Nicholas Bowman, Sonja Johnson-Yu, Kylie Jue
CS106AP

Section #3
July 9, 2019

Section #3: Solutions

1. Image Puzzles

```
def fix1(filename):
    """
    Fix the fix1 puzzle image by multiplying its pixels red,
    green, and blue values by 6, 2, and 4 respectively.
    """
    image = SimpleImage(filename)
    for pixel in image:
        pixel.red = pixel.red * 6
        pixel.green = pixel.green * 2
        pixel.blue = pixel.blue  * 4
    return image

def fix2(filename):
    """
    Fix the fix2 puzzle image by swapping its red and green
    pixel values.
    """
    image = SimpleImage(filename)
    for pixel in image:
        temp = pixel.red
        pixel.red = pixel.green
        pixel.green = temp
    return image

def fix3(filename):
    """
    Fix the fix3 puzzle image by swapping its blue and
    green pixel values.
    """
    image = SimpleImage(filename)
    for pixel in image:
        temp = pixel.red
        pixel.red = pixel.blue
        pixel.blue = temp
        pixel.green = pixel.green * 4
    return image
```

2. Image Range Loop Problems

- **`double_left`**

```
def double_left(filename):
    """
    Take the left half of the image and copy it on
    top of the right half. Assume the image width is even.
    A classic make-a-drawing situation.
    """
    image = SimpleImage(filename)
    # style idea: the math is hard enough to follow,
    # it's helpful to introduce a variable like this to name
    # a common value we need (identify mid_x on drawing too).
    mid_x = image.width // 2
    for y in range(image.height):
        for x in range(mid_x):
            pixel = image.get_pixel(x, y)
            pixel_right = image.get_pixel(mid_x + x, y)
            pixel_right.red = pixel.red
            pixel_right.green = pixel.green
            pixel_right.blue = pixel.blue
    return image
```

- **`double_left_up`**

```
def double_left_up(filename):
    """
    Copy the left half on top of the right half as
    before, except the right half should be upside down.
    """
    image = SimpleImage(filename)
    mid_x = image.width // 2
    for y in range(image.height):
        for x in range(mid_x):
            pixel = image.get_pixel(x, y)
            # the key line:
            pixel_right = image.get_pixel(mid_x + x,
                                          image.height - y - 1)
            pixel_right.red = pixel.red
            pixel_right.green = pixel.green
            pixel_right.blue = pixel.blue
    return image
```

- **`copies_2`**

```
def copies_2(filename):
    """
    Create a new `out` image twice as wide as the original.
    Place 2 copies of the original in `out` side-by-side,
    and return it.
    """
    image = SimpleImage(filename)
    # Creates a new white image with nothing in it
    out = SimpleImage.blank(width=image.width * 2, height=image.height)
    mid_x = image.width
    for y in range(image.height):
        for x in range(image.width):
            pixel = image.get_pixel(x, y)
            # left copy
            pixel_left = out.get_pixel(x, y)
            pixel_left.red = pixel.red
            pixel_left.green = pixel.green
            pixel_left.blue = pixel.blue

            # right copy
            pixel_right = out.get_pixel(mid_x + x, y)
            pixel_right.red = pixel.red
            pixel_right.green = pixel.green
            pixel_right.blue = pixel.blue
    return out
```

- **`squeeze_width`**

```
def squeeze_width(filename, n):
    """
    A sort of funhouse-mirror effect.
    Create a new `out` image the same height as the original,
    but with width divided by int n.
    Copy the original image to `out`, squeezing it horizontally
    by a factor of n, e.g. for n of 4, copy from the original
    at x 0, 4, 8, ... and so on.
    n=1 should copy the image without squeezing.
    """
    image = SimpleImage(filename)
    out = SimpleImage.blank(width=image.width // n, height=image.height)
    # strategy: iterate x,y over the out pixels, compute
    # scaled * n pixel in original image.
    for y in range(out.height):
        for x in range(out.width):
            pixel_out = out.get_pixel(x, y)
            pixel = image.get_pixel(x * n, y)

            pixel_out.red = pixel.red
            pixel_out.green = pixel.green
            pixel_out.blue = pixel.blue
    return out
```

