

Objectives

- Review
 - Linux
 - Why programming languages?
- Compiled vs. Interpreted Languages
- Programming in Python
 - Data types
 - Expressions
 - Variables
 - Comments
 - Arithmetic

Sep 12, 2007

Sprenkle - CS111

1

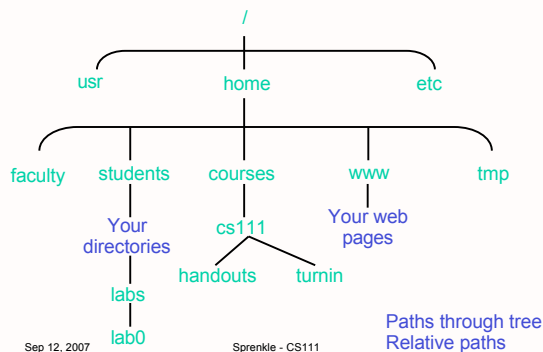
Review: Linux

- How do you ...
 - Learn more about a Linux command?
 - List the files in a directory?
 - Change your current directory?
 - Make a directory?
 - Find out the current directory?
- What is the shortcut for ...
 - The current directory?
 - The parent directory?

Sep 12, 2007

Sprenkle - CS111

Review: Linux File Structure



Sep 12, 2007

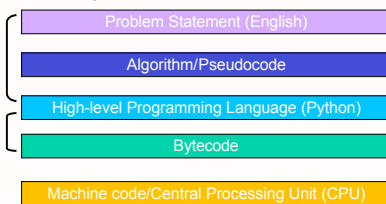
Sprenkle - CS111

Why Do We Need Programming Languages?

- Computers can't understand English
 - Too ambiguous (PB&J)
- Humans can't easily write machine code

Programmer (YOU!)
translates from
problem to algorithm
(solution) to program

Python **interpreter**
translates into
bytecode

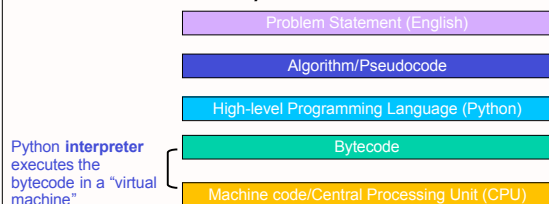


Sep 12, 2007

Sprenkle - CS111

Why Do We Need Programming Languages?

- Computers can't understand English
 - Too ambiguous (PB&J)
- Humans can't easily write machine code



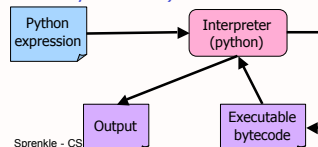
Sep 12, 2007

Sprenkle - CS111

Python Interpreter

1. Validates the Python programming language expression
 - Enforces Python syntax
 - Reports syntax errors
2. Simulates a computer (executes the expression)
 - Runtime errors (e.g., divide by 0)
 - Semantic errors (not what you *meant*)

- Good way to test expressions
- What you were doing in the "shell"

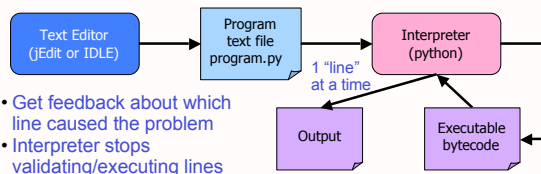


Sep 12, 2007

Sprenkle - CS

Our Programming Process

1. Programmer types a **program/script** into a **text editor** (jEdit or IDLE).
2. An **interpreter** turns each expression into **bytecode** and then executes each expression



- Get feedback about which line caused the problem
- Interpreter stops validating/executing lines

Sep 12, 2007

Sprenkle - CS111

7

Compiled Languages

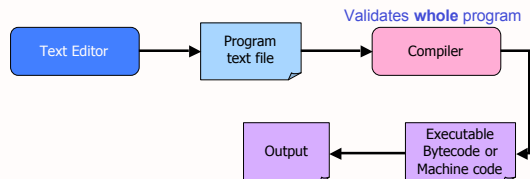
- Examples: Java, C++, C, etc.
- Compile whole program into bytecode/executable format
- Then, execute the bytecode/machine code

