Compsci 101
Selection, Lists, Sequences, Totem

Susan Rodger
January 20, 2022
E is for …

- Escape Sequence
  - Why \n is newline and \t is a tab
- Encryption
  - From Caesar Ciphers to SSL and beyond
- Enumerate
  - Iterating over data, counting
- Emoticon
  - 😄 😪
Luis von Ahn, Guatemalan entrepreneur
Duke BS Math 2000, CMU PhD CS

“I build systems that combine humans and computers to solve large-scale problems that neither can solve alone. I call this Human Computation, but others sometimes call it crowdsourcing.”

"In college, I thought my goal in life was to get a good GPA, but it's equally important to get involved with a good professor doing good research. Take advantage of what's going on around you."
Announcements

• APT-1 is due tonight! 11:30pm
  • Run each APT on the APT tester, 1 grace day

• QZ01-05 extended to Sat night 11:30pm (drop/add)
  • Remaining reading quizzes turn off 10:15am on due date

• Assignment 1 Faces is out, due Jan 27
  • Read the whole thing
  • Take assignment 1 quiz on Sakai – Due Jan 25

• Lab 2 Friday
  • Prelab 2 do before attending lab

• Always, Reading and Sakai quiz before next class

1/20/22 Compsci 101, Spring 2022
PFTD

• Assignment 1
• Selection continued
• Strings
  • Sequence of characters, “CompSci 101”
• Lists
  • Heterogenous sequences
• Sequences
  • len(...), indexing, and slicing
Finish WOTO-3 from last time
What does the animal say?

```python
import random

s += "What does a " + animal + " say?\n"
which = random.randint(0,1)

if which == 1:
    s += otherSound1 + "? No. "
    s += otherSound2 + "? No. "
else:
    s += otherSound2 + "? No. "
    s += otherSound1 + "? No. "

s += sound + "? Yes!\n"
```
Assignment 1: Faces
Learning Goals: Faces

• Understand differences and similarities:
  • Function definitions vs function calls
  • Functions with return statements vs those without
  • Functions with parameters vs those without
  • Functions can be arguments

• Be creative and learn lesson(s) about software design and engineering
  • Create a small, working program, make incremental improvements.
  • Read the directions and understand specifications!
## Function Name Format

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Returns</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>part_DESCRIPTION</td>
<td>No parameters</td>
<td>A string</td>
<td>part_smiling_mouth</td>
</tr>
<tr>
<td>DESCRIPTION_face</td>
<td>No parameters</td>
<td>No return value, only prints</td>
<td>happy_face</td>
</tr>
<tr>
<td>face_with_DESCRIPTION</td>
<td>1 or 2 parameters of type function</td>
<td>No return value, only prints</td>
<td>face_with_mouth</td>
</tr>
<tr>
<td>faces_DESCRIPTION</td>
<td>No parameters</td>
<td>No return value, calls face functions</td>
<td>faces_fixed, faces_selfie, faces_random</td>
</tr>
</tbody>
</table>

selfie_band, face_random – helper functions!
Creating your program

- Start small and build incrementally
- Use seven steps! Plan what to do!
With functions grow by…

Minimal code that does run and can be submitted

Where go from here?

• Add face part functions to create happy_face()
• Create the next face function for faces_fixed and any new face part functions
• Try a face_with function
• Go to the next group of faces
• etc.
Faces Assignment
What should you do …

• Read the assignment
• Do the Assignment 1 reading quiz
• Create project and start writing code (do not need to finish)

• Goal: Find your first question about how to do this assignment then ask on Ed Discussion (anonymously) or at consulting/office hours
Selection Syntax

- What is similar and different?
  - What other variations could work?
  - Could only `elif...else` work?
- `if` – required
- `elif` – optional, as many as needed
- `else` – optional, no condition

```
if BOOLEAN_CONDITION:
    CODE_BLOCK_A
else:
    CODE_BLOCK_B
elif BOOLEAN_CONDITION:
    CODE_BLOCK_B
else:
    CODE_BLOCK_C
```
Boolean condition (True/False)

```python
if BOOLEAN_CONDITION:
    CODE_BLOCK_A
```

• See `type(3 < 5)`
• Relational operators: `< <= > >= == !=`
• Boolean operators: `and or not`
Console on Booleans

```python
import sys; print('Python %s on %s'
sys.path.extend(['C:\Users\Susan'])
```
# Boolean Operations

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>A and B</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>A and B</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>A and B</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>A and B</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>A or B</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>A or B</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>A or B</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>A or B</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>not A</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>not A</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

