

Objectives

- User input
- Software development practices
 - Testing
 - Debugging
 - Iteration
- Broader Issue: Algorithms

Jan 21, 2022

Sprenkle - CSCI111

1

1

Review

- What is our development process?
 - Programming, in general
 - For lab work
- What are the two division operators?
- How should you “read” this expression? What does it mean?
 - `rem = num1 % num2`

Jan 21, 2022

Sprenkle - CSCI111

2

2

Division Terminology

- Dividend: number being divided
- Divisor: by how much the dividend is divided
- Quotient: the result

- Example Python statement:
 - `quotient = dividend/divisor`

Jan 21, 2022

Sprenkle - CSCI111

3

3

Brainstorm

- What useful thing does `% 10` do?
 - `3 % 10 = 3`
 - `51 % 10 = 1`
 - `40 % 10 = 0`
 - `678 % 10 = 8`
 - `12543 % 10 = 3`
- What useful thing does `// 10` do (integer division)?
 - `3 // 10 = 0`
 - `51 // 10 = 5`
 - `40 // 10 = 4`
 - `678 // 10 = 67`
 - `12543 // 10 = 1254`
- What useful thing does `% 2` do?

Jan 21, 2022

Sprenkle - CSCI111

4

4

Brainstorm

- What useful thing does % 10 do?

- 3 % 10 =
- 51 % 10 =
- 40 % 10 =
- 678 % 10 =
- 12543 % 10 =

Gives the last digit in the number

- What useful thing does // 10 do (integer division)?

- 3 // 10 =
- 51 // 10 =
- 40 // 10 =
- 678 // 10 =
- 12543 // 10 =

Shifts the number right by one

- What useful thing does % 2 do?

Even/odd; alternating pattern

Jan 21, 2022

Sprenkle - CSCI111

5

5

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 21, 2022

Sprenkle - CSCI111

6

6

Trick: Arithmetic Shorthands

- Called **extended assignment operators**
- Increment Operator
 - $x = x + 1$ can be written as $x += 1$
- Decrement Operator
 - $x = x - 1$ can be written as $x -= 1$
- Shorthands are similar for $*$, $/$, $//$:
 - `amount *= 1.055`
 - `x //= 2`


Jan 21, 2022

Sprenkle - CSCI111

7

7

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 21, 2022

Sprenkle - CSCI111

8

8

Interactive Programs

2.8 in Text Book

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

Jan 21, 2022

Sprenkle - CSCI111

9

9

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 21, 2022

Sprenkle - CSCI111

10

10

Getting Input From User

- Typically used in assignments
- Examples:
 - `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
 - Alternatively, could use `int` or `float` (conversion functions) instead of `eval`

Jan 21, 2022

Sprenkle - CSCI111

11

11

Getting Input From User

- Typically used in assignments
- Examples:
 - `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
 - Alternatively, could use `int` or `float` (conversion functions) instead of `eval`

Jan 21, 2022

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

12

12

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 21, 2022

Sprenkle - CSC111

input_demo.py

13

13

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 Zendaya? ") )
print("Cool! I like her", rating*1.8, "much!")
```

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

Jan 21, 2022

Sprenkle - CSC111

input_demo.py

14

14

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 Zendaya? ") )
print("Cool! I like her", rating*1.8, "much!")
```

} expression

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

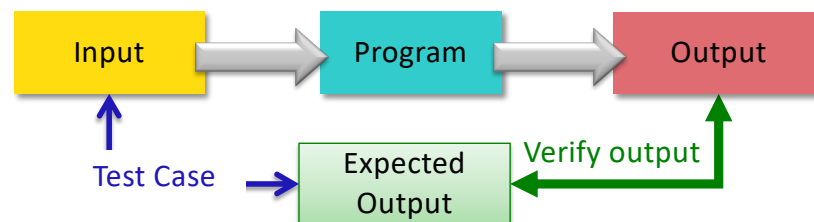
Jan 21, 2022

Sprenkle - CSCI111

15

15

Testing Process



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

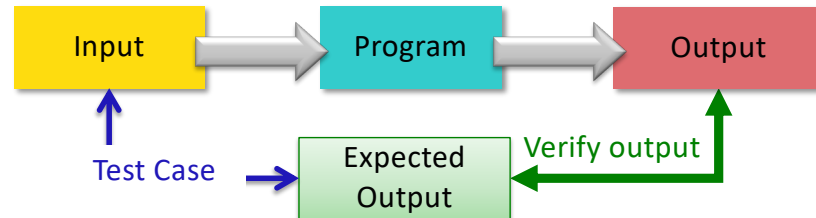
Jan 21, 2022

Sprenkle - CSCI111

16

16

Testing Process



- Test case:
 - input used to test the program
 - expected output given that input
- Verify if output is what you expected
- Goal: create *good* test cases that will reveal if there is a problem in your code

If output is not what you expect...

