

Objectives

- Software development practices
 - Testing
 - Debugging
 - Iteration
- User input

Jan 14, 2019

Sprenkle - CSCI111

1

Review

Get out handouts from Friday

- How can we tell our program to display output?
- How can we store information?
- What is the syntax to do the last step?
- What are the rules and conventions for variable names?
 - What is another word for “variable names”?
 - Describe what good variable names look like
- What are the types of information in Python?

Jan 14, 2019

Sprenkle - CSCI111

2

Review: 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 14, 2019

Sprenkle - CSCI111

3

What are the values?

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

➤ $r = 5$

➤ $s = -1 + r$

➤ $t = r + s$

➤ $s = 2$

➤ $r = -7$

Jan 14, 2019

Sprenkle - CSCI111

4

Programming Building Blocks

- Each type of statement is a building block

- Initialization/Assignment

- So far: Arithmetic

- Print

Assign.

print

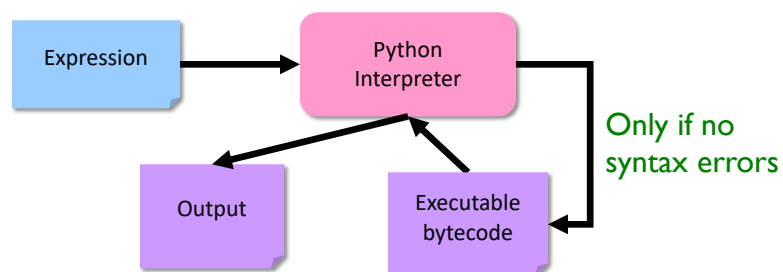
Jan 14, 2019

Sprenkle - CSCI111

5

Review: Python Interpreter

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



Jan 14, 2019

Sprenkle - CSCI111

6

Two Modes to Execute Python Code

- **Interactive/Shell**

- Try out Python expressions

- **Batch:** execute *scripts* (i.e., files containing Python code)

- What we'll write usually

Jan 14, 2019

Sprenkle - CSCI111

7

What are the values?

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

- $r = 5$

- $s = -1 + r$

- $t = r + s$

- $s = 2$

- $r = -7$

Try these expressions out in interactive mode!

Jan 14, 2019

Sprenkle - CSCI111

8

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 14, 2019

Sprenkle - CSCI111

9

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 14, 2019

Sprenkle - CSCI111

10

Programming Building Blocks

- Each type of statement is a building block

- Initialization/Assignment

- So far: Arithmetic

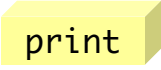
- Print

- We can combine them to create more complex programs

- Solutions to problems



Assign.



print



Assign.

print

Assign.

Assign.

print

Jan 14, 2019

Sprenkle - CSCI111

11

Bringing It All Together: A simple *program* or *script*

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

Comments: human-readable descriptions.
Computer does not execute.

```
x = 3
```

```
y = 5
```

```
print("x =", x)
```

```
print("y =", y)
```

```
result = x * y
```

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

arith_and_assign.py

Jan 14, 2019

Sprenkle - CSCI111

12

Bringing It All Together: A simple *program* or *script*

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

Comments: human-readable descriptions.
Computer does not execute.

```
x = 3
y = 5
```

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

```
# alternative to the previous program
print("x * y =", x * y)
```

arith_and_assign.py

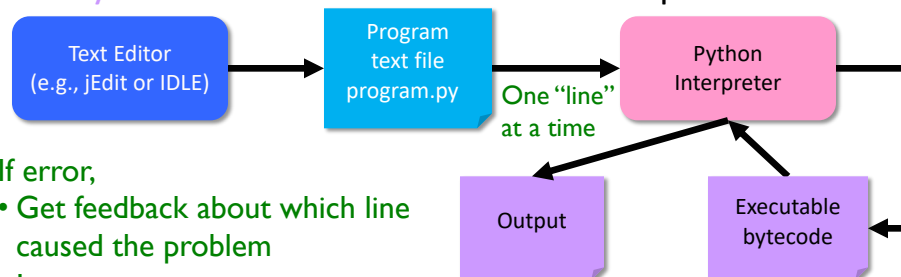
Jan 14, 2019

Sprenkle - CSCI111

13

Batch Mode: Execute Scripts

1. Programmer save a **program/script** into a **text file** using a **text editor**.
2. An **interpreter** turns each expression in file into **bytecode** and then executes each expression



