Reminders

• **Identity & Computing Lecture Series**
  - [https://identity.cs.duke.edu/speakerSeries.html](https://identity.cs.duke.edu/speakerSeries.html)
  - 9/20-Dr. Safiya Noble
  - 9/27-Dr. Michele Williams

• **Assignments**
  - APT-1 due today
  - Assign 1 out today
Key instructions

• Input
• Output
• Assignments* ✓
• Math/Logic ✓
• Conditionals ✓
• Repetition

*not listed in book
Python Data Types

• int, float, bool ✓
• Collections
  • Strings ←
  • Lists ←
  • Tuples
  • Sets
  • Dictionaries
PFTD

- Lists
- Sequences
- Debugging
  - PAY ATTENTION TO ERROR MESSAGES

“The mere imparting of information is not education.”
  - Dr. Carter G. Woodson
People to Know: 
Dr. Tessa Lau

- Cornell (BA, BS)
- University of Washington (MS, PhD)
- Founder/CEO, Dusty Robotics
- Co-founder/Chief Robot Whisperer, Savioke
Collection Data Type

- Collection of books, toys, shoes
  - Direct access to each item
- Comprised of smaller pieces
  - Strings and lists
- Strings
  - Smaller strings of size one char
  - Empty string- "" or ''
- Operations on strings
  - + → concatenation
  - * → repetition
Indexing a String

`string_name[index]`

- `string_name`
- index-character element directly accessing
  - Leftmost 0 to string_length-1

- What about `string_name[-1]`?
- **Whitespaces in a string count**
- `len()`-Python function

What is `result1[10]`?
Slicing Strings

- Real-world examples
  - Slicing bread, tomatoes, etc.
  - Substring (smaller part) of the larger string

\[ \text{string\_name}[n:m] \]

\( n \)-index of the first character in the substring
\( m \)-index of the character that immediately follows the last character in the substring

**Pro tip:** slicing only includes chars from \( n \) through \( m-1 \)
Comparing Strings

• Compares strings to determine the relationship between them
  • ==, >, <, >=, <=, !=

• string1 == string2

• Pro tip: Lexicographical order (A…Z, a…z)
  • ‘A’ < ‘a’
in and not in operators

• Is string1 a substring of string2?

```
string1 in string2
```

string can be a variable or a string literal (e.g., “This is literally an example of a string literal.”)
Activity 1: Strings
List

- Groceries, errands, names, etc.
- Collection of data values
  - Sequential
  - Directly access each element
  - Elements don’t have to be the same type

```
list_name=[item1, item2, ...item6]
```

**only top-level items in list**
List access and length

- Similar to strings
  \[ \text{list} \_ \text{name}[\text{index}] \]
- \text{list} \_ \text{name}
- index-character element directly accessing
  - leftmost 0 to list\_length-1
- What about \text{list} \_ \text{name}[-1]?

```python
if __name__ == '__main__':
    ages = [12, 44, 16, 21]
    names = ['Kim', 'Janay', 'TJ', 'Nia']
    combo = ['Tim', 13, 'Ashanti', [40, 'Pink']]

    # print list length
    print(len(ages))
    print(len(names))
    print(len(combo))

    # directly access elements
    print(ages[1])
    print(names[3])
    print(combo[-1])
```
Slicing Lists

- Sublist (smaller part) of the larger list

\[ \text{list\_name}[n:m] \]

- \( n \)-index of the first character in the sublist
- \( m \)-index of the character that immediately follows the last character in the sublist

**Pro tip: slicing only includes chars from \( n \) through \( m-1 \)**

```python
if __name__ == '__main__':
    ages = [12, 44, 19, 21]
    names = ['Kim', 'Janay', 'TJ', 'Nia']
    combo = ['Tim', 13, 'Ashanti', [40, 'Pink']]

    # slice lists
    print(ages[1:3])
    print(names[2])
    print(combo[1])
```
in and not in operators

• Is list1 a member of list2?

```
list1 in list2
list1 not in list2
```

```python
if __name__ == '__main__':
    ages = [12, 44, 10, 21]
    names = ['Kim', 'Janay', 'TJ', 'Nia']
    combo = ['Tim', 13, 'Ashanti', [40, 'Pink']]

    # check membership
    print(21 in ages)
    print('13' not in combo)
    print('Pink' in combo)
```
Activity 2: Lists
Functions Calling Other Functions

def function1(parameter):
    ...
    result=function2(parameter2)
    return result

def function2(parameter2):
    ...
    return result2

if __name__ == '__main__':
    output=function1(argument)
    print(output)

Example code(PyCharm)
Assignment 1: Totem Poles
Learning Goals: Totem Pole

• **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_head</td>
<td>No parameters</td>
<td>No return value, only prints</td>
<td>happy_head</td>
</tr>
<tr>
<td>head_with_DESCRIPTION</td>
<td>1 or 2 parameters of type function</td>
<td>No return value, only prints</td>
<td>head_with_mouth</td>
</tr>
<tr>
<td>totem_DESCRIPTION</td>
<td>No parameters</td>
<td>No return value, calls head functions</td>
<td>totem_fixed, totem_selfie, totem_random</td>
</tr>
<tr>
<td>selfie_band, head_random</td>
<td>- helper functions!</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Creating your program

Start small and build incrementally
With functions grow by...

- Minimal code that does run and can be submitted
- Where go from here?
  - Add head part functions to create happy_head()
  - Create the next head function for totem_fixed and any new head part functions
  - Try a head_with function
  - Go to the next totem
  - etc.
Totem Assignment by Tuesday

• At minimum…
• Read the assignment
• Create initial design
• 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 or at consulting/office hours
Reminders

• Work smarter, not harder
• Design first
• Try to identify where you are stuck
  • Identify resources to help solve problem
• Leverage your design and PythonTutor to understand program flow of control
  • http://pythontutor.com