Graphics and Computer Vision

Question 1

What does this code draw?

```
import turtle
alex = turtle.Turtle()

def mystery(side, listcols):
    for i in range(5):
        nextcol = listcols[i]
        alex.color(nextcol)
        alex.forward(side)
        alex.left(360/5)

mystery(100,["blue","red","green","orange","black"])
```

A. **Blue hexagon**
B. **5-coloured hexagon**
C. **5-coloured lines in row**
D. **Nothing**

Question 2

What does this code print?

```
def greet(greeting, name, repeat):
    print(greeting + ", " + name + "!"*repeat)


for i in range(3):
    greet("Hello", "Leo", i)
```

```
A. Hello, Leo!0
   Hello, Leo!1
   Hello, Leo!2
B. Hello, Leo!1
   Hello, Leo!2
C. Hello, Leo
   Hello, Leo!
   Hello, Leo!!
D. Hello, Leo!
   Hello, Leo!!
```

Question 3

What will this code do?

```
import turtle
import random
turtle.colormode(255)
alex = turtle.Turtle()

def myst(side,bluecomponent):
    for i in range(5):
        nextcol = (200,0,bluecomponent)
        alex.color(nextcol)
        alex.forward(side)
        alex.left(360/5)

myst(100, random.randint(0, 255))
```

A. **Draw five blue hexagons**

B. **Draw five random-colour hexagons**

C. **Draw a red-hue hexagon**

D. **Draw a hexagon with 5 random red-hues**

Question 4 (3-part)

Is there more red or more blue in each of these color representations (RGB)?

Expressed in hexadecimal numbers	FF0102	#1
Expressed in binary numbers	00000000 11111100 00111000	#2
Expressed as a three tuple with decimal (usual) numbers	(100,50,25)	#3

```
A. Yes, Red > Blue
B. No, Red < Blue
C. They are the same.
D. Don't know
```

Question 5

What does this code do? How could you fill in the blank to make various colors?

```
import turtle
import random
turtle.colormode(255)
billy = turtle.Turtle()

red_component = ___________________
green_component = _________________
blue_component = ___________________

color = (red_component, green_component, blue_component)
print("The amount of red is", color[0])
billy.color(color)
billy.forward(50)
```

```
A. random.randint(0,1)
B. random.randint(red, green, blue)
C. random.randint(0, 255)
D. random.randint(0, 256)
```

Functions

Functions

What are they useful for? Answers from last class:

- Functions allow the developer to **break a program** into **smaller**, easier to create **blocks**.
- Write less repetitive code.
  - **DRY: Don’t Repeat Yourself!**
- **Groups code** into meaningful blocks.

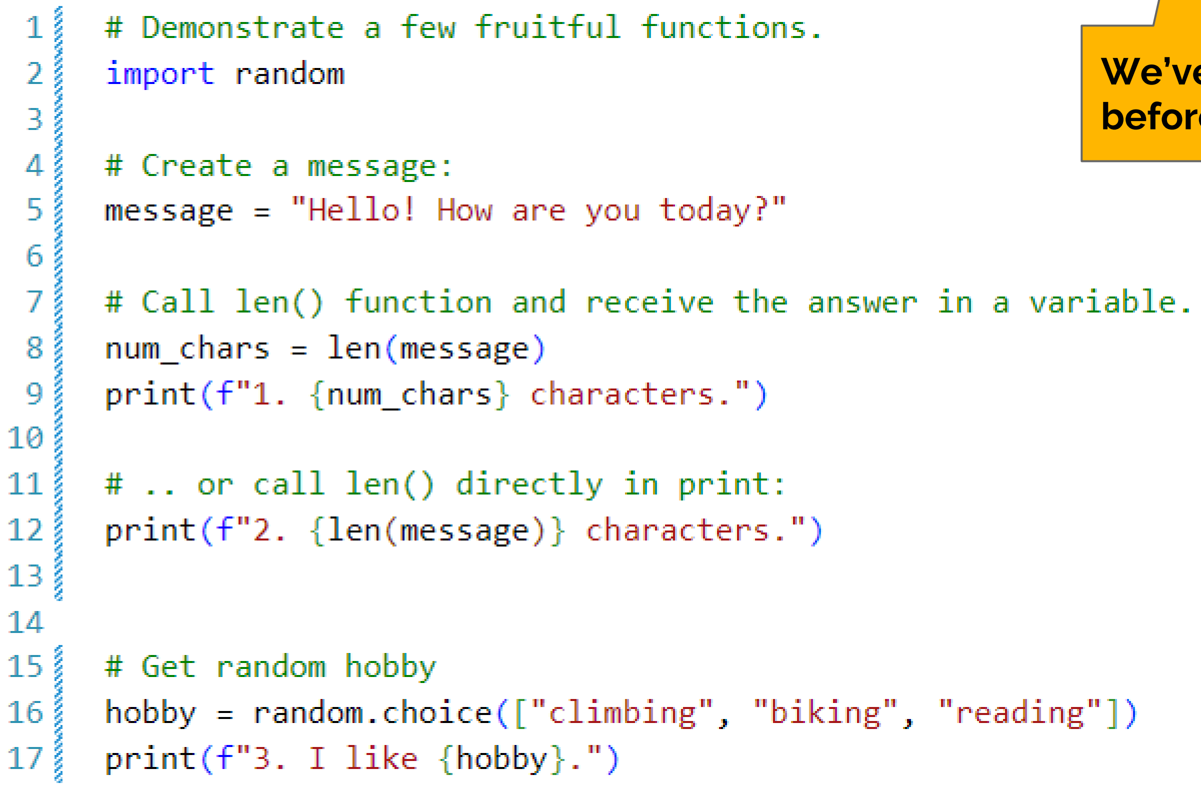
Functions are NOT meant to...

