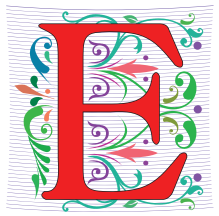
Compsci 101
Selection, Lists, Sequences, Faces

	A	B	Result
A and B	True	True	True
A and B	True	False	False
A and B	False	True	False
A and B	False	False	False
A or B	True	True	True
A or B	True	False	True
A or B	False	True	True
A or B	False	False	False
not A	True		False
not A	False		True

Susan Rodger

January 26, 2023

E is for ...



- **Escape Sequence**
  - Why `\n` is newline and `\t` is a tab
- **Encryption**
  - From Caesar Ciphers to SSL (https) and beyond
- **Enumerate**
  - Iterating over data, counting
- **Email**
  - a way to communicate

Luis von Ahn, Guatemalan entrepreneur
Duke BS Math 2000, CMU PhD CS

“I build systems that combine humans and computers to solve large-scale problem that neither can solve alone. I call this Human Computation, but others sometimes call it crowdsourcing.”

"In college, I thought my goal in life was to get a good GPA, but it's equally important to get involved with a good professor doing good research. Take advantage of what's going on around you."

ELIZABETH HOLMES: LA NUEVA ALIADA DEL MAGNATE CARLOS SLIM
Forbes
CENTROAMÉRICA
LUIS VON AHN
EL "PROFESOR" DE BILL GATES

duolingo

Announcements

- **APT-1 is due tonight!**
  - Run each APT on the APT tester, *1 grace day*
  - Check your grade – click *check submissions*
- **QZ01-05 turned off at 10:15am today!**
  - Be sure to do QZ06 by 10:15am on Thursday!
- **Assignment 1 Faces is out, program due Feb 2**
  - Read the whole thing
  - Assign1 Sakai Quiz – **Due Jan. 31 – no grace day**
- **Lab 2 Friday**
  - **Prelab 2 do before attending lab**
- **Always: Reading and Sakai quiz before next class**

Announcements

QZ01-03 1/28 10:15am
QZ04 1/29 10:15am
QZ05 1/30 10:15am

- **APT-1 is due tonight!**
  - Run each APT on the APT tester, 1 grace day
  - Check your grade – click *check submissions*
- **QZ01-05 turned off at ~~10:15am today!~~** EXTENDED!!!
  - Be sure to do QZ06 by 10:15am on Tuesday!
- **Assignment 1 Faces is out, program due Feb 2**
  - Read the whole thing
  - Assign1 Sakai Quiz – **Due Jan. 31 – no grace day**
- **Lab 2 Friday**
  - **Prelab 2 do before attending lab**
- **Always: Reading and Sakai quiz before next class**

Why is this person so important to this course?

Why is this person so important to this course?

- **Brad Miller, Runestone**
- **He built the Runestone infrastructure for online textbooks.**
- **Our Textbook is on his Runestone platform!**
- **Have you donated yet?**
  - *Everyone should give $10 donation*

Top 10 list for surviving in CompSci 101

10. Read the book and ask questions

9. Eat lots of pizza

8. Learn how to spell Rodger

7. Understand what you turn in

6. Visit your prof in her office hours and the UTAs in consulting hours

Top 10 list (cont)

