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**

What is a Function?

- Reusable and Organised block of code.
- Similar to functions in maths - $f(x)$ takes input x and produces some output. Eg. $f(x) = x + 1$
- Useful for abstraction
  - For example, “make a cup of tea” vs “boil water, add tea bag, add sugar, add milk, stir”.

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.

Functions in Python

- Python comes bundled with built-in functions.
- Examples:
  - **print(string)** - prints string to console.
    Eg. print("Hello World")
  - **input(string)** - prints string to console, then reads input as string. Eg. num = input("Please enter a number")
  - **len(list)** - finds the length of a list.
    Eg. print(len([1,2,4])) # Prints 3
  - **int(data)** - converts the value to an integer.
    Eg. num = int("5")

PEP doc: https://docs.python.org/3/library/functions.html

Importing Modules

- Let's take a look at the maths module. Let's say that you want to use pow(), which returns x(the base) raised to the power of y (exponent) the value of a number to the power of something. There are two ways to access this:
  - ```
    import math
    my_result = math.pow(x,y)
    ```
  - ```
    from math import pow
    my_result =pow(x,y)
    ```

https://docs.python.org/3/library/math.html

Self-defined functions

- Reusable and Organised block of code.
- The general syntax of a function:

def tells Python you are defining a function

```
def my_function(parameter1, parameter2):
    #statement
    local_variable = parameter1 * parameter2
    #expression
    return local_variable
```

Parameters can take **required positional input** or **optional keyword** input (default values)

Parameters - The defined input of a function.

return - if your function returns a value, then use this keyword to return it.

Calling Functions

- Declare a variable to store the return value
- Give arguments to the parameters of the function

```
answer = my_function(1, 9)
```

Arguments - The values passed to parameters.

- To display the output of the function you need to call print on the variable.

```
# Print the output of the function for the 'answer' instance
print(answer)
```

Scope

- Where is a variable accessible in Python? Generally, whenever code is executed, variables become accessible across the entire script.
- Functions are different, however. Variables declared within functions are not accessible outside the function.
  - This avoids variable names being overwritten.

```
def multiply(x,y):
    product = x * y
    return product

answer1 = multiply(2,3)

print(f"{x} times {y} is {answer1}")
```

```
    print(f"{x} times {y} is {answer1}")
NameError: name 'x' is not defined
```

CoGrammar

Questions around

CoGrammar

Thank you for joining