

Objectives

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

Jan 20, 2023

Sprenkle - CSC1111

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`
- How can we convert an integer to a string? Or a string to an integer?
- How should you “read” this statement? What does it mean?
 - `value += 2`
- How do we get input from a user?
 - Give example of getting input from a user, one where we want a string and one where we want a number
- Complete labeling the program from last time

Jan 20, 2023

Sprenkle - CSC1111

2

2

Review: 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) int("33")	3 33
float(<number or string>)	float(22)	22.0
str(<any value>)	str(99)	"99"

Jan 20, 2023

Sprenkle - CSC1111

3

3

Review: 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 $*$, $/$, $//$, $\%$, $**$:
 - $\text{amount} *= 1.055$
 - $x //= 2$

Jan 20, 2023

Sprenkle - CSC1111

4

4

Review: 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 20, 2023

Sprenkle - CSCI111

5

5

What Happens If ...?

Program:

```
str_age = input("Enter your age: ")
age = int(str_age)
```

Executing:

```
Enter your age: twelve
```

User enters a *string* but you were expecting an *integer*!

Jan 20, 2023

Sprenkle - CSCI111

6

6

What Happens If ...?

Program:

```
str_age = input("Enter your age: ")  
age = int(str_age)
```

Executing:

```
Enter your age: twelve  
Traceback (most recent call last):  
  File "/Users/sprenkles/Library/CloudStorage/Box-  
Box/CSCI111/inclass/03-input/int_input.py", line  
5, in <module>  
    age = int(str_age)  
ValueError: invalid literal for int() with base  
10: 'twelve'
```

Jan 20, 2023

Sprenkle - CSCI111

7

7

Restricting User's Inputs

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

Jan 20, 2023

Sprenkle - CSCI111

8

8

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 20, 2023

Sprenkle - CSC111

9

9

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 20, 2023

Sprenkle - CSC111

input_demo.py

10

10

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 20, 2023

Sprenkle - CSCI111

11

11

REFINING OUR DEVELOPMENT PROCESS

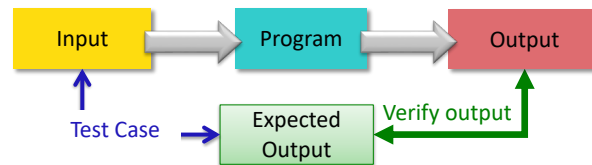
Jan 20, 2023

Sprenkle - CSCI111

12

12

Testing Process



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

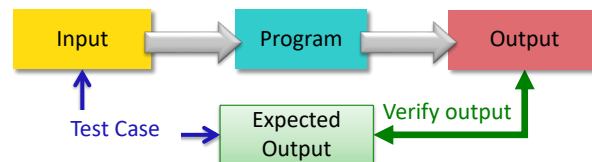
Jan 20, 2023

Sprenkle - CSCI111

13

13

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

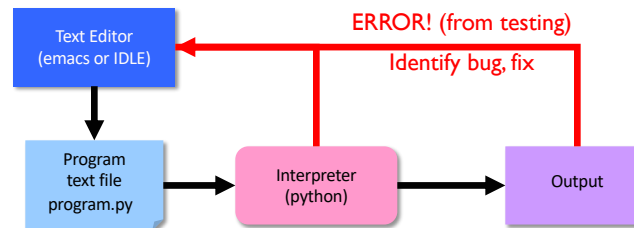
Jan 20, 2023

If output is not what you expect, we say that the program failed the test case.

14

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 20, 2023

Sprenkle - CSC1111

15

15

Practice: A Computational Algorithm

- Problem: Find the average of two numbers
- Process:
 1. Consider good test cases for the problem
 - Start thinking about expectations: “When user enters these inputs, this should be displayed.”
 2. Create a sketch of how to solve the problem (the algorithm)
 3. Fill in the details in Python

Jan 20, 2023

Sprenkle - CSC1111

16

16

Practice: Development Process

- Problem: Find the average of two numbers
- Test Cases

Input		Expected Output
num1	num2	

Jan 20, 2023

Sprenkle - CSC1111

17

17

Good Test Cases for Finding the Average

- Test both integers
- Test with at least one float
- Test numbers less than or equal to 0

Jan 20, 2023

Sprenkle - CSC1111

`average2.py`

18

18

Practice: Develop Algorithm

- Problem: Find the average of two numbers

Jan 20, 2023

Sprenkle - CSC111

19

19

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

Jan 20, 2023

Sprenkle - CSC111

`average2.py`

20

20

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.
- Consider problem 1 in Lab 1
 - You “hard coded” the values of i and j
 - You can (and will) modify the program to get user input for those variables in Lab 2.


Jan 20, 2023

Sprenkle - CSC1111

21

21

Formalizing Process of Developing Computational Solutions

- 
1. Think about expectations/test cases
 - “When user enters these values, this should happen.”
 2. Create a sketch of how to solve the problem (the algorithm)
 3. Fill in the details in Python
 4. Execute the program **with good, varied test cases** to try to **reveal errors**
 5. If output doesn’t match your expectation, debug the program
 - (Where is the problem? How do I fix it?)
 6. Iterate to improve your program
 - Better variable names, better input/output, more efficient, ...

Jan 20, 2023

Sprenkle - CSC1111

22

22

Design Patterns

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

Jan 20, 2023

Sprenkle - CSC1111

23

23

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

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

Jan 20, 2023

Sprenkle - CSC1111

24

24

Broader Issue: Typical Process

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

Jan 20, 2023

Sprenkle - CSCI111

25

25

Groups

Amanda
Brian
David
Jackson
Michelle

Alicia
Elizabeth
Ethan
Libby
Matt

Charlie
Harrison
Micah
Ricardo
Winter

Justin
Kyle
Samantha
Sambridhi

Claire
Elias
Tim
Tyler

Jan 20, 2023

Sprenkle - CSCI111

26

26

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 20, 2023

Sprenkle - CSCI111

27

27

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 20, 2023

Sprenkle - CSCI111

28

28

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 20, 2023

Sprenkle - CSCI111

29

29

Looking Ahead

- Pre Lab due Tuesday before lab
- Broader Issue: Algorithm Bias

Jan 20, 2023

Sprenkle - CSCI111

30

30