5. Check Ed Discussion every day

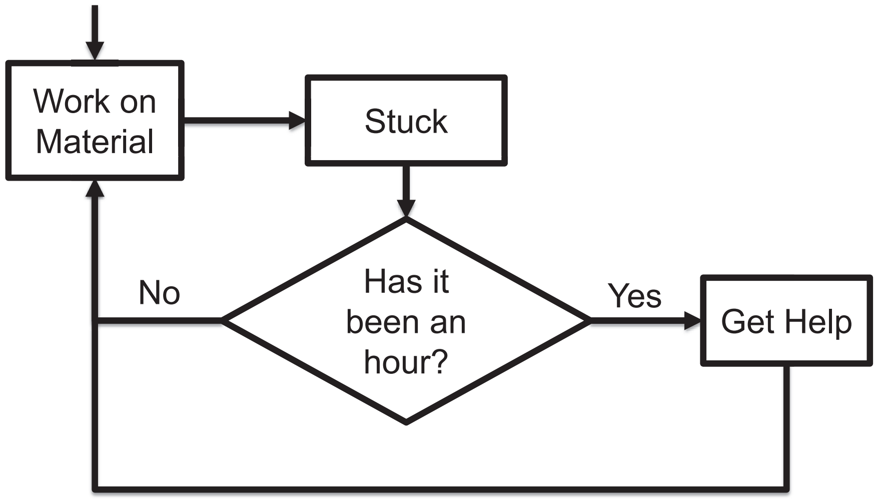
4. Learn how to debug your programs

3. Follow the 7-step process

2. Seek help (One Hour Rule!)

1. Start programming assignments early

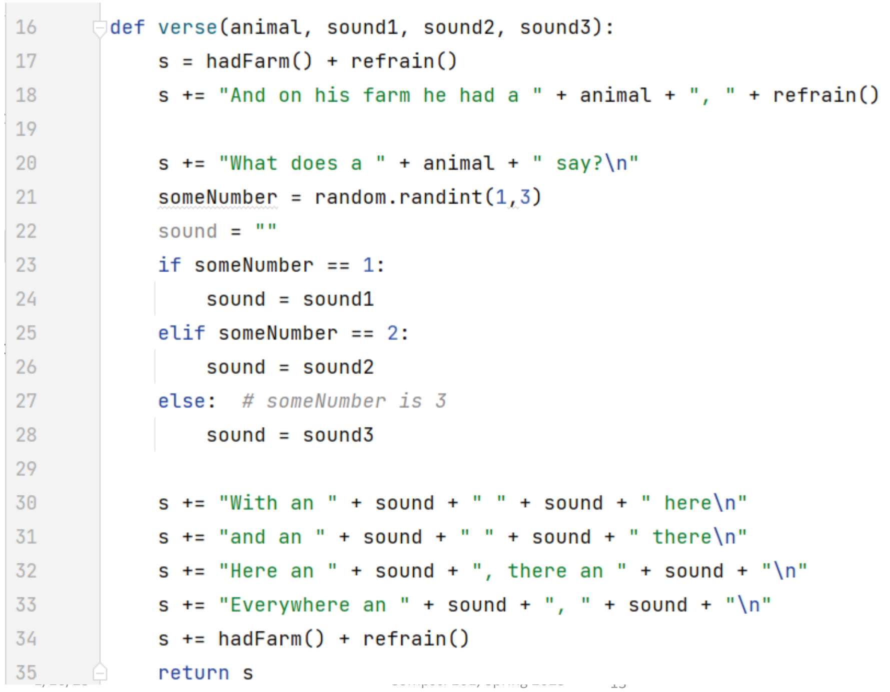
One Hour Rule for Getting Help



PFTD

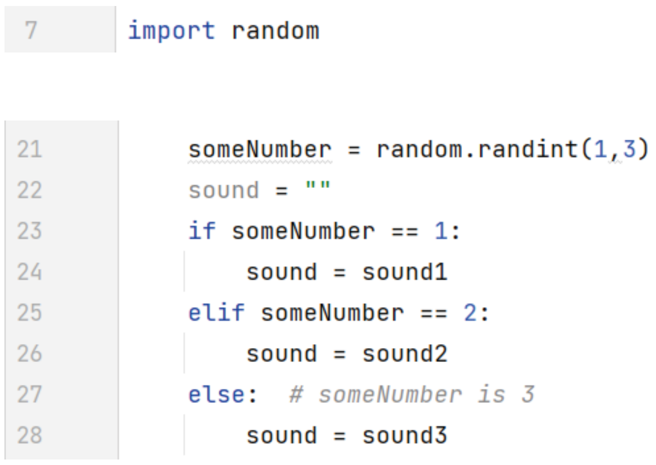
- **Finish WOTO from last time**
- **Assignment 1**
- **Strings**
  - Sequence of characters, “CompSci 101”
- **Lists**
  - Heterogenous sequences
- **Sequences**
  - len(…), indexing, and slicing
- **Functions as Parameters**

Go over WOTO-3 from last time



```python
def verse(animal, sound1, sound2, sound3):
    s = hadFarm() + refrain()
    s += "And on his farm he had a " + animal + ", " + refrain()

    s += "What does a " + animal + " say?\n"
    someNumber = random.randint(1,3)
    sound = ""
    if someNumber == 1:
        sound = sound1
    elif someNumber == 2:
        sound = sound2
    else:  # someNumber is 3
        sound = sound3

    s += "With an " + sound + " " + sound + " here\n"
    s += "and an " + sound + " " + sound + " there\n"
    s += "Here an " + sound + ", there an " + sound + "\n"
    s += "Everywhere an " + sound + ", " + sound + "\n"
    s += hadFarm() + refrain()
    return s
```

