CompSci 101
Fall 2021

Lecture 11

Reminders

- **Assignments**
  - Assign 2 due 10/7
  - APT 3 due 10/7
  - Assign 3 live

- **APT Quiz 1**
  - 10/8 (8am)-10/11 (11pm)
  - **TWO PARTS**
    - 1.5 hours each
    - Notes, lecture notes, **YOUR** old code, book, and web (for info on Python)
    - **MAY NOT** search for solutions online.
  - PM via Ed only.

- **Small-group tutoring**

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

- **List comprehensions**
- **Transform Assignment**
  - Global variable

“The mere imparting of information is not education.”

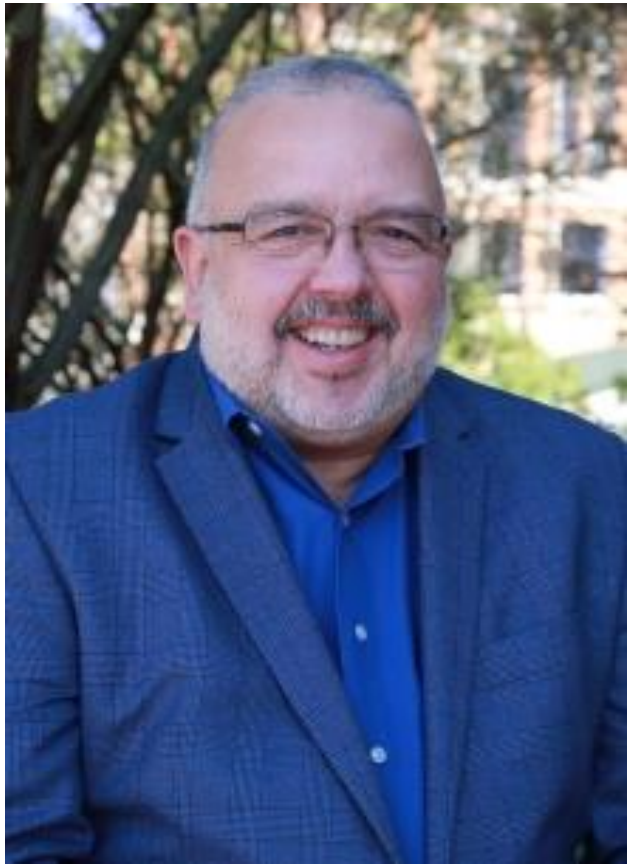
- Dr. Carter G. Woodson

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 PyCharm/PythonTutor IF YOU HAVE QUESTIONS!***

People to Know:
Dr. Manuel A. Pérez Quiñones

- DSc (CS)-The George Washington University
- BS/MS(CS)-Ball State University
- Professor, Software and Information Systems
  - UNC-Charlotte
- Personal information management, HCI, CS education, diversity in computing
- Co-founder-Hispanics in Computing
  - http://hispanicsincomputing.org/



Accumulator in one line

```
def onlyPos(nums):
    ret = []
    for n in nums:
        if n > 0:
            ret.append(n)
    return ret

print(onlyPos([1,2,3,-1,-2,-3]))
```

```
return [n for n in nums if n > 0]
```

- **List Comprehension**
  - We will use a complete, but minimal version of list comprehensions, much more is possible

List Comprehension Syntax

```
ret = []
for V in LIST:
    ret.append(V_EXP)
```

```
ret = [V_EXP for V in LIST]
```

```
ret = []
for V in LIST:
    if BOOL_EXP:
        ret.append(V_EXP)
```

```
ret = [V_EXP for V in LIST if BOOL_EXP]
```

- **V is any variable: all list elements in order**
- **V_EXP is any expression, often use V**

List Comprehension Syntax

```
ret = []
for V in LIST:
    ret.append(V_EXP)
```

```
ret = [V_EXP for V in LIST]
```

```
ret = []
for V in LIST:
    if BOOL_EXP:
        ret.append(V_EXP)
```

```
ret = [V_EXP for V in LIST if BOOL_EXP]
```

