Final exam on Wednesday, May 15
12:00–1:50, WWH 102
Introduction to Computer Programming
CSCI-UA 2

From Natural Language to Machine Language
by Way of Python
What is a program?

A sequence of instructions for a computer to follow

May be mathematical or symbolic

Basics include:
• input
• output
• math
• conditional execution
• repetition
Natural Languages and Programming Languages

Similarities

Syntax
Grammar
Parts of speech
Semantics
Syntax

Natural language syntax is the arrangement of words and phrases to create well-formed sentences.

Programming language syntax is the arrangement of words and characters to correctly structure programs.
Grammar

Natural language grammar refers to the whole system and structure of a language, such as sentences and paragraphs.

Programming languages also implement structure, such as tokens, blocks of code and statements within the blocks.
Parts of Speech

Natural languages incorporate different parts of speech, like nouns, verbs, and adjectives.

Programming languages also have parts of speech called “data types” that include different kinds of numbers and characters.
Semantics

In natural languages, semantics refers to the meaning of a word. “Cat” brings something specific to mind.

In programming languages, certain symbols, like + and =, have specific meaning as well as some key words.
Key Words

A primary difference between natural and programming languages

Python keywords:

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

Determining the logic of a program without regard for the language it will be written in

Best written out on paper or in a plain text editor

Pseudocode describes the steps of an algorithmic process
Program Design

Input
Processing
Output
Determine requirements
→ Write the source code
   ↓
   Convert source code to object code
      ↓
      Run the program
         ↓
         Check the output
Debugging

Syntax errors:
Program doesn’t run because structure isn’t correct or doesn’t follow rules of language

Runtime errors:
Happen when the program is running. Also called “exceptions”

Semantic errors:
Program runs without an error message, but not correctly because it’s not doing what you meant it to
Programming Languages
A contemporary list

Java
C
C++
PHP
C#
Visual Basic
Python
Objective-C
Perl
Javascript
Python

High-level programming language

Developed in the 1990s by Guido van Rossum

Actively maintained and documented by programmers around the world

Clear syntax

General purpose usage

Wide range of libraries available

Python 3
Python Interpreter

Like all high-level programming languages, Python programs must be compiled and/or interpreted to run.

Python scripts have to be processed by another program called the “Python interpreter”.

The interpreter does the following:
- Reads your script
- Compiles it into “bytecode”
- Executes bytecode to run program
IDLE
Interactive Development Environment

This is what we’ll be using to write, run, and debug our code.
Installing Python 3

Python comes preinstalled on Macs but it is version 2

python.org/download

Versions available for Mac, Windows, and Linux
Basic Data Types

- Integers
- Floating Point Numbers
- Strings
**Integer**

`int`

A whole number

Can be of unlimited size

Be careful not to use commas in your numbers, for example: 1,000
**Floating point number**

float

A number that contains a decimal point

Has minimum and maximum values

Limited precision

4 is not the same as 4.0
Arithmetic Operators
Lowest precedence to highest precedence

+    -
*    / 
//  
% **
String

str

A sequence of one or more characters

A string of letters and numbers

Enclosed in quotation marks

Double quoted strings can contain single quotes and vice versa

Triple quotes can span multiple lines
String Operators

+  
*  
#
Variable

A name that refers to a value

An “assignment statement” gives a value to a variable

Variables remember things

Variables can change, too

= is Python’s assignment token
Variable Names

Can be of any length
Characters must be letters, numbers, or the underscore (_)
First character cannot be a number
Case sensitive
Python keywords cannot be used as variable names
Multiple Assignment

Assigning more than one variable at a time is possible

\[ a, b, c = 'foo', 'bar', 'baz' \]

Swapping variable values

\[ x, y = y, x \]
Built-In Functions

A function is a reusable chunk of code

We’ve already been using functions
  - `print()`
  - `int()`
  - `type()`

Function name and arguments

Parentheses mean “execute this function”

Some functions take no input
Modules

A collection of related functions

To use a module, it must be imported:

```
import module
```

Then you can use any of the functions in that module:

```
module.function( )
```

You can also import all a module’s functions at once:

```
from module import *
```

This allows you to call the function without appending the module
Math Module
Contains many useful mathematical functions

- Square root
- Pi
- Exponentiation
- Degree conversion
- Tangents
Getting Help

Python has lots of documentation built in.

Functions and modules usually come with brief explanations.

