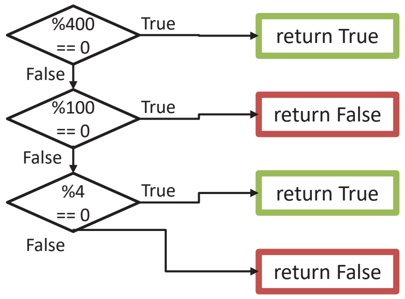
Compsci 101
Selection, Lists, Sequences, Totem



Susan Rodger

September 13, 2022

E is for …

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

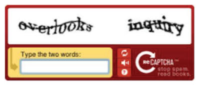
Luis von Ahn, Guatemalan entrepreneur
Duke BS Math 2000, CMU PhD CS

"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 Thur, Sept 15! 11:30pm**
  - 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 Sept 22**
  - Read the whole thing
  - Take assign1 quiz on Sakai – **Due Sept 20**
- **Lab 2 Friday**
  - **Prelab 2 do before attending lab, out today**
- **Always: Reading and Sakai quiz before next class**

Why is this person so important to this course?

PFTD

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

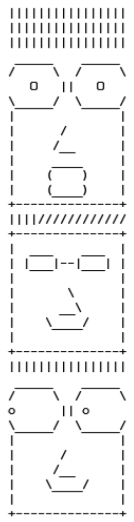
Go over WOTO-3 from last time

Assignment 1 and Pre-Lab 2

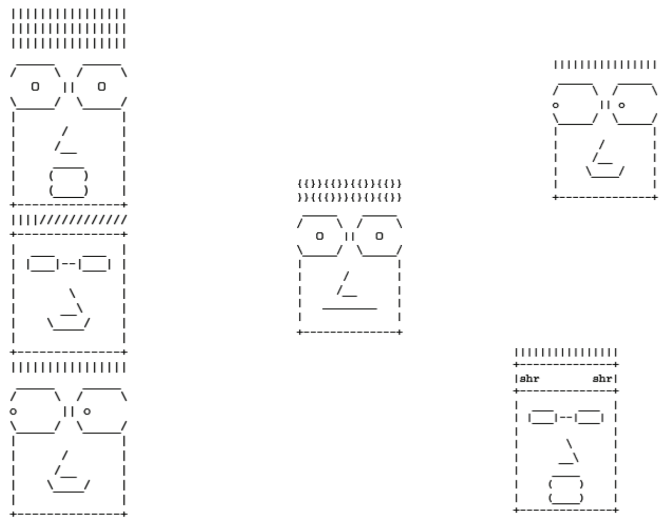
- **Assignment 1 Faces due Sept 22**

- **Sakai Quiz on Assignment 1**
  - Read through assignment 1
  - Take the quiz
  - Can take many times
  - Due Sept 20 (no grace day)!

- **Prelab 02 – before lab**
  - Read Assignment 1 and take its quiz once



Assignment 1: Faces



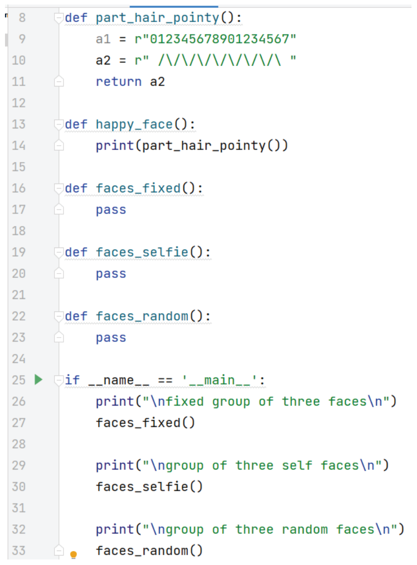
Learning Goals: Faces

- **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	Parameters	Returns	Example
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!			

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()
```

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
```

Could this else not be here?