Jan 21, 2022

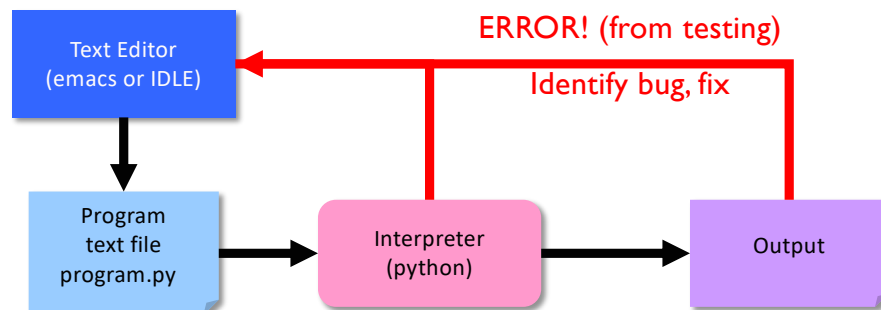
Sprenkle - CSCI111

17

17

Review: Debugging

- After executing program and output did not match what you expected
- 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 21, 2022

Sprenkle - CSCI111

18

18

Practice: A Computational Algorithm

- Find the average of two numbers
- Start the Process:
 1. Create a sketch of how to solve the problem (the algorithm)
 2. Fill in the details in Python
 3. Come up with good test cases for the problem

Jan 21, 2022

Sprenkle - CSCI111

19

19

Practice: A Computational Algorithm

- Find the average of two numbers
- Test Cases

Input		Expected Output
num1	num2	

Jan 21, 2022

Sprenkle - CSCI111

20

20

A Computational Algorithm

- Algorithm for finding the average of two numbers:
 1. “Hard-code” two numbers
 - Later: get the two numbers from user
 2. Calculate average
 3. 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 21, 2022

Sprenkle - CSCI111

average2.py

21

21

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. Execute the program **with good, varied test cases to try to reveal errors**
 4. If output doesn't match your expectation, debug the program
 - (Where is the problem? How do I fix it?)
-

Jan 21, 2022

Sprenkle - CSCI111

22

22

Suggested Approach to Development

- Input is going to become fairly routine.
- Wait to get user input until you have figured out the rest of the program/problem.

Jan 21, 2022

Sprenkle - CSCI111

23

23

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. Execute the program ***with good, varied test cases to try to reveal errors***
 4. If output doesn't match your expectation, debug the program
 - (Where is the problem? How do I fix it?)
 5. Iterate to improve your program
 - Better variable names, better input, more efficient, ...
-

Jan 21, 2022

Sprenkle - CSCI111

24

24

Design Patterns

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

Jan 21, 2022

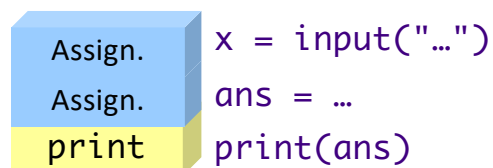
Sprenkle - CSCI111

25

25

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



Jan 21, 2022

Sprenkle - CSCI111

26

26

Broader Issue: Typical Process

1. Break into assigned groups
2. Introduce yourselves
3. Answer questions in groups
4. Discuss in class

Jan 21, 2022

Sprenkle - CSCI111

27

27

Groups

Cassandra
Declan
Matt
Patrick
Shelby

Aiden
Amanda
Ellie
Lakpa
Renan

Ford
Han
Jenna
Mac

Cole
Jack
Mary
Nick

Jan 21, 2022

Sprenkle - CSCI111

28

28

Broader CS Issues

- Good summaries!
 - Good English, complete sentences
 - Followed the specifications
- Good, thoughtful questions
 - A lot are teasers to what I hope we'll talk about later this term
- Interest scale is 0 to 9
 - Recall: Lab 0
 - Why we start at 0 will be clearer soon...

Jan 21, 2022

Sprenkle - CSCI111

29

29

Algorithms Everywhere

- How does knowing how your brain thinks about code affect how you think about code?
- Comment on these from articles:
 - "Because it's less familiar, *algorithm* tends to emphasize our uncertainty."
 - "An algorithm is, essentially, a brainless way of doing clever things."
- What are examples of algorithms that you do every day?
- What is machine learning useful for?
- What aren't algorithms useful for?
- What would be some useful algorithms, specific to W&L students?
 - What are problems that are difficult—but useful—to solve?

Jan 21, 2022

Sprenkle - CSCI111

30

30

My Corrections to Articles

- “In his book *The Master Algorithm*, Pedro Domingos offers a masterfully simple definition: ‘An algorithm is,’ Domingos writes, ‘a sequence of instructions telling a ~~computer~~ what to do.’”
- “An algorithm is, essentially, a ~~brainless~~ way of doing clever things.”

Jan 21, 2022

Sprenkle - CSCI111

31

31

Looking Ahead

- Pre Lab due Tuesday before lab
- Broader Issue:

Jan 21, 2022

Sprenkle - CSCI111

32

32