To list functions in a module:
```
dir(module)
```

To see all of Python’s built-in functions:
```
dir(__builtins__)
```

To get help with a specific function:
```
help(function)
```

Python’s general help utility:
```
help()
```
input()
Interactive Interpreter and Python Scripts

Interactive prompt doesn’t save files, just provides results as you go.

Best for experimenting with the language, gives immediate feedback.

Scripts are full programs that run a series of Python statements in order.

Most of what we’ll do from now on will be scripts, though we’ll still refer frequently to the interactive prompt.
Writing Programs

A program is just a text file containing Python statements.

A program can have two lines of code or thousands.

Any plain text editor can be used—although we will stick with IDLE.

Give your files the extension “.py”.

Python executes the file by running all the statements from top to bottom.
Basic Input

Reading strings from the keyboard

Input function

```
variable = input('Prompt' )
```

Input can be in the form of a string or numeric data type
More on the Print Function

One of the most common functions we’ve been using

Important in our standalone Python programs for providing output

Additional arguments to the print function include separator, end character

Separator default
sep = ' '  

End character default
end = '\n'
###Boolean Logic and If-Statements

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a == b</th>
<th>a != b</th>
<th>a and b</th>
<th>a or b</th>
<th>not a</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>
Boolean Logic

Programming languages make decisions in binary terms

Two boolean values: True and False

True and False can be assigned to variables, just like strings and numbers

Logical operators for boolean values: ==, !=, and, or, and not
**Equivalence**

\[ a == b \]

Evaluates to True when \( a \) and \( b \) are the same:

- Both \( a \) and \( b \) are True
- Both \( a \) and \( b \) are False
Negation

\[ a \neq b \]

Evaluates to True when \( a \) and \( b \) are not the same:

- \( a \) is True and \( b \) is False
- \( a \) is False and \( b \) is True
And

\( a \land b \)

Evaluates to True when both \( a \) and \( b \) are True:

\[ \begin{align*}
\text{• } a \text{ is True and } b \text{ is True}
\end{align*} \]
Or

\( a \lor b \)

Evaluates to True when \( a \) is True \( or \) \( b \) is True:

- \( a \) is True and \( b \) is True
- \( a \) is True and \( b \) is False
- \( a \) is False and \( b \) is True
Not

Evaluates to True when \( a \) is False and False when \( a \) is True:

- \( a \) is False
- \( b \) is False
### Truth Table

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a == b</th>
<th>a != b</th>
<th>a and b</th>
<th>a or b</th>
<th>not a</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>
Order of Execution
Highest to Lowest

a == b
a != b
not a
a and b
a or b
Conditional Statements

Allow you to change the flow of a Python program

Consist of if-statements

Introduce code blocks
If/Else-Statements

if
else

Begins with the keyword if
Followed by a Boolean expression
May be followed by an else block for alternate conditions
If/Else If-Statements

if
elif

If-statement with more than one condition

Multiple elif blocks are permitted

else block is optional
Semantic Indentation

Code blocks are meaningful in Python.

Indentation marks blocks of code and is not an optional format.

Code blocks must be indented consistently by the same amount.

As a result, Python code is clearer and more legible.
Introduction to Computer Programming
CSCI-UA 2

Repetition Structures
For-Loops and While-Loops
Loops

Introduce nonlinearity into programs
Repeatedly execute blocks of code
For-loops
While-loops
For-Loops

Repeats a block of code a specified number of times

Like printing a list of numbers

```python
for i in range(n):
    print(i)
```
While-Loops

While loops are more flexible, but also more error-prone

Must initialize variable and set increment

```python
i = 0

while i < 10:
    print(i)
    i = i + 1
```
Repetition Structures
For-Loops and While-Loops

↓
initialization_block
↓
Is condition true?
(yes) ↓
body_block
↓
after_block
↓
(no)
String Formatting Expressions

Python defines the % operator to work on strings

When applied to strings, % provides a compact way to code multiple string substitutions

Operator on left is conversion target

On the right, provide the object you want Python to insert in its place

```
print('%d %s %f' % (i, 'str'))
```

Most common: %s %d %f
Break Statement

Let's you jump out of a loop from within the loop body

Allows you to skip over unnecessary statements

break

Should only be used when it makes your code simpler

Related to: continue
Functions

Groups of statements that can be run more than once in a program

Reusable chunks of code

Take input, provide output

Can be reused in a variety of contexts

Maximize code reuse, minimize code redundancy

Encapsulate logic, splitting complex systems into manageable parts
Calling Functions

type(3.7)