If error,

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

Jan 15, 2019

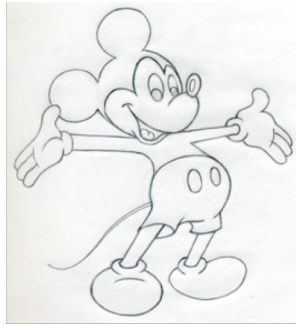
Sprenkle - CSCI111

14

Formalizing Process of Developing Computational Solutions

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

Use comments to describe the steps



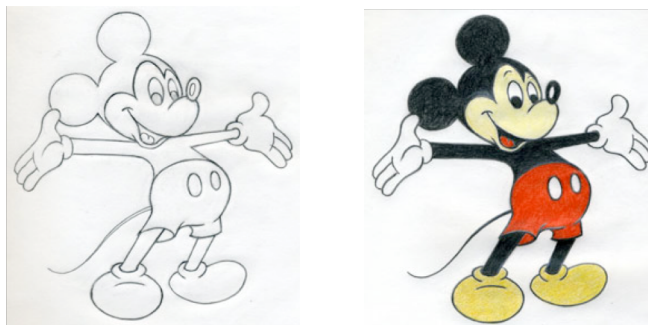
Jan 14, 2019

Sprenkle - CSCI111

15

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



Jan 14, 2019

Sprenkle - CSCI111

16

Errors

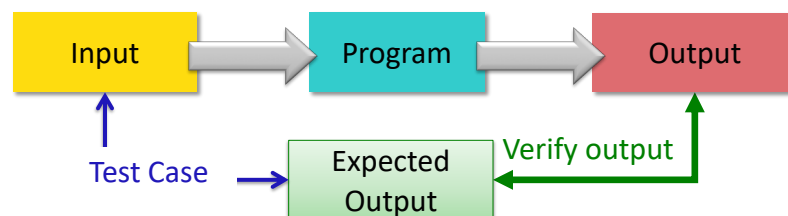
- Sometimes the program doesn't work
- Types of programming errors:
 - Syntax error
 - Interpreter shows where the problem is
 - Logic/semantic error
 - $\text{answer} = 2+3$
 - No, answer should be $2*3$
 - Exceptions/Runtime errors
 - $\text{answer} = 2/0$
 - Undefined variable name

Jan 14, 2019

Sprenkle - CSCI111

17

Testing Process



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

If output is not what you expect...

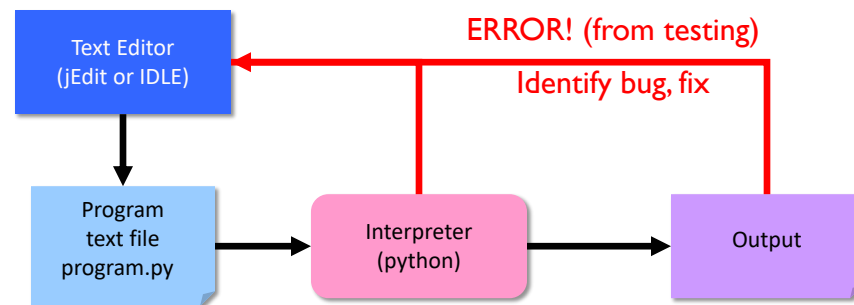
Jan 14, 2019

Sprenkle - CSCI111

18

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 14, 2019

Sprenkle - CSCI111

19

Practice: A Computational Algorithm

- Find the average of two numbers

Jan 14, 2019

Sprenkle - CSCI111

20

Practice: A Computational Algorithm

- Find the average of two numbers
- Test cases:

Input		Expected Output
num1	num2	

Jan 14, 2019

Sprenkle - CSCI111

21

A Computational Algorithm

- Algorithm for finding the average of two numbers:
 - Hard-code two numbers
 - Later: get the two numbers from user
 - Calculate average
 - Print average
- Test cases for finding the average
 - Test both integers
 - Test with at least one float
 - Test numbers less than or equal to 0

