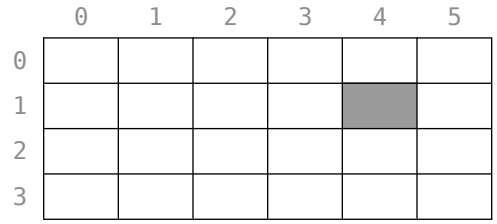
CSCI 145 Quiz 9 practice questions on images and objects **ANSWER KEY**

Images (Lecture 17)

1. A grayscale image `img` is 6 pixels wide and 4 pixels tall. The grid shows its pixels. `img[0][0]` is the top-left pixel.



a) How many pixels does `img` have? **24**

b) What expression gives the shaded pixel? **`img[4][1]`**

c) What expression gives the bottom-right pixel? **`img[5][3]`**

2. `img` is a grayscale image. Each line below runs on its own. Write the value of `img[0][0]` after the line runs. If the line causes an error, write `error`.

`img[0][0] = 400` **255**

`img[0][0] = -30` **0**

`img[0][0] = 99.5` **error**

`img[0][0] = int(99.5)` **99**

3. This code makes a new grayscale image and sets every pixel. Write each pixel's value in the grid.

```
img = middimage.new(3, 2, "L")
for x in range(img.width):
    for y in range(img.height):
        img[x][y] = 10 * x + y
```

	0	1	2
0	**0**	**10**	**20**
1	**1**	**11**	**21**

4. These lines run in the Shell, in order. Then write the value Python gives for each expression.

```
>>> img = middimage.new(2, 2)
>>> img[1][0] = (200, 120, 40)
>>> r, g, b = img[1][0]
```

`g` **120**

`img[1][0][2]` **40**

`img[0][0]` **(0, 0, 0)**

5. `img` is a color image. The line `new_img = mystery(img)` runs.

```
def mystery(img):
    result = middimage.new(img.width, img.height)
    for x in range(img.width):
        for y in range(img.height):
            r, g, b = img[x][y]
            result[x][y] = (r, g, 0)
    return result
```

a) If `img[0][0]` is `(90, 200, 30)`, what is `new_img[0][0]`?

(90, 200, 0)

b) Does `mystery` change `img`? (yes or no)

no

6. Write a function `count_dark(img)`. `img` is a grayscale image. The function returns the number of pixels whose value is less than 100. If there are none, it returns 0. Do not include a docstring.

For example, for the image `tiny` shown here, `count_dark(tiny)` returns 3.

	0	1	2
0	250	99	40
1	100	0	180

```
def count_dark(img):
    count = 0
    for x in range(img.width):
        for y in range(img.height):
            if img[x][y] < 100:
                count = count + 1
    return count
```

7. Questions 7 to 9 use the class `Trail` below.

```
class Trail:
    def __init__(self, name, miles):
        self.name = name
        self.miles = miles

    def __str__(self):
        return self.name + ": " + str(self.miles) + " miles"

    def extend(self, extra):
        self.miles = self.miles + extra

    def shorter_than(self, other):
        return self.miles < other.miles
```

a) What are the names of the two attributes of a `Trail`?

name, miles

b) Which method runs when a new `Trail` is made?

__init__

c) Which method changes one of a `Trail`'s attributes?

extend

8. These two lines run. Then write the value Python gives for each expression. Put quotes around a string.

```
t1 = Trail("Ridge", 6)
t2 = Trail("Pond Loop", 2)
```

`t2.name` **'Pond Loop'**

`t2.shorter_than(t1)` **True**

`str(t2)` **'Pond Loop: 2 miles'**

9. Draw the variables and objects after this code runs, as on the Lecture 18 worksheet: an arrow from each variable to its object, and each object as a box with its attributes inside. Then write what the code prints.

```
a = Trail("Ridge", 6)
b = Trail("Ridge", 6)
c = b
c.extend(4)
a.name = "Gap"
print(a)
print(b)
print(c)
```

Your picture

```
a ---> Trail
         name   'Gap'
         miles  6

b ---> Trail
c --->   name   'Ridge'
         miles  10

(b and c name the same Trail)
```

Output

```
Gap: 6 miles
Ridge: 10 miles
Ridge: 10 miles
```

10. Write a class `Team` with these three methods. Do not include docstrings.

- `__init__(self, name, wins)` stores its parameters as the attributes `name` and `wins`.
- `__str__(self)` returns a string such as `"Otters: 7 wins"`.
- `better_than(self, other)` takes another `Team`. If this team has more wins than `other`, it returns `True`. Otherwise, it returns `False`.

For example:

```
>>> otters = Team("Otters", 7)
>>> owls = Team("Owls", 4)
>>> print(otters)
Otters: 7 wins
>>> otters.better_than(owls)
True
>>> owls.better_than(otters)
False
```

```
class Team:
    def __init__(self, name, wins):
        self.name = name
        self.wins = wins

    def __str__(self):
        return self.name + ": " + str(self.wins) + "
wins"

    def better_than(self, other):
        return self.wins > other.wins
```