CoGrammar

Functions

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Software Engineering Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(FBV: Mutual Respect.)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes. You can submit these questions here: **Open Class Questions**

Software Engineering Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Lecture Objectives

1. Recall functions and define what a function in programming is.
2. Expand on functions by exploring Higher Order Functions.
3. Apply the above knowledge to improve data management in programs.

What is a Function?

- ★ Reusable and Organised block of code.
- ★ Sometimes called a ‘method’.
- ★ Similar to functions in maths - f(x) takes input x and produces some output.
- ★ Useful for abstraction.
- ★ For example, “make a cup of tea” vs “boil water, add tea bag, add sugar, add milk, stir”.

Calling Functions

- Functions with one required positional input:
  - `my_function(input1)`
- Functions with two required positional inputs:
  - `my_function(input1, input2)`
- Functions with one required positional input and one optional keyword input:
  - `my_function(input1, keyword_arg=input2)`

Functions in Python

- ★ Python comes bundled with in-built functions.
- ★ Examples:
  - print**(string)** - prints string to console.
  - input**(string)** - prints string to console, then reads input from the console as string.
  - len**(array)** - returns the length of an array.
  - int**(data)** - converts the value to an integer.

...but wait! There's more!

- ★ The list of functions that you can use in Python doesn't just stop with what is built in.
- ★ Using Pip (python package manager), you can install various packages containing modules.
- ★ Note: Some packages are already installed by default in Python, such as the Math package.
- ★ These modules can be imported into your script using an import statement.

Importing Libraries

★ Let's take a look at the maths module. Let's say that you want to use round(), which rounds a number off. There are two ways to access this:

a.
```
import math
my_result = math.round(my_num, 2)
```

b.
```
from math import round
my_result = round(my_num, 2)
```

Defining our own Functions

- Uses the def keyword (for define):
  - `def add_one(x): # function called add_one`
  - `    y = x + 1`
  - `    return y`
- Important keywords:
  - def - tells Python you are defining a function
  - return - if your function returns a value, then use this keyword to return it.

Why Functions?

- ★ **Reusable code** - Sometimes you need to do the same task over and over again.
- ★ **Error checking/validation** - Makes this easier, as you can define all rules in one place.
- ★ **Divide code up into manageable chunks** - Makes code easier to understand.
- ★ **More rapid application development** - The same functionality doesn't need to be defined again.
- ★ **Easier maintenance** - Code only needs to be changed in one place.

What is a Higher order Function (HOF)

- ★ A function that takes other functions as arguments (or returns a function) is called a higher order function
- ★ Higher-order functions operate by accepting a function as an argument, altering it, and then returning the altered function. More modular and reusable code can be produced as a result.

Built-in HOF

★ A few useful higher-order functions are map(), filter(), and reduce(). map() and filter() are built-in functions(reduce() is contained in functools() module).

```
numbers = [1,2,3,4,5]
square = map(lambda i: i**2 , numbers)
print (square)

print (list(square))
#Output: [1, 4, 9, 16, 25]
```

Best Practices

★ **Descriptive Function Names:**
Instead of foo() or bar(), let's name our functions so that anyone reading our code knows exactly what's going on.

```
def calculate_area(radius):
    # Code for calculating area
```

★ **Single Responsibility Principle:**
One function, one responsibility. Break down your code into smaller, focused functions.

```
# Step 1:
def fetch_data(url):
    print("Code to fetch data from the URL.")

# Step 2:
def process_data(data):
    print("Code to process this data.")
```

★ **Avoiding Global Variables:**

Global variables can be tricky. Stick to local scope and keep your functions pure.

```
count = 0
def increment_count():
    global count
    count += 1
    return count

#   Better:
def increment_count(count):
    return count + 1
```

★ **Docstrings:**
A form of documenting your functions. Think of these as user manuals for each function.

```
def calculate_area(radius):
    """
    Calculate the area of a circle.

    :param radius: The radius of the circle.
    :type radius: float
    :return: The area of the circle.
    :rtype: float
    """
    # Code for calculating area
```

Poll:

Assessment

Wrapping Up

Libraries

We can use Python's built in functions or import functions from different modules to use within our code.

User defined functions

We can create our own functions with their own behaviour to use within our programs.

Higher Order Functions

Higher Order Functions allow us to receive functions as input arguments or return functions as output.

CoGrammar

Questions around Functions

CoGrammar

Thank you for joining