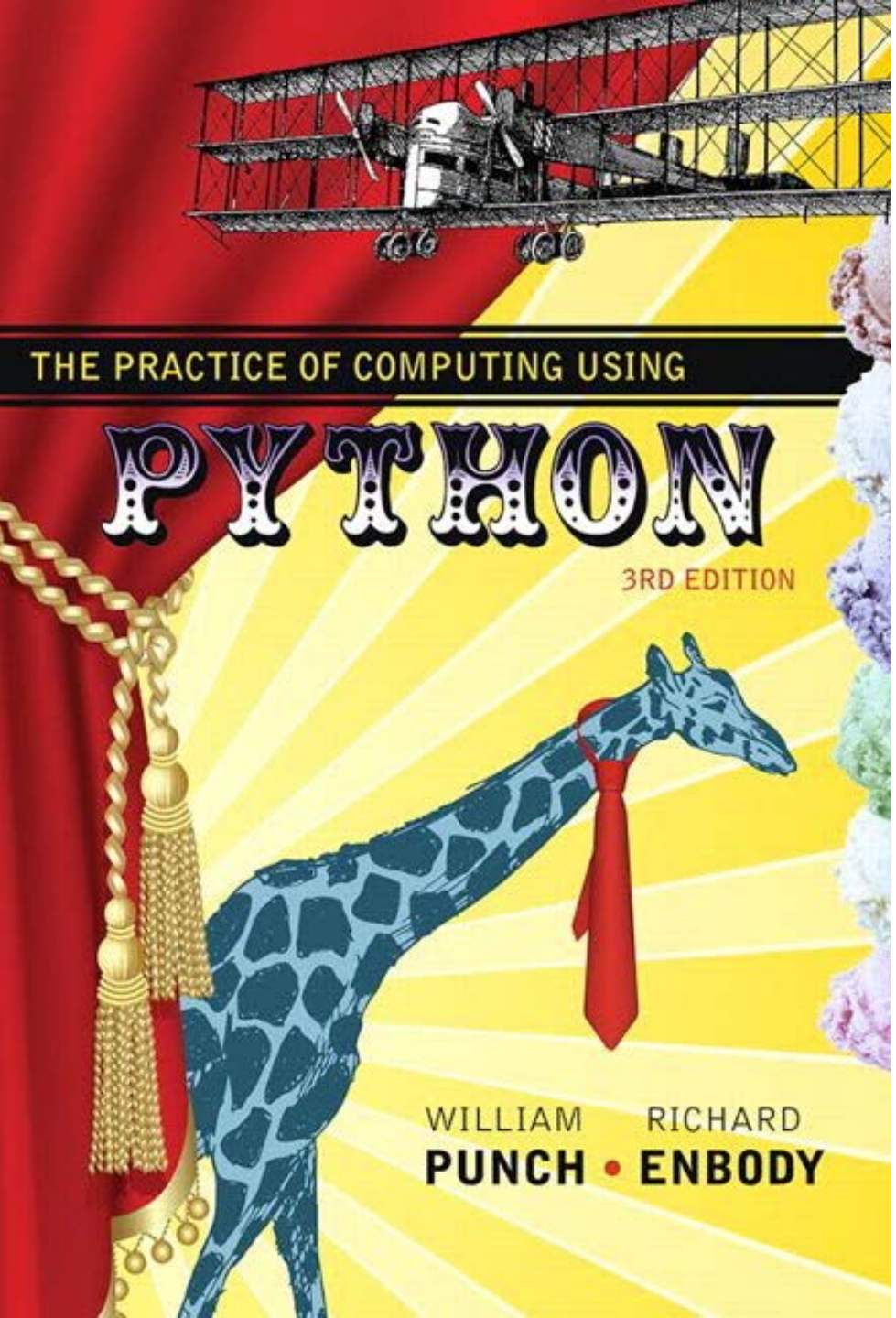
guanglan@bu.edu

Some slides adapted from Prof. Eugene Pinsky and some from Brian Gregor (BU research computing)

Table of Content

Course Learning Objectives

- Learn to use Python programming language constructs to implement a variety of analytical and computational methods
- Understand trade-offs of different Python methods and data structures in computation
- Apply acquired skills in diverse settings by completing a course project
- Present both symbolic and visual results on the project
- Learn advantages and limitations of using Python



Required textbook

Punch, W. and Enbody, R. (2016). *The Practice of Computing Using Python* (3rd ed.). Pearson.

ISBN-13: 978-0-13-437976-0

This book can be purchased from Barnes and Noble at Boston University. An e-book is available at Vitalsource.com.

Note: You do not need to purchase the textbook "with access," also referred to as the "lab" portion of this text. It will not be used in this course.

