CompSci 101
Fall 2021

Lecture 13

Reminders

- Assignments
  - APT 4-live
- Exam 2
  - 10/26
  - Review session (10/21)
- 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

- Images & Tuples cont.
- Sets and APTs

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. Helen Chavez

- PhD-Arizona State University
- BS/MS-Tecnológico de Monterrey, Guadalajara
- Lecturer
  - Arizona State University
  - Affective computing, intelligent tutoring systems, engineering education
- Co-Chair, Celebrations Committee
  - Association for Computing Machinery's Council on Women in Computing, North America Committee (ACM-W North America)

Tuple: What and Why?

- Similar to a list in indexing starting at 0
  - Can store any type of element
  - Can iterate over
- Immutable - Cannot mutate/change its value(s)
  - Efficient because it can't be altered
- Consider `x = (5,6)` and `y = ([1,2],3.14)`
  - What is `x[0] = 7`? `y[0].append(5)`?
  - https://goo.gl/ooyHPQ

Activity 1: https://bit.ly/101f21-10-14-1

Make Gray: Notice the Tuples!

```
13  def grayByPixel(img, debug=False):
14      width = img.width
15      height = img.height
16      new_img = img.copy()
17      if debug:
18          print("creating %d x %d image" % (width,height))
19      for x in range(width):
20          for y in range(height):
21              (r,g,b) = img.getpixel((x,y))
22              grays = getGray(r,g,b)
23              new_img.putpixel((x,y),grays)
```

TPS: How does this code make a grey image?

TPS: New stuff here, what and where?

Revisiting nested Loops

- What is printed here? y varies first
  - Value of x as inner loop iterates?

```
>>> for x in range(5):
...     for y in range(3):
...         print(x, y)
```

```
0 0
0 1
0 2
1 0
1 1
1 2
2 0
2 1
2 2
3 0
3 1
3 2
4 0
4 1
4 2
```

TPS: Why is the first column have the number repeated like that? What if the print became: `print(y, x)`?

Make Gray cont.

```
13  def grayByPixel(img, debug=False):
14      width = img.width
15      height = img.height
16      new_img = img.copy()
17      if debug:
18          print("creating %d x %d image" % (width,height))
19      for x in range(width):
20          for y in range(height):
21              (r,g,b) = img.getpixel((x,y))
22              grays = getGray(r,g,b)
23              new_img.putpixel((x,y),grays)
```

Tuple

Tuple

Nested Loops

Tuple

TPS: How many parameters does putpixel have?

Accessing Individual Pixels is Inefficient

- Accessing each one one-at-a-time is inefficient
  - Python can do better "under the hood"

- PIL provides a function **`img.getdata()`**
  - Returns list-like object for accessing all pixels
  - Similar to how file is a sequence of characters
  - Symmetry: **`img.putdata(sequence)`**

Processing all Pixels at Once

- Treat **`img.getdata()`** as list, it's not quite a list
  - Iterable: object use in "`for … in …`" loop

```
27  def grayByData(img, debug=False):
28      pixels = [getGray(r,g,b) for (r,g,b) in img.getdata()]
29      new_img = Image.new("RGB", img.size)
30      new_img.putdata(pixels)
```

TPS: An image is 2D and putdata(seq) takes a 1D sequence. How did we get an image?

Hint: What type are the elements in the list comprehension?

Hint: What do we know about the length of that sequence and the sequence putdata(…) needs?

Summary of Image functions

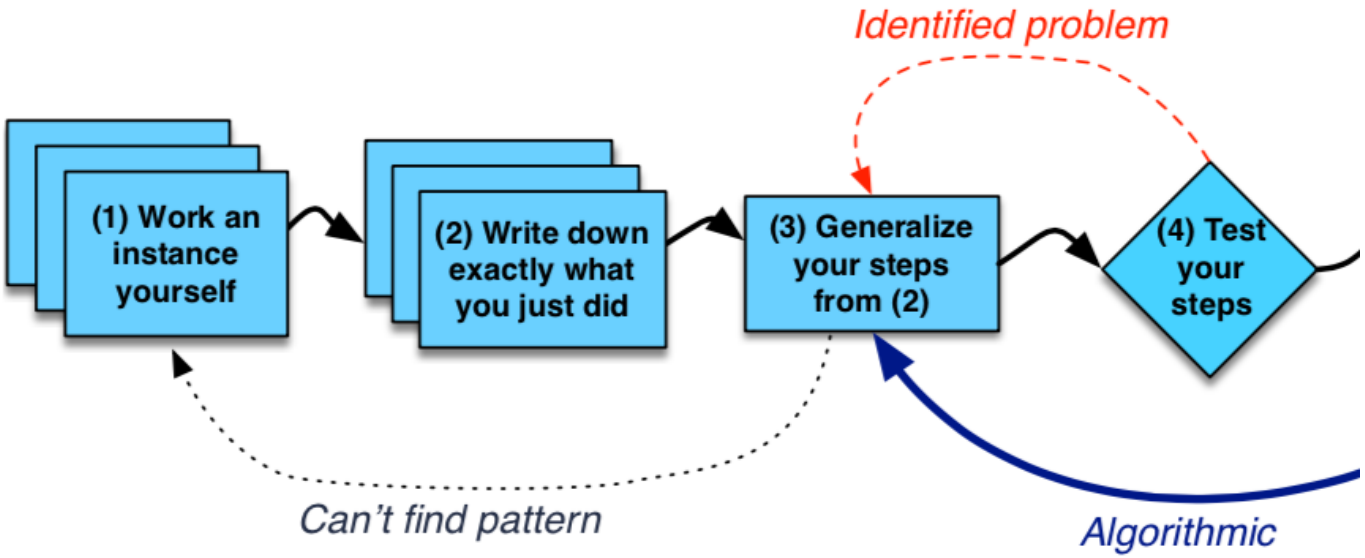
- Many, many more
  - http://bit.ly/pillow-image

