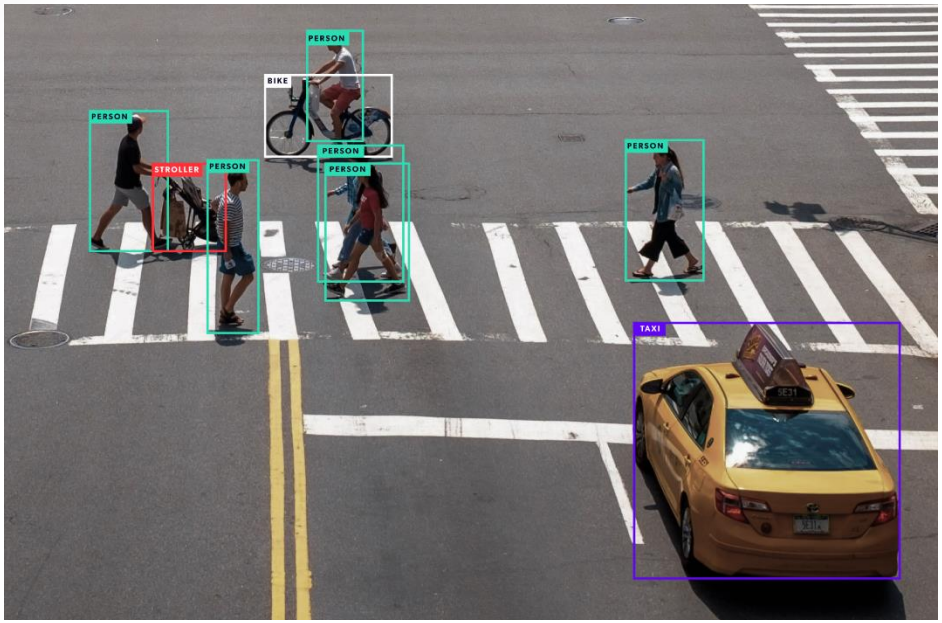


Source: https://tryolabs.com/guides/introductory-guide-computer-vision

CMPT 120

Computers that can **see**!

Lecture 24 – Computer Vision

Python – File IO and Lists of Lists!

Last Lectures

- Solved our **factorial** problem **recursively**
- **Visualized** the execution of our solution
- Solved the **palindrome** problem **recursively**
- Closed our unit on **Computer Graphics** looking into drawing **trees iteratively** as well as **recursively** using **turtle**
- Solved the problem of drawing a tree for every season using **recursion** and **dictionary**

- Midterm

Review: Challenge

- How would you modify the re-cursive functior **drawTree** to ge this tree?

```
# Define a recurisve function that draws a tree
def drawTree(aTurtle, aLevel, aBranchLength):
    '''Draws a tree recursively where
       "aTurtle" is the turtle drawing the tree,
       "aLevel" is the number of levels of branches and
       "aBranchLength" the length of branch to draw.
    '''

    # Base Case:
    # If we are at the leaf level (level == 0), draw a green leaf!
    if aLevel == 0:
        aTurtle.color("green")
        aTurtle.stamp()
        aTurtle.color("brown")

    else:
        # Recursive Case:
        # Draw a branch
        aTurtle.forward(aBranchLength)

        # Turn left and draw a smaller tree
        aTurtle.left(40)
        drawTree(aTurtle, aLevel - 1, aBranchLength/1.5)

        # Challenge: Come back straight and draw a smaller tree
        aTurtle.right(40)
        drawTree(aTurtle, aLevel - 1, aBranchLength/1.5)

        # Turn right and draw a smaller tree
        aTurtle.right(40)
        drawTree(aTurtle, aLevel - 1, aBranchLength/1.5)

        # Go back
        aTurtle.left(40)
        aTurtle.back(aBranchLength)

    return
```

Today's Menu

- Start investigating another field in Computing Science: **Computer Vision!**
- Let's have another look at **Lists**!
- Introduce **file IO** -> **I**nput/**O**utput
  - **text file**

Computers that can see!

Images and videos

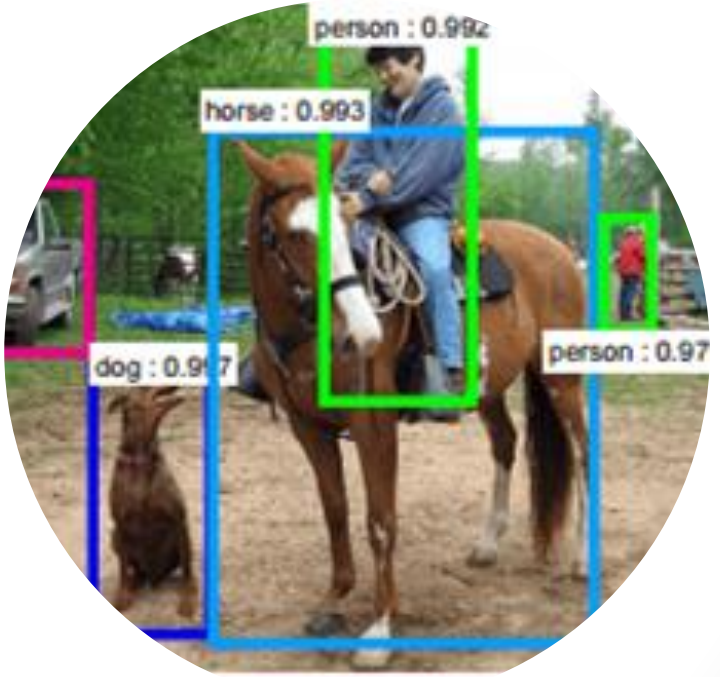
- We can make **computers** understand our world **visually**