Grading Structure

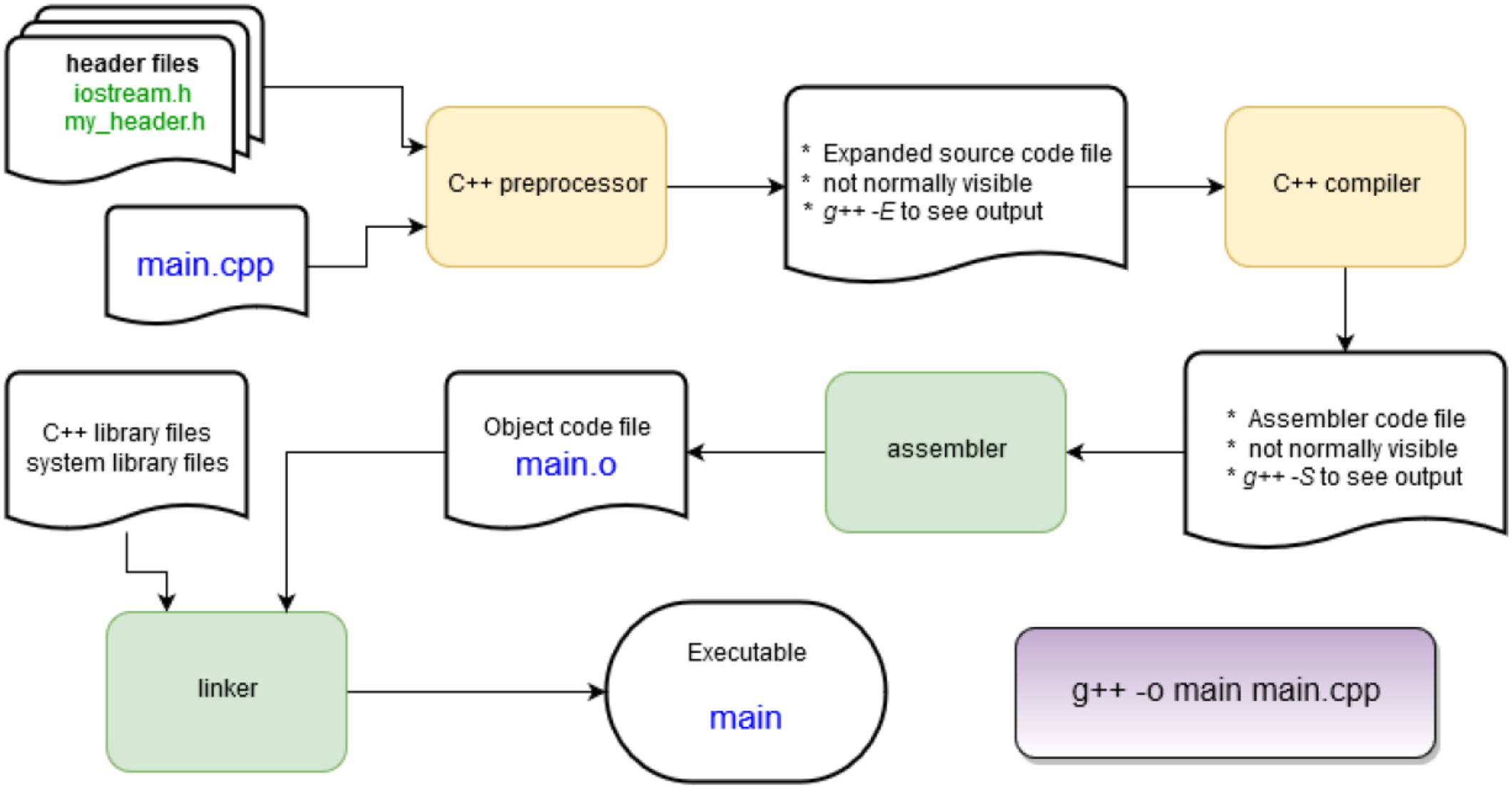
- Six 30-min online quizzes
- Six assignments
- A course project
- A proctored close-book final exam