**Example Sentences**

- **IF** it is raining **OR** it might rain today, **I will** carry an umbrella.
- **IF** my cat is hungry **AND** she likes the food, **she will** eat dinner.
- **IF** I did **NOT** have dessert yesterday, **I may** have dessert today.
Example with And and Or

```python
x = 3
y = 8
if x < 2 or y > 2:
    print("first")
elif x > 2 and y < 2:
    print("second")
else:
    print("third")

x = 3
y = 2
if x < 2 or y > 2:
    print("first")
elif x > 2 and y < 2:
    print("second")
else:
    print("third")
```
WOTO-1 Review Functions and Booleans

• In your groups:
  • Come to a consensus

<table>
<thead>
<tr>
<th>A and B</th>
<th>A</th>
<th>B</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
When is a leap year?

- [https://en.wikipedia.org/wiki/Leap_year](https://en.wikipedia.org/wiki/Leap_year)

- “years which are multiples of four (with the exception of years divisible by 100 but not by 400)”

- $2004/4 = 501$, $2004/100 = 20.04$, $2004/400 = 5.01$
  - Leap year

- $2200/4 = 550$, $2200/100 = 22$, $2200/400 = 5.5$
  - Not Leap Year

- $2000/4 = 500$ and $2000/100 = 20$, $2000/400 = 5$
  - Leap Year
WOTO-2: Which LeapYear correct?

• is_leap_one
• is_leap_two
Which LeapYear correct?

• Is 1900 a leap year?

• Which program is correct?
• What is wrong with the program that is not correct?
if’s or if…elif…else?

• Remember steps 1-4 do not involve code!
• After have plan, choose based on what works best
  • There could be multiple ways to implement it
Strings

• $x = \text{“chair”}$
• $y = \text{“desk”}$
• $w = \text{len}(x)$
• $v = x[\text{len}(y)]$
• $t = x[\text{len}(x)]$
Lists

- **Syntax**: `[ITEM_1, ITEM_2, ITEM_3, ...]`
  - Starts and ends with square brackets: `[ ... ]`
  - Elements in the list are divided by commas “,”
- Lists can be *heterogenous* sequence
  - Strings, ints, lists, anything

```
[1, 2, 3]
["hello", "world"]
["count", "off", 1, 2, 3.0, "done"]
```
Python Sequences

• Types String and List are both sequences
• A sequence in Python has
  • Length - `len(...)`
  • Membership – `in`
  • Indexing and slicing – `[n], [n:m]`
• Difference:
  • String is immutable – cannot change
  • List is mutable – can change
len(...) for Python Sequences

• Length – the number of *elements* in a sequence
• len(...) – returns the length of a sequence

s="hello world"  l=["hello", "world"]
  • What is len(s)?

  • What is len(l)?
in for Python Sequences

- `in` checks for membership in the sequence
  - True/False – if `element in seq`

- `s="hello world"` `lst=["hello", "world"]`
  - What is an element for the string `s`? List `lst`?
    - What is ‘h’ in `s`?
    - What is ‘h’ in `lst`?
    - “hello” in `lst`?
Indexing Python Sequences

- `s="hello world" l=["hello", "world"]`
- Indexing provides access to individual elements
  - Compare `s[0]` and `l[0]`
    - Start with 0 offset, what is last valid positive index?
  - Compare `s[-1]` and `l[-1]`
    - What is negative index of second to last element?
  - Index \(-n\) is the same as index `len(seq) - n`
Slicing Python Sequences

- \( s = "hello\ world" \)
- \( lst = ["my", \ "big", \ "beautiful", \ "world"] \)
- Slicing provides sub-sequence (string or list)
  - \( \text{seq}[n:m] \) – all elements \( i \), s.t. \( n \leq i < m \)
  - Compare \( s[0:2] \) and \( lst[0:2] \)
    - \( s[0:2] \) is "he"
    - \( lst[0:2] \) is ["my", "big"]
  - What is length of subsequence? \( \text{len}(lst[1:3]) \)
    - \( lst[1:3] \) is ["big", "beautiful"]
    - \( \text{len}(lst[1:3]) \) is 2
Slicing Python Sequences (more)

• s = "hello world"
• lst=["my", "big", "beautiful", "world"]
• Slicing provides sub-sequence (string or list)
  • Compare s[4:-1] and lst[2:-1]
    • s[4:-1] is
    • lst[2:-1] is
  • Is last index part of subsequence?

• Omit last value. Compare s[2:] , s[:3]
  • s[2:] is
  • s[:3] is
WOTO-3 Sequence Length Indexing

• In your groups:
  • Come to a consensus