CompSci 101
Fall 2021

Lecture 6

Reminders

- **Identity & Computing Lecture Series**
  - 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.”

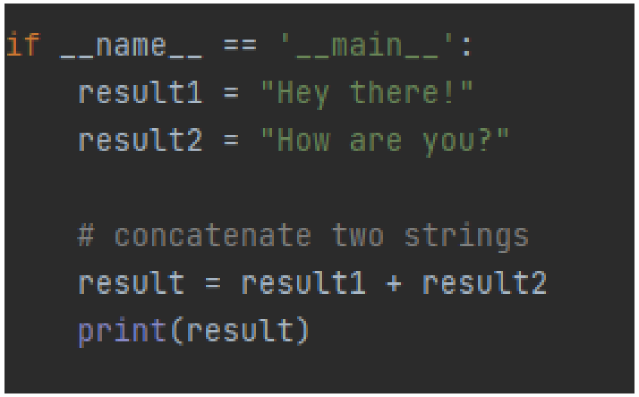
  - Dr. Carter G. Woodson

People to Know: Dr. Tessa Lau

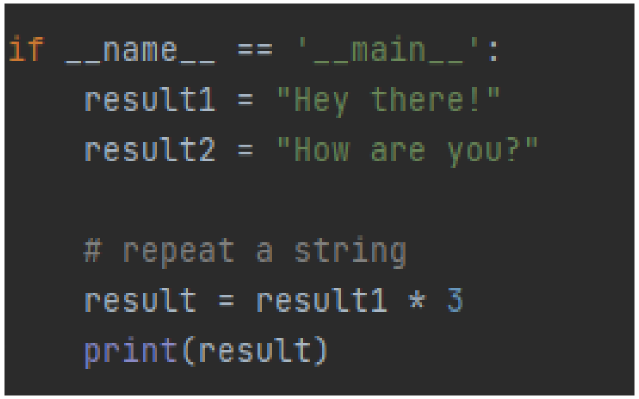
- Cornell (BA, BS)
- University of Washington (MS, PhD)
- Founder/CEO, Dusty Robotics
- Co-founder/Chief Robot Whisperer, Savioke

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



```python
if __name__ == '__main__':
    result1 = "Hey there!"
    result2 = "How are you?"

    # concatenate two strings
    result = result1 + result2
    print(result)
```



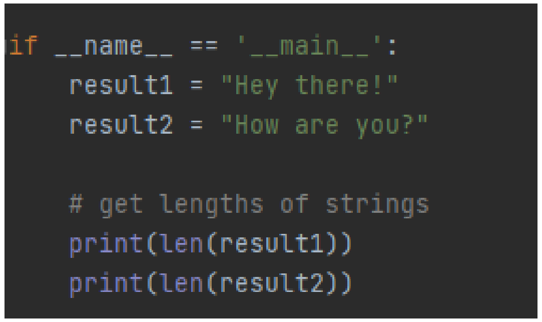
```python
if __name__ == '__main__':
    result1 = "Hey there!"
    result2 = "How are you?"

    # repeat a string
    result = result1 * 3
    print(result)
```

Indexing a String

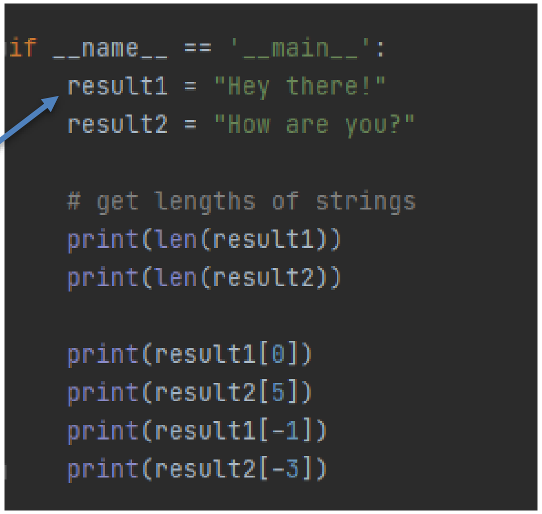
string_name[*index*]

