

Strings and Formatting

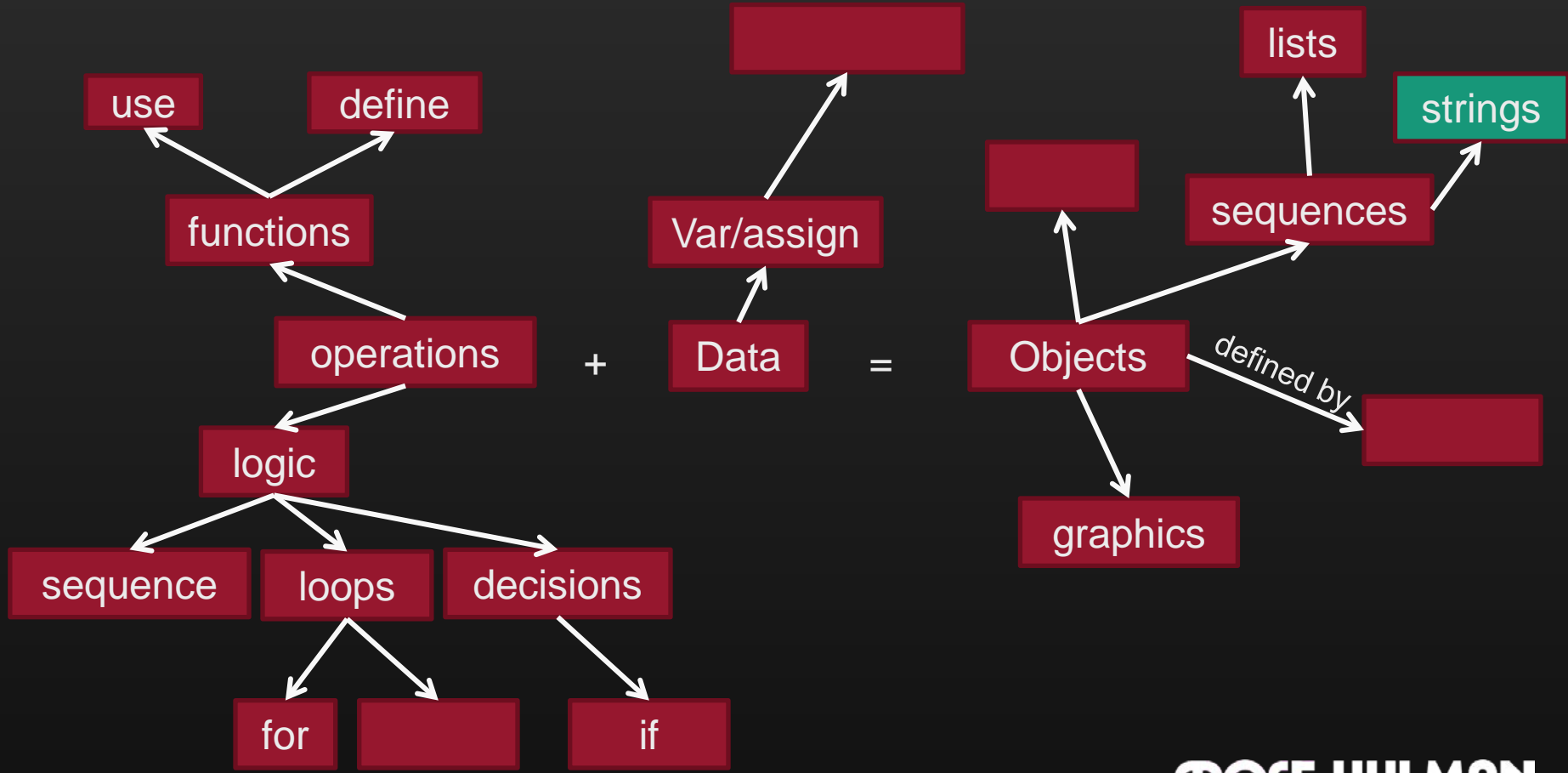
Rose-Hulman Institute of Technology

Computer Science and Software Engineering

Check out **09-StringsFormattingAndFiles** from SVN

Work here until dismissed at 9:55

Exam 1/2 during second half



Strings (sequences of characters)

- String literals (constants):
 - `"One\nTwo\nThree"`
 - `"Can't Buy Me Love"`
 - `'I say, "Yes." You say, "No." '`
 - `"'A double quote looks like this \", ' he said."`
 - `'''I don't know why you say, "Goodbye,"
I say "Hello." '''`

Q1,2

Operating on Strings

Reference

Operations/Methods	What does each of these operation/method do?
<code>s1 + s2</code>	Concatenates two strings e.g. "xyz" + "abc" ⇒ "xyzabc"
<code>s * <int></code>	Replicates string s <int> times e.g. "xyz" * 4 ⇒ "xyzxyzxyzxyz"
<code>s.capitalize()</code>	Copy of s with only 1 st letter capitalized
<code>s.lower()</code>	Copy of s with all lower case characters
<code>s.split()</code>	List of the words in s (split on spaces by default)
<code>s.rjust(width)</code>	Copy of s with enough spaces to start to make len(copy) = width
<code>s.center(<int>, [<str>])</code>	Copy of s padded on either end to be <int> characters long (padded with spaces by default, or else optional <str>)