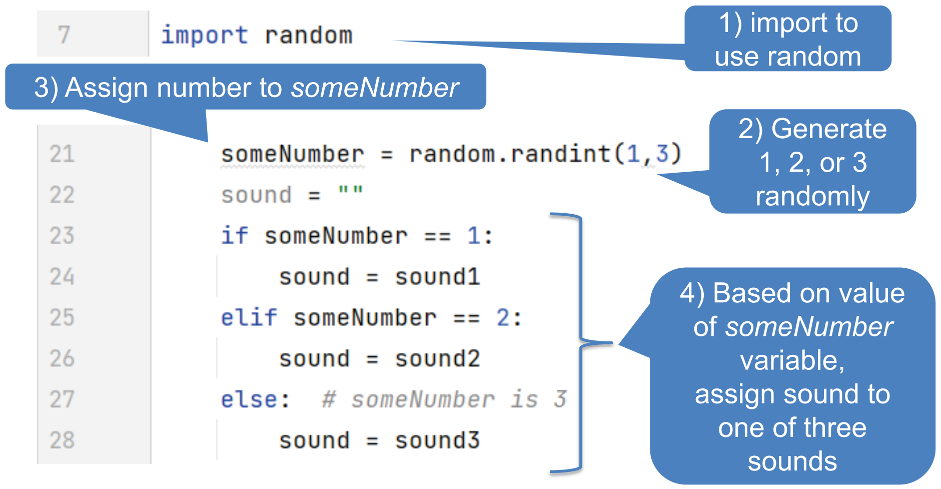
Old MacDonald random



```python
import random
```

```python
    someNumber = random.randint(1,3)
    sound = ""
    if someNumber == 1:
        sound = sound1
    elif someNumber == 2:
        sound = sound2
    else:  # someNumber is 3
        sound = sound3
```

Old MacDonald random

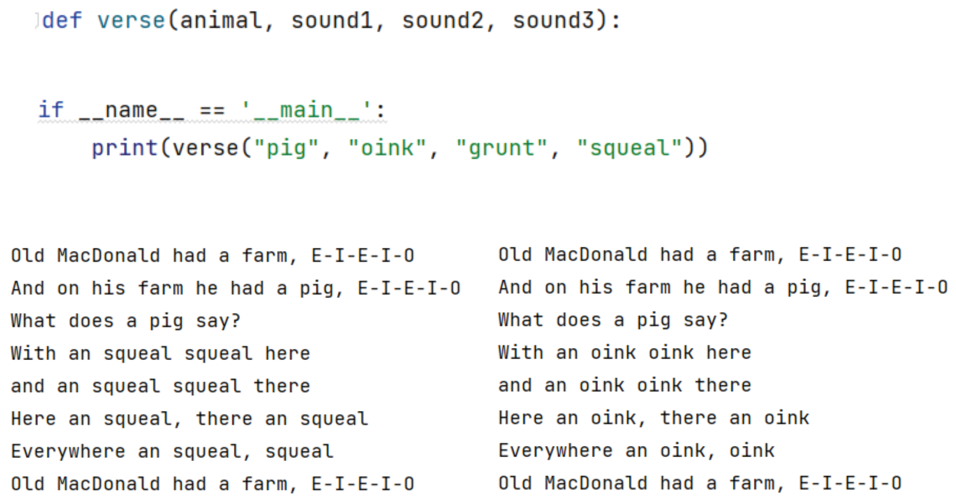

1) import to use random
3) Assign number to *someNumber*
2) Generate 1, 2, or 3 randomly
4) Based on value of *someNumber* variable, assign sound to one of three sounds

```python
import random
```

```python
    someNumber = random.randint(1,3)
    sound = ""
    if someNumber == 1:
        sound = sound1
    elif someNumber == 2:
        sound = sound2
    else:  # someNumber is 3
        sound = sound3
```

Do in Assignment 1: Randomly pick one of three eyes

Run Twice - Different Output



```python
def verse(animal, sound1, sound2, sound3):


if __name__ == '__main__':
    print(verse("pig", "oink", "grunt", "squeal"))
```

```
Old MacDonald had a farm, E-I-E-I-O
And on his farm he had a pig, E-I-E-I-O
What does a pig say?
With an squeal squeal here
and an squeal squeal there
Here an squeal, there an squeal
Everywhere an squeal, squeal
Old MacDonald had a farm, E-I-E-I-O
```

```
Old MacDonald had a farm, E-I-E-I-O
And on his farm he had a pig, E-I-E-I-O
What does a pig say?
With an oink oink here
and an oink oink there
Here an oink, there an oink
Everywhere an oink, oink
Old MacDonald had a farm, E-I-E-I-O
```