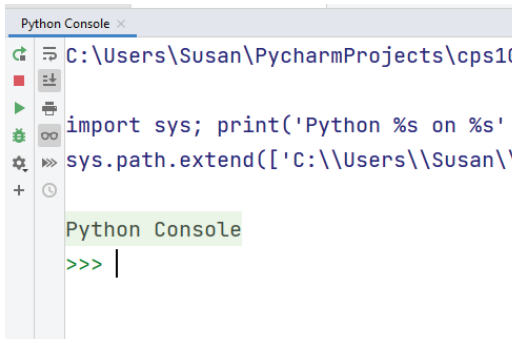
- **What is similar and different?**
  - What other variations could work?
  - Could only `elif…else` work?
- **if – required**
- **elif – optional, as many as needed**
- **else – optional, no condition**

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

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:

WOTO-1 Review Functions and Booleans
http://bit.ly/101f22-0913-1

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

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

When is a leap year?

- https://en.wikipedia.org/wiki/Leap_year
- **"years which are multiples of four (except NOT if years divisible by 100 but not by 400)"**
- **2004/4 = 501, 2004/100=20.04, 2004/400=5.01**
- **2200/4 = 550, 2200/100=22, 2200/400 = 5.5**
- **2000/4 =500 and 2000/100 = 20, 2000/400 = 5**

WOTO-2: Which LeapYear correct? http://bit.ly/101f22-0913-2

- **is_leap_one**
- **is_leap_two**

Which LeapYear correct?

- **Is 1900 a leap year?**

- **Which program is correct?**
- **What is wrong with the program that is not correct?**

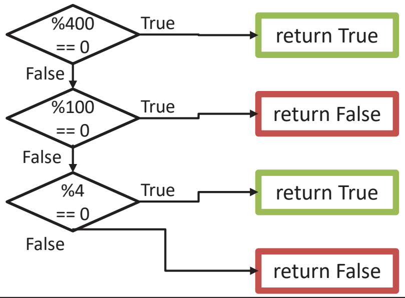
Wikipedia Leap Year Algorithm

- **See algorithm section**
  - https://en.wikipedia.org/wiki/Leap_year

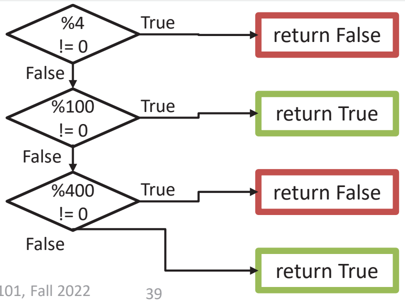
```
def is_leap(year):
    if year % 4 != 0:
        return False        # not leap
    elif year % 100 != 0:   # 1968
        return True
    elif year % 400 != 0:
        return False        #1968
    else:
         return True            #2000
```

Flowchart: if vs if...elif...else

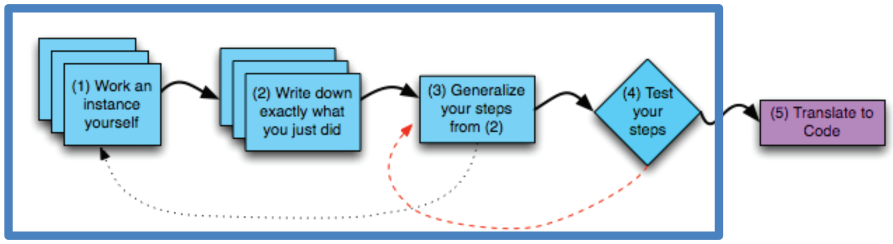
```
def is_leap_one(year):
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    if year % 4 == 0:
        return True
    return False
```



```
def is_leap(year):
    if year % 4 != 0:
        return False
    elif year % 100 != 0:
        return True
    elif year % 400 != 0:
        return False
    else:
        return True
```



if's or if…elif…else?


(1) Work an instance yourself
(2) Write down exactly what you just did
(3) Generalize your steps from (2)
(4) Test your steps
(5) Translate to Code

- **Remember steps 1-4 do not involve code!**
- **After have plan, choose based on what works best**
  - There could be multiple ways to implement it

Strings

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

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)?`

`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`?
  - `"hello" in lst?`

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 (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`**

WOTO-3 Sequence Length Indexing http://bit.ly/101f22-0913-3

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