Compsci 101
Lists, Mutation, Objects

Susan Rodger
September 15, 2022

F is for ...

- Function
  - Key to all programming
- Floating Point
  - Decimal numbers aka Python float
- File
  - Sequence of stored bits

Genesis Bond ‘16

- Struggled at Duke
  - 5 years
- Revature
  - Trainer Full Stack Development
  - She worked smarter
- Meta (Facebook)
  Engineer, big success!

“Poor preparation promotes poor performance. In anything you do, your preparation will show.”

Announcements

- Assign 1 Faces, Sakai QZ due Sept 20 (no grace day)
  - Program is due Sept 22!
- Lab 2 Friday, Do Prelab 2 before lab
- Sakai QZ due by lecture time each day
- Exam 1 – Tuesday, Sept 27
  - In person during class
  - Go over next time, see old exams
- Need SDAO letters for exams!
  - Email them to Ms. Velasco
    yvelasco@cs.duke.edu
Sage Learning Communities (Optional)

- STEM Advancement through Group Engagement (SAGE) Learning Communities are now available for CompSci 101!
- Strengthen programming skills!
- Hands on collaborative programming activities
- Sign up at:  https://arc.duke.edu/sage

PFTD

- Strings, Lists, Sequences, Slicing
- Functions as Parameters
- Debugging
- List concatenation and nesting
- Mutability
- Objects and what that means

Strings

- x = "chair"
- y = "desk"
- w = len(x)
- v = x[ len(y) ]
- t = x[ len(x) ]

Strings

- x = "chair"
- y = "desk"
  z is  "ask"
- w = len(x)  
  w is  5
- v = x[ len(y) ]  
  v is  "r"
- t = x[ len(x) ]  
  t is  ERROR !!!!!!!
Strings

- \( x = \text{“chair”} \)
- \( y = \text{“desk”} \)
- \( w = \text{len}(x) \) \( w \text{ is 5} \)
- \( v = x[\text{len}(y)] \) \( v \text{ is “r”} \)
- \( t = x[\text{len}(x)] \) \( t \text{ is ERROR !!!!!!!!} \)

Lists

- Syntax: \([\text{ITEM}_1, \text{ITEM}_2, \text{ITEM}_3, \ldots]\)
- Starts and ends with square brackets: \([\ldots]\)
- Elements in the list are divided by commas \(,\)
- Lists can be \textbf{heterogenous} sequence
  - Strings, ints, lists, anything

\[
\begin{bmatrix}
0 & 1 & 2 & 3 & 4 \\
\end{bmatrix}
\]

Python Sequences

- Types String and List are both sequences
- A sequence in Python has
  - Length - \texttt{len(...)}
  - Membership – in
  - Indexing and slicing – \([n], [n:m]\)
- \textbf{Difference}:
  - String is immutable – cannot change
  - List is mutable – can change

\texttt{len(...)} for Python Sequences

- Length – the number of \texttt{elements} in a sequence
- \texttt{len(...)} – returns the length of a sequence

\[
\begin{bmatrix}
1, 2, 3 \\
“hello”, “world” \\
“count”, “off”, 1, 2, 3.0, “done”
\end{bmatrix}
\]

\[
\begin{bmatrix}
s=“hello world” \quad l=“hello”, “world”
\end{bmatrix}
\]

- What is \texttt{len(s)}?
- What is \texttt{len(l)}?
### len(...) for Python Sequences

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

```python
s = "hello world"  l = ["hello", "world"]
- What is `len(s)`?
  - 11
- What is `len(l)`?
  - 2
```

### in for Python Sequences

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

```python
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

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

```python
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`?
```

```python
Indexing provides access to individual elements
- Compare `s[0]` and `l[0]`  "h" vs "hello"
  - Start with 0 offset, what is last valid positive index?
- Compare `s[-1]` and `l[-1]`  "d" vs "world"
  - What is negative index of second to last element?
- Index `-n` is the same as index `len(seq) - n`
  -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
  -0 1 2 3 4 5 6 7 8 9 10
  | HELLO WORLD |
  | -11 -10 -9 -8 -7 -6 -5 -4 -3 2 1 |
```
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`

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>E</td>
<td>L</td>
<td>L</td>
<td>O</td>
<td>W</td>
<td>O</td>
<td>R</td>
<td>L</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>-11</td>
<td>-10</td>
<td>-9</td>
<td>-8</td>
<td>-7</td>
<td>-6</td>
<td>-5</td>
<td>-4</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

Slicing Python Sequences

- `s = "hello world"
- lst=["my", "big", "beautiful", "world"]

- Slicing provides sub-sequence (string or list)
  - `seq[n:m]` – all elements i, s.t. `n <= 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? `len(lst[1:3])`
    - `lst[1:3]` is  
      - ["big", "beautiful"]
    - `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  
      - "lo world"
    - `lst[2:-1]` is  
      - ["beautiful"]
  - Is last index part of subsequence?
    - NO
  - Omit last value. Compare `s[2:]`, `s[:3]`
    - `s[2:]` is  
      - ["lo world"]
    - `s[:3]` is  
      - ["hel", "hel"]

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 "o worl"
    - `lst[2:-1] is ["beautiful"]
  - Is last index part of subsequence?
    - NO, in `s[2:4] we go up to but not including 4
  - Omit last value. Compare `s[2:], `s[:3]
    - `s[2:] is "llo world"
    - `s[:3] is "hel"

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!

WOTO-1 Sequence Length Indexing

- **In your groups:**
  - Come to a consensus

Name vs Value vs Type
What are the arrows?

- **Name:** Enzo’s Pizza Co.
- **Address (arrow):** 2608 Erwin Rd # 140, Durham, NC 27705
- **Value:** Physical Store

Functions can be arguments