Some more string methods

Reference

Methods	What does each of these operation/method do?
<code>s.count(sub)</code>	Returns the number of occurrences of sub in s
<code>s.find(sub)</code>	Returns the first position (index, 0-based) where sub occurs in s
<code>s.title()</code>	Copy of s with first character of each word capitalized
<code>s.replace(old, new)</code>	Copy of s where all occurrences of old in s have been replaced with new
<code>s.lstrip()</code>	Copy of s with leading white space removed
<code>s.strip()</code>	Copy of s with leading and trailing white space removed
<code>s.join(list)</code>	Concatenate list into a string, using s as the separator between items in the list

Practice with string operations

- `>>> franklinQuote = 'Who is rich? He who is content. ' + 'Who is content? Nobody.'`
- `>>> franklinQuote.lower()`
- `'who is rich? he who is content. who is content? nobody.'`
- `>>> franklinQuote.replace('He', 'She')`
- `'Who is rich? She who is content. Who is content? Nobody.'`
- `>>> franklinQuote.find('rich')`

Q3,4

Review: Strings are immutable

- Lists are mutable:
 - `>>> colors = ["red", "white", "blue"]`
 - `>>> colors[1] = "grey"`
 - `>>> colors.append("cyan")`
- A string is an immutable sequence of characters
 - `>>> building = "Taj Mahal"`
 - `>>> building[2]`
 - `>>> building[1:4]`
 - `>>> building[4] = "B" # Error!`

Q5,6

Strings and Lists

- A String method: `split` breaks up a string into separate words
 - `>>> franklinQuote = 'Who is rich? He who is content.' + 'Who is content? Nobody.'`
 - `>>> myList = franklinQuote.split(' ')`
`['Who', 'is', 'rich?', 'He', 'who', 'is', 'content.', 'Who', 'is', 'content?', 'Nobody.']`
- A string method: `join` creates a string from a list
 - `>>> '#'.join(myList)`
 - `>>>`
`'Who#is#rich?#He#who#is#content.#Who#is#content?#Nobody.'`
- What is the value of `myList[0][2]`? `myList[2][0]`?

Lists of Strings

```
>>> beatles = ['John', 'Paul']
>>> beatles.append('George')
>>> beatles
['John', 'Paul', 'George']
>>> beatles + ['Ringo']
['John', 'Paul', 'George', 'Ringo']
>>> beatles
['John', 'Paul', 'George']
```

```
>>> beatles = beatles + ['Ringo']
>>> beatles
['John', 'Paul', 'George', 'Ringo']
>>> beatles[1]
'Paul'
>>> beatles[1][2]
'u'
```

Getting a string from the user

```
>>> name = input('Enter your name:')
Enter your name:John
>>> name
'John'
>>>
```

You'll need input,
string methods, lists,
and print!

Q8

String Representation

- Computer stores 0s and 1s
 - Numbers are stored as 0s and 1s
 - What about text?
- Text also stored as 0s and 1s
 - Each character has a code number
 - Strings are sequences of characters
 - Strings are stored as sequences of code numbers
 - Does it matter what code numbers we use?
- Translating: **ord(<char>)** and **chr(<int>)**

Q9,10

Consistent String Encodings

- Examples:
 - ASCII—American Standard Code for Info. Interchange
 - “Ask-ee”
 - Standard US keyboard characters plus “control codes”
 - 7 bits per character
 - Extended ASCII encodings (8 bits)
 - Add various international characters
 - Unicode (16+ bits)
 - Tens of thousands of characters
 - Nearly every written language known

String Formatting

- Use `format()` for complex text output
 - Uses a *template* string with *slots*
 - `t = "All {0} wants for {2} is his {1} front teeth"`
 - Values to format are plugged into each slot
 - The string method `format` does the plugging
 - `<template-string>.format(<values>)`
 - `t.format("Delvin", 2, "Christmas")`

Returns a string, rather than printing it.

String Formatting

- What does each slot look like?
 - {<index>:<format-specifier>}
 - <index> tells which of the parameters is inserted in slot
 - Optional <format-specifier> lets us do fancier formatting

Format Specifiers

- Syntax:
 - **<width>.<precision><typeChar>**
- **<width>**: total spaces to use. Notes:
 - 0 (or omitted) means as many as needed
 - 0*n* means pad with leading 0s to *n* total spaces
 - <*n* means “left justify” in the *n* spaces
 - ^*n* means “center justify” in the *n* spaces
- **<precision>** gives digits after decimal point, rounding if needed.
- **<typeChar>** is:
 - **f** for float, **s** for string, or **d** for decimal (i.e., int)

Q12,13

No communication with others about exam until noon

Exam 1/2.