CS 106
Introduction to Computer Science I

07 / 06 / 2021

Instructor: Michael Eckmann

Today's Topics

- Introduction
- Review the syllabus
  - including the policies on academic dishonesty and improper collaboration
- Introductory comments on programming languages
- An example of a simple Python program
- Printing

Who is your instructor?

- I'm Mike Eckmann, an Associate Professor in the Computer Science Dept., Skidmore College. I have been at Skidmore since 2004. Before coming to Skidmore I was at Lehigh University in PA.
- I studied Mathematics and Computer Engineering and Computer Science all at Lehigh University.
- I was employed as a programmer (systems analyst) for eight years.

1st Homework

- Read the syllabus and review the improper collaboration policy (both will be available on our course webpage.)
- Read chapter 1 of text.
- Will send course webpage and a questionnaire via email later this class.

Syllabus

- Office hours
  - By appointment
- Text book
- Assignments
  - Programs & HW
- Collaboration policy
- Grading scheme
- Workload
- Student preparation before class

Note: The most up-to-date syllabus will be found on the course web page.

This semester we will ...

- Be introduced to computer science.
- Learn programming (in Python)!
- Solve problems and learn to think like programmers.
- Hopefully have a fun learning experience.

Computer Science is ...

- more than computer programming.

Programming Languages

- Machine
- Assembly
- High-level
  - in no particular order
  - Pascal, C, C++, Basic, Fortran, Java, Python and many, many more ...
- Procedural vs. Object-oriented

Python version

- A note about which version of Python we will focus on.
- Python 2.x vs. Python 3.x
- python --version

Syntax vs. semantics

- Time flies like an arrow
  - 3 meanings (semantics) for 1 syntax
  - That's why English is not a programming language!
- Errors
  - Run time (crashes)
  - Compile time (syntax errors)
  - Logic (semantic errors)
- Have any of you encountered any of these with software that you use? Which kinds of errors? Examples?

Hello, World! program

```
# Hello, World program
# CS106 Intro to CS I
# Skidmore College

print('Hello, World!')
```

Discussion of “Hello, World!”

- comments in source code
- The built-in function named: print

Common programming mistake (bug)

- Case matters --- print is different from Print
- Text can be enclosed in single quotes or double quotes
- Python programs must be in plain text files (not a Word document or any other file type)

Comments

- Comments are ignored by the compiler
- Single line comments start with # and the comment continues until a newline
- One common way to make multi-line comments are to use three single quotes to start them and end the comment with three single quotes
- What's the purpose of comments if the compiler ignores them?

Writing and running python programs

We can use the python shell to type and execute code line by line (good when learning, experimenting, etc.)

We can use a text editor to write, save and edit our python programs (then we can run at the command line like:

python myprog.py OR python3 myprog.py

We can use Idle or PyCharm to write, save, edit and run python programs

More about print

- Print can take more than one thing inside the paretheses (must separate them by commas)
- help(print) # look at file, sep, and end parameters

Print examples

print(“I love programming”)
print(“in Python.”)

Output:
I love programming
in Python.

Print examples

```
print("I love programming", end="")
print("in Python.")
```

Output:

I love programmingin Python.

Print examples

print(“I love programming”, end=””)
print(“in Python.”, end=””)

How will the above output?

Escape sequences

- A string's *value* is specified between quotes.
- Within the quotes the backslash character \ is special.
- The backslash character followed by another character is called an escape sequence.
- Escape sequences are handled differently than regular characters.
- **\n** is an escape sequence to embed a newline
- **\t** is an escape sequence to embed a tab
- E.g. **print(**“Hello**\n**World”**)** prints:

Hello

World

Escape sequences

If we do

print("Hamilton has a "great" lacrosse team"**)**

- there will be a syntax error (pointing at g in great)

Escape sequences

print("Hamilton has a "great" lacrosse team"**)**

- The string will be

– Hamilton has a

- and then the python compiler will see a *g* and will generate an error because the second double quote (the one before great) causes the string to end.
- What can we do?

Escape sequences

- There is an escape sequence to represent a ". It is \"

```
print("Hamilton has a \"great\" lacrosse team")
```

- Now, the string literal contains the double quotes around great. The first double quote and last double quote mark the beginning and ending of the String literal.
- The above line of code will do what we want now.
- Any other ways you can think of to get the same result (without escape sequences)?

Escape sequences

- The \ character is special inside a string. Python will interpret the \ and the next character as an escape sequence
- Let's search internet for: python 3 escape sequences
- How might we add a backslash character to a string?
- Let's say we want to print a string like:
  - C:\My Documents
- Will this work?
  - **print(**"C:\My Documents"**)**

Let’s try this problem

- Write an application that displays the numbers 1 to 4 on the same line, with each pair of adjacent numbers separated by one space. Write the program using the following methods:
  - A) Using one print
  - B) Using four prints
- Now let's do the same thing but on 4 different lines.
  - A) Using one print
  - B) Using four prints

String concatenation

- To concatenate two strings together, python provides the operator +

e.g.

print("Hey" + "now.");

prints

Heynow.

What would

print("Hey", "now.")

print?

Variables and Type

- A program often needs to be able to store and change data as it is running. For example, a program might need to store a user's name, or the user's age, or the air temperature, or whatever.
- Variables allow names to be given to data that can be stored and changed.

Variables, Values and Type

- A *variable* is a location in memory (RAM) with a

- ***name*** --- given by the programmer --- try to be descriptive of the data the variable will hold
- ***value*** --- assigned to a variable name with =
  - value can be stored / changed / read
  - Values have a data type
  - The data type is tied to the value NOT the variable
  - One can change the value of a variable e.g.

    x = 5

    x = ‘hey’

Some python data types

- **int** --- integer type --- hold *whole* numbers like – 31, 4256, 56, 2632, 755, -901
- **float** --- numeric type that holds numbers with decimal values like -5.5, 98.6, 1002.99995 etc.
- **bool** --- holds one of True or False
- **str** --- string type --- holds text like ‘Hey’

Values and Type

```
>>> type(True)
<class 'bool'>
>>> type(3)
<class 'int'>
>>> type(3.444)
<class 'float'>
>>> type('adsf')
<class 'str'>
```

Variables and Type

- To be able to specify to our program that we want to use a variable we simply make up a variable name and type it followed by = and a value to be assigned to it
- e.g.

  age = 21

Variables and Type

- age is a name we made up, what type is 21?