3. Photoshop Image Algorithms

- Rotate left

```
def rotate_image_left(image):
    """
    Rotates an image counter-clockwise 90 degrees.
    """
    out = SimpleImage.blank(width=image.height, height=image.width)
    for y in range(image.height):
        for x in range(image.width):
            source_pixel = image.get_pixel(x, y)
            dest_pixel = out.get_pixel(y, image.width - 1 - x)
            dest_pixel.red = source_pixel.red
            dest_pixel.green = source_pixel.green
            dest_pixel.blue = source_pixel.blue

    return out
```

- Flip vertical

```
def flip_vertical(image):
    """
    Flips an image across its horizontal axis.
    """
    out = SimpleImage.blank(width=image.width, height=image.height)
    for y in range(image.height):
        for x in range(image.width):
            source_pixel = image.get_pixel(x, y)
            dest_pixel = out.get_pixel(x, image.height - 1 - y)
            dest_pixel.red = source_pixel.red
            dest_pixel.green = source_pixel.green
            dest_pixel.blue = source_pixel.blue

    return out
```

4. Range and List Problems

```
def negative(n):
    """
    Returns a list that counts down from `n` to negative `n`,
    where `n` is located at index 0.
    """
    nums = []
    for i in range(n, (-1 * n) - 1, -1):
        nums.append(i)
    return nums
    # Note: unary - has the highest precedence, so the 2nd param
    # can be written as -n - 1, but the parentheses look fine too.

def mirror(lst):
    """
    Given a list `lst`, returns a "mirrored list" that
    includes the original elements in order, followed by
    the elements in reverse order.
    """
    mirrored_list = []
    mirrored_list.extend(l)
    for i in range(len(lst) - 1, -1, -1):
        current_elem = lst[i]
        mirrored_list.append(current_elem)
    return mirrored_list

def unique_numbers(nums):
    """
    Given a list `nums` of integers, returns the number
    of unique numbers.
    """
    if len(nums) == 0:
        return 0
    curr_num = nums[0]
    num_unique = 1
    for i in range(1, len(nums)):
        if curr_num != nums[i]:
            num_unique += 1
            curr_num = nums[i]
    return num_unique
```

5. Calculator

```
"""
Calculator.py
-------------
Implements the calculator program, as specified in the
section 3 handout.
"""

import sys # so we have access to argv
```

```
def exp(base, power):
    """
    Returns `base` multiplied by itself `power` times.
    """
    total = 1
    for i in range(power):
        total *= base
    return total

def square(base):
    """
    Returns `base` times `base`.
    """
    return exp(base, 2)

def add(args):
    """
    Given a list of string representations of integers,
    returns the sum of the integers.
    """
    total = 0
    for i in range(len(args) - 2):
        total += int(args[i + 2])
    return total

def main(args):
    """
    This program checks for the operation specified by the
    user and then calls the appropriate function to perform
    the mathematical operation.
    """
    operation = args[1]     # args[0] is calculator.py

    if operation == '-exp':
        base = int(args[2]) # convert the argument to an int
        power = int(args[3])
        result = exp(base, power)
        print(result)

    if operation == '-square':
        base = int(args[2])
        result = square(base)
        print(result)

    if operation == '-add':
        result = add(args)
        print(results)
```

```
if __name__ == "__main__":
    main(sys.argv)
```

6. Word Count

```
def count_words(line):
    """
    Given a string `line` of words delimited by spaces,
    returns the number of words in `line`.
    """
    words_list = line.split()
    num_words = len(words_list)
    return num_words

def main():
    """
    This console program asks the user for a file and then
    prints the number of lines, words, and characters in the
    specified file.
    """
    file_name = input('File: ')
    lines = 0
    words = 0
    chars = 0
    with open(file_name, 'r') as f:
        for line in f:
            stripped_line = line.strip()
            lines += 1
            words += count_words(stripped_line)
            chars += len(stripped_line)
    print('Lines = ' + str(lines))
    print('Words = ' + str(words))
    print('Characters = ' + str(chars))

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