CompSci 101
Fall 2021

Lecture 12

Reminders

- Assignments
  - APT 4 live
- Exam 2
  - 10/26
- 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

- DeMorgan's Law
- Short Circuiting
- Images & Tuples

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. Rocío Aldeco-Pérez

- PhD-University of Southampton
- BS/MS-Benemérita Universidad Autónoma
- Research Associate Professor
  - Universidad Nacional Autónoma de México
  - Privacy of information, information security, applications of blockchain to improve services
- Vice Chair
  - Association for Computing Machinery's Council on Women in Computing, North America Committee (ACM-W North America)

Activity 2: https://bit.ly/101f21-10-12-2

- This helps with APT-4

Index without error?

```
["a","b","c","a"].index("b") == 1
["a","b","c","a"].index("B") ERROR!
["a","b","c","a"].index("B") ??? -1
```

- Use while loop to implement index.
- TPS: What is the while loop's Boolean condition?

```
i = 0
while BOOL_CONDITION:
   i += 1
```

Index without error?

```
["a","b","c","a"].index("b") == 1
["a","b","c","a"].index("B") ERROR!
["a","b","c","a"].index("B") ??? -1
```

- Use while loop to implement index.
- TPS: What is the while loop's Boolean condition?
  - Whether found value: `lst[dex] == elm`
  - Whether reach end of list: `dex >= len(lst)`

  - `Note: dex→i`

DeMorgan's Law

- While loop stopping conditions, stop with either:
  - `lst[dex] == elm`
  - `dex >= len(lst)`
- While loop needs negation: DeMorgan's Laws

  `not (A and B)` equivalent to **`(not A) or (not B)`**

  `not (A or B)` equivalent to **`(not A) and (not B)`**

```
while not (lst[dex] == elm or dex >= len(lst)):

while lst[dex] != elm and dex < len(lst):
```

Why did >= become < ?

TPS: DeMorgan's Law

Fill in the blanks

A	B	not (A and B)	(not A) or (not B)
True	True		False
True	False	True	
False	True		True
False	False	True	

A	B	not (A or B)	(not A) and (not B)
True	True	False	
True	False		False
False	True	False	
False	False		True

TPS: DeMorgan's Law

Fill in the blanks

A	B	not (A and B)	(not A) or (not B)
True	True	False	False
True	False	True	True
False	True	True	True
False	False	True	True

A	B	not (A or B)	(not A) and (not B)
True	True	False	False
True	False	False	False
False	True	False	False
False	False	True	True

TPS: Will this work?

- If not, what input will not work?

```
1  def index(lst, elm):
2      dex = 0
3      while lst[dex] != elm and dex < len(lst):
4          dex += 1
5      if dex < len(lst):
6          return dex
7      else:
8          return -1
```

Short Circuit Evaluation

- Short circuit evaluation, these are not the same!

```
3       while lst[dex] != elm and dex < len(lst):
```

```
3       while dex < len(lst) and lst[dex] != elm:
```

- As soon as truthiness of expression known
  - Stop evaluating
  - In `(A and B)`, if `A` is false, do not evaluate `B`

Example: To sit in the student section of a game you need to "have a ticket" and "be a student"

Python Logic Summarized

- A and B is True only when A is True and B is True
- A or B is False only when A is False and B is False

- Short-circuit evaluation of A or B ?
  - If A is true, do not evaluate B

A	B	Evaluate B with and?	Evaluate B with or?
True	True	Yes	No
True	False	Yes	No
False	True	No	Yes
False	False	No	Yes

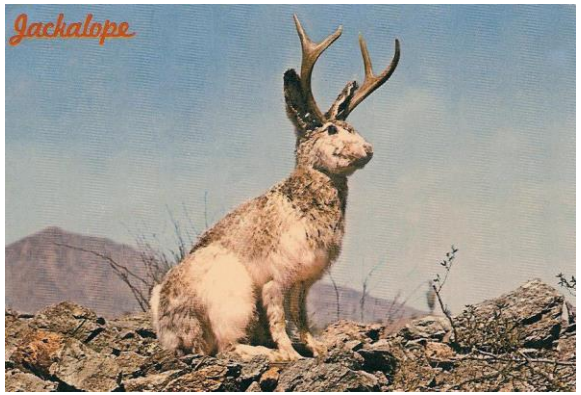
Activity 1: https://bit.ly/101f21-10-12-1

Images

What is photoshop?

