Compsci 101
Selection, Lists, Sequences, Totem

Susan Rodger
September 13, 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

• Email
  • a way to communicate

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 Thur, Sept 15! 11:30pm
  • Run each APT on the APT tester, 1 grace day
  • Check your grade – click check submissions

• QZ01-05 turned off at 10:15am today!
  • Be sure to do QZ06 by 10:15am on Thursday!

• Assignment 1 Faces is out, program due Sept 22
  • Read the whole thing
  • Take assign1 quiz on Sakai – Due Sept 20

• Lab 2 Friday
  • Prelab 2 do before attending lab, out today
  • Always: Reading and Sakai quiz before next class

duolingo

Forbes
Why is this person so important to this course?

• Brad Miller, Runestone
• He built the Runestone infrastructure for online textbooks.
• Our Textbook is on his Runestone platform!
• Have you donated yet?
  • Everyone should give a $10-$20 donation

Top 10 list for surviving in CompSci 101

10. Read the book and ask questions
9. Eat lots of pizza
8. Learn how to spell Rodger
7. Understand what you turn in
6. Visit your prof in her office hours on zoom and the UTAs in consulting hours

Top 10 list (cont)

5. Check Ed Discussion every day
4. Learn how to debug your programs
3. Follow the 7 step process
2. Seek help (One Hour Rule!)
1. Start programming assignments early
One Hour Rule for Getting Help

- Work on Material
- Stuck
- Has it been an hour?
- Yes: Get Help
- No

PFTD

- Finish WOTO from last time
- Assignment 1
- Selection continued
- Strings
  - Sequence of characters, “CompSci 101”
- Lists
  - Heterogenous sequences
- Sequences
  - len(...), indexing, and slicing

Go over 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"
```
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"
```

Run Twice - Different Output

```python
def verse(animal, sound, otherSound1, otherSound2):
    if __name__ == '__main__':
        print(verse("pig", "oink", "bark", "baa"))

Old MacDonald had a farm, E-I-E-I-O
And on his farm he had a pig, E-I-E-I-O
What does a pig say?
bark? No. baa? No. oink? Yes!
With an oink oink here and an oink oink there
Here an oink, there an oink
Everywhere an oink, oink
Old MacDonald had a farm, E-I-E-I-O

Old MacDonald had a farm, E-I-E-I-O
And on his farm he had a pig, E-I-E-I-O
What does a pig say?
baa? No. bark? No. oink? Yes!
With an oink oink here and an oink oink there
Here an oink, there an oink
Everywhere an oink, oink
Old MacDonald had a farm, E-I-E-I-O
```

Assignment 1 and Pre-Lab 2

- **Assignment 1 Faces due Sept 22**
- **Sakai Quiz on Assignment 1**
  - Read through assignment 1
  - Take the quiz
  - Can take many times
  - Due Sept 20 (no grace day!)
- **Prelab 02 – before lab**
  - Read Assignment 1 and take its quiz once
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>

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 Sakai 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

Review Selection Syntax

```python
if BOOLEAN_CONDITION:
    CODE_BLOCK_A
else:
    CODE_BLOCK_B
```

- 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

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

Boolean Operations

<table>
<thead>
<tr>
<th>A and B</th>
<th>A and B</th>
<th>A and B</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A or B</th>
<th>A or B</th>
<th>A or B</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>not A</th>
<th>not A</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

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")
```

OUTPUT:

```
third
```
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")
```

OUTPUT:
```
f
```

Example with And and Or

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

OUTPUT:
```
first
```

WOTO-1 Review Functions and Booleans

• In your groups:
  • Come to a consensus

<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>
</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 (except NOT if 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
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 (except NOT if years divisible by 100 but not by 400)”

- \(2004/4 = 501, \ 2004/100 = 20.04, \ 2004/400 = 5.01\)
- \(2200/4 = 550, \ 2200/100 = 22, \ 2200/400 = 5.5\)
- \(2000/4 = 500 \text{ and } 2000/100 = 20, \ 2000/400 = 5\)


- is_leap_one
- is_leap_two

```
def is_leap_one(year):
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    if year % 4 == 0:
        return True
    return False
```

```
def is_leap_two(year):
    if year % 4 == 0:
        return True
    if year % 100 == 0:
        return False
    if year % 400 == 0:
        return True
    return False
```

Which LeapYear correct?

- Is 1900 a leap year?
- Which program is correct?
- What is wrong with the program that is not correct?

- If statements in a different order!
  - The order matters!
Which LeapYear correct?

- Is 1900 a leap year? **NO**
  - 1900/4 = 475
  - 1900/100 = 19
  - 1900/400 = 4.75 Not divisible by 400

- Which program is correct? **Is_leap_one**
- What is wrong with the program that is not correct?
  - If statements in a different order!
  - The order matters!

Wikipedia Leap Year Algorithm

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

```python
def is_leap(year):
    if year % 4 != 0:
        return False       # not leap
    elif year % 100 != 0:  # 1968
        return True
    elif year % 400 != 0:
        return False       #1968
    else:
        return True           #2000
```

Flowchart: if vs if…elif…else

![Flowchart](image)

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