Image function/method	Purpose
`im.show()`	Display image on screen
`im.save("foo.jpg")`	Save image with filename
`im.copy()`	Return copy of im
`im.getdata()`	Return iterable pixel sequence
`im.load()`	Return Pixel collection indexed by tuple (x,y)

Activity 2: https://bit.ly/101f21-10-14-2

Sets: Eating Good

- https://www2.cs.duke.edu/csed/pythonapt/eatinggood.html



- TPS: Do steps 1-4 of 7-steps
  - Especially think on: How do we guard against "double adding"

Lists or Set?

```
if name not in names:
    names.append(name)
```

PYCHARM SYNTAX

- For EatingGood, we must avoid adding the same element more than once
  - Lists store duplicates →[ ]
  - Sets do not store duplicates→ { }

List and Set, Similarities/Differences

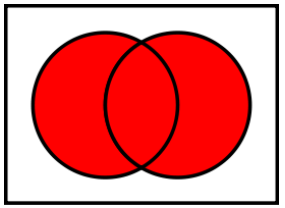
	Function for List	Function for Set
Adding element	`x.append(elt)`	`x.add(elt)`
Size of collection	`len(x)`	`len(x)`
Combine collections	`x + y`	`x | y`
Iterate over	`for elt in x:`	`for elt in x:`
Element membership	`elt in x`	`elt in x`
Index of an element	`x.index(elt)`	`CANNOT DO THIS`

- Lists are ordered and indexed, e.g., has a first or last
- Sets are **not** ordered, very fast, e.g., **`if elt in x`**

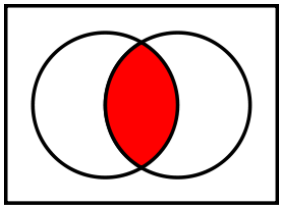
- **`*Python console examples*`**

Python Set Operators
Demo

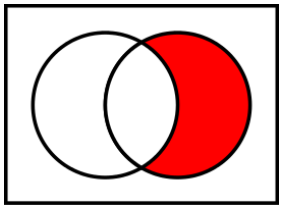
- Using sets and set operations often useful
- A | B, set union
  - Everything



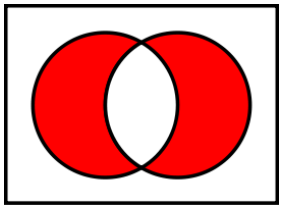
- A & B, set intersection
  - *Only* in both



- B – A, set difference
  - In B *and* not A



- A ^ B, symmetric diff
  - Only in A *or* only in B



Activity 3: https://bit.ly/101f21-10-14-3

Another Trip to the SandwichBar

- https://www2.cs.duke.edu/csed/pythonapt/sandwichbar.html
- What does this problem look like without sets?


```
for dex in range(len(orders)):
    if canmake(orders[dex], available):
        return dex
```

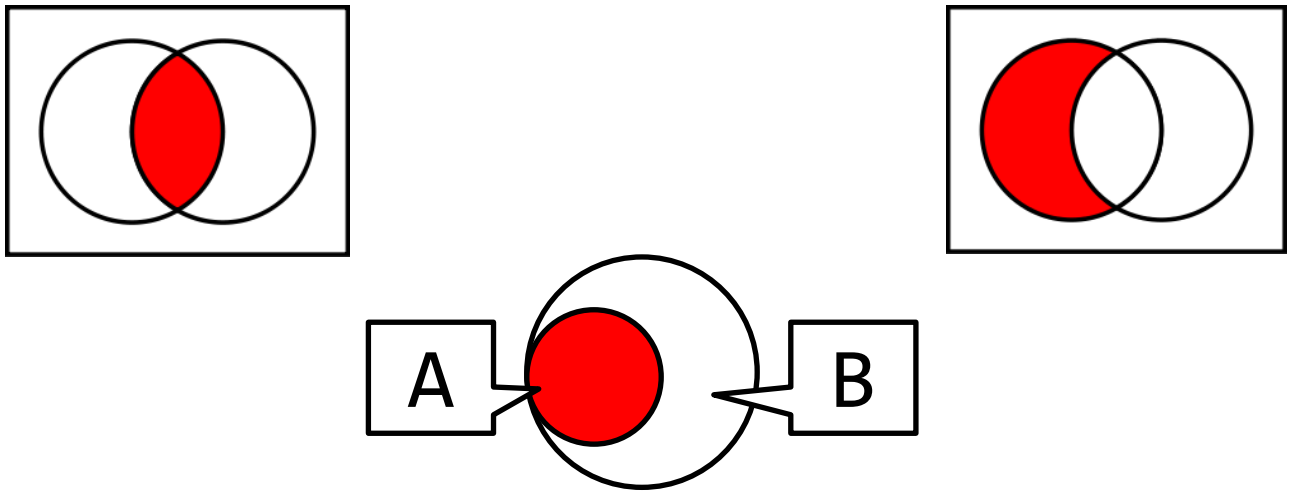
Given two lists A and B

- Determine if all elements in A are also in B
  - Examine each element in A
    - If not in B? False
  - After examining all elements? True

- TPS: Could we use sets instead?

Given two sets A and B

- Determine if all elements in A are also in B
  - `if len(A & B) == len(A)`
  - `if len(A - B) == 0`



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