- 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



```
if __name__ == '__main__':
    result1 = "Hey there!"
    result2 = "How are you?"

    # get lengths of strings
    print(len(result1))
    print(len(result2))
```



```
if __name__ == '__main__':
    result1 = "Hey there!"
    result2 = "How are you?"

    # get lengths of strings
    print(len(result1))
    print(len(result2))

    print(result1[0])
    print(result2[5])
    print(result1[-1])
    print(result2[-3])
```

What is result1[10]?

Slicing Strings

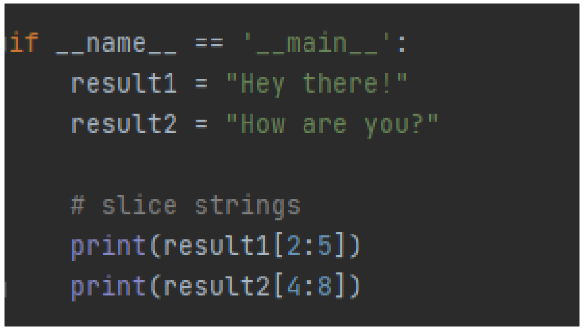
- Real-world examples
  - Slicing bread, tomatoes, etc.
  - Substring (smaller part) of the larger string

string_name[*n*:*m*]

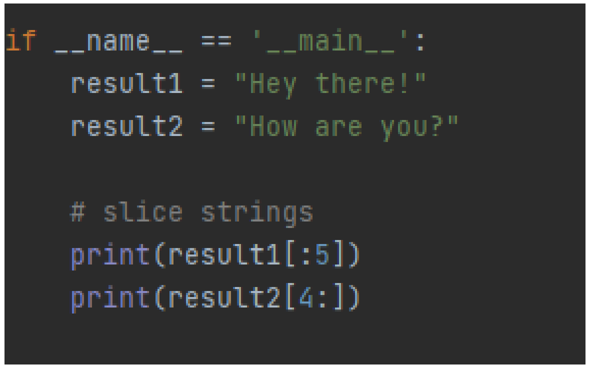
n-index of the first character in the substring

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*



```
if __name__ == '__main__':
    result1 = "Hey there!"
    result2 = "How are you?"

    # slice strings
    print(result1[2:5])
    print(result2[4:8])
```



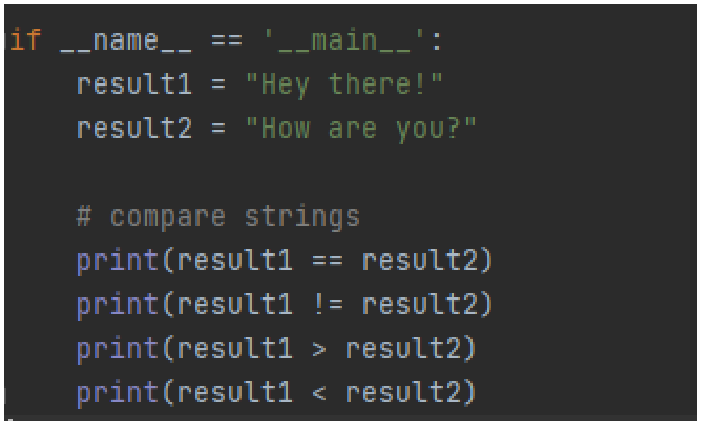
```
if __name__ == '__main__':
    result1 = "Hey there!"
    result2 = "How are you?"

    # slice strings
    print(result1[:5])
    print(result2[4:])
```