- **`if` part optional - `BOOL_EXP` is a Boolean expression usually using `V`**

TPS: List Comprehension Examples

```
[w for w in words if w.count('e') == 0]
```

- **What does this represent? When is the if true/false?**

```
[v*2 for v in range(6) if v % 2 == 1]
```

- **What does this represent? What is the length?**

```
sum([1 for x in words if len(x) > 4])
```

- **What does this represent?**

Activity 1: https://bit.ly/101f21-10-7-1

Assignment 3: Transform

- **Reading and writing files**
  - We've seen how to read, writing is similar
  - Open, read, and close
  - Open, write, and close - `.write(…)`

- **Apply a function to every word in a file**
  - Encrypt and decrypt
  - Respect lines, so resulting file has same structure

Encrypting and Decrypting

- **We give you:**
  - Transform.py
  - Vowelizer.py - Removes vowels

- **You implement**
  - Pig Latin
  - Caesar cipher

- **Challenge: Shuffleizer**

Concepts in Starter Code

- **Global variables**
  - Generally avoided, but very useful
  - Accessible in all module functions

- **FileDialog and tkinter**
  - API and libraries for building UI and UX

- **Docstrings for understanding!**

Global Variables (Best Practice)

- **Best practice = help other humans read the code**

- **All variables that will be global are created with an initial assignment at the top of the file**
- **When used in a function, variable is declared global at the beginning of the function**

What is global?

- **Accessible everywhere in the file (or "module")**
- **Variable is in the global frame**
  - First frame in Python Tutor
- **If declared global in a function:**
  - The variable in the global frame can also be reassigned in that function's
  - Despite Python being in a different frame!
- **Eliminates the need to pass this value to all the functions that need it**
- **Demo-Pycharm**

When reading code with globals

- **When checking the value of a variable, ask:**
  - Is this variable local to the function or in the global frame?
- **When in a function and assigning a value to a variable, ask:**
  - Has this variable been declared global?
    - If yes, reassign the variable in the **global frame**
    - If no, create/reassign the variable in the **function's local frame**
  - **Demo-global vs. variable**

TPS: What will this print?

```
1  s = 'top of the file'
2
3  def change():
4      global s
5      s = 'I am'
6      t = 'changing'
7
8      print('Inside change:', s, t)
9
10 if __name__ == '__main__':
11     print('Beginning of main:', s)
12     s = 'inside main'
13     t = 'text'
14
15     print('Before change:', s, t)
16     change()
17     print('After change:', s, t)
```

Remember: When considering a variable ask is this variable global?
- If yes: look in global (first) frame for value AND when reassigning do it in the global frame

TPS: What will this print?

```
1  s = 'top of the file'
2
3  def change():
4      global s
5      s = 'I am'
6      t = 'changing'
7
8      print('Inside change:', s, t)
9
10 if __name__ == '__main__':
11     print('Beginning of main:', s)
12     s = 'inside main'
13     t = 'text'
14
15     print('Before change:', s, t)
16     change()
17     print('After change:', s, t)
```

https://goo.gl/rewukN

Remember: When considering a variable ask is this variable global?
- If yes: look in global (first) frame for value AND when reassigning do it in the global frame

What, where, read, write? (in 101)

What is it?	Where first created?	Where accessible? (read)	Where reassign-able? (write)
Regular variable in main	In main	In main only (technically anywhere, but don't do that)	In main only
Regular local function variable	In function	In function only	In function only
Global variable	Top of file	If not reassigning the value, in main and all functions	In main or in any function that first declares it global

Python will have an error if it is not declared global and it is used and then there is a variable with the same name being assigned

Can avoid this by ALWAYS declaring the variable global in the function (best practice) if that is the variable you are using

Activity 2: https://bit.ly/101f21-10-7-2

Tkinter and FileDialog

- **This library and API is useful for creating GUIs**
  - Difficult and not always about the big picture
  - Debugging can be frustrating
  - Tedium of making things right versus exultation in creating wonderful programs!

