CompSci 101
Fall 2021
Reminders

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

• **No lab Friday, 9/24**

• **Assignments**
  • Assign 1 due today
  • Assign 2 live 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

• Loops (for)
  • range()
  • Traversing strings/lists
• Accumulators
• Split/join

“The mere imparting of information is not education.”
• Dr. Carter G. Woodson
KISS Principle

• Think of the non-computing context for any word/terms
• KISS model
  • Work smarter, not harder!!
• “Good programmers are simply good designers.”
  • -Dr. Washington
• Design first and always!
• Importance of reusability
• USE PYTHON TUTORS IF YOU HAVE QUESTIONS!
People to Know: Dr. Richard Tapia

- UCLA (BA, MA, PhD, Math)
- University Professor, Rice University
  - 6th person w/title in 100-year history
- Research
  - Computational/mathematical sciences
  - Education/outreach
- National Academy of Engineering
  - 1st Hispanic person elected.
- National Medal of Science
- Conferences in Honor
  - Tapia Celebration of Diversity in Computing
  - Blackwell-Tapia Conference
Split vs. Join

Break single string into list of strings

```
var = name.split()
var = name.split(arg)
```

arg ➔ variable or string

“Glue” strings in a list together to form a single string

```
var = glue.join()
```

glue ➔ variable or string
Split vs. join examples

```python
if __name__ == '__main__':
    phrase = "Hi! There! Neighbor!"
    lst = phrase.split()
    print(lst)

    lst2 = phrase.split('!')
    print(lst2)

test = ['Give', 'me', 'a', 'chance', '!']
result = ' '.join(test)
print(result)
result2 = '*'.join(test)
print(result2)
```
Anatomy of a for loop

```python
if __name__ == '__main__':
    for number in [0, 1, 2, 3]:
        print(number)
```

for v in seq:
    CODE_BLOCK

```python
for v in seq:
    CODE_BLOCK
```

Start

More elements in seq?

Yes

v = next element in seq

CODE_BLOCK

No

Finished
range() function

- What about larger numbers?

- range(stop)
  - 0 up to (not including) stop

- range(start, stop)
  - Specify start value (increment by 1)

- range(start, stop, step)
  - Specify step value

```python
if __name__ == '__main__':
    for number in [0, 1, 2, 3]:
        print(number)
```
Activity 1:
Why use loops?

• Repetition
  • Keeping a running total (counter)
  • Summing (other repetitive calculations)

• Accumulators
  • “Accumulate”-acquire an increasing number of quantity of.

• Rules for accumulators
  • Initialize the “running total”
  • Don’t initialize inside the loop
  • Increase the total with each iteration
Another way to use accumulators

def square(x):
    # raise x to the second power
    runningtotal = 0
    for counter in range(x):
        runningtotal = runningtotal + x
    return runningtotal
Activity 2
Traversing strings

Print each character in the string

```python
if __name__ == '__main__':
    name = 'Tiana'
    for i in range(5):
        print(name[i])
```

Can this be simplified?
What about printing the characters in reverse order?
Accumulators with Strings

• How is “+” used with strings?
  • Concatenation
  • result = “string1” + “string2”

• Still require initialization
  • Empty string (“”) instead of 0

• Still “acquiring/increasing quantity.”
  • Appending to string
Designing Solution

2. What did we do?
   a. Paper and pencil, write it down!
3. Generalize
4. Test: “Computer” -> “Cmprtr”?
• Write a function `isVowel(ch)`
  • Returns true if the input is a vowel and false otherwise
  • Input: string of a single character (length 1)
  • Return: bool
Why would we use “not in” instead of “in” for Activity 3?

- KISS
- Which is simpler to use?
  - What’s required to use “in”?
  - What’s required to use “not in”?
  - Which is simpler to design/implement?
Which is better to traverse list?

fruits = ["apple", "orange", "banana", "cherry"]
fruits = ["apple", "orange", "banana", "cherry"]

for position in range(len(fruits)):  # by index
    print(fruits[position])
for afruit in fruits:  # by item
    print(afruit)
Remember lists are mutable...

```python
numbers = [1, 2, 3, 4, 5]
print(numbers)

for i in range(len(numbers)):
    numbers[i] = numbers[i] ** 2

print(numbers)
```
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