WELCOME TO CS1!

- Chanathip Namprempre, lecturer and lab instructor
- Several 121 alums as TAs

REED COLLEGE CSCI 121 SPRING 2023

COURSE OVERVIEW

LECTURE 01-1

CHANATHIP NAMPREMPRE, REED COLLEGE CSCI 121

COURSE OVERVIEW

- There is a course webpage at https://nchanath.github.io/121-S23
  - It has the syllabus and a schedule of topics covered.
  - There I'll post readings, assignments, lecture materials.

ASSIGNMENT SUBMISSION THRU GRADESCOPE

- There is a **Gradescope** "course" for submitting completed assignments.
  - You should have received an invitation to join it.
  - Homework and project hand-ins will be posted there.
- Start **Homework 1** as soon as possible. It is due by Sunday at 11:00pm.
  - set up your computer.
  - practice submitting assignments.

- I post homework descriptions at https://nchanath.github.io/121-S23 under tab “Homework”

CS1 IS AN INTRODUCTION TO SEVERAL THINGS

- Course topics:
  - An **introduction** to programming. We will use the **Python** language.
  - An **introduction** to the discipline of computer science.
  - An **introduction** to object-oriented programming.
  - An **introduction** to data structures and algorithms.

No prerequisites.
No prior programming experience expected.

WHY PYTHON?

- It's a good first language.
  - It's easy to learn, loose, feature-rich.
  - Has features from several language families.

- It has a large programmer base.
  - Used by the open source community for scripting, glue.
  - Used by several scientific communities:
    - bioinformatics, computational chemistry, SAGE math.
  - Programming tools and docs are freely available.

- It's great fun.

A PYTHON PROGRAM

```
name = input("Enter your name: ")
print("Hello, " + name + ".")

course = int(input("What's the course's #? "))
print("Ahh, yes, CSCI " + str(course) + "!")
```

```
square = 0
count = 0
while square + 2 * count + 1 <= course:
    square += 2 * count + 1
    count += 1

if course % square == 0:
    print("Did you know " + str(course))
    print("equals "+ str(count) + " squared?")
```

ANOTHER PYTHON PROGRAM

```
import time

def newton(guess, target):
    time.sleep(0.5)
    next = guess - (guess * guess - target) / (2 * guess)
    while abs(next - guess) > 0.001:
        print(guess)
        guess = next
        next = guess - (guess * guess - target) / (2 * guess)

course = 121
name = input("Enter your name: ")
print("Okay, " + name + " let me think...")
approx = newton(course/2.0, course)
print("Did you know " + str(course))
print("is roughly "+ str(approx) + " squared?")
```

WEEKLY PROGRAMMING TOPICS

- Here are the first several weeks of programming topics:
  - **WEEK 1:** scripting; program input and output; calculating things
  - **WEEK 2:** defining functions and procedures; checking conditions
  - **WEEK 3:** loops
  - **WEEK 4:** lists and dictionaries
  - **WEEK 5:** recursion
  - ...

The schedule can be found at https://nchanath.github.io/121-S23 under tab “Syllabus”

A PYTHON PROGRAM

WEEK 1: SCRIPTING

```
name = input("Enter your name: ")
print("Hello, " + name + ".")

course = int(input("What's the course's #? "))
print("Ahh, yes, CSCI " + str(course) + "!")

square = 0
count = 0
while square + 2 * count + 1 <= course:
    square += 2 * count + 1
    count += 1

if course % square == 0:
    print("Did you know " + str(course))
    print("equals "+ str(count) + " squared?")
```

WEEK 1: INPUT

WEEK 1: OUTPUT

WEEK 1: CALCULATING

A PYTHON PROGRAM

```
name = input("Enter your name: ")
print("Hello, " + name + ".")

course = int(input("What's the course's #? "))
print("Ahh, yes, CSCI " + str(course) + "!")
square = 0
count = 0
while square + 2 * count + 1 <= course:
    square += 2 * count + 1
    count += 1

if course % square == 0:
    print("Did you know " + str(course))
    print("equals "+ str(count) + " squared?")
```

WEEK 3: LOOPS

WEEK 2: CHECKING CONDITIONS

ANOTHER PYTHON PROGRAM

```
import time

def newton(guess, target):
    time.sleep(0.5)
    next = guess - (guess * guess - target) / (2 * guess)
    while abs(next - guess) > 0.001:
        print(guess)
        guess = next
        next = guess - (guess * guess - target) / (2 * guess)

course = 121
name = input("Enter your name: ")
print("Okay, " + name + " let me think...")
approx = newton(course/2.0, course)
print("Did you know " + str(course))
print("is roughly "+ str(approx) + " squared?")
```

