

Objectives

- Continuing fundamentals of programming
 - Numeric Operations
- Introduction to design patterns
- Software development practices
 - Testing
 - Debugging
 - Iteration

Jan 19, 2016

Sprengle - CSCI111

1

Review

- What are the two ways we can use Python?

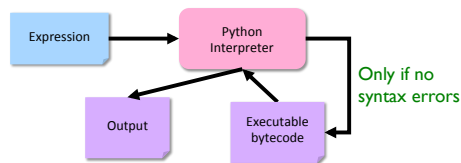
Jan 20, 2016

Sprengle - CSCI111

2

Python Interpreter

1. Validates Python programming language expression(s)
 - Enforces Python syntax rules
 - Reports syntax errors
2. Executes expression(s) ← Have a lot of these early on!



Jan 20, 2016

Sprengle - CSCI111

3

Two Modes to Execute Python Code

- **Interactive:** using the interpreter
 - Try out Python expressions
- **Batch:** execute *scripts* (i.e., files containing Python code)
 - What we'll write usually

Jan 20, 2016

Sprengle - CSCI111

4

Parts of an Algorithm

- Input, Output
- 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

Jan 19, 2016

Sprengle - CSCI111

5

Review

- What are Python's primitive data types and what do they represent?
- How do we name variables?
 - What is another word for "variable name" in programming?
- How do we give variables values?

Jan 19, 2016

Sprengle - CSCI111

6

Recap of Programming Fundamentals

- Most important data types (for us, for now): **int, float, str, bool**
 - Use these types to represent various information
- Variables have identifiers, (implicit) types
 - Should have “good” names
 - Names: start with lowercase letter; can have numbers, underscores
- Assignments
 - $x = y$ means “x set to value y” or “x is assigned value of y”
 - Only variable on LHS of statement changes

Jan 19, 2016 Sprenkle - CSCI111 7

Review: Assignment statements

- Assignment statements are NOT math equations!


```
count = count + 1
```
- These are commands!


```
x = 2
y = x
x = x + 3
```

What is the value of y?

Jan 19, 2016 Sprenkle - CSCI111 8

Numeric Arithmetic Operations

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

Jan 19, 2016 Sprenkle - CSCI111 9

Arithmetic & Assignment

- You can use the assignment operator (=) and arithmetic operators to do calculations
 1. Calculate right hand side
 2. Assign value to variable
- Remember your order of operations! (PEMDAS)
- Examples:


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

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

Jan 19, 2016 Sprenkle - CSCI111 10

Arithmetic & Assignment

- Examples:


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

Computer Memory	
x	34
y	1.5
z	35.5

- For 3rd statement
 - need to “lookup” values of X and y
 - computer remembers the result of the expression, not the expression itself

Jan 19, 2016 Sprenkle - CSCI111 11

What are the values?

- After executing the following statements, what are the values of each variable?


```
> r = 5
> s = -1 + r
> t = r + s
> s = 2
> r = -7
```

How can we verify our answers?

Jan 19, 2016 Sprenkle - CSCI111 12

What are the values?

- After executing the following statements, what are the values of each variable?
 - `a = 5`
 - `y = a + -1 * a`
 - `z = a + y / 2`
 - `a = a + 3`
 - `y = (7+x)*z`
 - `x = z*2`

Jan 19, 2016 Sprenkle - CSCI111 13

What are the values?

- After executing the following statements, what are the values of each variable?
 - `a = 5`
 - `y = a + -1 * a`
 - `z = a + y / 2`
 - `a = a + 3`
 - `y = (7+x)*z`
 - `x = z*2`

Runtime error:
 x doesn't have a value yet!
 • We say "x was not initialized"
 • Can't use a variable on RHS until seen on LHS!*

Jan 19, 2016 Sprenkle - CSCI111 14

Bringing It All Together: A simple program

```
# Demonstrates arithmetic operations and
# assignment statements
# by Sara Sprenkle

x = 3
y = 5

print("x =", x)
print("y =", y)

print("x * y =", x*y)

# alternatively:
# result = x * y
# print("x * y =", result)
```

arith_and_assign.py

What does this program display?

Jan 19, 2016 Sprenkle - CSCI111 15

Bringing It All Together: A simple program

```
# Demonstrates arithmetic operations and
# assignment statements
# by Sara Sprenkle

x = 3
y = 5

print("x =", x)
print("y =", y)

print("x * y =", x*y)

# alternatively:
# result = x * y
# print("x * y =", result)
```

arith_and_assign.py

x = 3
 y = 5
 x * y = 15

Note: doesn't print out x (literally)
 But the value of x

Jan 19, 2016 Sprenkle - CSCI111 16

Two Division Operators

<p>/ Float Division</p> <ul style="list-style-type: none"> Result is a float Examples: <ul style="list-style-type: none"> <code>6/3 → 2.0</code> <code>10/3 → 3.3333333333333335</code> <code>3.0/6.0 → 0.5</code> <code>19/10 → 1.9</code> 	<p>// Integer Division</p> <ul style="list-style-type: none"> Result is an int Examples: <ul style="list-style-type: none"> <code>6//3 → 2</code> <code>10//3 → 3</code> <code>3.0//6.0 → 0.0</code> <code>19//10 → 1</code>
---	--

