

## CompSci 101 Conditionals (Cont'd), Collections, Strings, Lists

1

## Announcements

- Upcoming due dates
  - All Sakai quizzes due @145pm day of lecture
  - Assignment 1-due 2/11 @1130pm
- Piazza channel
  - Direct questions here

2

## **E** is for ...

- Escape Sequence
  - Why `\n` is newline and `\t` is a tab
- Encryption
  - From Caesar Ciphers to SSL and beyond
- Enumerate
  - Adding counters to iterable
- Emoticon
  -



01/23/2020

CompSci 101, Spring 2020, Selection, Lists,  
Sequences, Tuples

3

3

## Computer Scientists to Know

- Dr. Clarence "Skip" Ellis
  - 1<sup>st</sup> Black person to earn a Ph.D. in computer science
- Dr. Timnit Gebru
  - Co-founder, Black in AI
  - Ethical AI researcher



4

## PFTD

- Selections/Conditionals continued
- Strings
- Lists
- “The mere imparting of information is not education.”
  - Dr. Carter G. Woodson

5

## Selection/Conditionals: if...elif...else

```
if BOOLEAN_CONDITION:
    CODE_BLOCK_A
else:
    CODE_BLOCK_B

if BOOLEAN_CONDITION:
    CODE_BLOCK_A
elif BOOLEAN_CONDITION:
    CODE_BLOCK_B
else:
    CODE_BLOCK_C
```

```
if __name__ == '__main__':
    num1 = 5

    if num1 == 5:
        print("The number is 5")
    else:
        if num1 < 5:
            print("The number is less than 5")
        else:
            print("The number is greater than 5")
```

```
if __name__ == '__main__':
    num1 = 5

    if num1 == 5:
        print("The number is 5")
    elif num1 < 5:
        print("The number is less than 5")
    else:
        print("The number is greater than 5")
```

6

## WOTO 4: Review

- Extending your program from WOTO 3
- Simulate rolling two dice
  - “Roll” two dice
  - Results of the rolls sent to function named *sum*
    - Sum dice values
    - If sum is 7 or 11, then output “You win!”
    - Otherwise, output “Next time!”

7

## 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)
```

8

## 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

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

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

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

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

9

## Indexing a String

`string_name[index]`

- Index: 0 to (string\_length-1)

- **\*\*\*Whitespaces count\*\***

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

10

## Slicing Strings

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

`string_name[n:m]`

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

    # slice strings
    print(result1[0:5])
    print(result2[4:8])
```

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

    # slice strings
    print(result1[0])
    print(result2[6])
```

11

WOTO 1: <http://bit.ly/101s21-0204-1b>

12

## Comparing Strings

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

- `string1 == string2`

**\*\*need to output this or store the result\*\***

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

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

13

## *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 = "hey are you?"

    # check if result1 is result2
    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!")
```

14

WOTO 2: <http://bit.ly/101s21-0204-2>

15

## Lists

- Groceries, errands, names, etc.

- Lists are:

- Ordered
- Mutable
- Duplicate elements allowed
- 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, 18, 21]
    names = ["Mia", "Jung", "Li", "Mia"]
    combo = ["Li", 13, "Kameli", [40, "Pink"]]

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

16

## List access and length

- Similar to strings
- `list_name[index]`
- list\_name-your variable name
- index-character element directly accessing
  - leftmost 0 to list\_length-1
- What about list\_name[-1]?

```
if __name__ == '__main__':
    ages = [12, 44, 18, 21]
    names = ['Tim', 'Johnny', 'JP', 'Pina']
    combo = ['Tim', 15, 'Ashanti', [40, 'Pina']]

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

    # directly access elements
    print(ages[1])
    print(names[2])
    print(combo[-1])
```

17

## 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**

```
if __name__ == '__main__':
    ages = [12, 44, 18, 21]
    names = ['Tim', 'Johnny', 'JP', 'Pina']
    combo = ['Tim', 15, 'Ashanti', [40, 'Pina']]

    # slice lists
    print(ages[1:3])
    print(names[:2])
    print(combo[1:2])
```

18

## in and not in operators

- Is list1 a member of list2?

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

```
if __name__ == '__main__':
    ages = [12, 44, 18, 21]
    names = ['Tim', 'Johnny', 'JP', 'Pina']
    combo = ['Tim', 15, 'Ashanti', [40, 'Pina']]

    # check membership
    print(12 in ages)
    print('JP' not in names)
    print('Pina' in combo)
```

19

WOTO 3: <http://bit.ly/101s21-0204-3>

20



## With functions grow by...

```

7 def part_simple_hair():
8     a = r"012345678901234567"
9     a = r" /\/\/\/\/\/\/\/\/"
10    return a
11
12 def happy_head():
13    print(part_simple_hair())
14
15 def totem_fixed():
16    happy_head()
17
18 def totem_selfie():
19    pass
20
21 def totem_random():
22    pass
23
24 if __name__ == "__main__":
25    print("tofixed totem\n")
26    totem_fixed()
27
28    print("to selfie totem\n")
29    totem_selfie()
30
31    print("torandom totem\n")
32    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.

25

## Totem Assignment by Tuesday

- At minimum...
- Read the assignment
- Do the Totem reading quiz
- 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 Piazza or at consulting/office hours

26

## Remember

- 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>

27