Exam 1 on Monday, March 4
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 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
Introduction to Computer Programming
CSCI-UA 2
From Natural Language to Machine Language by Way of Python
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
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

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

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
print()
Multiple Assignment

Assigning more than one variable at a time is possible

```python
a, b, c = 'foo', 'bar', 'baz'
```

Swapping variable values

```python
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

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.
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

$\neg a \neq b$

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:

- \( a \) is True and \( b \) is True
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
not a

Evaluates to True when a is False and False when a is True:

• a is False
• b is False
## Truth Table

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>a == b</td>
<td>a != b</td>
<td>a and b</td>
<td>a or b</td>
<td>not a</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---------</td>
<td>---------</td>
<td>----------</td>
<td>---------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td></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-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