Comparing Strings

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

- *string1* == *string2*

- *Pro tip: Lexicographical order (A…Z, a…z)*
    - *'A' < 'a'*



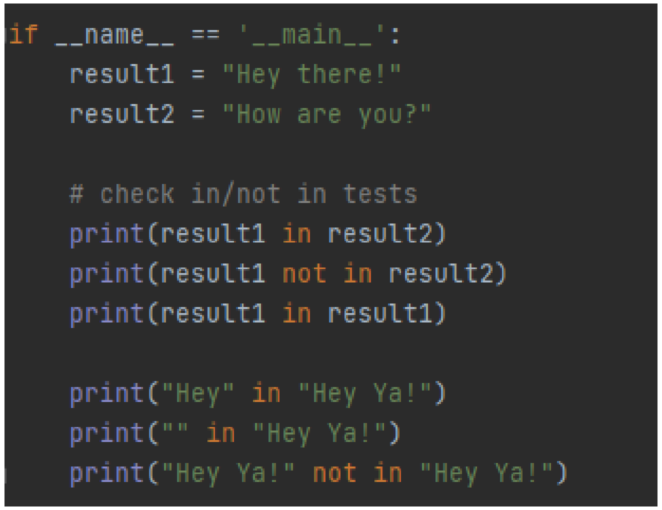
```
if __name__ == '__main__':
    result1 = "Hey there!"
    result2 = "How are you?"

    # compare strings
    print(result1 == result2)
    print(result1 != result2)
    print(result1 > result2)
    print(result1 < result2)
```

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.”)



```
if __name__ == '__main__':
    result1 = "Hey there!"
    result2 = "How are you?"

    # check in/not in tests
    print(result1 in result2)
    print(result1 not in result2)
    print(result1 in result1)

    print("Hey" in "Hey Ya!")
    print("" in "Hey Ya!")
    print("Hey Ya!" not in "Hey Ya!")
```

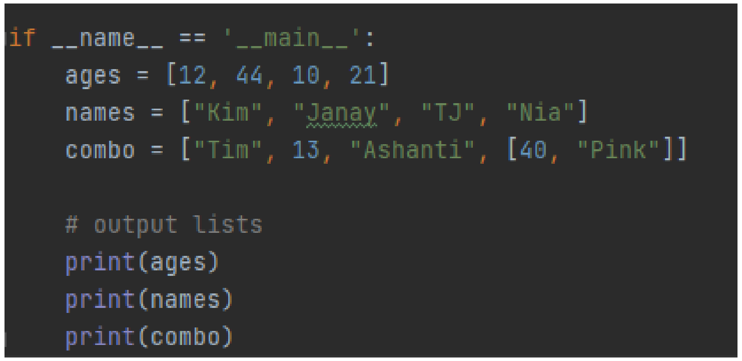
Activity 1: Strings
http://bit.ly/101f21-09-09-1

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****



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

    # output lists
    print(ages)
    print(names)
    print(combo)
```

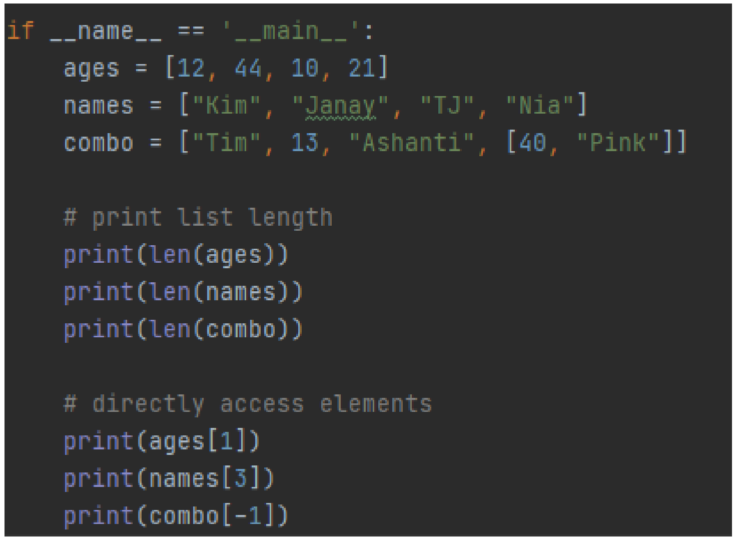
List access and length

- Similar to strings

list_name[*index*]

- list_name
- index-character element directly accessing
  - leftmost 0 to list_length-1

- What about list_name[-1]?



