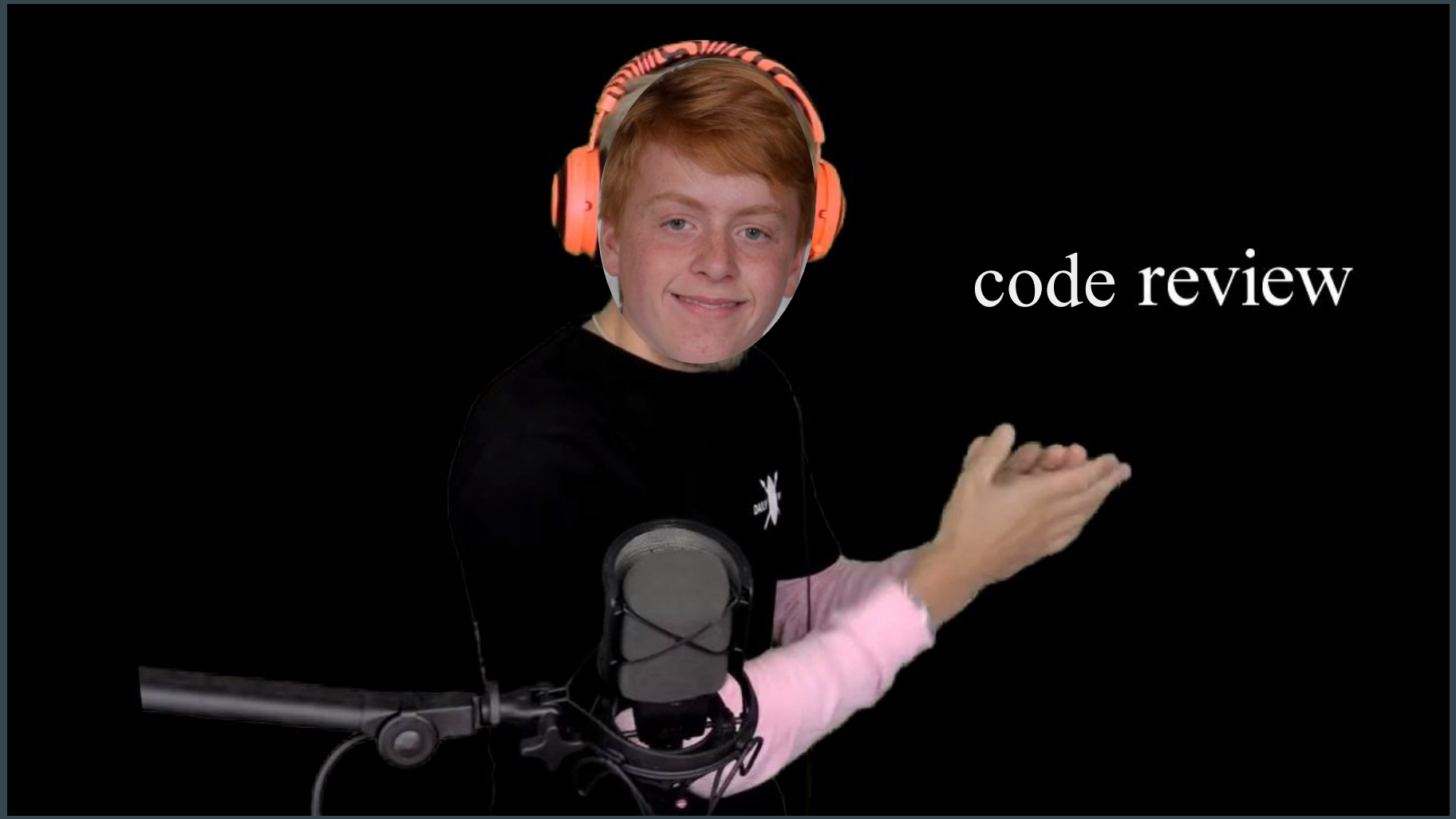
Introduction to Functions

Hawken Coding Club
Educational Meeting No. 3

Slido

Join our Slido link

- Go to slido.com
- Enter code #G694
- Submit code anonymously throughout the meeting
- No pressure


code review

Review

Review Exercise

In the world of Dlrow, the words *kcuf*, *nmad*, and *tihs* are considered bad words.
The government of Dlrow requests your assistance in curing their swearing epidemic.

Write a script that repeatedly prompts the user for a word.
Stop if the user enters a period, question mark, or exclamation point.
Print out all words that are not one of the bad words in Dlrow.
You can assume that the user enters only one word at a time.
Bonus: yell at the user if they use a bad word / congratulate them if they don't..