Jan 14, 2019

Sprenkle - CSCI111

average2.py

22

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

average2.py

Jan 14, 2019

Sprenkle - CSCI111

23

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 14, 2019

Sprenkle - CSCI111

24

Trick: Type Conversion

- You can convert a variable's type
 - Use the type's **constructor**


Conversion Function/Constructor	Example	Value Returned
int(<number or string>)	int(3.77)	3
	int("33")	33
float(<number or string>)	float(22)	22.0
str(<any value>)	str(99)	"99"

Jan 14, 2019

Sprenkle - CSCI111

25

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 14, 2019

Sprenkle - CSCI111

26

Interactive Programs

2.8 in Text Book

- Meaningful programs often need input from users
- Demo: `input_demo.py`

Jan 14, 2019

Sprenkle - CSCI111

27

Getting Input From User

- `input` is a *function*
 - **Function:** A command to do something
 - A “subroutine”
- Syntax:
 - `input(<string_prompt>)`
- Semantics:
 - Display the prompt `<string_prompt>` in the terminal
 - Read in the user’s input and *return* it as a string/text

Jan 14, 2019

Sprenkle - CSCI111

28

Getting Input From User

- Typically used in assignments
- Examples:

Prompt displayed to user

- `name=input("What is your name? ")`
 - `name` is assigned the string the user enters
- `width=eval(input("Enter the width:"))`
 - What the user enters is evaluated (as a number) and assigned to `width`
 - Use `eval` function because expect a number from user

What do you think the code looks like for `input_demo.py`?

Jan 14, 2019

Sprenkle - CSCI111

29

Getting Input from User

```
color = input("What is your favorite color? ")
```

Semantics: Sets the variable **color** to the user's input

Terminal:

Grabs every character up to the user presses "enter"

```
> python3 input_demo.py
What is your favorite color? blue
Cool! My favorite color is _light_ blue !
```

Jan 14, 2019

Sprenkle - CSCI111

`input_demo.py`

30

Restricting User's Inputs

```
>>> x = 7
>>> yourVal = input("My val is: ")
My val is: x
>>> print(yourVal)
x
```

Jan 14, 2019

Sprenkle - CSCI111

31

Restricting User's Inputs

```
>>> x = 7
>>> yourVal = input("My val is: ")
My val is: x
>>> print(yourVal)
x
>>> yourVal = eval(input("My val is: "))
My val is: x
>>> print(yourVal)
7
What happened here?
>>> yourVal = int(input("My val is: "))
My val is: x
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10:
'x'
```

Jan 14, 2019

Sprenkle - CSCI111

32

Identify the Parts of a Program

```
# Demonstrate numeric and string input
# by Sara Sprenkle for CS111
#

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

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

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

Jan 14, 2019

Sprenkle - CSCI111

input_demo.py

33

Identify the Parts of a Program

```
# Demonstrate numeric and string input
# by Sara Sprenkle for CS111
#

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

rating = eval(input("On a scale of 1 to 10, how much do
you like Chadwick Boseman? "))
print("Cool! I like him", rating*1.8, "much!")
                        expression
```

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

Jan 14, 2019

Sprenkle - CSCI111

34

Improving average2.py

- With what we just learned, how could we improve `average2.py`?
- Example of suggested approach to development
 - Input is going to become fairly routine.
 - Wait on input until you have figured out the rest of the program/problem.

Jan 14, 2019

Sprenkle - CSCI111

35

Design Patterns

- General, repeatable solution to a commonly occurring problem in software design
 - Template for solution

Jan 14, 2019

Sprenkle - CSCI111

36

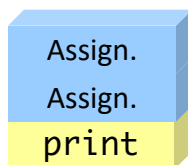
Design Patterns

- General, repeatable solution to a commonly occurring problem in software design

➤ Template for solution

- Example (Standard Algorithm)

- Get input from user
- Do some computation
- Display output



```
x = input("...")
ans = ...
print(ans)
```

Jan 14, 2019

Sprenkle - CSCI111

37

Looking Ahead

- Prelab 1 due tomorrow before lab
- Lab 1 due Friday
- Broader Issue due Friday

Jan 14, 2019

Sprenkle - CSCI111

38