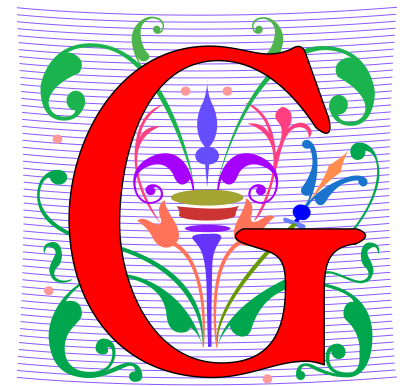


G is for ...



- **Google**
 - How to find the answer to everything
- **Global Variable**
 - Accessible everywhere, typically do not do
- **GIGO**
 - Garbage In, Garbage Out
- **Git**
 - Working Together or Solo

Sir Tim Berners-Lee

- Invented World Wide Web
 - Turing award 2016
- HTTP vs. TCP/IP
 - Just protocols?



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

“The Web as I envisaged it, we have not seen it yet. The future is still so much bigger than the past.”

“We need diversity of thought in the world to face the new challenges.”

Did you sign up for compsci@duke.edu mailing list?

- **Mailing list to get the CompSci weekly newsletter**
 - Events, research and job opportunities
- **To add yourself:**
 - Go to lists.duke.edu
 - Authenticate and then add compsci@duke.edu
- **Sample item:**
 - Duke Women in Tech looking for new members and to get our mailing lists. Fill out this form: <https://tinyurl.com/witspring22>

Announcements

- **Assignment 1 Faces**
 - Assign 1 QZ due today 11:30pm (no grace day!)
 - Program due Thursday (has one grace day)
 - Also REFLECT Form due same time
 - Remember, no consulting hours on Friday
- **APT-2 out today, due Sept 29**
- **Lab 3 Friday**
 - Do prelab 3 before attending!
- **Exam 1 on Tuesday, Sept 27**

Lecture for Thursday Sept 22

- **Lecture will be asynchronous**
 - There is no in-person lecture
- **Watch the videos (there are 4), do WOTOs**
- **Complete it before going to lab**
- **WOTOs will turn off by Friday night 11:30pm**

- **Prof. Rodger is out of town Tues-Friday, no office hours today or the rest of the week**
- **We have fantastic UTAs in consulting hours! And there is always Ed Discussion!**

PFTD

- **Exam 1**
- **Lists continued**
- **String methods and more**
- **For Loops**

Exam 1 – Sept 27, 2022

- **All lecture/reading topics through Tues. Jan 20**
 - Topics on Jan 20 at simpler level
 - Loop over list, loop over characters in a string
- **Understand/Study**
 - Reading, lectures
 - Assignment 1, APT-1, (APT-2 helpful, not required)
 - Labs 0-3
 - Very Important! Practice writing code on paper
- **Logistics:**
 - In person, in lecture

Exam 1 – Sept 27, 2022 (cont)

- **What you should be able to do**
 - Read/trace code
 - Determine output of code segment
 - Write small code segments/function
- **Look at old test questions**
 - We will look at some in Lab 3
- **Exam 1 is your own work!**
 - Do not consult with anyone else.
 - Closed book, no notes, no paper, no calculators
 - See Exam 1 Reference sheet (will be on exam)

Python Reference Sheet, is attached to your exam (see link on calendar page, under 9/27)

Python Reference Sheet for CompSci 101, Exam 1, Fall 2022


On this page we'll keep track of the Python types, functions, and operators that we've covered in class. You can also review the online [Python References](#) for more complete coverage, BUT NOTE there is way more python in the there then we will cover! The reference page below is all you should need to complete the exam.