Review Exercise - Solution

```python
# Define our constants as lists
BAD_WORDS = ["kcuf", "nmad", "tihs"]
STOP_PUNCTUATION = [".", "?", "!"]

# Hold user input for later
acceptable_input = []

# An easy way to implement "loop until..." loop
while True:

  word = input()

  # Stop accepting input if input is stop punctuation
  if word in STOP_PUNCTUATION:
    break # Stop the infinite loop!

  if word not in BAD_WORDS:
    # BONUS: print("Thank you for not swearing.")
    acceptable_input.append(word)
  else:
    # BONUS: print("This is a family-friendly zone.")
    pass # don't do anything

# Print all acceptable words
for acceptable_word in acceptable_input:
  print(acceptable_word)
# Alternative: print("\n".join(acceptable_input))
```

https://colab.research.google.com/drive/1XmMIIbur7mlZJRuG26n5z8BGC65o8j_o?usp=sharing

`def`ining Functions

How To Make Your Own Functions

```python
def funct():
  print("Functions are cool")
funct()
```

```
Functions are cool
```

```python
myCoolNumber = 7
def coolNumberDouble():
  print(myCoolNumber*2)
coolNumberDouble()
```

```
14
```

A function **definition** defines what the function does when it is ran.

Nothing actually takes place until the function is called!

Parameters and Arguments

Parameters are the inputs that a particular function accepts.

```
def name(first, last):
  print("You're name is", first, last)
name("John", "Shin")
```

```
You're name is John Shin
```

Parameters

Arguments

The function is called with **arguments**, which are the values that are passed into a function.

A function call with different arguments usually produces different results.

Returning

Functions can not only be used to print out something, they can also **`return`** a value back to the code that called it.

```
def double(x):
  return 2*x
y = 1
print(y)
print(double(y))
```

```
1
2
```

```
def funct(name, num):
  if (name == "John"):
    print("Shin")
    return
  if (num == 1):
    print(1)
funct("John", 1)
```

```
Shin
```

Blank return statements can be used to stop a function from going any further

If a function ends without a `return` statement, it automatically `returns` `None`.

slido

Submit a function called diameter_to_area. The function will receive an int parameter named diameter. The function should then return the area of a circle with that diameter. (Hint a = pi*r^2)

ⓘ Start presenting to display the poll results on this slide.

Scope

Variables defined inside a function are **local** to that function. They have **local scope**.

They cannot be accessed from outside of the function unless explicitly `return`ed.

```
def funct():
  x = 10
#Theres an error because x is only stored inside of the function
print(x)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-dd49dffba692> in <module>()
      2   x = 10
      3 #Theres an error because x is only stored inside of the function
----> 4 print(x)
      5
      6

NameError: name 'x' is not defined
```

Common Functions

A lot of basic functionality is provided for you in the form of **built-in functions**.

This saves time, because you don't have to re-write that code yourself.

Common Functions: What you already know

print()

input()

type()

Common Functions: Sort

```
lst = [4,2,6,1,8,9,12,5]
def sort(lst):
  i=0
  while i < len(lst):
    temp = lst[i]
    j = i-1
    while j >= 0 and lst[j] > temp:
      lst[j+1] = lst[j]
      j -= 1
    lst[j+1] = temp
    i+=1
  return lst

sort(lst)
```

```
[1, 2, 4, 5, 6, 8, 9, 12]
```

Common Functions: Max

```python
def findMax(myList):
  highest = myList[0]
  x = 0
  while x < len(myList):
    if myList[x] > highest:
      highest = myList[x]
    x += 1
  return highest
findMax([5, 18, 2, 19, 25, 46, 4])
```

```
46
```

Common Functions: I wanna learn more!

A full list of built-in functions are found here:

https://**docs.python.org**/3/library/functions.html

There are even more functions available in separate **modules**.

However, modules are a more advanced topic, so we will discuss them later.

Recursion

Recursion is the use of a function that calls itself.

In other words, the function is defined in terms of itself.

```python
def add1(num):
  if num <= 10:
    print(num)
    add1(num+1)
add1(0)
```

```
0
1
2
3
4
5
6
7
8
9
10
```

slido

Submit a function called factorial. It should take num as a parameter and return num factorial. Use recursion to make this process easier.factorial(n) = n * n-1 * n-2 * … * 1

ⓘ Start presenting to display the poll results on this slide.

Optional Parameters

Optional parameters have a default value. The code calling the function can simply leave out an argument to use the default value or supply an argument to override the default value.