Run Twice - Different Output

```
def verse(animal, sound1, sound2, sound3):


if __name__ == '__main__':
    print(verse("pig", "oink", "grunt", "squeal"))
```

Generate 1, 2 or 3
1 use sound1
2 use sound2
3 use sound3

Which random number was generated for this verse?

```
Old MacDonald had a farm, E-I-E-I-O
And on his farm he had a pig, E-I-E-I-O
What does a pig say?
With an squeal squeal here
and an squeal squeal there
Here an squeal, there an squeal
Everywhere an squeal, squeal
Old MacDonald had a farm, E-I-E-I-O
```

3

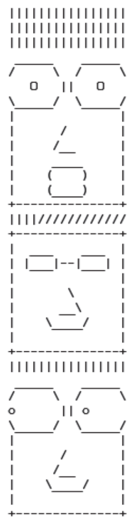
Which random number was generated for this verse?

```
Old MacDonald had a farm, E-I-E-I-O
And on his farm he had a pig, E-I-E-I-O
What does a pig say?
With an oink oink here
and an oink oink there
Here an oink, there an oink
Everywhere an oink, oink
Old MacDonald had a farm, E-I-E-I-O
```

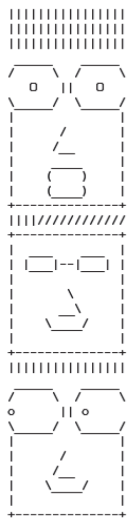
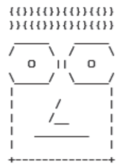
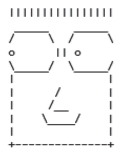
1

Assignment 1 and Pre-Lab 2

- **Assignment 1 Faces due Feb 2**
- **Sakai Quiz on Assignment 1**
  - Read through assignment 1
  - Take the quiz
  - Can take many times
  - Due Jan 31 (no grace day)!
- **Prelab 02 – before lab**
  - Read Assignment 1 and take its quiz once



Assignment 1: Faces

Learning Goals: Faces

- **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!

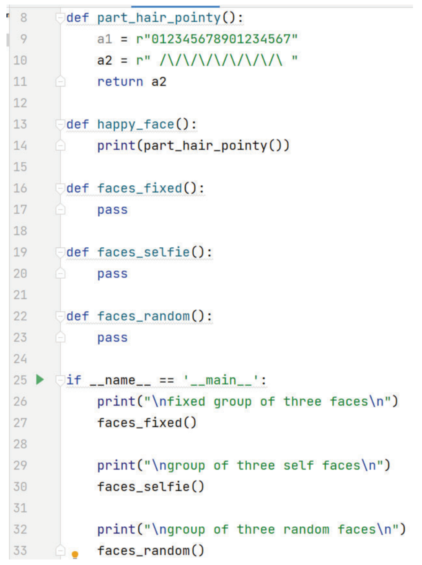
Function Name Format

Function Name Template	Parameters	Returns	Example: Function names
`part_DESCRIPTION`	No parameters	A string	`part_smiling_mouth`
`DESCRIPTION_face`	No parameters	No return value, only prints	`happy_face`
`face_with_DESCRIPTION`	1 or 2 parameters of type function	No return value, only prints	`face_with_mouth`
`faces_DESCRIPTION`	No parameters	No return value, calls face functions	`faces_fixed`, `faces_selfie`, `faces_random`
`selfie_band`, `face_random` – helper functions!			

