

## ECS 10

11/24

### Assignment

- Due Dec. 3. Make some progress on it this week before you go home.
- Problems installing PIL? Please email me if you figure out how to fix something, we will put tips on the assignment Web page.
- No lecture Wds!

### Pastel Program

- Define a function to fade a color.
- Use the function on every pixel.
- Write a for loop that will do something to every pixel in a square.

(0,0)	(1,0)	(2,0)	...	
(0,1)	(1,1)	(2,1)	...	
...				

### Double for Loop

```

1 for x in range(5):
2     for y in range(3):
3         print (x,y) # print the tuple

```

- Block under line 1 contains another for loop.
- For every value of x, line 2 generates every value of y.
- The for loop in line 2 starts over for every new value of x.

### Add Loop to Program

- We know width and height from the im.size variable.
- Use getpixel() and putpixel() methods to read and write the color of a pixel.
- Call the function we wrote to change the color.

### Repetition

```

red = color[0]
diff = 255 - red
red = red+3*diff/4
green = color[1]
diff = 255 - green
green = green + 3*diff/4

```

- Replace repetition with a function

## Why Replace Repetition

- Makes it easier to improve the program – if the code is just in one place, you only have to change it in one place.
- Fewer bugs!
- Neater, shorter, easier to understand.

## Functions calling Functions

- This happens all the time.
- In a large, complicated program, you have to go from one function to another to follow is going on.
- Comments describing what every function does are very important.
- Two lines, sometimes more, at the top of a function.

## Parameters and Arguments

```
def pastel(color):
    .....
    outColor = pastel(turkeyColor)
```

- Function **pastel** is called with the variable **turkeyColor** as argument.
- The parameter **color** is a different variable than **turkeyColor**, but it gets the same value.

## Local variables

- The variables inside a function are invisible to the main program.
- This is to prevent functions from having unintended side effects in the main program.
- So variables inside a function are called local variables.
- The parameters of the function are local variables.

## Example

```
def addTwo(y):
    y = y+2
    return y

x = 10
z = addTwo(x)
```

Local variable **y** is not defined in the main program.

## Global variables

- Variables in the main program can be seen inside a function. So they are called global variables.
- Functions cannot change global variables, just see them.
- This again is to prevent unintended side effects.

### Global variable version

```
def addTwo():
    y = x+2
    return y
```

```
x = 10
z = addTwo()
print 'x,z =', x,z
```

x is a global variable, so it can be seen inside the function.

The function can't change x, so it still needs local variable y.

### Two variables with same name

- Python will let you write a program with a local variable named x and also a global variable named x.

```
def addTwo(x):
    x = x+2
    return x
x = 10
z = addTwo(x)
```

A perfectly legal Python program, but a very bad idea!

### Try to avoid this!

- It can be really confusing.
- You need to know that it can be done because you will end up doing it by mistake.
- When you figure it out, fix it by changing the name of one of the variables.
- Happens when you re-use variable names, for example when you have lots of for loops using i as the variable:

```
for i in range(len(s)):
```

### Is it local or global?

- If it is a parameter of the function, it's local.
- If it appears on the left in an assignment in a function, it's local.
- If it appears on the left in an assignment in the main program, but only on the right in the function, it's global.
- If it appears on the left in **both** the main program and the function, you have two variables with the same name! Danger!