Sep 12, 2007

Sprenkle - CS111

Compiled Language Programming Process

1. **Compiler** compiles program
 - Validate program, report syntax errors
 - Creates **executable** (bytecode or machine code)
2. Execute **executable**



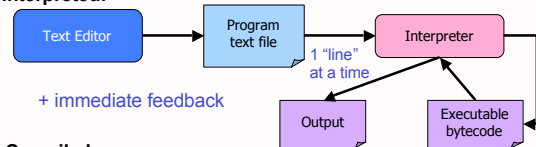
Sep 12, 2007

Sprenkle - CS111

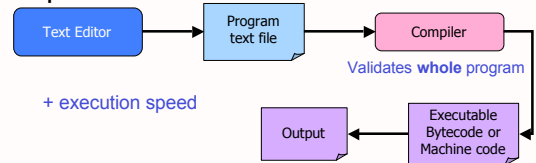
9

Compiled vs. Interpreted Languages

Interpreted:



Compiled:



Sep 12, 2007

Sprenkle - CS111

Parts of an Algorithm

- ➔ Primitive operations
 - What data you have, what you can do to the data
- Naming
 - Identify things we're using
- Sequence of operations
- Conditionals
 - Handle special cases
- Repetition/Loops
- Subroutines
 - Call, reuse similar techniques

An overview for the semester!

Sep 12, 2007

Sprenkle - CS111

11

Primitive Types

- Variable **types** are the kind of thing a variable can hold
- Broadly, the categories of primitive types are
 - Numeric
 - Boolean
 - Strings

Sep 12, 2007

Sprenkle - CS111

12

Numeric Primitive Types

Data Type	Description	Examples
int	Plain integers (32-bit precision)	-214, -2, 0, 2, 100 Range: -2^{31} to $2^{31}-1$
float	Real numbers	.001, -1.234, 1000.1, 0.00, 2.45
long	Bigger integers (neg or pos, precision limited by computer memory)	2147483648L
complex	Imaginary numbers (have real and imaginary part)	$1j * 1j \rightarrow (-1+0j)$

Sep 12, 2007

Sprenkle - CS111

Copy on board

13

How big (or small or precise) can we get?

- We cannot represent all values
- Problem: Computer has a finite capacity
 - The computer only has so much memory that it can devote to one value.
 - Eventually, reach a cutoff
 - Limits size of value
 - Limits precision of value

PI has more decimals, but we're out of space!

0 0 0 0 0 3 . 1 4 1 5 9 2 6 5

In reality, computers represent data in binary, using only 0s and 1s

Sep 12, 2007

Sprenkle - CS111

14

Strings: `str`

- Indicated by double quotes `""` or single quotes `' '`
- Treat what is in the `""` or `' '` literally
 - Known as **string literals**
- Examples:
 - `"Hello, world!"`
 - `"c"`
 - `"That is Sara's cat"`

Sep 12, 2007

Sprenkle - CS111

15

Booleans: `bool`

- 2 values
 - `True`
 - `False`
- More on these later...

Sep 12, 2007

Sprenkle - CS111

What is the value's type?

Value	Type
52	
-0.01	
$4+6j$	
<code>"int"</code>	
4047583648L	
True	
<code>'false'</code>	

Sep 12, 2007

Sprenkle - CS111

17

Introduction to Variables

- Variables have names, called **identifiers**
- A variable name (identifier) can be any **one** word that:
 - Consists of letters, numbers, or `_`
 - Cannot start with a number
 - Cannot be a Python **reserved word**
 - like `for`, `while`, `def`, etc.
- Python is case-sensitive:
 - `change` isn't the same as `Change`

Sep 12, 2007

Sprenkle - CS111

18

Variable Name Conventions

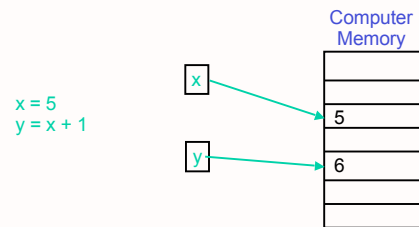
- Variables start with lowercase letter
- Constants (values that won't change) are in all capitals
- Example: Variable for the current year
 - > currentYear
 - > current_year
 - > ~~current year~~
 - > CURRENT_YEAR