Deliverable	Weight
Quizzes	15%
Assignments	35%
Project	20%
Final Exam	30%

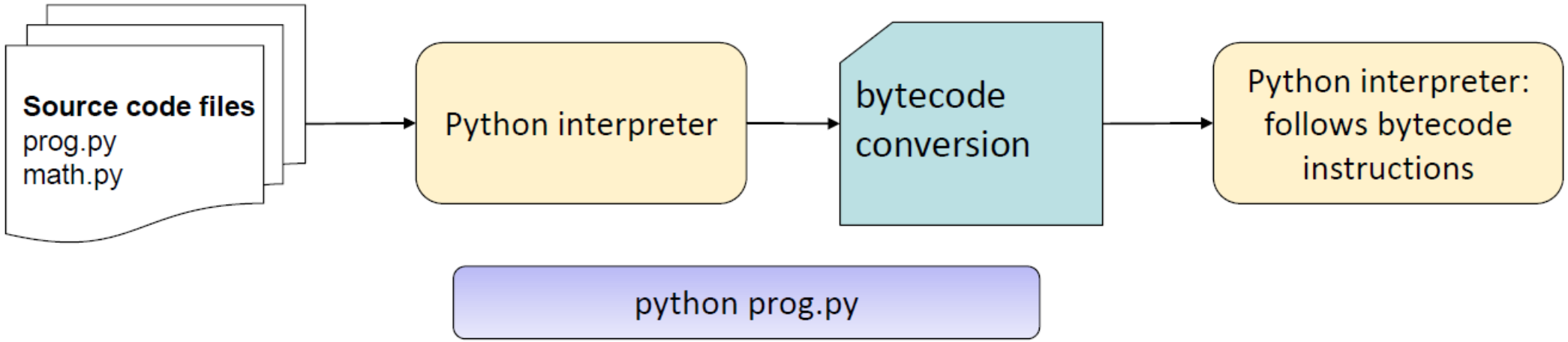
Resources

- Our required textbook and Blackboard lecture material
- http://www.pythontutor.com/visualize.html - run simple Python programs and visualize the execution. We will use it in class.
- https://docs.python.org/2/tutorial
- https://www.tutorialspoint.com/python
- https://www.learnpython.org
- https://www.python.org/community/sigs/current/edu-sig/ - contains links to learning resources, including free books
- Ask questions in Blackboard site -> Class Discussion -> General course questions/comments
- Email your facilitators or me

Compiled Languages (ex. C++ or Fortran)



Interpreted Languages (ex. Python or R)



- Clearly, a lot less work is done to get a program to start running compared with compiled languages!
- Bytecodes are an internal representation of the text program that can be efficiently run by the Python interpreter.
- The interpreter itself is written in C and is a compiled program.

Comparison

Interpreted

- Faster development
- Easier debugging
  - Debugging can stop anywhere, swap in new code, more control over state of program
- (almost always) takes less code to get things done
- Slower programs
  - Sometimes as fast as compiled, rarely faster
- Less control over program behavior

Compiled

- Longer development
  - Edit / compile / test cycle is longer!
- Harder to debug
  - Usually requires a special compilation
- (almost always) takes more code to get things done
- Faster
  - Compiled code runs directly on CPU
  - Can communicate directly with hardware
- More control over program behavior

BOSTON UNIVERSITY

Python is

- An open-source general purpose interpreted programming language
- Freely usable and distributable, even for commercial use
- a language that supports object-oriented programming
- a language that is dynamically typed
- a language that provides automatic memory management and garbage collection

A simple example program

- A program is a human-readable essay on problem solving that also execute on a computer

- A simple program
  - ✓ Gets input(s)
  - ✓ Performs some computation(s)
  - ✓ Outputs result(s)

```
# Calculate the area of a square
in_str = input("Enter a numeric value as the side of a square:")
x = float(in_str)
y = x**2
print(“Area of the square is", y)
```

A simple example program (cont.)

- The input is a string
- Need to cast it into float
- When the user input a letter, an error showed up “could not convert string to float
- Need to improve the program

Print output (drag lower right corner to resize)

```
Enter a numeric value as the side of a squar
Area of the square is 25.0
```

Frames Objects

Global frame	
in_str	"5"
x	5.0
y	25.0

```
Enter a numeric value as the side of a square:a
<class 'str'>
Traceback (most recent call last):

  File "<ipython-input-23-e52153af90be>", line 5, in <module>
    x = float(in_str)

ValueError: could not convert string to float: 'a'
```

A simple example program (cont.)

```
in_str = input("Enter a numeric value as the side of a square:")
if in_str.isnumeric() is True:
    x = float(in_str)
    y = x**2
    print("Area of the square is", y)
else:
    print("Wrong input. Cannot cast: ", in_str)
```

```
Enter a numeric value as the side of a square:a
Wrong input. Cannot cast:  a
```

Python operators

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Exponentiation
//	Floor division (rounds the result down to the nearest whole number)

Assignment Operator	Example
=	x = 1
+=	x += 1 same as x = x+1
-=	x -= 1 same as x = x-1
*=	x *= 2 same as x = x*2
/=	x /= 2 same as x = x/2
**=	x **= 2 same as x = x**2
//=	x //= 2 same as x = x//2