Integer division is the division used in most programming languages

Jan 19, 2016 Sprenkle - CSCI111 17

Integer Division Practice

- What is the result?
 - `x = 6//4`
 - `y = 4 // 6 * 5.0`
 - `a = 6/12`
 - `b = 6.0//12`
 - `z = x / a`

What is integer division good for?

Jan 19, 2016 Sprenkle - CSCI111 18

More on Arithmetic Operations

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

Precedence rules: P E - DM% AS
 ↑
 negation

Associativity matters when you have the same operation multiple times

Jan 19, 2016 Sprenkle - CSCI111 19

NOT Math Class

- Need to write out all operations explicitly
 - In math class, $a(b+1)$ meant $a * (b+1)$

Write this way in Python

Jan 19, 2016 Sprenkle - CSCI111 20

Math Practice

```


5 + 3 * 2
2 * 3 ** 2
-3 ** 2
2 ** 3 ** 3
    
```

How should we verify our answers?

Jan 19, 2016 Sprenkle - CSCI111 21

Formalizing Process of Developing Computational Solutions

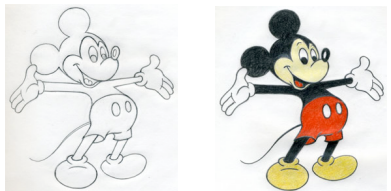
- Create a sketch of how to solve the problem (the algorithm)



Jan 19, 2016 Sprenkle - CSCI111 22

Formalizing Process of Developing Computational Solutions

- Create a sketch of how to solve the problem (the algorithm)
- Fill in the details in Python



Jan 19, 2016 Sprenkle - CSCI111 23

Errors

- Sometimes the program doesn't work
- Types of programming errors:
 - Syntax error
 - Interpreter shows where the problem is
 - Logic/semantic error
 - answer = 2+3
 - No, answer should be 2*3
 - Exceptions/Runtime errors
 - answer = 2/0
 - Undefined variable name

Expose errors when **Testing**

Jan 19, 2016 Sprenkle - CSCI111 24

Testing Process

- Test case: **input** used to test the program, **expected output** given that input
- Verify if **output** is what you expected

Jan 19, 2016 Sprenkle - CSCI111 25

Testing Process

- Need **good test cases** to help determine if program is correct
 - Tester plays devil's advocate
 - Want to expose **all errors!**
 - Find before customer/professor!

If output is not what you expect...

Jan 19, 2016 Sprenkle - CSCI111 26

Debugging

- After identifying errors during *testing*
- Identify the problems in your code
 - Edit the program to fix the problem
 - Re-execute/test until all test cases pass
- The error is called a "bug" or a "fault"
- Diagnosing and fixing error is called **debugging**

Jan 19, 2016 Sprenkle - CSCI111 27

Formalizing Process of Developing Computational Solutions

1. Create a sketch of how to solve the problem (the algorithm)
2. Fill in the details in Python
3. Test the Python program with **good** test cases
 - a. If errors found, debug program
 - b. Repeat step 3

Jan 19, 2016 Sprenkle - CSCI111 28

Practice: Our First Computational Algorithm

- Find the area of a rectangle, which has a width and height
- Test cases:

Input			Expected Output
width	height		

Jan 19, 2016 Sprenkle - CSCI111 29

Our First Computational Algorithm

- Algorithm for finding the area of a rectangle:
 - Optional: get the width and height from user
 - Alternative: "hard-code" width and height
 - Calculate area
 - Print area
- Test cases for finding the area of a rectangle
 - Test both integers
 - Test with at least one float for width, height
 - Test numbers less than or equal to 0
 - Shouldn't compute area for those

area.py

Jan 19, 2016 Sprenkle - CSCI111 30

Good Development Practices

- Design the algorithm
 - Break into pieces
- **Implement and Test** each piece *separately*
 - Identify the best pieces to make progress
 - Iterate over each step to improve it
- Write comments **FIRST** for each step
 - Elaborate on what you're doing in comments when necessary

Jan 19, 2016

Sprenkle - CSCI111

31

When to Use Comments

- Document the author, high-level description of the program at the top of the program
- Provide an outline of an algorithm
 - Separates the steps of the algorithm
- Describe difficult-to-understand code

Jan 19, 2016

Sprenkle - CSCI111

32

Looking Ahead

- Lab tomorrow
- No Broader Issue article for this week

Jan 19, 2016

Sprenkle - CSCI111

33