WEEK 2: FUNCTIONS

WEEK 3: LOOPS

AND ANOTHER

```
import time

def newton(guess, target):
    time.sleep(0.5)
    print(guess)
    next = guess - (guess * guess - target) / (2 * guess);
    if abs(next - guess) < 0.001:
        return next
    else:
        return newton(next, target)

course = 121
name = input("Enter your name: ")
print("Okay, " + name + " let me think...")
approx = newton(course/2.0, course)
print("Did you know " + str(course))
print("is roughly "+ str(approx) + " squared?")
```

ANOTHER PYTHON PROGRAM

```
import time

def newton(guess, target):
    time.sleep(0.5)
    print(guess)
    next = guess - (guess * guess - target) / (2 * guess);
    if abs(next - guess) < 0.001:
        return next
    else:
        return newton(next, target)

course = 121
name = input("Enter your name: ")
print("Okay, " + name + " let me think ..")
approx = newton(course/2.0, course)
print("Did you know " + str(course))
print("is roughly "+ str(approx) + " squared?")
```

WEEK 2: FUNCTIONS

WEEK 5: RECURSION

WEEKLY PROGRAMMING TOPICS

- Here are the first several weeks of programming topics:
  - **WEEK 1:** scripting; program input and output; calculating things
  - **WEEK 2:** defining functions and procedures; checking conditions
  - **WEEK 3:** loops
  - **WEEK 4:** lists and dictionaries
  - **WEEK 5:** recursion
  - ...

The schedule can be found at https://nchanath.github.io/121-S23 under tab “Syllabus”

- We move somewhat quickly, but it has worked well to do so!
- Though we use **Python**, these concepts are universal.
  - You are learning the structure of algorithms; ***algorithmic problem solving***.

WEEKLY PROGRAMMING TOPICS; ASSIGNMENTS

- The remaining weeks provide a transition to more advanced programming.

  - **WEEK 6**: object-oriented programming
  - **WEEK 7**: higher order functions
  - ***SPRING BREAK***
  - **WEEK 8**: algorithmic efficiency
  - **WEEK 9**: sorting and searching
  - **WEEK 10**: linked lists
  - **WEEK 11**: binary search trees

  ...

ASSIGNMENTS

- There will be ***weekly lab homework***.
  - Assigned on Friday, due the following Friday before lab (except for the very first lab).

- There will be four ***programming projects***.
  - We'll give you 2-4 weeks to complete each.

- There will be several ***in-class quizzes***
  - Starting in a few weeks, then every week, or every other week.
  - Write code on paper, no use of a computer.

FOUR PROGRAMMING PROJECTS

- Project 1: grid
  - simulate a "cellular automaton" and do image processing

- Project 2: tweets / ciphers / stats and chats
  - analyze data in complex ways

- Project 3: hawks and doves / boids
  - simulate a population of birds

- Project 4: adventure / arcade
  - build an 80s-style game, either text-based or graphical

These will be due on occasional Tuesdays.

MEETING TIMES

- **LECTURE**: Mondays and Wednesdays, 80 minutes
  - 1:10-2:30pm in ETC 208

- **LAB MEETING**: Fridays, 80 minutes
  - 1:10-2:30pm in ETC 208

- **EVENING TUTORING**: Sunday through Thursdays, 7-9pm, ETC 208

- **MY OFFICE HOURS**: TBA

RESOURCES

In addition to me and the TAs...

- I will post all my slides and code examples

- I'll suggest supplemental readings from **two textbooks**:
  - **Python: how to think like a computer scientist** from Green Tea Press
    - Follows the topics of the course fairly closely
  - **CoMPoSING PRoGRAMS** from UC Berkeley
    - This is their **Python** rewrite of MIT's famous Structure & Interpretation of Computer Programs ("SICP"; uses **Scheme**)
    - Interesting supplement. Only use Chapters 1 and 2.
  - Both are freely available on-line.

CS1 IS AN INTRODUCTION TO SEVERAL THINGS

- Course topics:
  - An introduction to programming. We will use the **Python** language.
  - **An introduction to the discipline of computer science.**
  - An introduction to object-oriented programming.
  - An introduction to data structures and algorithms.