- **If you don't see the rocket-ship? Oops**
  - What happens when you run Transform?

Caesar Cipher

- **https://en.wikipedia.org/wiki/Caesar_cipher**
- **A to Z -> 0 to 25**
  - 23rd letter is X

Index	0	1	2	3	4	5	6	7	8	9	10
Plain text	A	B	C	D	E	F	G	H	J	K	L
Cipher text (shift key = 23)	X	Y	Z	A	B	C	D	E	F	G	H

`findShift(…)`

- **Input: cipher text (a.k.a. encrypted)**
- **Output: shift used for encryption**
- **How: Brute force**
  - Have a list of real words
  - Check all possible shift values for cipher text
  - Shift value resulting in the most real words is the decryption key (a.k.a. shift number)
  - Encryption_shift_key = 26 - decryption_key

TPS: Why is this 26 and not 25?

Eyeball Decryption Example

```
st = "Bxvncrvnb rc'b njbh cx lxdwc oaxv 1-10, kdc wxc jufjhb"
```

with this code

```
for sh in range(26):
    setShift(sh)
    print(sh, encrypt(st))
```

results in this output:

```
0  Bxvncrvnb rc'b njbh cx lxdwc oaxv 1-10, kdc wxc jufjhb
1  Cywodswoc sd'c okci dy myexd pbyw 1-10, led xyd kvgkic
2  Dzxpetxpd te'd pldj ez nzfye qczx 1-10, mfe yze lwhljd
3  Eayqfuyqe uf'e qmek fa oagzf rday 1-10, ngf zaf mximke
4  Fbzrgvzrf vg'f rnfl gb pbhag sebz 1-10, ohg abg nyjnlf
5  Gcashwasg wh'g sogm hc qcibh tfca 1-10, pih bch ozkomg
6  Hdbtixbth xi'h tphn id rdjci ugdb 1-10, qji cdi palpnh
7  Iecujycui yj'i uqio je sekdj vhec 1-10, rkj dej qbmqoi
8  Jfdvkzdvj zk'j vrjp kf tflek wifd 1-10, slk efk rcnrpj
9  Kgewlaewk al'k wskq lg ugmfl xjge 1-10, tml fgl sdosqk
10 Lhfxmbfxl bm'l xtlr mh vhngm ykhf 1-10, unm ghm teptrl
11 Migyncgym cn'm yums ni wiohn zlig 1-10, von hin ufqusm
12 Njhzodhzn do'n zvnt oj xjpio amjh 1-10, wpo ijo vgrvtn
13 Okiapeiao ep'o awou pk ykqjp bnki 1-10, xqp jkp whswuo
14 Pljbqfjbp fq'p bxpv ql zlrkq colj 1-10, yrq klq xitxvp
15 Qmkcrgkcq gr'q cyqw rm amslr dpmk 1-10, zsr lmr yjuywq
16 Rnldshldr hs'r dzrx sn bntms eqnl 1-10, ats mns zkvzxr
17 Sometimes it's easy to count from 1-10, but not always
18 Tpnfujnft ju't fbtz up dpvou gspn 1-10, cvu opu bmxbzt
19 Uqogvkogu kv'u gcua vq eqwpv htqo 1-10, dwv pqv cnycau
20 Vrphwlphv lw'v hdvb wr frxqw iurp 1-10, exw qrw dozdbv
21 Wsqixmqiw mx'w iewc xs gsyrx jvsq 1-10, fyx rsx epaecw
22 Xtrjynrjx ny'x jfxd yt htzsy kwtr 1-10, gzy sty fqbfdx
23 Yuskzosky oz'y kgye zu iuatz lxus 1-10, haz tuz grcgey
24 Zvtlaptlz pa'z lhzf av jvbua myvt 1-10, iba uva hsdhfz
25 Awumbquma qb'a miag bw kwcvb nzwu 1-10, jcb vwb iteiga
```

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