Paola Celio | Pietro Corsi | Sergio Lins

PYTHON BASICS

Dipartimento di Matematica e Fisica - Dipartimento di Scienze

- Last time we've introduced the image manipulation and today we'll do more example about this focusing on histograms.
- We're also working on files manipulation.
- But let me spend few minutes in something that is a little test for us:
    i. Working on image file,
    ii. Working on lists

Examples

- First example on images

  We may suppose here to be able to do a practical application so let me ask I've an image called test.png in a directory recognize a part of the image (rectangle cause we're at Python basics course) and put that part inside a rectangle then hide other part of the image and save the evidenced one in a new file.

```
import matplotlib.pyplot as plt

from matplotlib.patches import Rectangle

from PIL import Image


im = Image.open('./test.png')

# Display the image

plt.imshow(im)

plt.show()


# Get the current reference

ax = plt.gca()


# Create a Rectangle patch
```

```
rect = Rectangle((50,100),40,30,linewidth=1,edgecolor='r',facecolor='none')


# Add the patch to the Axes

ax.add_patch(rect)

plt.imshow(im)

#plt.show()


plt.savefig('./test_modified.png')

plt.savefig('./test_modified.pdf')
```

- Second example on function and lists[1]

```
# a. Given a list of numbers, return a list where

# all adjacent == elements have been reduced to a single element,

# so [1, 2, 2, 3] returns [1, 2, 3]. You may create a new list or

# modify the passed in list.

def remove_adjacent(nums):

  # +++your code here+++

  return
```

[1] https://developers.google.com/edu/python/exercises/basic

```
# b. Given two lists sorted in increasing order, create and return a merged
# list of all the elements in sorted order. You may modify the passed in lists.
# Ideally, the solution should work in "linear" time, making a single
# pass of both lists.
def linear_merge(list1, list2):
  # +++your code here+++
  return

# Note: the solution above is kind of cute, but unfortunately list.pop(0)
# is not constant time with the standard python list implementation, so
# the above is not strictly linear time.
# An alternate approach uses pop(-1) to remove the endmost elements
# from each list, building a solution list which is backwards.
# Then use reversed() to put the result back in the correct order. That
# solution works in linear time, but is more ugly.

# Simple provided test() function used in main() to print
# what each function returns vs. what it's supposed to return.
```

```
def test(got, expected):
  if got == expected:
        prefix = ' OK '
  else:
        prefix = '  X '
  print '%s got: %s expected: %s' % (prefix, repr(got), repr(expected))


# Calls the above functions with interesting inputs.
def main():
  print 'remove_adjacent'
  test(remove_adjacent([1, 2, 2, 3]), [1, 2, 3])
  test(remove_adjacent([2, 2, 3, 3, 3]), [2, 3])
  test(remove_adjacent([]), [])

  print
  print 'linear_merge'
  test(linear_merge(['aa', 'xx', 'zz'], ['bb', 'cc']),
        ['aa', 'bb', 'cc', 'xx', 'zz'])
  test(linear_merge(['aa', 'xx'], ['bb', 'cc', 'zz']),
```

```
        ['aa', 'bb', 'cc', 'xx', 'zz'])
  test(linear_merge(['aa', 'aa'], ['aa', 'bb', 'bb']),
        ['aa', 'aa', 'aa', 'bb', 'bb'])


if __name__ == '__main__':
  main()
```

For people that is experiencing troubles with matplotlib

pip uninstall matplotlib

python -m pip install --upgrade pip

pip install matplotlib