- **Q:** What is computer science as an academic discipline?

Q: WHAT IS COMPUTER SCIENCE?

- **A:** It's programming.
- **A:** It's about programming.
- **A:** It's about "about programming."
- Etc.

Q: WHAT IS COMPUTER SCIENCE?

- **A:** It's programming.
- **A:** It's about programming.
- **A:** It's about "about programming."
- Etc.
- You will learn to be reflective about programming, and also to be reflective about the tools that run programs.
- If you continue studying CS, you will learn to make tools that help people write programs. And make tools that help tools that run programs. etc.

SETTING THINGS UP

CONFIGURING YOUR SYSTEM FOR THE COURSE

CHANATHIP, REED COLLEGE CSCI 121

INSTALL VSCODE

- **MAC**
  - Download and install vscode from https://code.visualstudio.com/

- **MS Windows**
  - Install vscode from Microsoft Store

- **Linux**
  - Use apt
  - Follow instructions from, e.g., https://linuxize.com/post/how-to-install-visual-studio-code-on-ubuntu-20-04/

RUNNING A TERMINAL

- **MAC**
  - Go to Launchpad and launch the “terminal” app

- **MS Windows**
  - Go to the start menu and launch “Powershell”

- **Linux**
  - Launch “terminal” application

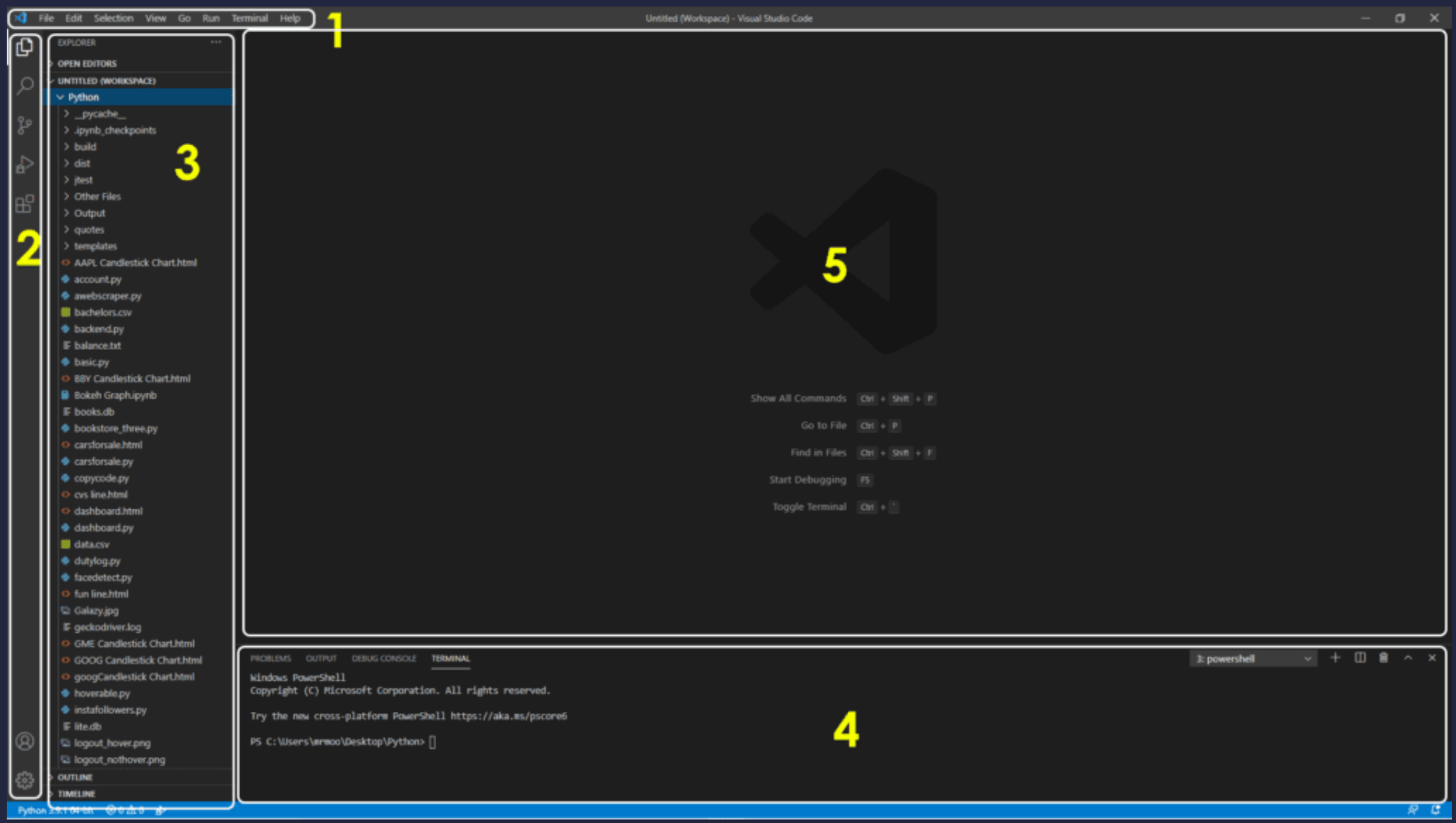
INSTALL PYTHON3

- **MAC**
  - First run python --version on the command line
  - If your version is 2.x, then download and install python3 from www.python.org

- **MS Windows**
  - Download and install python3 from www.python.org or from MS Store (choose Python 3.10)

- **Linux**
  - Check python --version on the command line
  - If python3 isn’t install, sudo apt install -y python3-pip

VSCODE USER INTERFACE



https://python.land/creating-python-programs/vscode-gui-tour

CONFIGURE VSCODE

- **Install extensions**
  - “Python” by Microsoft
  - “Pylint” by Microsoft
  - “Live Share Extension Pack” by Microsoft (sign up for a github account if you don’t already have a Microsoft account)

- **Explore vscode**
  - Open a terminal panel with Ctrl+` (back tick)
  - Navigate directories in your computer using the terminal
  - Create and save a file, then find that file on your computer using Windows Explorer

PYTHON SCRIPTING

LECTURE 01-1:
THE ANATOMY OF A PYTHON SCRIPT

JIM FIX, REED COLLEGE CSCI 121

PYTHON SCRIPTING

- We start by looking at **Python scripting**:
  - A script is a text file containing lines of Python code.
  - Each line is a Python ***statement***.