Mathematical Operators		
Symbol	Meaning	Example
+	addition	4 + 5 = 9
-	subtraction	9 - 5 = 4
*	multiplication	3*5 = 15
/ and //	division	6/3 = 2.0 6/4 = 1.5 6//4 = 1
%	mod/remainder	5 % 3 = 2
**	exponentiation	3**2 = 9, 2**3 = 8
String Operators		
+	concatenation	"ab"+"cd"="abcd"
*	repeat	"xo"*3 = "xoxoxo"
Comparison Operators		
==	is equal to	3 == 3 is True
!=	is not equal to	3 != 3 is False
>=	is greater than or equal to	4 >= 3 is True
<=	is less than or equal to	4 <= 3 is False
>	is strictly greater than	4 > 3 is True
<	is strictly less than	3 < 3 is False
Boolean Operators		

Nested Lists

- **Lists are heterogenous, therefore!**
 - `lst = [1, 'a', [2, 'b']]` is valid
 - `len(lst) == 3`
 - `[2, 'b']` is one element in list `lst`

`lst[2][1]`



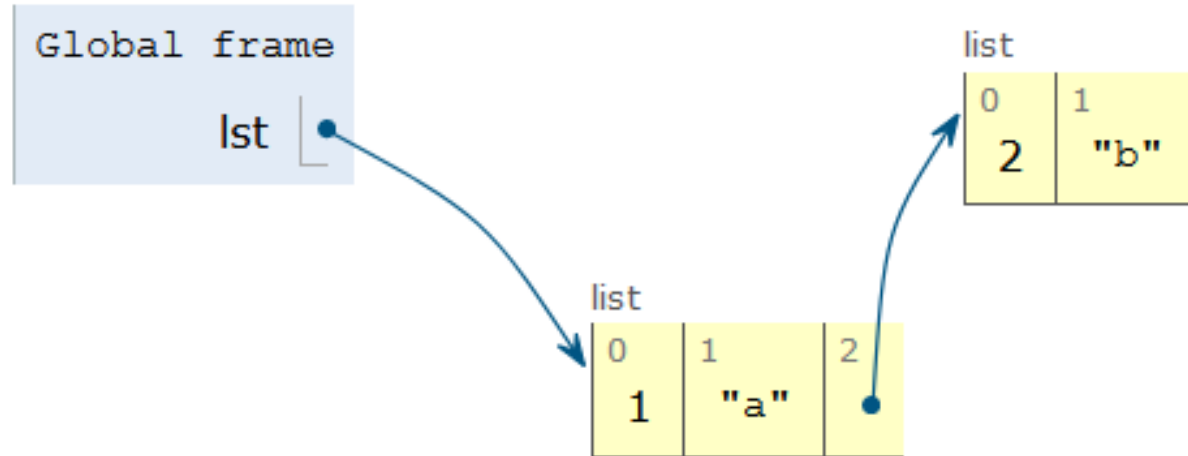
`[2, 'b'] [1] == 'b'`

- **How to index?**
 - `[...]` all the way down
 - `lst[2][1]` returns `'b'`

Nested Lists with Python Tutor

Frames

Objects



```
➔ 1 lst= [1, 'a', [2, 'b']]
   2 print(len(lst))
   3 print(type(lst[2]))
   4 print(lst[2])
   5 print(lst[2][1])
```

Mutating Lists

- `lt = ['Hello', 'world']`
 - Change to: `['Hello', 'Ashley']`
- **Two ways: 1. Build new list or 2. modify list**
 1. Concatenation: `lt = [lt[0]] + ['Ashley']`
 2. Index: `lt[1] = 'Ashley'`
- **How change 'b' in `lt = [1, 'a', [2, 'b']]`?**
 - `lt[2][1] = 'c'`

Mutating Lists code

```
lst1 = ['Hello', 'world']  
print(lst1)  
lst2 = [lst1[0]] + ['Ashley']  
print(lst2)  
print(lst1)  
lst1[1] = 'Ashley'  
print(lst1)
```

```
lst3 = [1, 'a', [2, 'b']]  
print(lst3)  
lst3[2][1] = 'c'  
print(lst3)
```

Immutable built-in Types

- **In python string, int, float, boolean - Immutable**
 - Once created cannot change
 - These are still objects in Python3!!
- **PythonTutor gets this wrong**
 - Everything should be in Objects area
- **Objects don't change**
 - Value associated with variable changes

```
val = 0
bee = val
val = val + 20
```

Immutable built-in Types

- **In python string, int, float, boolean - Immutable**
 - Once created cannot change
 - These are still objects in Python3!!
- **PythonTutor gets this wrong**
 - Everything should be in Objects area
- **Objects don't change**
 - Value associated with variable changes

```
val = "apple"  
bee = val  
val = val + "sauce"
```