Creating your program

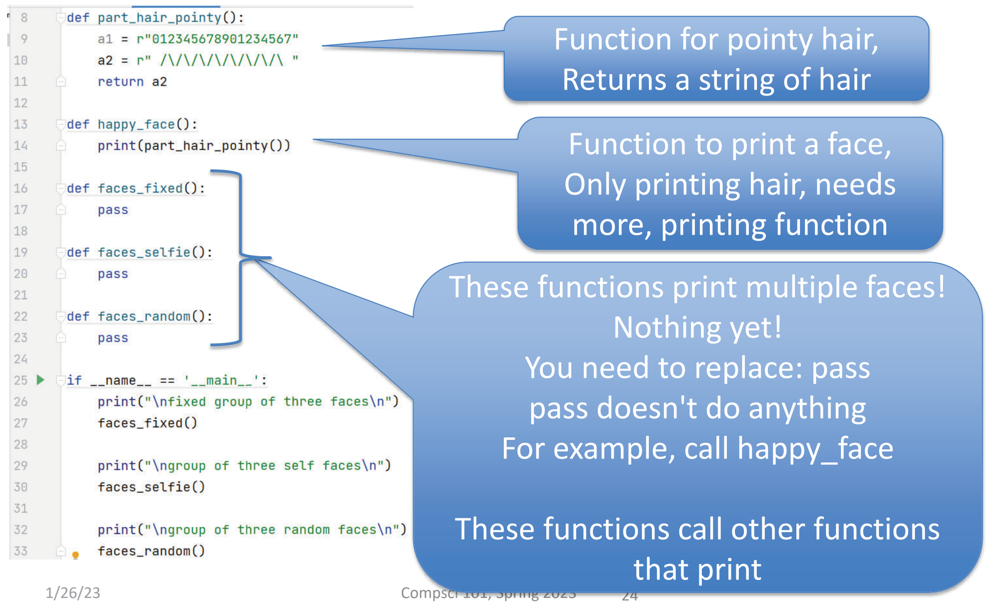
Start small and build incrementally
Use seven steps! Plan what to do!
…

With functions grow by…



```
def part_hair_pointy():
    a1 = r"012345678901234567"
    a2 = r" /\/\/\/\/\/\/\/\ "
    return a2

def happy_face():
    print(part_hair_pointy())

def faces_fixed():
    pass

def faces_selfie():
    pass

def faces_random():
    pass

if __name__ == '__main__':
    print("\nfixed group of three faces\n")
    faces_fixed()

    print("\ngroup of three self faces\n")
    faces_selfie()

    print("\ngroup of three random faces\n")
    faces_random()
```

With functions grow by…



```
def part_hair_pointy():
    a1 = r"012345678901234567"
    a2 = r" /\/\/\/\/\/\/\/\ "
    return a2

def happy_face():
    print(part_hair_pointy())

def faces_fixed():
    pass

def faces_selfie():
    pass

def faces_random():
    pass

if __name__ == '__main__':
    print("\nfixed group of three faces\n")
    faces_fixed()

    print("\ngroup of three self faces\n")
    faces_selfie()

    print("\ngroup of three random faces\n")
    faces_random()
```

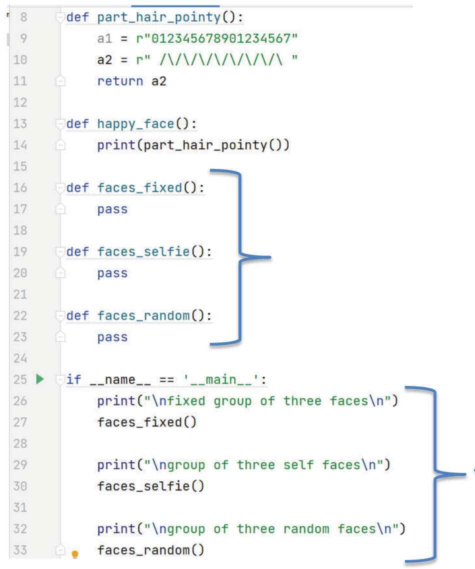
Function for pointy hair, Returns a string of hair

Function to print a face, Only printing hair, needs more, printing function

These functions print multiple faces!
Nothing yet!
You need to replace: pass
pass doesn't do anything
For example, call happy_face

These functions call other functions that print

With functions grow by…



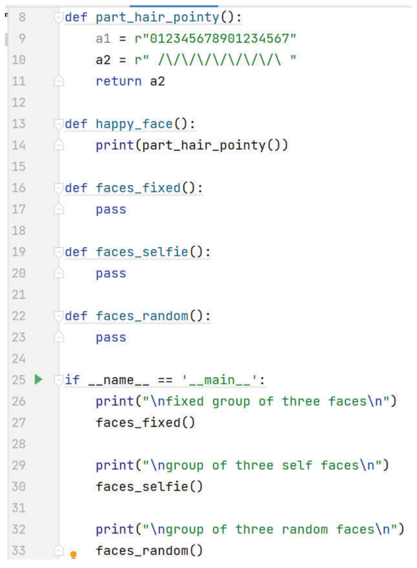
```
8  def part_hair_pointy():
9      a1 = r"012345678901234567"
10     a2 = r" /\/\/\/\/\/\/\ "
11     return a2
12
13 def happy_face():
14     print(part_hair_pointy())
15
16 def faces_fixed():
17     pass
18
19 def faces_selfie():
20     pass
21
22 def faces_random():
23     pass
24
25 if __name__ == '__main__':
26     print("\nfixed group of three faces\n")
27     faces_fixed()
28
29     print("\ngroup of three self faces\n")
30     faces_selfie()
31
32     print("\ngroup of three random faces\n")
33     faces_random()
```

Program starts here!