  - The Python ***interpreter*** (the `python3` command) executes each statement, line by line, from top to bottom.
  - A statement directs that an action be made by the interpreter, which has a *state-changing* effect.

RUNNING A SCRIPT

- The script text below was saved as **hello_calc.py** within a folder:

```
print("Hello there, everyone!")
print("This is our second Python program.")
print("Did you know that 78687 times 89798 is this?")
print(78687 * 89798)
```

- On MS Windows, within Powershell, after the **prompt**, I enter the **command**:

```
PS C:\Users\Chanathipn> python3 hello_calc.py
Hello there, everyone!
This is our second Python program.
Did you know that 78687 times 89798 is this?
7065935226
PS C:\Users\Chanathipn>
```

- The Python interpreter outputs those **four lines of text**.

PYTHON SCRIPTING

Each Python statement directs that an action be taken, which has an effect on the ***runtime system***.

- Some examples of effects:
  - some text gets ***output*** (printed) to the *console*
  - some typed console ***input*** is read
  - some named ***variable*** gets assigned a newly computed value
  - a window is displayed, a file is read, a URL's content is fetched, the program connects to a database or a network service, a noise is made, etc., etc.

ANOTHER EXAMPLE: VARIABLES

- Here is that same program, slightly modified:

```
print("Hello there, everyone!")
print("This is our third Python program.")
result = 78687 * 89798
print("Did you know that 78687 times 89798 is this?")
print(result)
```

- The third line is an assignment statement.
  - It introduces a variable named **`result`** and associates it with a **`value`**.
  - That computed value is saved in Python's memory and can then be used later in the script.

- In line 5, we tell Python to **output the value** of **`result`**.

ANOTHER EXAMPLE: VARIABLES AND VARIABLE ASSIGNMENT

- Here is that same program, slightly modified:

```
print("Hello there, everyone!")
print("This is our third Python program.")
result = 78687 * 89798
print("Did you know that 78687 times 89798 is this?")
print(result)
```

- A variable can be reassigned in a subsequent statement
  - It can even be computed from its prior value, for example:

```
amount = amount + 10
```

- Don’t reverse the order! Don’t write:

```
78687 * 89798 = result
```

  - Assignment statements are not logical/mathematical assertions!
  - They are actions to taken by the Python interpreter at that moment.