Images and videos

Examples of such applications

- Facebook's automatic photo captions for the blind
- Facial recognition on smartphones
- Medical imaging
- Gesture recognition

- In CS at SFU, we have [Visual Computing M.Sc.]



Computer Vision

*"**Computer vision** is the field of computing science that focuses on replicating parts of the complexity of the human vision system and enabling computers to identify and process objects in images and videos in the same way that humans do."*

Thank you **towardsdatascience.com**

- Let's check it out!
  - https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e

Computer Vision

1. APPLICATIONS
   - In this unit, we'll learn about the computing science field of **computer vision**, that allows computers to **process images** and **understand** them

2. ALGORITHMS
   - We'll learn about nested loops, etc…

3. Python
   - We'll learn about file input/output (I/O), lists of lists, etc…

List of Lists

Step 1 - Problem Statement

- What if a problem statement asks us to manipulate a **matrix**?

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Step 2 - Design

- In our solution (Python program), we could use a **list of lists** to represent a **matrix**
  - The Python code representing the above data would look like:

```
myMatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Accessing elements in a list of lists (Indexing a list of lists)

```
myMatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

>>>myMatrix[0]
>>>[1, 2, 3]

>>>myMatrix[2][0]
>>>7

>>>myMatrix[0][3]
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    myMatrix[0][3]
IndexError: list index out of range
```

This is the Python syntax to access a list inside a list.

This is the Python syntax to access an element of a list inside a list.

Modifying elements in a list of lists

```
myMatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
stdInfo = ["Mike", [112, "B Street"], "YVR"]
```

Slicing a list of lists

```
myMatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
stdGrades = [[3,4.5,4],[3.5,5,"-"],[4,4,3]]
```

Review: List methods (functions)

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Source: https://www.w3schools.com/python/python_ref_list.asp

Another use for List of Lists

- Using a list of lists, we can represent a **grid** or a **maze** in our Python program:

```
W W W W W W W W W W W W W W W W W W W W W W W W
W W W 0 0 0 0 W W W 0 0 0 0 W W W W 0 0 0 0 W W
W W 0 0 0 0 0 W W 0 0 0 0 0 0 W W 0 0 0 0 0 0 W
W 0 0 0 0 0 0 W 0 0 0 0 0 0 0 0 0 0 0 W W W 0 0 W
E 0 0 0 0 0 0 W 0 0 0 W 0 0 0 0 W 0 0 W W W 0 0 W
W 0 0 0 0 0 0 W 0 0 0 W 0 0 0 0 W 0 0 W W W 0 0 W
W W W 0 0 0 0 W W W W 0 0 0 0 W W 0 0 W W W 0 0 W
W W W 0 0 0 0 W W W 0 0 0 0 W W W 0 0 W W W 0 0 W
W W W 0 0 0 0 W W 0 0 0 0 W W W W 0 0 W W W 0 0 W
W W W 0 0 0 0 W 0 0 0 0 W W W W W 0 0 W W W 0 0 W
W W W 0 0 0 0 W 0 0 0 0 0 0 0 0 W 0 0 W W W 0 0 W
W W W 0 0 0 0 0 0 0 0 0 0 0 0 0 W 0 0 0 0 0 0 0 S
W W W 0 0 0 0 W 0 0 0 0 0 0 0 0 W W 0 0 0 0 0 W W
W W W W W W W W W W W W W W W W W W W W W W W W
```

How to create a grid – Take 1

```
# Set variables
row = 5
column = 3
symbol = " - "
grid = list()
# Create a grid
for aRow in range(row):
    listRow = list()
    for aColumn in range(column):
        listRow.insert(aColumn, symbol)
    grid.insert(aRow, listRow)
```

Nested **for** loop!

How to print a list of lists

Take 1:

```
print(grid)
```

Take 2:

```
for aRow in range(row):
    print(grid[aRow])
```

Take 3:

```
# Print the list using join() method
for aRow in range(len(grid)):
    print( '  '.join(grid[aRow]))
```

Text Files

- Contain characters (ASCII code)
- Can be created/edited using a text editor
- We first need to **open** a text file in our Python program in order to …
  - read its content => `open(filename, 'r')`
  - write into the file => `open(filename, 'w')`
- Then we read its content or write into the file
- Then we must **close** the text file once we are done with it => `fileVariable.close()`

`fileVariable` represents the file in our Python program

Let's give a go!

Step 1 - Problem Statement

- Read and print the content of **words_7_a.txt**

Let's try again!

Step 1 - Problem Statement

- Read and print the content of **grid_2.txt**

Image Processing

Combining Images

Let's get started!

Step 1 - Problem Statement

- Write a program that combines two images **kid-green.jpg** and **beach.jpg** to form one image in which the kid is jumping on the beach.

- Let's get started by downloading these two files from our course web site:

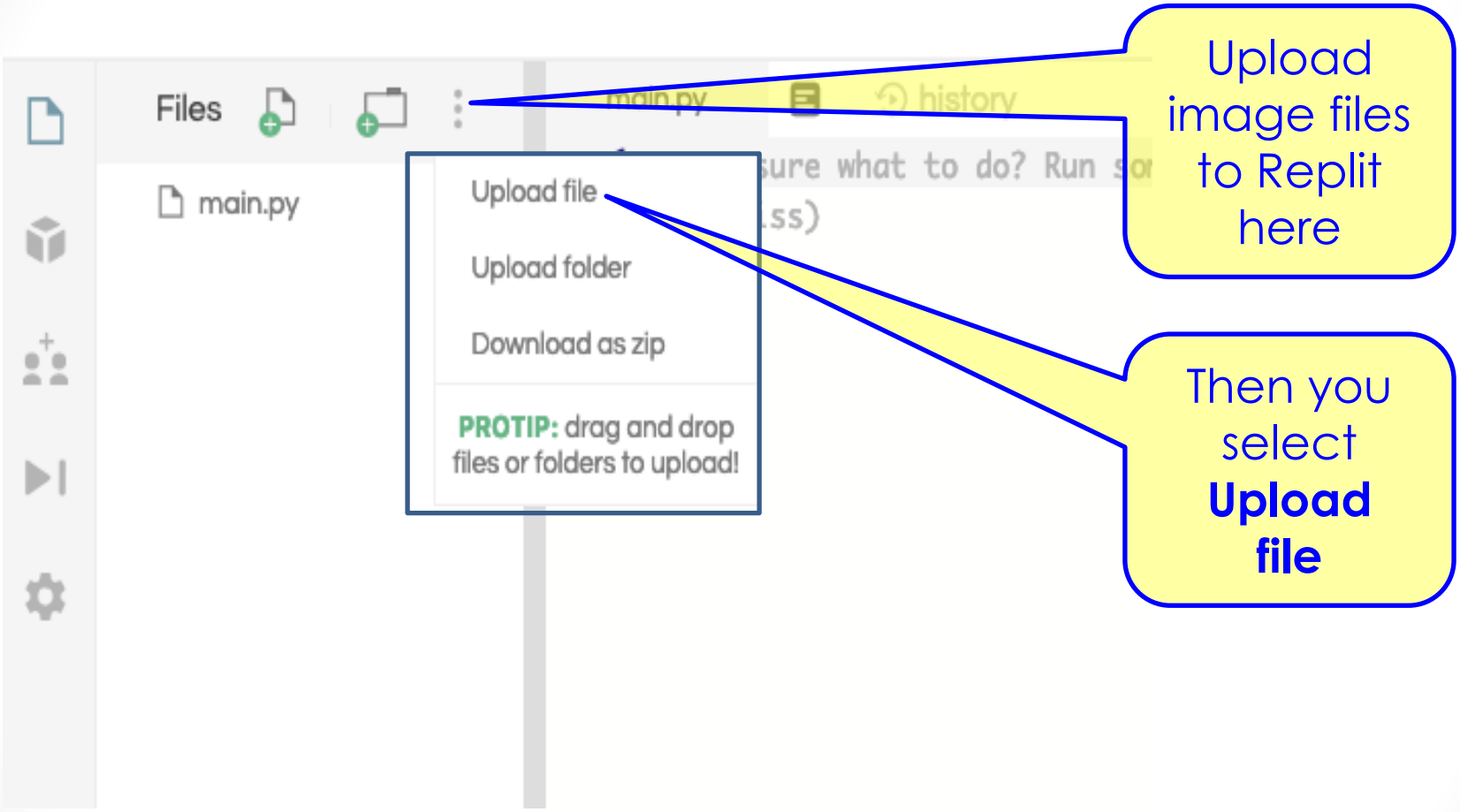
1. **kid-green.jpg**
2. **beach.jpg**

The resulting image:

replit!

- Over the next few weeks, we shall be using **replit** - https://replit.com
- Create yourself an account

Working with images on replit



Next lectures

- Solve our **image processing** problem
- And in the process, have a closer look at
  - Using the **PIL** module
  - How to open an **image file**
  - How to read the content of an **image file**
    - **Pixels**
    - **RGB** colour scheme
  - How to merge two images together
    - Nested for loops
    - Tuples