```
if __name__ == '__main__':
    ages = [12, 44, 10, 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

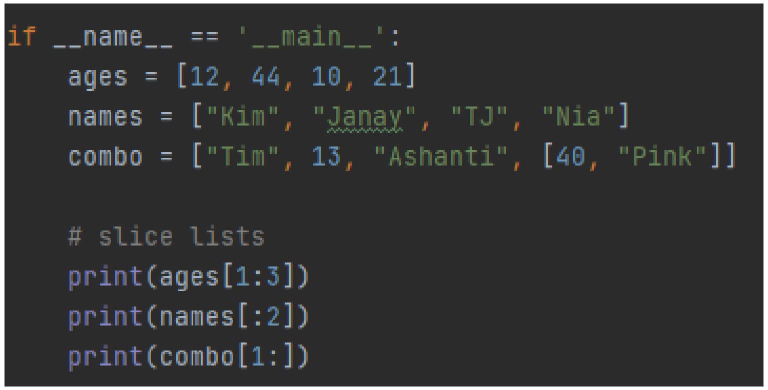
- Sublist (smaller part) of the larger list

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*



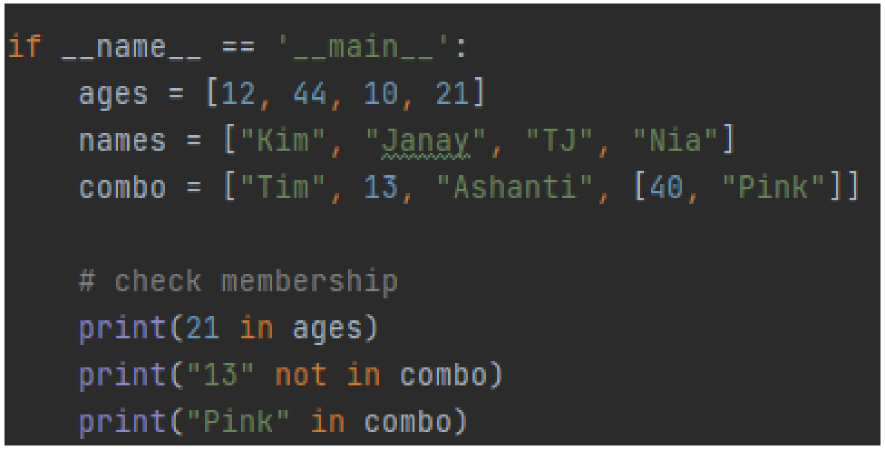
```
if __name__ == '__main__':
    ages = [12, 44, 10, 21]
    names = ["Kim", "Janay", "TJ", "Nia"]
    combo = ["Tim", 13, "Ashanti", [40, "Pink"]]

    # 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*



```
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
http://bit.ly/101f21-09-09-2