Image Processing

- Convert image into format for manipulating the image
  - Visualization, Sharpening, Restoration, Recognition, Measurement, more
  - Resizing, Red-eye Removal, more
  - CrashCourse: Navigating Digital Info
    - http://bit.ly/dukecs101-cc-images



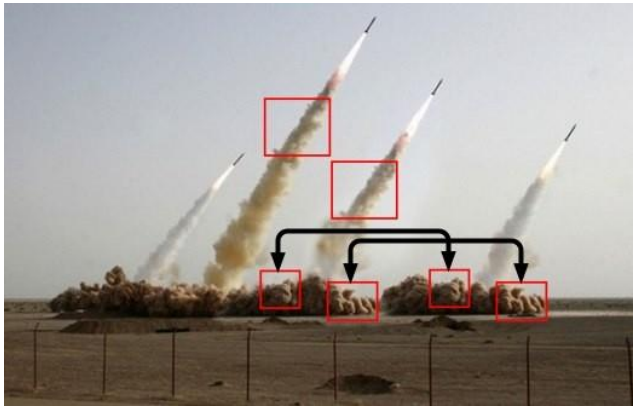


Image Library

- PIL: Python Image Library -> Pillow
  - To install run the command below in a terminal
    - Terminal in PyCharm, not “Python Console”
  - `python3 -m pip install Pillow`
    - OR
  - `pip install Pillow`

- Library has extensive API, far more than we need
  - Concepts often apply to every image library
  - Realized in Python-specific code/functions

SimpleDisplay.py

- Access to PIL and Image module
  - What type is img?
  - https://pillow.readthedocs.io/en/latest/

```
6   from PIL import Image
7
8   if __name__ == '__main__':
9       img = Image.open("images/bluedevil.png")
10      img.show()
11      print("width %d, height %d" % (img.width, img.height))
```

What is a class in Python?

- Class ≈ module ≈ library (for CS101)
- Class – Also blueprint/factory for creating objects
  - We've used int, float, str
  - **`<class 'int'>, <class 'list'>`**
  - Everything is a class in Python3

- Use . dot notation to access object's innards
  - **`"Hello".lower()`** is a function aka method
  - **`img.width`** is an attribute aka field/property

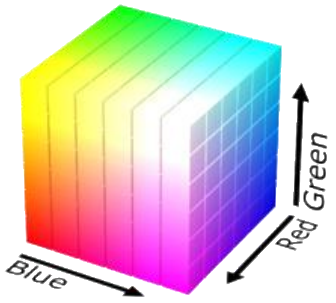
Image Library Basics

- Library can create/open images in different formats, e.g., .png, .jpg, .gif, …
- Images have properties: width, height, type, color-model, and more
  - Functions and fields access these properties, e.g., **`im.width`**, **`im.format`**, and more
- Pixels are formed as triples (255,255,255), (r,g,b)
  - In Python these are tuples: immutable sequence

Color Models

- Cameras, Displays, Phones, JumboTron: RGB
  - Additive Color Model: Red, Green, Blue
  - https://en.wikipedia.org/wiki/RGB_color_model

- Contrast Printers and Print which use CMYK
  - Subtractive: Cyan, Magenta, Yellow, Key/Black

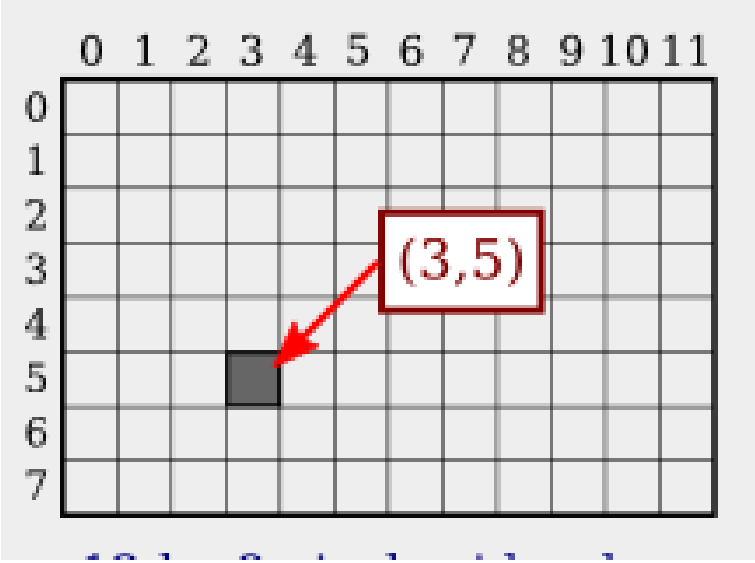


Example: Convert Color to Gray

Process each pixel
Convert to gray

First View of Image for Grayscale

- Image is a collection of pixels
  - Organized in rows: # rows is image height
  - Each row has the same length: image width
- Pixels addressed by (x, y) coordinates
  - Upper-left (0,0), Lower-right (width-1,height-1)
  - Typically is a single (x, y) entity: tuple
- Tuple is immutable, indexed sequence (a, b, c)



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)`
  - TPS What is `x[0] = 7`? `y[0].append(5)`?
  - https://goo.gl/ooyHPQ

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