- Use fewer variables
  (extra variables do not hurt in general)
- Make the program run faster
  (but they might, and they do **save programming time**!)
- Make code harder to understand
  (but they are an extra thing to learn)

Fruitful Functions

Find the 3 fruitful function calls

We've used them before!



```python
# Demonstrate a few fruitful functions.
import random

# Create a message:
message = "Hello! How are you today?"

# Call len() function and receive the answer in a variable.
num_chars = len(message)
print(f"1. {num_chars} characters.")

# .. or call len() directly in print:
print(f"2. {len(message)} characters.")



# Get random hobby
hobby = random.choice(["climbing", "biking", "reading"])
print(f"3. I like {hobby}.")
```

Creating fruitful functions

```python
# Define a fruitful function
def power(x, y):
    result = x ** y
    return result

# Use the fruitful function
#     - Must receive (or catch) the answer in a variable
calc = power(2, 3)
print(f"power(2,3) returned {calc}")

# Calling without receiving answer does nothing
power(42, 3)
```

To make fruitful: Just use the keyword **return** in your function

Fruitful functions

Notice the difference between these pieces of code:

```python
def power1(a,b):
    result = a**b
    print(result)
```

```python
def power2(a,b):
    result = a**b
    return result
```

Fruitful functions

Return does two things:

- **Return the value**
- **Exit the function** immediately, even if it’s inside a loop!

Remember to receive what is returned!

Fruitful functions

```python
'''
    Input: list of integers
    Returns: maximum number in list
'''
def listMax(numList):
    maxNum = numList[0]     # Initialize maxNum
    for val in numList:     # Go through list
        if val > maxNum:
            maxNum = val    # Update maxNum
    return maxNum

print(listMax([1,5,3,5,2]))
```

However, noramlly we don't re-write built-in Python functions.

Here, just call max():
`print( max([1,5,3,5,2]) )`

Fun creating functions

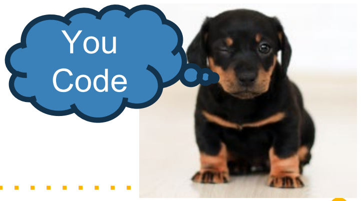
- Write a fruitful function which is passed a string and a character. It must count how many times the character appears in a string.

Fun creating functions (2)

- Write a function name factorial which is given an integer n and computes n!
  - Ex: 4! = 4 * 3 * 2 * 1

Fun creating functions (3)

- Write a function which is given a string (*s*) and a number (*n*) which prints the string *s* to the screen *n* times.



More function fun

- Write a well-named function which is passed a list of strings and returns true if all strings in the list are already lower case. Returns false otherwise.
    - Hint: Use a return statement before end of function.

More function fun (2)

- Write a function named only_even_digits which is passed an integer.
  - Return true if it contains only even digits; false otherwise.