Compare assign with integers, strings and lists – 1

Python 3.6
([known limitations](#))

```
→ 1 x = 6
   2 y = x
   3 x = 3
   4 m = "pink"
   5 n = m
   6 m = "red"
   7 a = ["pig", "cow", "dog"]
   8 b = a
   9 a[-1] = "ant"
```

[Edit this code](#)

→ line that just executed

→ next line to execute

Frames

Objects

List Cloning (or copying)

```
lst1 = ['a', 'b', 1, 2]
```

```
lst2 = lst1
```

```
lst3 = lst1[:]
```

WOTO-1 Cloning

<http://bit.ly/101f22-0220-1>

List Concatenation Steps

1. Calculate the length of the new list
2. Create list of that length
3. Copy values from first list
4. Copy values from second list
5. Assign the variable to the new list



Brand
new list!

```
1 lst0 = [1, 2]
2 lst1 = [3, 4, 5]
3 lst2 = lst0 + lst1
```

Concatenation:

length, create, copy, copy, assign

```
1 lst0 = [1,2]
2 lst1 = [3, 4, 5]
3 lst2 = lst0 + lst1
```

Concatenation: Makes new List

```
1 lst0 = [1,2]
2 tmp = lst0
3 lst0 = lst0 + [4]
```

Concatenation:

length, create, copy, copy, assign

- **How is the inner list copied?**

```
1 lst0 = [1, ['b', 3.0]]
2 lst1 = [4]
3 lst2 = lst0 + lst1
```

What will Python Tutor Display? How many copies of ['b', 3.0] will be present?

List Mutation: .append(...)

- **.append()** – list function that adds element to end of list
 - Mutates list to left of “.”
 - “.” – call function to the right of the dot on the thing to the left of the dot (LEFT.RIGHT)

x = [6, 2, 4]

x.append(3)

x.append([5,2])

List Mutation: .append(...)

```
1 lst0 = [1, 2, 3]
2 tmp = lst0
3 lst0.append(4)
```

What will Python
Tutor Display? One or
two lists?

WOTO-2 – Mutable and Append

<http://bit.ly/101f22-0220-2>

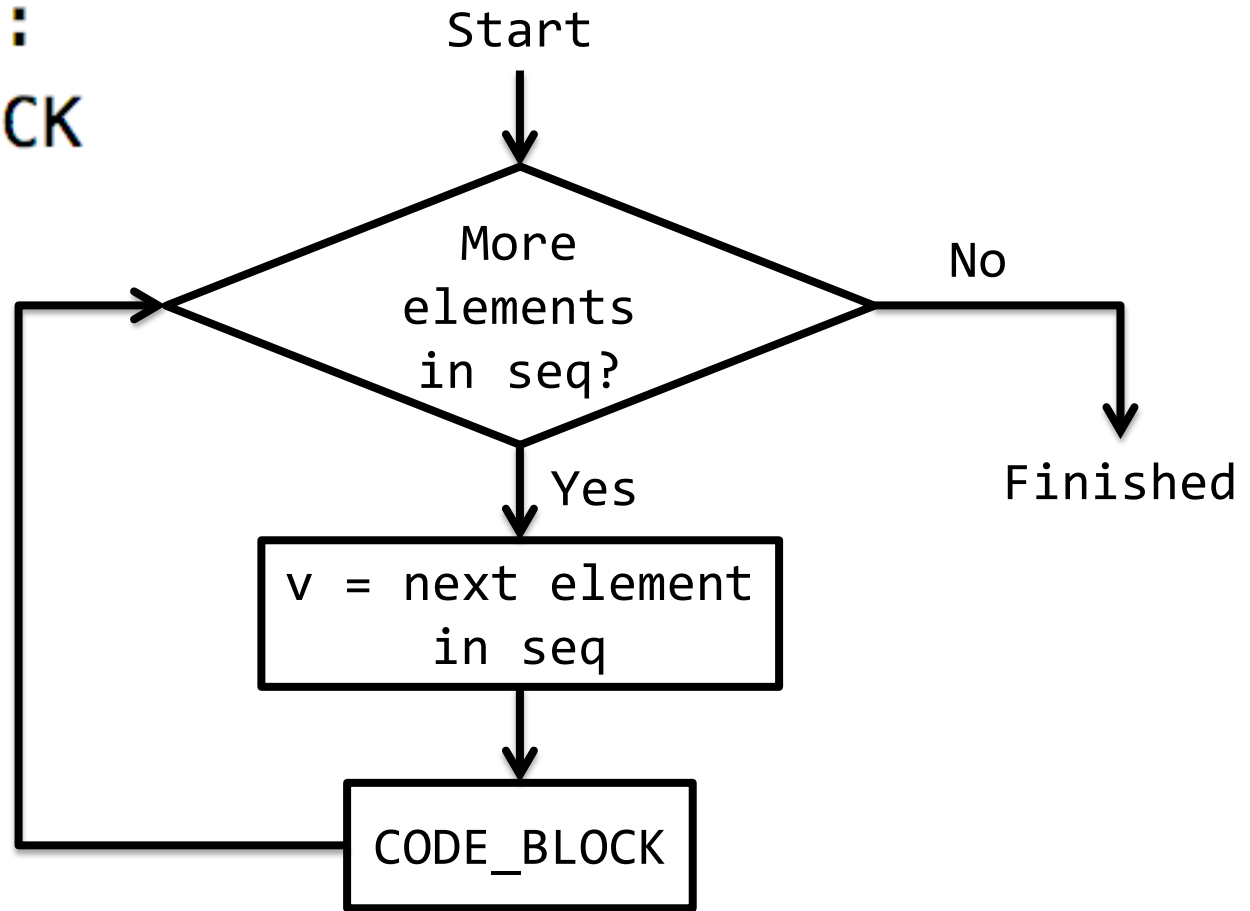
Anatomy of a for loop

```
for VARIABLE in SEQUENCE:  
    CODE_BLOCK
```

