Hyperiondev

Recap on Functions & Function Scope

Objectives

1. Recap on:
    a. What is a function?
    b. The general syntax of a function
    c. How to call a function
    d. How do we assign default values
2. Creating self-defined functions
    a. User input examples
    b. Using functions to simplify code

Lecture – Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- 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.
- You can also submit questions here: http://hyperiondev.com/sbc4-se-questions
- For all non-academic questions, please submit a query: www.hyperiondev.com/support
- Report a safeguarding incident: http://hyperiondev.com/safeguardreporting
- We would love your feedback on lectures: https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0g/

Github Repository - Lecture Examples

https://github.com/HyperionDevBootcamps/C4_SE_lecture_examples

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 Function?

- 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)

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

Parameters - The defined input of a function.

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)
```

Default Values

- Remember **optional keyword** arguments? These are made with default values.
- def multiply(num1, num2 = 5):
- This can be called with multiply(10), for example.
- The default value can be overwritten with multiply(10, num2=6).

```
def multiply(num1,num2 = 5):
    sum = num1 * num2
    return sum

answer1 = multiply(10)
answer2 = multiply(10,num2 = 6)

print(answer1)  #prints 50
print(answer2)  #prints 60
```

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}")
```

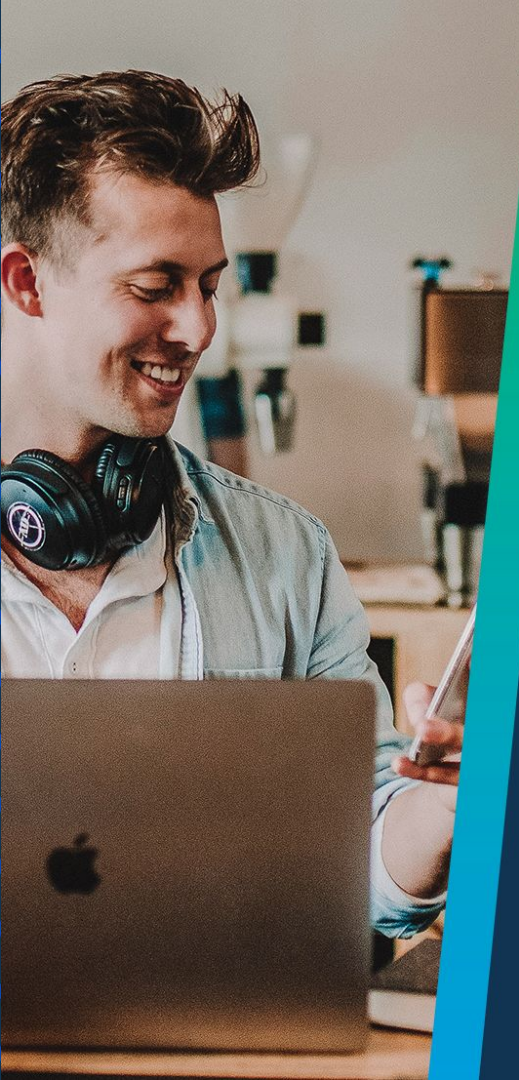
```
    print(f"{x} times {y} is {answer1}")
NameError: name 'x' is not defined
```

Let’s have a look at some more examples in VS code :)

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any



Hyperiondev

Thank you for joining us