Sep 12, 2007

Sprenkle - CS111

19

Variable Semantics



Sep 12, 2007

Sprenkle - CS111

Assignments

- Variables can be given any value using the "=" sign
 - > called an **assignment** statement
- Syntax: `<variable> = <expression>`
- After a variable is given a value, the variable is said to be **initialized**.
- These aren't equations! Read "=" as "gets"
 - `current_year = 2007;`
 - `my_num = 3.4;`
 - `option = 'q';`

Sep 12, 2007

Sprenkle - CS111

21

Variables: The Rules

- Only the variable(s) to the left of the = change
- You should initialize a variable before using it on the righthand side (rhs) of a rule.
- You can only have one variable with any given name in a particular block.

Sep 12, 2007

Sprenkle - CS111

22

Literals

- Pieces of data that are not variables are called **literals**.
- Ex:
 - 4
 - 3.2
 - 'q'
 - "books"

Sep 12, 2007

Sprenkle - CS111

23

Arithmetic Operations

Symbol	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder ("mod")
**	Exponentiation (power)

Sep 12, 2007

Sprenkle - CS111

Arithmetic

- You can use the assignment operator (=) and arithmetic operators to do calculations.
- Remember your order of operations! (PEMDAS)
- The thing on the left gets changed.

```
x = 4+3*10;  
y = 3.0/2.0;  
z = x+y;
```

The right-hand sides are **expressions**, just like in math.

Sep 12, 2007

Sprengle - CS111

25

Printing Output

- **print** is a function
 - Function: A command to do something
 - Displays the result of expression(s) to the terminal
- `print "Hello, class"`
 - ← **print** automatically adds a '\n' (carriage return) after it's printed
 - string literal
- `print "Your answer is", answer`
 - Use commas to print multiple "things" in one line

Sep 12, 2007

Sprengle - CS111

26

Getting Input From User

- **input** and **raw_input** are functions
 - Prompts user for input, gets the user's input
 - `input`: for numbers
 - `raw_input` for strings
- Typically used in assignments
 - `width = input("Enter the width: ")`
- In execution, terminal displays
 - "Enter the width: "
 - Assigns `width` the value the user enters

Sep 12, 2007

Sprengle - CS111

Getting Input from User

- `color = raw_input("What is your favorite color? ")`

Terminal:

Grabs every character up to the user presses "enter"

```
> python input_demo.py  
What is your favorite color? blue  
Cool! My favorite color is _light_ blue !
```

Assigns variable `color` that input

Sep 12, 2007

Sprengle - CS111

Documenting Your Code

- Use English to describe what your program is doing in **comments**
 - Everything after a `#` is a comment
 - Color-coded in IDLE, jEdit
 - Python does not execute comments
- How to Use Comments
 - Document the author, high-level description of the program at the top of the program
 - Provide an outline of an algorithm
 - Identifies the steps of the algorithm
 - Describe difficult-to-understand code

Sep 12, 2007

Sprengle - CS111

Identify the Pieces of a Program

```
# Demonstrate numeric and string input  
# by Sara Sprengle for CS111 on 9/12/07  
#
```

```
color = raw_input("What is your favorite color? ")  
print "Cool! My favorite color is _light_", color, "!"
```

```
scale = input("On a scale of 1 to 10, how much do you like Matt Damon? ")  
print "Cool! I like him ", scale*1.8, " much!"
```

Identify the comments, variables, functions, expressions, assignments

Sep 12, 2007

Sprengle - CS111

Identify the Pieces of a Program

```
# Demonstrate numeric and string input  
# by Sara Sprenkle for CS111 on 9/12/07  
#
```

```
color = raw_input( "What is your favorite color? " )  
print "Cool! My favorite color is _light_ ", color, "!"  
  
scale = input( "On a scale of 1 to 10, how much do you like Matt Damon? " )  
print "Cool! I like him ", scale*1.8, " much!"  
                        {  
                        expression
```

Identify the **comments**, **variables**, **functions**, **expressions**,
assignments, **literals**

Practice

- Average three numbers