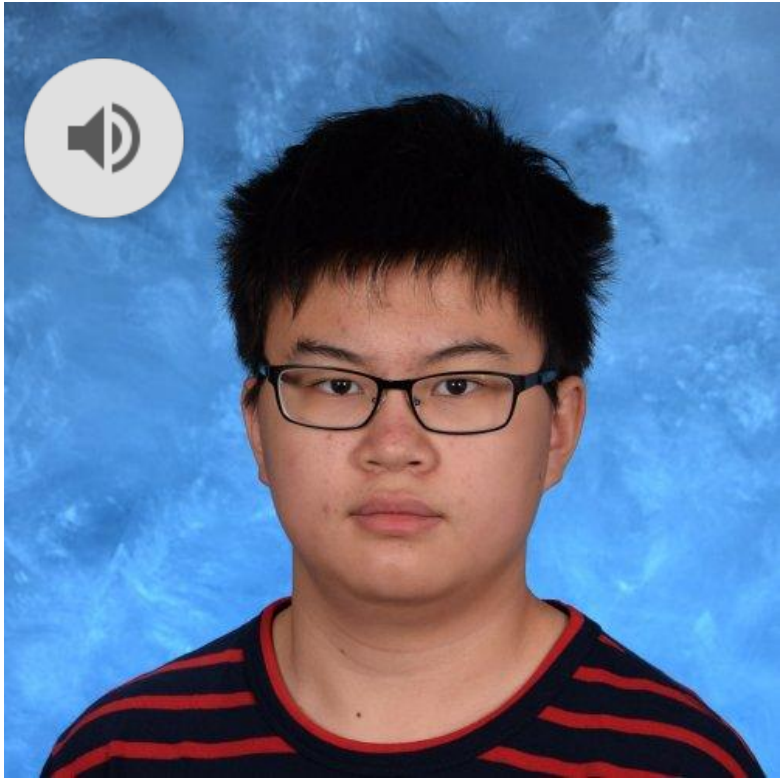
```
def newStudent(firstName, school, lastName=None, grade=None):
  if lastName == None and grade == None:
    print("Hello " + firstName + " Welcome to " + school + " school!")
  elif grade == None:
    print("Hello " + firstName + " " + lastName + " Welcome to " + school + " school!")
  else:
    print("Hello " + firstName + " Welcome to " + school + " school! Welcome to " + grade + " grade!")
newStudent("John", "Hawken")
newStudent("John", "Hawken", "Shin")
newStudent("John", "Hawken", "Shin", "Tenth")
```

```
Hello John Welcome to Hawken school!
Hello John Shin Welcome to Hawken school!
Hello John Welcome to Hawken school! Welcome to Tenth grade!
```

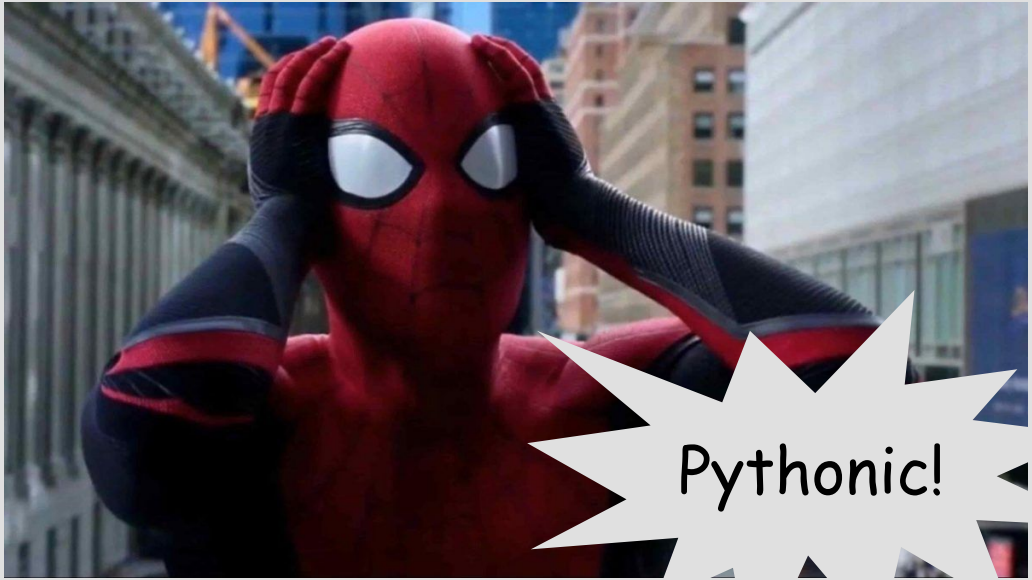
I was really sad I couldn't actually join the meeting today, but I was determined to make myself heard anyway. Using the power of Python programming, I have transformed myself into . . .

Virtual Raymond

>>> click on the icon to listen <<<



Best Practices



Best Practices

“With great power comes great responsibility.” - French Revolutionaries / Spider-Man

Best practices are principles to keep code efficient, organized, maintainable, etc.

They are recommendations, not rules, but don’t break them without good reason.

