Week 2 – variables and expressions - functions

Returning multiple values in a functionA screen shot of a computer program

Description automatically generated

It is possible for return statement to print nothing, if there is nothing to return

A computer screen shot of a computer code

Description automatically generated

Principle of code reuse

A close up of a text

Description automatically generated

**Study Guide: Functions**

This study guide provides a quick-reference summary of what you learned in this lesson and serves as a guide for the upcoming practice quiz.

In the Functions segment, you learned how to define and call functions, utilize a function’s parameters, and return data from a function. You also learned how to differentiate and convert between different data types utilizing variables. Plus, you learned a few best practices for writing reusable and readable code.

**Terms**

* **return value** - the value or variable returned as the end result of a function
* **parameter (argument)** -  a value passed into a function for use within the function
* **refactoring code** - a process to restructure code without changing functionality

**Knowledge**

* The purpose of the **def()** keywordis to define a new function.
* Best practices for writing code that is readable and reusable:
  + **Create a reusable function** - Replace duplicate code with one reusable function to make the code easier to read and repurpose.
  + **Refactor code** - Update code so that it is self-documenting and the intent of the code is clear.
  + **Add comments** - Adding comments is part of creating self-documenting code. Using comments allows you to leave notes to yourself and/or other programmers to make the purpose of the code clear.

**Coding skills**

**Skill Group 1**

* Use a function that accepts multiple parameters
* Return a result value

1

2

3

4

5

6

7

8

9

10

11

12

13

14

# This function calculates the number of days in a variable number of

# years, months, and days. These variables are provided by the user and

# are passed to the function through the function’s parameters.

def find\_total\_days(years, months, days):

# Assign a variable to hold the calculations for the number of days in

# a year (years\*365) plus the number of days in a month (months\*30) plus

# the number of days provided through the "days" parameter variable.

    my\_days = (years\*365) + (months\*30) + days

# Use the "return" keyword to send the result of the "my\_days"

# calculation to the function call.

    return my\_days

# Function call with user provided parameter values.

print(find\_total\_days(2,5,23))





RunReset

903

**Skill Group 2**

* Use a function to return the result of a measurement conversion
* Use arithmetic operators to perform a calculation
* Combine text with a function call within a print() statement
* Convert the return value from a float data type to a string for the print() function
* Call the function and perform a calculation on the return value within a print() statement

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

# This function converts fluid ounces to milliliters and returns the

# result of the conversion.

def convert\_volume(fluid\_ounce):

# Calculate value of the "ml" variable using the parameter variable

# "fluid\_ounce". There are approximately 29.5 milliliters in 1 fluid

# ounce.

    ml = fluid\_ounce \* 29.5

# Return the result of the calculation.

    return ml

# Call the conversion from within the print() function using 2 fluid

# ounces. Convert the return value from a float to a string.

print("The volume in millimeters is " + str(convert\_volume(2)))

# Call the function again and double the 2 fluid ounces from within

# the print function.

print("The volume in millimeters is " + str(convert\_volume(2)\*2))

# Alternative calculation:

# print("The volume in millimeters is " + str(convert\_volume(4))

Logical operators

And, or, not

dfds