```python
1 def enzospizzaco():
2     print("Pizza!")
3     return "2608 Erwin Rd # 140, Durham, NC 27705"
4
5 def eatfood(where):
6     print("Let's go eat!")
7     address = where()
8     print("The address is", address)
9
10 if __name__ == '__main__':
11     eatfood(enzospizzaco)
```

Pizza2.py - Pass multiple functions to eatfood

```python
7 def naanstop():
8     print("Indian cuisine!")
9     return "2812 Erwin Road, Durham, NC 27705"

10 def enzospizzaco():
11     print("Pizza!")
12     return "2608 Erwin Rd # 140, Durham, NC 27705"

13 def eatfood(where):
14     print("Let's go eat!")
15     address = where()
16     print("The address is", address)
17
18 if __name__ == '__main__':
19     eatfood(enzospizzaco)
20     eatfood(naanstop)
```
In Assignment 1 Faces

Two parameters that are functions!

```python
def face_with_mouthAndEyes(mouthfunc, eyefunc):
    print(part_hair_squiggly())
    print(eyefunc())
    print(part_nose_up())
    print(mouthfunc())
    print(part_chin_simple())
```

```python
def face_random():
    eyefunc = part_eyes_sideways
    mouthfunc = part_mouth_oh
    x = random.randint(1, 4)
    if x == 1:
        mouthfunc = part_mouth_frown
        eyefunc = part_eyes_ahead
    # now call the function
    face_with_mouthAndEyes(mouthfunc, eyefunc)
```

Variables whose values are function names

def face_random():
    eyefunc = part_eyes_sideways
    mouthfunc = part_mouth_oh
    x = random.randint(1,4)
    if x == 1:
        mouthfunc = part_mouth_frown
        eyefunc = part_eyesAhead
    <code not shown>

# now call the function
face_with_mouthAndEye(mouthfunc, eyefunc)

• Finding what is wrong + fixing it
  • Finding is its own skill set, and many find difficult
  • Fixing: revisit Step 1—5

How Not To Debug

• Bad (but tempting) way to debug
  • Change a thing. Does it work now?
  • No ... another change ... how about this?
• Trust doctor if they say?
  • “Ok try this medicine and see what happens?”
• Trust mechanic if they say?
  • “Let’s replace this thing and see what happens”

It may be easy, but that doesn’t make it a good idea!

Debugging Steps

1. Write down exactly what is happening
   1. input, output, what should be output
   2. ____ happened, but ____ should happen
2. Brainstorm possible reasons this is happening
   1. Write down list of ideas
3. Go through list
4. Found it?
   1. Yes, fix it using the 7-steps
   2. No, go back to step 2

This is what experts do!

Remember: One-hour rule
Debugging Steps

- Write down what is happening
- Brainstorm
- Go through list
- Found problem?
  - No
  - Yes! Fix it!

Relate W’s to Debugging

- Who was involved?
  - Which variables are involved?
- What happened?
  - What kind of error/bug is it?
- Where did it take place?
  - Where in the code did this happen?
- When did it take place?
  - Does it happen every time? For certain cases?
- Why/How did it happen?
  - Given the answers to the above, how did the error/bug happen?

Step 7 -> Steps 1-4 or 5
Which year is a leap year?

• A Leap Year must be divisible by four.
• But Leap Years don't happen every four years ... there is an exception.
  • If the year is also divisible by 100, it is not a Leap Year unless it is also divisible by 400.

WOTO: Buggy Leap Year

• Who? (Which variables)
  • year (only one)
• What kind of bug is it?
  • Semantic error
• Where in the code?
  • One of the places it returns True
• When does it happen?
  • Input: 1900, but not 2016 nor 2019
• Why/How did it happen?
  • A property 1900 has but not 2016 and 2019
Buggy Leap Year – add print tests

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

Output:
Is 2016 a leap year? (should be True) True
Is 2019 a leap year? (should be False) False
Is 1900 a leap year? (should be False) False

Buggy Leap Year – Which “return true”?

```python
def is_leap_year(year):
    if year % 4 == 0:
        print("DEBUG: if year % 4 == 0:")
        return True
    if year % 100 == 0:
        return False
    if year % 400 == 0:
        print("DEBUG: if year % 400 == 0:")
        return True
    return False
```

Output:
DEBUG: if year % 4 == 0:
Is 2016 a leap year? (should be True) True
Is 2019 a leap year? (should be False) False
DEBUG: if year % 4 == 0:
Is 1900 a leap year? (should be False) True

Correct Leap Year – ifs correct order

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

Why Leap Year Buggy?

- **Why:** Should not always return True if year is divisible by 4
- **Solution:** Check first for %400, then %100, and finally %4

![Diagram showing the steps of algorithmic problem-solving](steps.png)
List Concatenation

- String concatenation:
  - “hi” + “there” == “hi there”

- List concatenation:
  - [1, 2] + [3, 4] == [1, 2, 3, 4]

List examples

- [1, 2] + [3, 4]
- lst1 = ['a', 'b']
- lst2 = [5, 6]
- lst1 + lst2
- lst1 + “c”
- lst1 + [“c”]

Nested Lists

- Lists are heterogenous, therefore!
  - lst = [1, ‘a’, [2, ‘b’]] is valid
  - len(lst) == 3
    - [2, ‘b’] is one element in list lst
  - lst[2][1] returns ‘b’

- How to index?
  - [...] all the way down
  - lst[2][1] returns ‘b’