ANOTHER EXAMPLE: FORMATTING OUTPUT

- Here is that same program, modified even more:

```
print("Hello there, everyone!")
print("This is our fifth Python program.”)
result = 78687 * 89798
print("78687 times 89798 equals ” + str(result) + ”.”)
```

- In line 4 we use the function **`str`**, feeding it the result.
  - It converts that number into a ***string of characters.***
- We then use the ***string concatenation*** operation “**+**” to stick three strings together into a single string:

```
"78687 times 89798 equals 7065935226."
```

- Then we print that whole string of characters.

The plus sign means different things for different ***types*** of data values.

STRING ARITHMETIC: PLUS

- The plus sign means different things for different types of values:

```
print("Hello there, everyone!")
print("This is our sixth Python program.”)
result = 78687 * 89798
print("78687 times 89798 equals ” + str(result) + ”.”)
print(result + result)
print(str(result) + str(result))
```

- Here is its execution within Terminal:

```
PS C:\Users\Chanathipn> python3 hello6.py
Hello there, everyone!
This is our sixth Python program.
78687 times 89798 equals 7065935226.
14131870452
706593522667065935226
PS C:\Users\Chanathipn>
```

STRING ARITHMETIC: TIMES

- Another sample program:

```
name = input("Could someone volunteer their name? ")
print("Hello there, "+name+"!")
print("Thanks for volunteering like that.")
print("This is our eighth Python program.”)
repeated = (name + “, “) * 3 + name
print(“Below is your name repeated four times:”)
print(repeated)
```

- Its execution within Terminal

```
PS C:\Users\Chanathipn> python3 shoutout4x.py
Could someone volunteer their name? Audrey Bilger
Hello there, Audrey Bilger!
Thanks for volunteering like that.
This is our eighth Python program.
Below is your name repeated four times:
Audrey Bilger, Audrey Bilger, Audrey Bilger, Audrey Bilger
PS C:\Users\Chanathipn>
```

INTERACTING WITH THE USER

- This program interacts with the program's user:

```
name = input("Could someone volunteer their name? ")
print("Hello there, " + name + "!")
print("Thanks for volunteering like that.")
print("This is our seventh Python program.")
```

- Here is one such interaction within Terminal:

```
PS C:\Users\Chanathipn> python3 shoutout.py
Could someone volunteer their name? Audrey Bilger
Hello there, Audrey Bilger!
Thanks for volunteering like that.
This is our seventh Python program.
PS C:\Users\Chanathipn>
```

- The program has an assignment statement followed by 3 print statements.
- The assignment's right hand side uses a function named `input`
- That function first outputs a **prompt string** to the console...
  - And then it reads a **string of input** typed into the console.

STRING ARITHMETIC

- Another sample program:

```
name = input("Could someone volunteer their name? ")
print("Hello there, "+name+"!")
print("Thanks for volunteering like that.")
print("This is our eighth Python program.”)
repeated = (name + “, “) * 3 + name
print(“Below is your name repeated four times:”)
print(repeated)
```

- Its execution within Terminal

```
PS C:\Users\Chanathipn> python3 shoutout4x.py
Could someone volunteer their name? Audrey Bilger
Hello there, Audrey Bilger!
Thanks for volunteering like that.
This is our eighth Python program.
Below is your name repeated four times:
Audrey Bilger, Audrey Bilger, Audrey Bilger, Audrey Bilger
PS C:\Users\Chanathipn>
```

ANOTHER EXAMPLE

- Consider this Python program:

```
pi = 3.14159
area = float(input("Circle area? "))
radius = (area / pi) ** 0.5
print("The radius of that circle is "+str(radius)+" units.")
```

- This has is 3 assignment statements and a print statement.
  - The first defines/assigns the variable named **`pi`**.
  - The second gets a floating point value (a “calculator number”) as input, assigned to **`area`**. We compute that using an arithmetic formula.
  - The functions **`float`** and **`str`** convert values of one type to values of another type.

IMPORTING LIBRARIES

- Consider this Python program:

```
from math import pi, sqrt
area = float(input("Circle area? "))
radius = sqrt(area / pi)
print("The radius of that circle is "+str(radius)+" units.")
```

- Here we **`import`** some definitions from a Python package named **`math`**.
  - **`pi`** is the name of a floating point constant.
  - **`sqrt`** is the name of a floating point function.
  - There are packages for all sorts of useful Python libraries.