Function name
Arguments passed into function
Return Value

( ) tells Python to execute the function

Even if a function takes no input, the brackets are still required

Some functions do not return a value
Defining Functions

Just like writing a Python program but with some extra syntax

Function header:

• Begins with `def`
• Followed by name of function
• Function parameter list
• Ends with a `:`
Documentation String

Doc String
Optionally follows function header
Explains what function will do
May include example(s)
Almost always good to include for clarity and as a reminder
Function Body

All the code that follows the header

Simply an indented block of code with necessary statements

This code can use the variables from the function header

Function should return a value with the keyword return

After return, Python jumps out of the function and back to the program
Function Syntax

def name(parameters):
    statements

A return value is not required

return None

Anything besides return are called “side effects”

A print() statement is an example of a side effect
Variables
Local and Global

Variable scope is something we must pay attention to when using functions. When and where is a variable accessible to your program?

Local variables are only usable within the function they are local to.

Global variables are usable by any function or code within your program.
Local Variables

def area(rad):
    import math
    return math.pi * rad ** 2

When this function ends, rad is automatically deleted because it is local
Global Variables

rad = 5

def area():
    import math
    return math.pi * rad ** 2

def change_rad(new_rad):
    rad = new_rad

This program will not allow rad to be changed because it is global

To change the radius variable:
    global rad
Main Function

main()

It is both common and a good idea to use a main function in your programs. This is usually the starting point of a program and is run by typing: main(). This simplifies rerunning programs and as well as passing input values.
Function Parameters

Used to pass input into a function

Python passes values by reference

Default values can be set for parameters

```
def name(parameter = 'default')
```
Modules

A module is a group of related functions.

Different from a regular Python program in that it acts like a toolbox.

A module usually does not have a main() function.

To use a module, simply import it.
Importing Modules

Import functions individually:

```python
import module_name

module_name.function()
```

Import functions all at once:

```python
from module_name import *

function()
```
Namespaces

Functions within a module are available when you import them.

Modules form namespaces.

Different modules with the same function name will not clash in the same program.

module1.function(parameter)
module2.function(parameter)

The only time functions may conflict is when you import all using `*`.
Strings

Strings and Characters

s[i]
Strings

Strings are one of Python’s primary data types.

Strings can be used to represent anything that can be encoded as text.

Examples: symbols, words, text files, Python programs, and Web pages.

We’ve already been using strings quite a bit and now will focus on accessing and processing them.
Indexing

$s[i]$ 

Strings are defined as ordered collections of characters.

Therefore, we can access these characters by position with “indexing.”

Offsets start at 0 and end at one less than the length of the string.

Offset can also be specified negatively.
### Strings and Characters

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>y</td>
<td>t</td>
<td>h</td>
<td>o</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>-6</td>
<td>-5</td>
<td>-4</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>
Characters

ord('a')

All characters have a corresponding character code

Python uses Unicode to assign character codes

Unicode is a common standard for encoding thousands of symbols and characters from many languages
Escape Characters

“Escape characters” are a special notation for whitespace and other unprintable characters.

Used for characters without a standard symbol or symbols Python sets aside.

Escape characters begin with a backslash.

Escape sequences are considered one character.
Slicing Strings

Slicing is for extracting substrings from a string.

To slice a string, indicate the first character you want and one past the last character.

First index location is always 0.

Last index location is always the length of the string minus one.
String Functions

s.function()
Regular Expressions

A “regular expression” is a special sequence of characters that helps you match or find other strings.

Python has a regular expressions module, “re,” in its standard library.

Used to efficiently and compactly describe a set of strings.

Regular expressions operators and functions can be combined to perform powerful string processing tasks.
Regular Expressions

Operators

xy?
Describes either x or xy (y is optional)

x | y
Describes x or y

x*
Describes an infinite set of strings

x+
Describes an infinite set of one or more strings

Use round brackets to indicate what substring an operator should apply to
Turtle Graphics

Turtle is a standard Python module for programming graphics on a Cartesian plane

The “turtle” is a cursor that moves around a canvas drawing lines and shapes

Many of the the programming concepts we’ve learned so far are applicable to drawing

We simply need to add some turtle syntax to our vocabulary
Turtle Graphics

Creating a new turtle

Import module: import turtle

The turtle module introduces two new types: Turtle and Screen

turtle.Screen() creates a window with a canvas for drawing

turtle.Turtle() creates an object that does the drawing
Turtle Graphics

Methods

“Methods” allow us to modify the turtle and the window objects

forward(distance)
backward(distance)
left(angle)
right(angle)
circle(radius)
color()
pensize(width)
begin_fill()
end_fill()
title('Title')
Turtle Graphics
Attributes

- Location
- Orientation
- Drawing tool (pen)
Data Structures

Also known as objects, data structures are collections of values and functions.