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

```
elf totem

|||||///////////
|||||///////////
+--------------+
|jn131    jn131|
+--------------+
 _____    _____
/     \  /   o \
 o     ||
\_____/  \_____/
|      /       |
|     /___     |
|     ____     |
|    |____|    |
|              |
|______()______|
((((((((((((((((
))))))))))))))))
+--------------+
|jn131    jn131|
+--------------+
+--------------+
|  ___    ___  |
| |___|--|___| |
|              |
|              |
|     /__\     |
|              |
|    ~~~~~~~   |
|     ||||     |
|______________|
/\/\/\/\/\/\/\/\
/\/\/\/\/\/\/\/\
+--------------+
|jn131    jn131|
+--------------+
 _____    _____
/     \  /     \
   o   ||   o
\_____/  \_____/
|       \      |
|     ___\     |
|              |
|    \____/    |
|              |
|______________|
```

```
selfie totem

\/\/\/\/\/\/\/\/
+--------------+
|nrt13    nrt13|
+--------------+
|  ___    ___  |
| /   \  /   \ |
 |   0 || 0   |
| \___/  \___/ |
|       \      |
|     (__)     |
|   ________   |
|   \      /   |
|    \____/    |
+--------------+
\/\/\/\/\/\/\/\/
+--------------+
|nrt13    nrt13|
+--------------+
|  ___    ___  |
|_|   |--|   |_|
| |___|  |___| |
|       \      |
|     (__)     |
|   ________   |
|   \      /   |
|    \____/    |
+--------------+
\/\/\/\/\/\/\/\/
+--------------+
|nrt13    nrt13|
+--------------+
|  ___    ___  |
| /   \  /   \ |
 |   0 || 0   |
| \___/  \___/ |
|       \      |
|     (__)     |
|     ___      |
|              |
+--------------+
```

```
self totem

|\     ||     /|
||\    ||    /||
|||\   ||   /|||
||||\  ||  /||||
|||||\ || /|||||
||||||\/\/||||||
+--------------+
|jw542    jw542|
+--------------+
|___   ___  ___|
/   \ /   \/   \
  o  |  o |   o
\___/ \___/\___/
|      /\      |
|  __________  |
|  \  / \  /   |
|   \/   \/    |
|   \______/   |
+--------------+
|\     /\     /|
||\   /||\   /||
|||\ /||||\ /|||
+--------------+
|jw542    jw542|
+--------------+
|_____    _____|
/_____\  /_____\
 __0___||__0___
\_____/  \_____/
|      /\      |
|     _____    |
|    /     \   |
+--------------+
|\     /\     /|
||\   /||\   /||
|||\ /||||\ /|||
+--------------+
|jw542    jw542|
+--------------+
|___   ___  ___|
/   \ /   \/   \
  o  |  o |   o
\___/ \___/\___/
|      /\      |
|  __________  |
|  \  / \  /   |
|   \/   \/    |
|   \______/   |
+--------------+
```