SOME ISSUES I'D LIKE TO ADDRESS

- values versus variables versus expressions
- functions, *calling* functions, *defining* functions (next week)
- different types: `int` versus `float` versus `str`
- operations on each type (and the "overloaded" meanings of `+` and `*`)
- built-in functions for each type
- managing print output carefully
  - special characters (tab, end of line, quote, …)

Let's switch modes in how we use the Python interpreter…

PYTHON SCRIPTING

LECTURE 01-1 (CONT.):
PYTHON INTERPRETER

JIM FIX, REED COLLEGE CSCI 121

PYTHON INTERPRETER: BASICS

- Python can be used to "live script":

```
PS C:\Users\Chanathipn> python3
>>> print(“hello”)
hello
>>> print(6 * 7)
42
>>> result = 6 * 7
>>> print(result)
42
>>>
```

- We can try a Python coding by interacting directly with the interpreter.
- We type in Python statements one at a time.
- Each line gets executed immediately.

THE INTERPRETER AS CALCULATOR: EVALUATION

- Python can be used to evaluate expressions:

```
PS C:\Users\Chanathipn> python3
>>> "hello"
hello
>>> 6 * 7
42
>>> result = 6 * 7
>>> result
42
>>>
```

- We enter Python expressions instead.
  - Python evaluates it and shows its value on the next line.
- A Python statement describes an action to be performed.
- A Python expression describes a value to be calculated.
  - This evaluation is different than printing.

THE INTERPRETER AS CALCULATOR: AUTOMATIC CONVERSION

- Python can be used to evaluate expressions:

```
PS C:\Users\Chanathipn> python3
>>> "hello"
hello
>>> 6 * 7
42
>>> result = 6 * 7
>>> result
42
>>>
```

- Here, Python is acting differently. It calculates the value of the expression, then (quietly) converts that to a string of text, then reports that text representing the value.

THE INTERPRETER AS CALCULATOR: REPL

- Python can be used to evaluate expressions:

```
PS C:\Users\Chanathipn> python3
>>> "hello"
hello
>>> 6 * 7
42
>>> result = 6 * 7
>>> result
42
>>>
```

- It follows three steps:
  - **READs**: it looks at the expression entered after >>>
  - **EVALUATEs**: it performs that calculation, obtaining a value, including looking up variables' values
  - **PRINTs**: it converts that value to a string; displays it.

THE INTERPRETER AS CALCULATOR: REPL

- Python can be used to evaluate expressions:

```
PS C:\Users\Chanathipn> python3
>>> "hello"
hello
>>> 6 * 7
42
>>> result = 6 * 7
>>> result
42
>>>
```

- This is the "**READ** - **EVALUATE** - **PRINT LOOP**" (or "**REPL**").
- Having access to a **REPL** for a programming language is wonderful!
- It's a big reason we teach programming in Python.

PYTHON PROVIDES SOME USEFUL FUNCTIONS…

```
>>> pow(2,3)
8
>>> abs(-3)
3
>>> abs(4 + 2)
6
>>> min(3,7)
3
>>> max(4, 10.5 + 8.3, 6)
18.8
>>> from math import sqrt, pow
>>> sqrt(2.0)
1.4142135623730951
>>> pow(2.0,4.5)
22.627416997969522
```

PYTHON PROVIDES ARITHMETIC

```
>>> 3 + 7
10
>>> 4 + 2 * 3
10
>>> (4 + 2) * 3
18
>>> 4 / 16
0.25
>>> 2 ** 4
16
>>> 0.1 + 0.2
0.30000000000000004
```

ARITHMETIC: FLOATING POINT IMPRECISION

```
>>> 3 + 7
10
>>> 4 + 2 * 3
10
>>> (4 + 2) * 3
18
>>> 4 / 16
0.25
>>> 2 ** 4
16
>>> 0.1 + 0.2
0.30000000000000004
>>> type(4)
<class 'int'>
>>> type(0.25)
<class 'float'>
```

INTEGERS VERSUS FLOATING POINT NUMBERS

- Python has two types of number values: `int` and `float`
- With integers, computation is exact.
- With floating point numbers ("*floats*"), computation is approximate.

```
>>> 10 / 2
5.0
>>> 3 + 4.0
7.0
>>> int(8.7)
8
```

