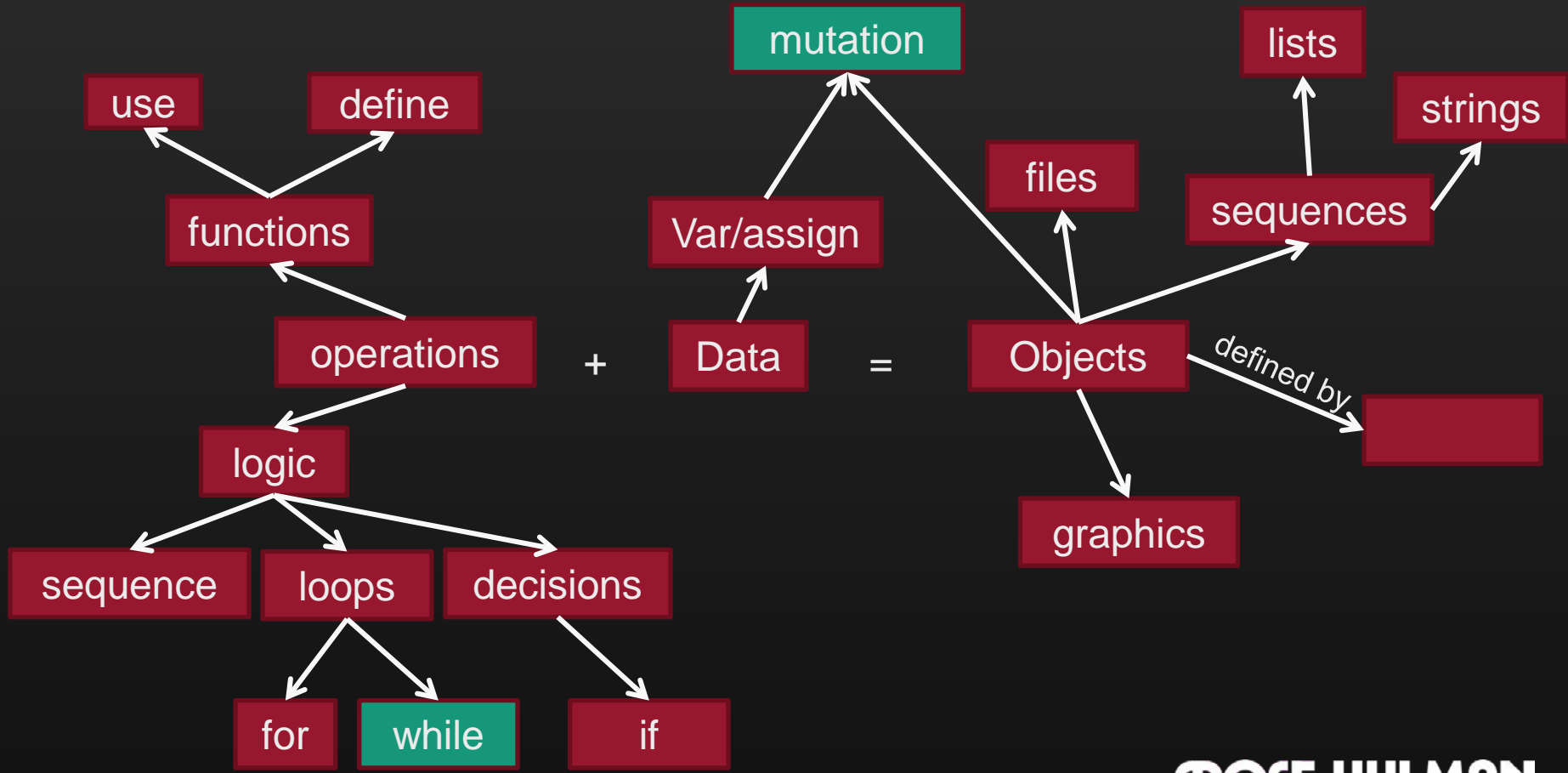


Loop Patterns

Rose-Hulman Institute of Technology

Computer Science and Software Engineering

Check out 13-LoopPatterns from SVN



Today's Goal

- Get better at solving problems that involve repetition by:
 - Seeing some different common loop patterns
 - Practicing with them!

Recap: Two main types of loops

- *Definite Loop*
 - We know at the beginning of the loop how many times its body will execute
 - Implemented in Python as a for loop
 - Cannot be an infinite loop
- *Indefinite loop*
 - The body executes as long as loop condition is True
 - Implemented in Python as a while statement
 - Can be an infinite loop if the condition never becomes False

Q1-2

Some Indefinite Loop Patterns

- Interactive loops
- Sentinel loops
- File loops

Today's
focus

- Post-test loops
- “Loop and a half”

Section 8.5
in Zelle

Q3

Interactive: Make the user count

- Open and run `averageUserCount.py`
- When does the loop terminate?
- Is this a user friendly way to get input?
 - Why?
 - Why not?

Interactive: More?

- Open and run `averageMoreData.py`
- Better?

Sentinel loop

- Open module **averageSentinel.py**
 - Study the code
 - Run it
- User signals end of data by a special *sentinel value*



Not used in calculation

Q5

Non-numeric Sentinel

- Suppose negative numbers are legitimate?
- Open module **averageOtherSentinel.py**
 - Study the code
 - Run it
 - What's the sentinel?

Still not used in calculation

File Loop

- Open and run `averageFile.py`
 - Use input file `numbers.txt`

Escaping From a Loop

- **break** statement ends the loop immediately
 - Does not execute any remaining statements in loop body
- **continue** statement skips the rest of this iteration of the loop body
 - Immediately begins the next iteration (if there is one)
- **return** statement ends loop and function call
 - May be used with an expression
 - within body of a function that returns a value
 - Or without an expression
 - within body of a function that just does something

Q7

Interactive Loop with Graphics

- Implement in module **clickInsideCircle.py**:
 - Display a window that contains a circle and a message saying "**Click inside Circle**".
 - Whenever the user clicks outside the circle, display "**You missed!**". Continue accepting clicks
 - If the user clicks inside the circle, display "**Bull's eye!**". Then pause and close the window.

Individual Exercise on Using loops

- Define function **listAndMax()** in module **listMax.py** that
 - Prompts the user to enter numbers, one at a time
 - Uses a blank line (<ENTER>) as sentinel to terminate input
 - Accumulates the numbers in a list
 - Uses a loop to calculate the maximum value of the numbers
 - Returns two values:
 - the list of numbers entered in the order they were entered
 - the maximum value
- Define function **main()** that
 - Calls **listAndMax()**
 - Prints the list of numbers entered
 - Prints the maximum value of the list of numbers

Q8, 9

Start Homework

- When you are through with your individual exercise commit your solutions to your SVN repository
- Finish homework 11 if you haven't
- Continue working on homework 13