- **Think of as:**
 - “For each element in the SEQUENCE put it in the VARIABLE and execute the CODE_BLOCK.”
 - Also called: *Iterate* over the sequence
- **What type(s) are sequences?**
 - Strings, Lists
- **Will VARIABLE likely be in CODE_BLOCK?**

Anatomy of a for loop

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




Example for loop with a list

- What does this for loop do?

```
1 lst = [5, 3, 2]
2 sum = 0
3 for num in lst:
4     sum = sum + num
5 print(sum)
```

- What is first value of **num**?
- What is final value of **num**?

Trace through for loop – 1



```
1 lst = [5, 3, 2]
2 sum = 0
3 for num in lst:
4     sum = sum + num
5 print(sum)
```


Example for loop with a string

- What does this for loop do?

```
1 word = 'cat'  
2 for ch in word:  
3     word = word + ch  
4 print(word)
```

- What is first value of **ch**?
- What is final value of **ch**?

Trace through for loop – 1



```
1 word = 'cat'  
2 for ch in word:  
3     word = word + ch  
4 print(word)
```

String's split(...)

- **Strings have functions too!**
- **TYPE_STRING.FUNCTION(PARAMETERS)**
 - **“.”** means apply function to what is on the left
- **'one fish two fish'.split()** returns a list
- What did it divide the string by?
 - When no parameter, default whitespace
- **'one fish, two fish'.split(',')**

String's join(...)

- **TYPE_STRING.join(SEQ_OF_STRINGS)**
 - Opposite of .split()
 - Creates string from sequence's items separated by the string to the left of join
- ' '.join(['one', 'fish', 'two', 'fish'])
- '+'.join(['one', 'fish', 'two', 'fish'])
- 'ish'.join(['f', 'w', 'd', 'end'])

More Methods

String

<code>.find(s)</code>	index of first occurrence of s
<code>.rfind(s)</code>	index of last occurrence of s (from Right)
<code>.upper()/ .lower()</code>	uppercase/lowercase version of string
<code>.strip()</code>	remove leading/trailing whitespace
<code>.count(s)</code>	number of times see s in string
<code>.startswith(s)</code>	bool of whether the string begins with s
<code>.endswith(s)</code>	bool of whether the string ends with s

List

<code>sum(lst)</code>	sum of the elements in lst
<code>max(lst)</code>	maximum value of lst
<code>min(lst)</code>	minimum value of lst
<code>.append(elm)</code>	Mutates the list by adding elm to the end of the list
<code>.count(elm)</code>	Number of times see elm in the list

WOTO-3 – Split and Join

<http://bit.ly/101f22-0220-3>