SPECIAL CHARACTERS

- A backslash character `\` followed by a second character expresses special characters
  - a tab is `\t`, a new line is `\n`, a quote is `\'`, a backslash is `\\`

```
>>> z = input('What\'s your name?')
What's your name?John
>>> x + ” “ + z
'Hello John'
>>> print(“I\’ve “+str(19)+“ characters.\nSee?”)
I've 19 characters.
See?
>>> len(“I\’ve “+str(19)+“ characters.\nSee?”)
24
>>> print(“\thello\nthere”)
    hello
there
>>> print("/\\/\\/\\/\\"
/\/\/\/\
```

AN INFORMAL QUIZ

```
>>> z = 7
>>> x = 5 + z
>>> z = z + 1
>>> print(str(z) + str(z))
??????????
>>> 0.2 + 0.1
??????????
>>> 0.2 - 0.1
??????????
>>> len('Jim\'s example:\t  done.\n')
??????????
>>> print("abc\n"*4)
??????????
??????????
...?
>>> "hello" - "llo"
??????????
```

AN INFORMAL QUIZ (CONT'D)

```
   >>> int(-1.375)
??????????
>>> 40 / 5
??????????
>>> float(40 / 5)
??????????
>>> print(input("hello") + input(“goodbye”))
??????????
>>> ??????????
     * * * * * *
       * * * * *
     * * * * * *
       * * * * *
     * * * * * *
       * * * * *
     * * * * * *
       * * * * *
     * * * * * *
>>>
```

I then hit
the 6 key,
the RETURN key,
the 7 key, and
the RETURN key.