Learning Goals: Totem Pole

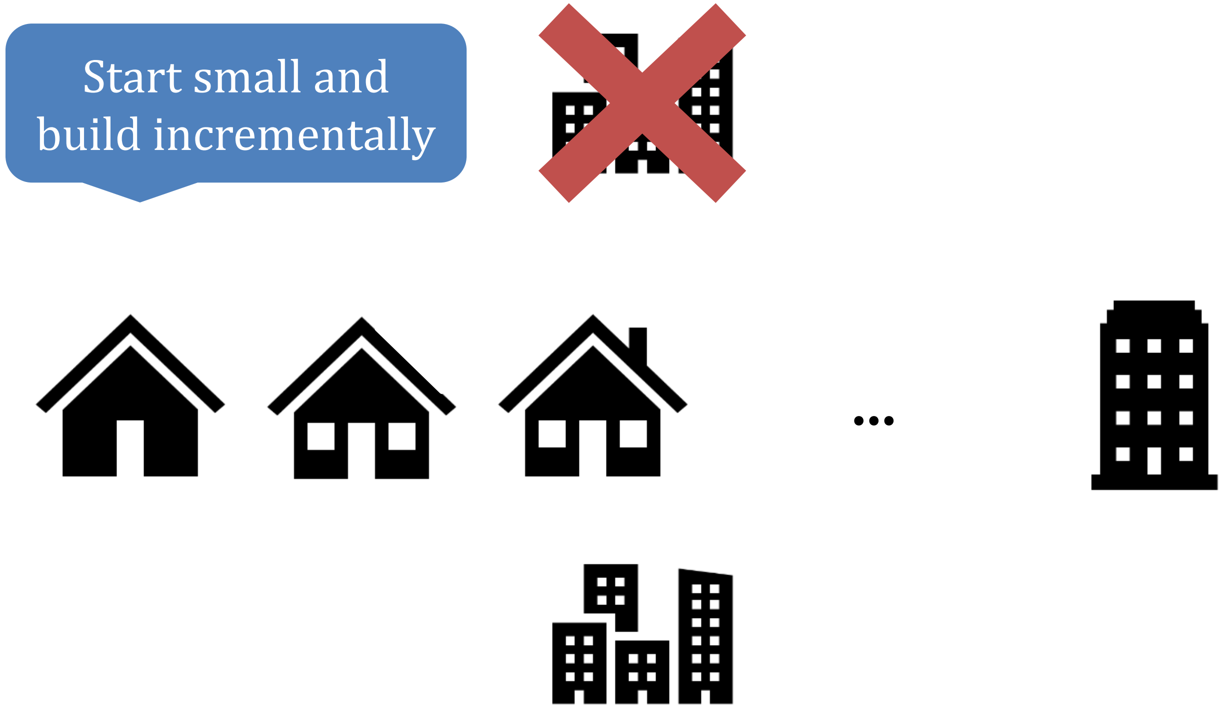
- **Understand differences and similarities:**
  - Function definitions vs function calls
  - Functions with return statements vs those without
  - Functions with parameters vs those without
  - Functions can be arguments

- **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!

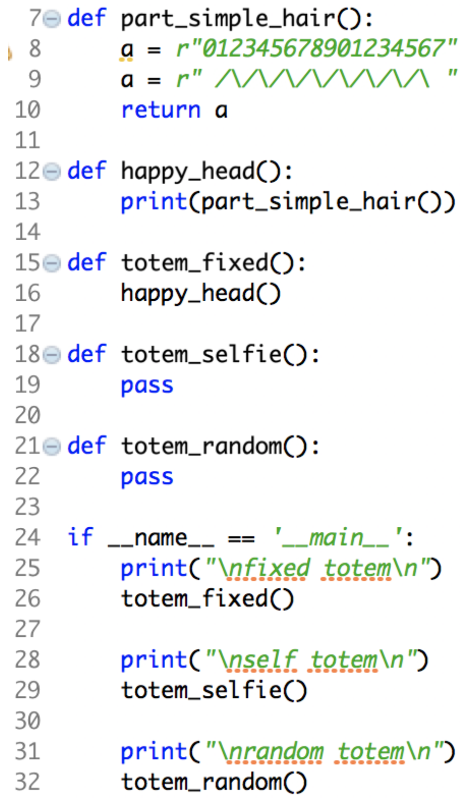
Function Name Format

Function	Parameters	Returns	Example
`part_DESCRIPTION`	No parameters	A string	`part_smiling_mouth`
`DESCRIPTION_head`	No parameters	No return value, only prints	`happy_head`
`head_with_DESCRIPTION`	1 or 2 parameters of type function	No return value, only prints	`head_with_mouth`
`totem_DESCRIPTION`	No parameters	No return value, calls head functions	`totem_fixed`, `totem_selfie`, `totem_random`
`selfie_band`, `head_random` – helper functions!			

Creating your program



With functions grow by…



```python
def part_simple_hair():
    a = r"012345678901234567"
    a = r" /\/\/\/\/\/\/\/\ "
    return a

def happy_head():
    print(part_simple_hair())

def totem_fixed():
    happy_head()

def totem_selfie():
    pass

def totem_random():
    pass

if __name__ == '__main__':
    print("\nfixed totem\n")
    totem_fixed()

    print("\nself totem\n")
    totem_selfie()

    print("\nrandom totem\n")
    totem_random()
```

- **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