COMP120: Creative Computing: Tinkering

.

The end result is that a bunch of different colours, get set to a few colours
Beware of naive solutions with a large number of ‘if’ statements
[plain] [remember picture, overlay] [at=(current page.center)] [width=]field$_{n}ormal$;
[plain] [remember picture, overlay] [at=(current page.center)] [width=]field$_{s}epia$;
Sepia Tone
First, we’re calling greyScaleNew (the one with weights).
We then manipulate the red (increasing) and the blue (decreasing) channels to bring out more yellows and oranges.
It’s perfectly okay to have one function calling another.
Why are we doing the comparisons on the red? Why not? After greyscale conversion, all channels are the same!
The end result is that a bunch of different colours, get set to a few colours
Why these values? Trial-and-error: Tinker the values!
[fragile] Source Code: Sepia (1)
def sepiaTint(picture): Convert image to greyscale makeGreyscale(picture)
loop through picture to tint pixels for p in getPixels(picture): red = getRed(p) blue = getBlue(p)
tint shadows if (red ¡ 63): red = red*1.1 blue = blue*0.9 ...
Note: This source code excerpt will not work in PyGame.
[fragile] Source Code: Sepia (2)
... tint midtones if (red ¿ 62 and red ¡ 192): red = red*1.15 blue = blue*0.85
tint highlights if (red ¿ 191): red = red*1.08 if (red ¿ 255): red = 255
blue = blue*0.93
set the new color values setBlue(p, blue) setRed(p, red)
Note: This source code excerpt will not work in PyGame.
Activity #6: Sepia Tone
In pairs:

Setup a basic project in PyGame
Refer to the following documentation
Refactor the function: `sepiaTint(picture) to use constants rather than literals`
Tinker with the values of the constants to test your solution
Then, post your solution on Slack