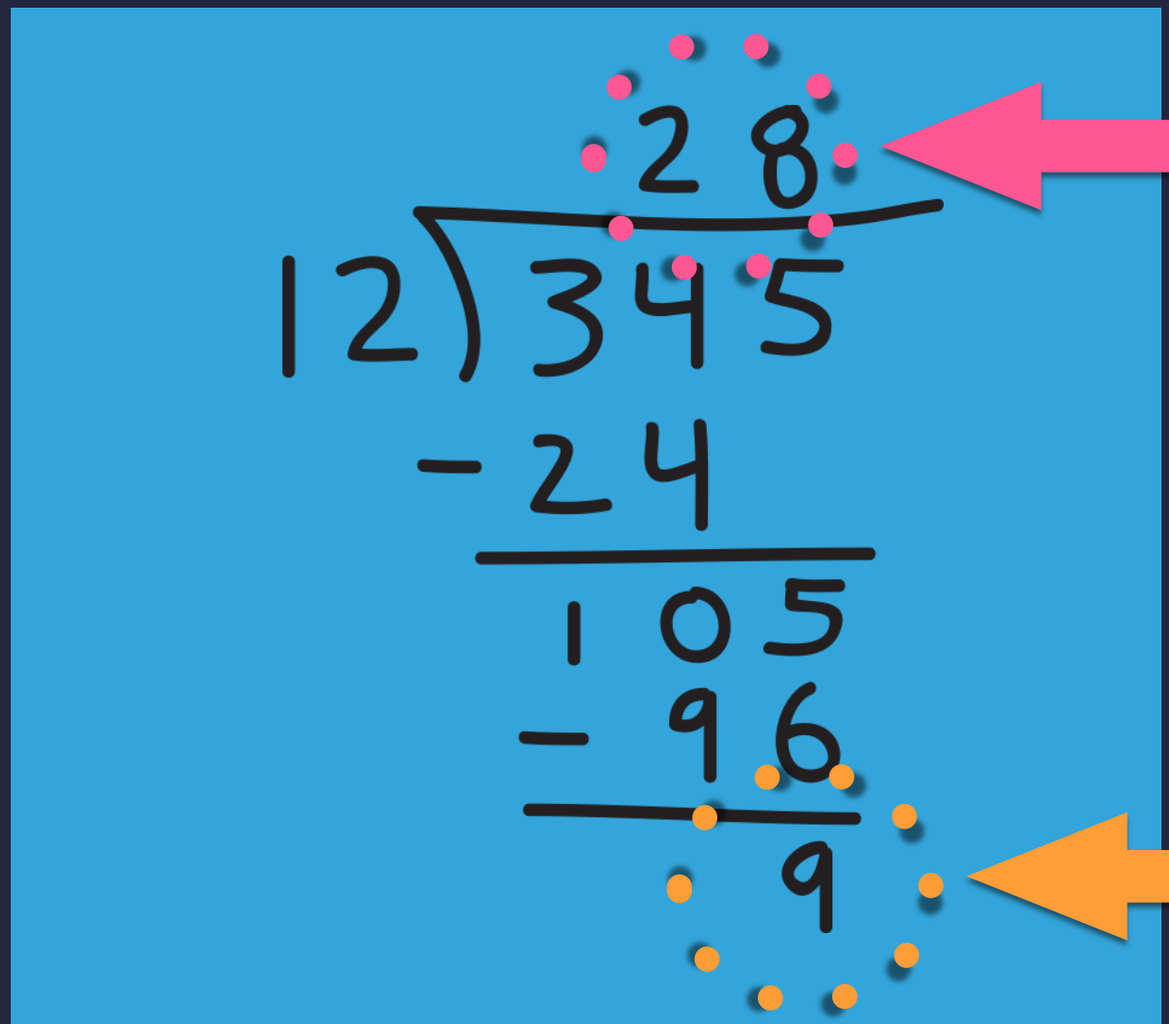
INTEGER VERSUS FLOATING POINT DIVISION

- With the normal division operation, the slash **/**, you get a floating point division.

```
>>> 10.2 / 2.0
5.1
>>> 10 / 2
5.0
>>> 10 // 2
5
>>> 87 / 10
8.7
>>> 87 // 10
8
```

- There is also an integer division operation, the double slash operator **//**.
  - This gives the integer quotient.
  - The remainder due to the division is discarded.

RECALL: LONG DIVISION



PYTHON HAS // AND % OPERATORS

- The `//` operation ("div") gives the integer quotient due to the division of two integers:

```
>>> 345 // 12
28
```

- The `%` operation ("mod") gives the integer remainder due to the division of two integers:

```
>>> 345 % 12
9
```

- This property always holds: **`n == q * d + r`**

```
>>> 28 * 12 + 9
345
```

EXAMPLE USES

```
>>> 345 % 10
????????
>>> 345 // 10
????????
>>> 6789 % 2
????????
>>> 6790 % 2
????????
>>> -26 % 2
????????
>>> -76 % 10
????????
>>> -26 // 2
????????
>>> -76 // 10
????????
```

EXAMPLE USES

```
>>> 345 % 10
5
>>> 345 // 10
34
>>> 6789 % 2
1
>>> 6790 % 2
0
>>> -26 % 2
0
>>> -76 % 10
4
>>> -26 // 2
-13
>>> -76 // 10
-8
```

SUMMARY

- So far, three kinds of statements:
  - `print` statement
  - assignment statement
  - `import` statement
- Several built-in functions
  - `input`
  - conversions: `str`, `int`, `float`
  - `abs`, `min`, `max`, `pow`, and many more from the `math` library
  - `len`
  - `type`

SUMMARY (CONT'D)

- Binary operations (so far)
  - for integers: `+` `-` `*` `//` `%` `**`
  - for floats: `+` `-` `*` `/` `**`
  - for strings: `+` `*` `%`

SUMMARY (CONT'D)

- The Python interpreter can be run *interactively* or *not*.

  - When **interactive**, you type in a statement or an expression.
    - When a statement is entered, it gets executed.
      - If there is any output, it appears on subsequent lines.
    - When an expression is entered, it gets evaluated.
      - The value that results is displayed on the next line.

  - When **not interactive**, Python just loads and runs a script.
    - Its code is executed, line by line (statement followed by statement).

HOMEWORK 1 AND LAB

- Don't forget *you have homework 1 due Sunday by 11:00pm!*

- We'll have a lab for **Homework 1** assignments:
  - the description will be at https://nchanath.github.io/121-S23 under tab “Homework”
  - you'll write several Python scripts much like these examples
  - bring your laptop to the lab

"Think Python" text

READINGS; NEXT WEEK

- This week's lecture material can be supplemented with:
  - **Reading:** TP Ch. 1 and 2; CP Ch 1.1-1.2
- Next week we'll look at
  - defining functions (i.e. `def` ...)
  - the conditional statement (i.e. `if`)

"Composing Programs" text

- **Reading:**
  - TP Ch. 3, 6 (functions); TP Chs 4.1-4.8 (conditionals)
  - CP 1.3-1.4

TO DO

- We'll continue our exploration of Python next time.

Meanwhile, here are some things you need to do:

- Carefully read the syllabus at the course website.
- Look at the assigned readings for this week on the schedule.
- Join the **Gradescope 121 course**.
- Attend lab Friday to finish work on Homework 1.
- (Homework 1 includes setting up your computer to write/run Python code.)
- Finish Homework 1 by Sunday at 11:00pm.