These call the functions in lines 16-23

Nothing to do here

With functions grow by…



```
8  def part_hair_pointy():
9      a1 = r"012345678901234567"
10     a2 = r" /\/\/\/\/\/\/\ "
11     return a2
12
13 def happy_face():
14     print(part_hair_pointy())
15
16 def faces_fixed():
17     pass
18
19 def faces_selfie():
20     pass
21
22 def faces_random():
23     pass
24
25 if __name__ == '__main__':
26     print("\nfixed group of three faces\n")
27     faces_fixed()
28
29     print("\ngroup of three self faces\n")
30     faces_selfie()
31
32     print("\ngroup of three random faces\n")
33     faces_random()
```

Minimal code that does run and can be submitted

Where go from here?

- Add face part functions to create happy_face()
- Create the next face function for faces_fixed and any new face part functions
- Try a face_with function
- Go to the next group of faces
- etc.

Faces Assignment
What should you do …

- **Read the assignment**
- **Do the Assignment 1 Sakai quiz**
- **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 Discussion (anonymously) or at consulting/office hours**

Review Selection Syntax

```
if BOOLEAN_CONDITION:
    CODE_BLOCK_A
```

```
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
```

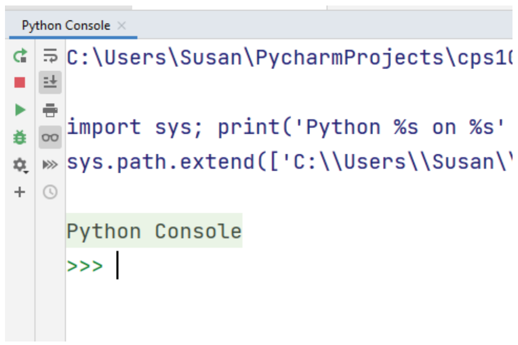
- **What is similar and different?**
  - What other variations could work?
  - Could only `elif…else` work?
- **if – required**
- **elif – optional, as many as needed**
- **else – optional, no condition**

Boolean condition (True/False)

```
if BOOLEAN_CONDITION:
    CODE_BLOCK_A
```

- See `type(3 < 5)`
- Relational operators: `<  <=  >  >=  ==  !=`
- Boolean operators: `and  or  not`

Console on Booleans



```
C:\Users\Susan\PycharmProjects\cps1(

import sys; print('Python %s on %s'
sys.path.extend(['C:\\Users\\Susan\'

Python Console
>>> |
```

Boolean Operations

	A	B	Result
A and B	True	True	True
A and B	True	False	False
A and B	False	True	False
A and B	False	False	False
A or B	True	True	True
A or B	True	False	True
A or B	False	True	True
A or B	False	False	False
not A	True		False
not A	False		True

Boolean Operations

	A	B	Result
A and B	True	True	True
A and B	True	False	False
A and B	False	True	False
A and B	False	False	False
A or B	True	True	True
A or B	True	False	True
A or B	False	True	True
A or B	False	False	False
not A	True		False
not A	False		True

IF my cat is hungry ***AND*** she likes the food, she will eat dinner.

IF it is raining ***OR*** it might rain today, I will carry an umbrella.

IF I did ***NOT*** have dessert yesterday, I may have dessert today.