Remember, code is:

1. primarily a set of instructions for the computer -> good syntax
2. also a form of communication between programmers -> good style

Best Practices: DRY

DRY = Don't Repeat Yourself

Using functions helps keep your code DRY by not repeating similar pieces of code.

[pretend there's an example here]

DRY code is easier to read and understand both for you and other programmers.

Maintainability - changes are made in just one location, which prevents errors.

Best Practices: KISS

KISS = Keep It Simple, Stupid

Don't define functions just because you can (or want to).

Additional complexity is a cost - you should have some benefit in mind (like DRY).

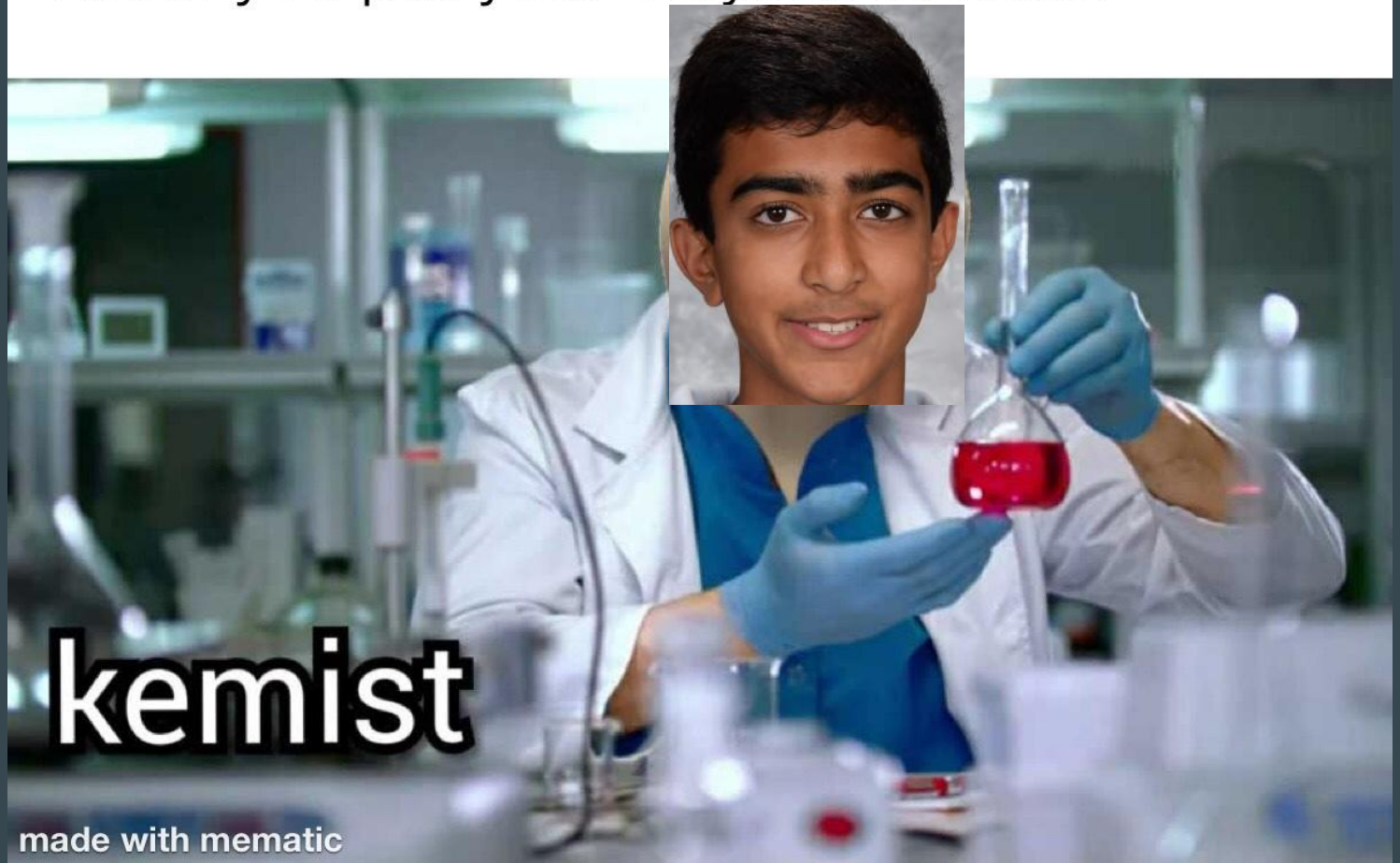
Code with a lot of function calls requires the reader to jump from place to place.

Make your own judgement as to what option is simpler to understand overall.

Thanks for making it to the end.

Here is your reward.

When you put your crayons in water



If you ever need help, use a powerful search engine!