Python program structure

- Python programs have extension .py
- There are two ways to run a program
  - ✓ Shell mode – “interactive” use

```
In [28]: in_str = input("Enter a numeric value as the side of a square:")

Enter a numeric value as the side of a square:45

In [29]: if in_str.isnumeric() is True:
    ...:     x = float(in_str)
    ...:     y = x**2
    ...:     print("Area of the square is", y)
    ...: else:
    ...:     print("Wrong input. Cannot cast: ", in_str)
    ...:
Area of the square is 2025.0
```

  - ✓ Script mode – save code in a file and run the file

```
In [25]: runfile('C:/Users/guanglan/Dropbox/BUwork/Python/CS521/square_area.py', wdir='C:/
Users/guanglan/Dropbox/BUwork/Python/CS521')

Enter a numeric value as the side of a square:45
Area of the square is 2025.0
```

Python conventions and syntax

- A module is a Python file containing a set of Python commands to solve some problems

- A module can be *imported* into the Python shell
  Example: > import math

- Python differentiates code into two categories
  - ✓ Expression: a combination of values and operations that creates a new value that we call a return value
    Example: > y = x + 5

  - ✓ Statement: performs some task, but does not return a value
    Example: > print(‘Hello’)

  - ✓ Some statements may have a side effect, which is some change that results from executing the statement
    Example: > x = 5

Python conventions and syntax (cont.)

- Each statement ends with newline
- Or split a long statement into multiple line and indicate continuation by placing a backslash “\” at the end of a line
- When putting multiple statements in one line, separate them by “;”
- Indentation is the leading whitespace (whitespace at the beginning of a line)
- Indentation is treated uniquely in Python. They are used for grouping code
- Other programming languages encourage indentation
- Python requires consistency in whitespace indentation (Need to pay special attention to indentation in Python)

Exercise 1

```
x = 0; y = 0; z = 10
if z > 25:
    x = z**2
    y = z**3
print (x, y)
```

```
x = 0; y = 0; z = 10
if z > 25:
    x = z**2
y = z**3
print (x, y)
```

What will these programs output?

Add comments in Python

- # is the Python comment character
- On any line, everything after the # character is ignored by Python
- To add block comment in Spyder
  - Select the lines that you want to comment out, click Edit -> Add Block comment
  - Or select the lines that you want to comment out and press Ctrl + 4
- To remove block comment in Spyder
  - Select the comments of interest, click Edit -> Remove Block comment
  - Or select the comments of interest and press Ctrl + 5

Python keywords

- These are special words in Python that cannot be used by you to name things
- Except the three keywords (True, False, None), all others are lowercase
- True and False are used as Boolean values in Python code
- None represents no value. It is also the default value returned by a function if it doesn’t have a return statement.

```
In [34]: help("keywords")

Here is a list of the Python keywords.  Enter any keyword to get more help.

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

Objects and types

- In Python, everything in the system is considered to be an object

- An object in Python has the following
  - ✓ An identity - Whenever an object is created, it receives an identification number. View the number using id()

  - ✓ Some attributes – information about the object, such as the object's type, type()

  - ✓ Zero or more names – used by programmers to make the code more readable

Naming objects

- A name may contain letters, numbers, and “_”
- A name cannot be the keywords listed in the previous slide
- Every name must begin with a letter or “_”
- A number is not allowed as the first character
- Multi-word names can be linked together using “_”
- A name starting with “_” is often used to denote a variable with special characteristics
- Names are case sensitive – x and X are two different names
- Never use the characters 'l', 'O', or 'I' as single character variable names
- Variable names and function names should be lowercase, with words separated by underscores as necessary to improve readability
- Class names should normally use the CapWords convention.

Exercise 2

Which are illegal statements?

a = 5
for = 5
a2 = 5
a 2 = 5
a_2 = 5
_a_2 = 5
a-5 = 5
a$2 = 5
a2$ = 5
$a2 = 5

Python types

- Python has two groups of types
  - ✓ Primitive types
    - ✓ int - Integers
    - ✓ float - Floating-point numbers
    - ✓ bool – Boolean: True, False
    - ✓ char
    - ✓ complex
  - ✓ Collections
    - ✓ str
    - ✓ list
    - ✓ tuple
    - ✓ dict
    - ✓ set
- Python has two special types
  - ✓ None type
  - ✓ range type

Get help in Python

Get help on a function

> help(print)
OR
> print?
OR
> ?print

If no argument is given, the interactive help system starts on the console
> help()
> print

Type q to quit the help session
> q

Some commands to help you work efficiently

```
# View the history of what you typed
> hist
> hist -no

# re-run a line
> recall line-number
```

Key takeaways

- Python is an open-source interpreted programming language
- No variable declaration is needed
- Code blocks are identified by indentation
- A module is a Python file containing a set of Python commands to solve some problems
- Large programs are split into modules