Example with And and Or

```
x = 3
y = 8
if x < 2 or y > 2:
    print("first")
elif x > 2 and y < 2:
    print("second")
else:
    print("third")
```

OUTPUT:

```
x = 3
y = 2
if x < 2 or y > 2:
    print("first")
elif x > 2 and y < 2:
    print("second")
else:
    print("third")
```

OUTPUT:

Example with And and Or

```
x = 3
y = 8
if x < 2 or y > 2:
    print("first")
elif x > 2 and y < 2:
    print("second")
else:
    print("third")
```

OUTPUT:
first

Example with And and Or

```
x = 3
y = 8
if x < 2 or y > 2:
    print("first")
elif x > 2 and y < 2:
    print("second")
else:
    print("third")
```

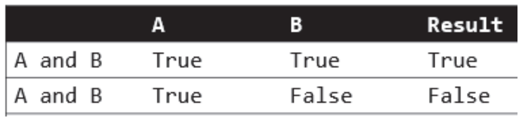
OUTPUT:
first

```
x = 3
y = 2
if x < 2 or y > 2:
    print("first")
elif x > 2 and y < 2:
    print("second")
else:
    print("third")
```

OUTPUT:
third

WOTO-1 Review Functions and Booleans http://bit.ly/101s23-0126-1

- **In your groups:**
  - Come to a consensus



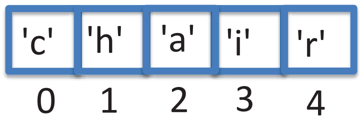
	A	B	Result
A and B	True	True	True
A and B	True	False	False

Strings - indexing

- **x = "chair"**
- **y = "desk"**
- **z = x[2] + y[2] + y[3]**
- **w= len(x)**
- **v = x[len(y)]**
- **t = x[len(x)]**

Strings - indexing

- **x = "chair"**
- **y = "desk"**
- **z = x[2] + y[2] + y[3]**
- **w= len(x)**
- **v = x[len(y)]**
- **t = x[len(x)]**

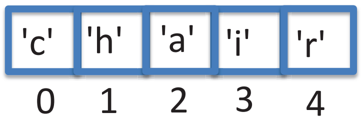


A string is a sequence of characters, numbered starting at 0

Strings - indexing

What are the values of z, w, v and t?

- **x = "chair"**
- **y = "desk"**
- **z = x[2] + y[2] + y[3]** **z is "ask"**
- **w= len(x)** **w is 5**
- **v = x[len(y)]** **v is "r"**
- **t = x[len(x)]** **t is ERROR !!!!!!!!**



Lists

- **Syntax: `[ITEM_1, ITEM_2, ITEM_3, …]`**
  - Starts and ends with square brackets: […]
  - Elements in the list are divided by commas "**,**"
- **Lists can be *heterogenous* sequence**
  - Strings, ints, lists, anything

```
[1, 2, 3]
["hello", "world"]
["count", "off", 1, 2, 3.0, "done"]
```

Python Sequences

- **Types String and List are both sequences**
- **A sequence in Python has**
  - Length - `len(…)`
  - Membership – `in`
  - Indexing and slicing – `[n], [n:m]`
- **Difference:**
  - String is immutable – cannot change
  - List is mutable – can change

`len(…)` for Python Sequences

- **Length – the number of *elements* in a sequence**
- **`len(…)` – returns the length of a sequence**

- **`s="hello world"   l=["hello", "world"]`**
  - What is `len(s)`?

  - `What is len(l)?`

`len(…)` for Python Sequences

- **Length – the number of *elements* in a sequence**
- **`len(…)` – returns the length of a sequence**

- **`s="hello world"   l=["hello", "world"]`**
  - What is `len(s)`?
    - **11**
  - `What is len(l)?`
    - **2**

`in` for Python Sequences

- **`in` checks for membership in the sequence**
  - True/False – if `element in seq`

- **`s="hello world"  lst=["hello", "world"]`**
  - What is an element for the string `s`? List `lst`?

  - What is: `'h' in s`?
  - What is: `'h' in lst`?
  - What is: `"hello" in lst`?

`in` for Python Sequences

- **`in` checks for membership in the sequence**
  - True/False – if `element in seq`
- **`s="hello world"  lst=["hello", "world"]`**
  - What is an element for the string `s`? List `lst`?

    s has 'h', 'e', etc,   lst has "hello", "world"
  - What is: `'h' in s`? True
  - What is: `'h' in lst`? False
  - What is: `"hello" in lst`? True

Indexing Python Sequences

- **`s="hello world" l=["hello", "world"]`**
- Indexing provides access to individual elements
  - Compare **`s[0]`** and **`l[0]`** "h" vs "hello"
    - Start with 0 offset, what is last valid positive index?
  - Compare **`s[-1]`** and **`l[-1]`** "d" vs "world"
    - What is negative index of second to last element?
    - Index **-n** is the same as index **`len(seq) - n`**

      -2 11 – 2 is 9

0	1	2	3	4	5	6	7	8	9	10
H	E	L	L	O		W	O	R	L	D
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Indexing Python Sequences