Tuples, lists, dictionaries, sets

Unlike strings, these data structures can contain any kind of data.

The primary two we’ll focus on are lists and dictionaries.
Tuples

( )

(1, 2.0, 'three')

A tuple is an immutable sequence of 0 or more values

Enclosed in round brackets, items separated by a comma

Tuples with a single item must be followed by a comma: (x,)

Once created, it cannot be changed, which can help prevent errors

Use indexing and slicing to access individual elements
Tuples

Functions

x in tuple
len(tUPLE)
tuple.count(x)
tuple.index(x)
Lists

[ ]

A list is a mutable sequence of 0 or more values

Enclosed in square brackets, items separated by a comma

List elements can be added, removed, or modified

More common in practice than tuples

Use indexing and slicing to access individual elements
Lists

Functions

- `s.append()`
- `s.count(x)`
- `s.extend(list)`
- `s.index(x)`
- `s.insert(i, x)`
- `s.pop(i)`
- `s.remove(x)`
- `s.reverse()`
- `s.sort()`
Lists

List Comprehensions

List comprehensions is a special notation for creating lists

Similar to regular expressions, it makes your code more efficient

Commonly used for modifying a list in some way
Dictionaries

{}  

A dictionary is a data structure for storing pairs of values

Values can be accessed by their keys

Like lists, dictionaries are mutable

Keys are unique and cannot be repeated within a dictionary

Keys must be immutable and cannot be a list or dictionary

Values are, however, mutable
Dictionaries

Functions

d.items()
d.keys()
d.values()
d.get(key)
d.pop(key)
d.popitem()
d.clear()
d.copy()
d.fromkeys(s, t)
Sets

set()

A set is a collection of 0 or more items with no duplicates.

A good way to remove duplicates from a sequence.

Two categories: mutable sets and immutable frozensets.
Input and Output
Reading and Writing Text Files
Input and Output

I/O

Console I/O involves printing to the screen and reading strings via `input`

File I/O involves reading and writing external files

More on string formatting

Reading from the Web
String Interpolation
format % values

Simple way to format strings
Conversion specifiers tell Python how to render values
Most common are: d, f, s
Supply one value for each specifier
String Formatting

`format(value, spec)`

Use format strings with the string function, `format`.

Anything within `{}` is replaced.

Formatting codes also allow you to specify format: `{x:.3f}`
Files

Named collection of bits on a storage device

Text files
- Strings
- Python source code
- HTML files
- Can be viewed with a text editor

Binary files
- Not human readable
- Examples include image files
- Information grouped in chunks

Web pages are essentially strings
Directories are essentially folders on your computer.

Python will, by default, access and write to files in the “current working directory”.

Pathname to a particular file allows you to access it regardless of location.

Use Python’s os module to get and change the current working directory.
Directory Functions

os

- `os.getcwd()`
- `os.listdir()`
- `os.chdir()`
- `os.path.isfile(path)`
- `os.path.isdir(path)`
- `os.stat(file)`
Reading Text Files

\[ f = \text{open}(\text{file}) \]

Open file in read mode

\[ \text{open}(\text{file\_name}, \ 'r') \]

Iterate over each line of the file to read

\[ \text{for line in file:} \]

Alternatively, a text file can be read directly into computer memory (RAM)

\[ \text{file\_read()} \]
Writing Text Files

'w'

Open file in write mode

open(file_name, 'w')

Write to the file

file.write()

If another file of the same name exists, this will overwrite it

Append mode 'a' allows you to add lines to the end of a file
File Modes
open(file, mode)

' r ' Open for reading
' w ' Open for writing
' a ' Open for appending to file
' b ' Binary mode
' t ' Text mode
' + ' Open a file to read and write
Reading Web Pages

urllib.request.open

Python supports reading from the Web with the urllib.request module.

XML documents are easier to parse than HTML documents.

Basic string formatting functions can be used to locate and retrieve specific information within a Web page.
Final exam on Wednesday, May 15
12:00–1:50, WWH 102