- **`s="hello world" l=["hello", "world"]`**
- Indexing provides access to individual elements
  - Compare **`s[0]`** and **`l[0]`**
    - Start with 0 offset, what is last valid positive index?
  - Compare **`s[-1]`** and **`l[-1]`**
    - What is negative index of second to last element?
    - Index **-n** is the same as index **`len(seq) - n`**

0	1	2	3	4	5	6	7	8	9	10
H	E	L	L	O		W	O	R	L	D
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Slicing Python Sequences

- **`s="hello world"`**
- **`lst=["my", "big", "beautiful", "world"]`**
- Slicing provides sub-sequence (string or list)
  - `seq[n:m]` – all elements `i`, s.t. `n <= i < m`
  - Compare **`s[0:2]`** and **`lst[0:2]`**
    - **`s[0:2] is`**
    - **`lst[0:2] is`**
  - What is length of subsequence? **`len(lst[1:3])`**
    - **`lst[1:3] is`**
    - **`len(lst[1:3]) is`**

Slicing Python Sequences

- `s="hello world"`
- `lst=["my", "big", "beautiful", "world"]`
- Slicing provides sub-sequence (string or list)
  - `seq[n:m]` – all elements `i`, s.t. n `<= i < m`
  - Compare `s[0:2]` and `lst[0:2]`
    - `s[0:2] is "he"`
    - `lst[0:2] is ["my", "big"]`
  - What is length of subsequence? `len(lst[1:3])`
    - `lst[1:3] is ["big", "beautiful"]`
    - `len(lst[1:3]) is 2`

Slicing Python Sequences (more)

- `s = "hello world"`
- `lst=["my", "big", "beautiful", "world"]`
- Slicing provides sub-sequence (string or list)
  - Compare `s[4:-1]` and `lst[2:-1]`
    - `s[4:-1] is`
    - `lst[2:-1] is`
  - Is last index part of subsequence?
  - Omit last value. Compare s[2:] , s[:3]
    - `s[2:] is`
    - `s[:3] is`

Slicing Python Sequences (more)

- `s = "hello world"`
- `lst=["my", "big", "beautiful", "world"]`
- Slicing provides sub-sequence (string or list)
  - Compare `s[4:-1]` and `lst[2:-1]`
    - `s[4:-1] is "o worl"`
    - `lst[2:-1] is ["beautiful"]`
  - Is last index part of subsequence?
    - **NO, in `s[2:4]` we go up to but not including 4**
  - Omit last value. Compare s[2:] , s[:3]
    - `s[2:] is "llo world"`
    - `s[:3] is "hel"`

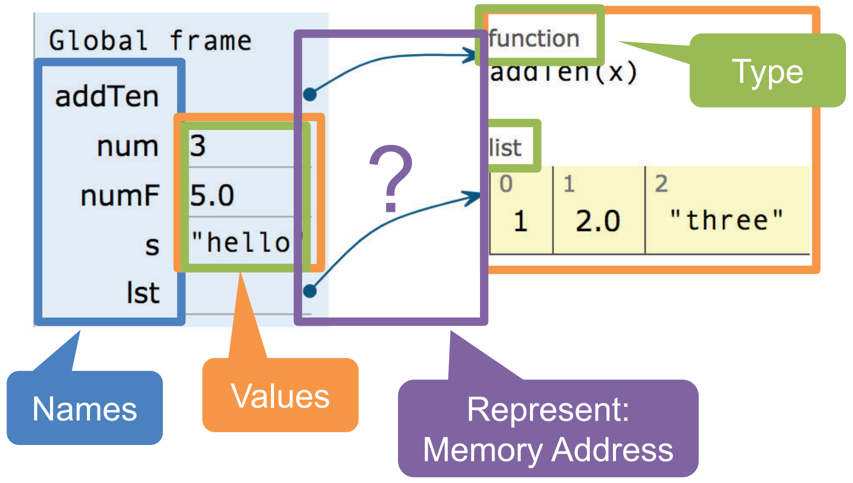
WOTO-2 Sequence Length Indexing http://bit.ly/101s23-0126-2

- **In your groups:**
  - Come to a consensus

Learning Goals: Faces

- **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!

Name vs Value vs Type


Global frame
addTen
num
numF
s
lst
3
5.0
"hello
?
function
addTen(x)
list
0 1 2
1 2.0 "three"
Type
Names
Values
Represent: Memory Address