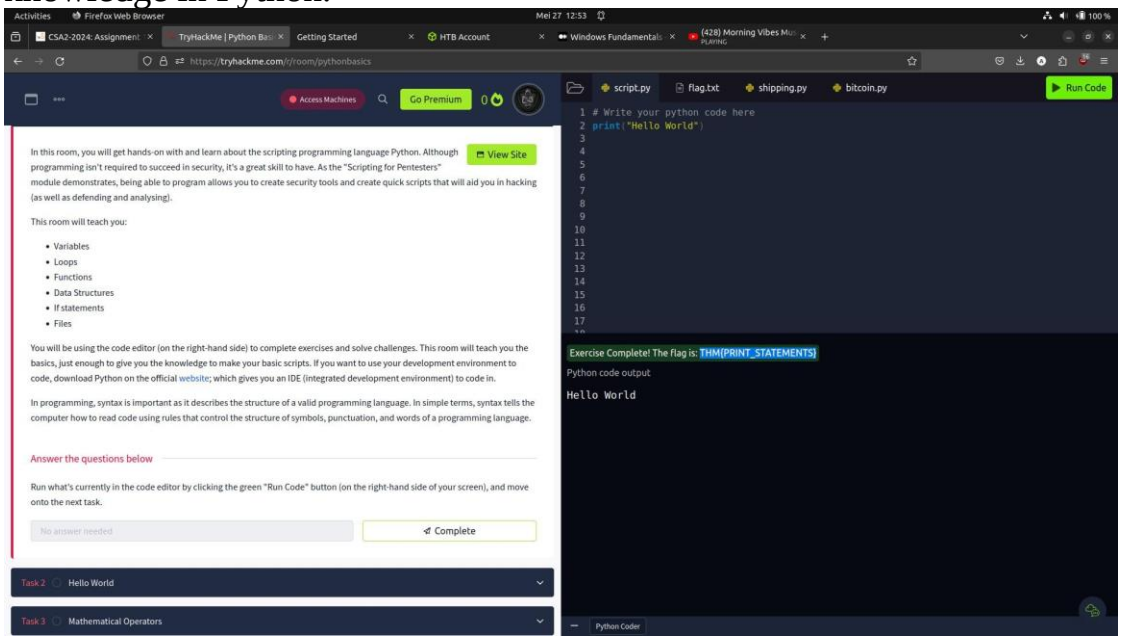
John-mbithi-mutave

Python Basics

The Python Basics module on TryHackMe introduces participants to Python programming through a web-based code editor. It emphasizes the importance of programming skills in cybersecurity, enabling the creation of security tools and quick scripts. Topics covered include variables, loops, functions, data structures, if statements, and files. Participants can run code in the editor and proceed through exercises to build foundational knowledge in Python.



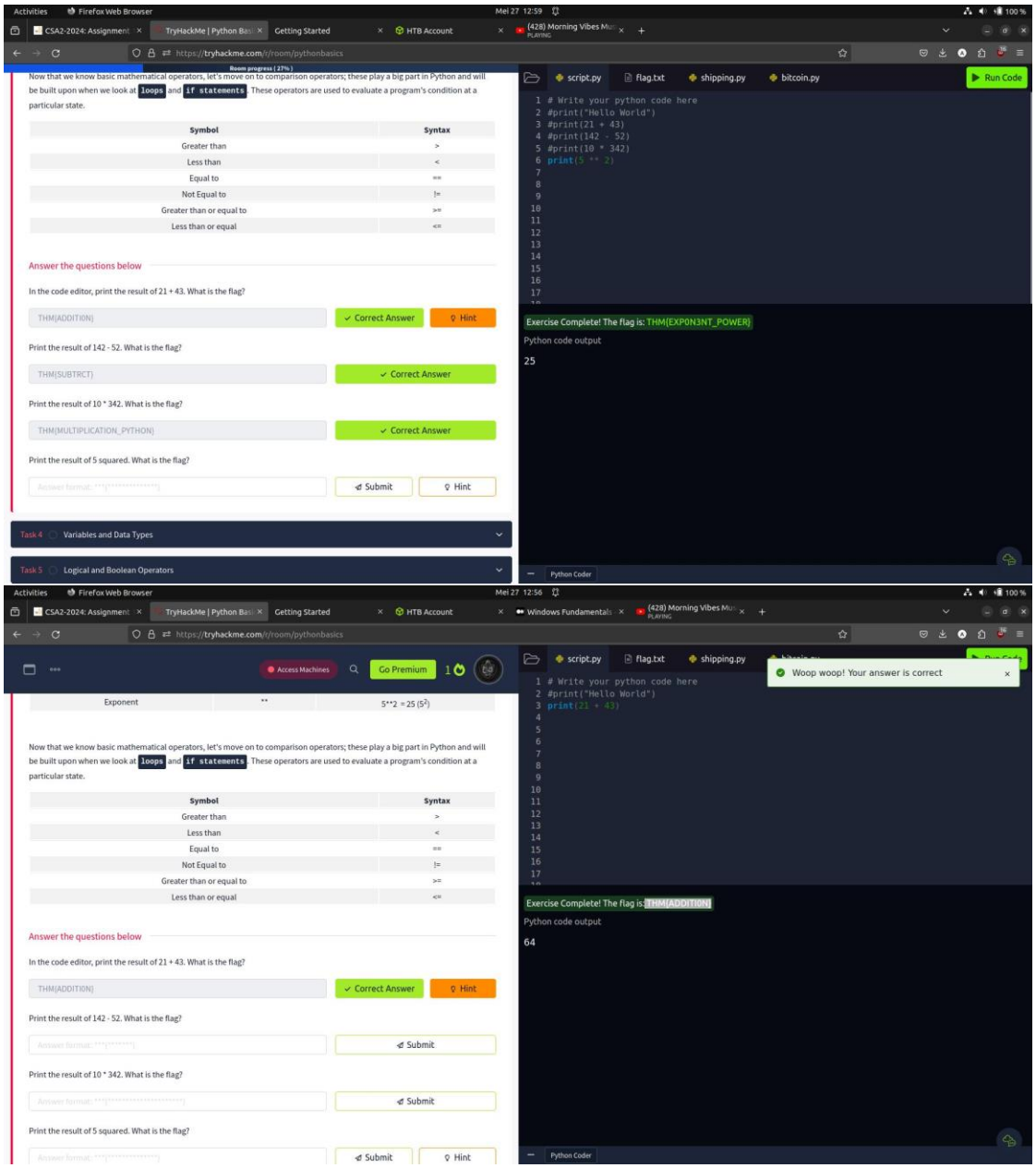
Mathematical Operators

Mathematical operators in Python function similarly to a calculator, allowing for addition, subtraction, multiplication, division, modulus, and exponentiation. These operations are represented by symbols such as "+", "-", "*", "/", "%", and "**" respectively. Understanding these operators is essential for performing mathematical computations in Python.

Comparison Operators

Comparison operators are pivotal in Python as they evaluate conditions within a program. These operators include symbols such as ">", "<", "==", "!=", ">=", and "<=", indicating greater than, less than, equal to, not equal to, greater than or equal to, and less than or equal to, respectively. Mastery of these operators is crucial for implementing logic in Python scripts, especially within loops and if statements.

John-mbithi-mutave



Variables and Data Types

Variables in Python allow for the storage and manipulation of data within a program. Each variable is assigned a name and can hold various types of data.

For instance:

python
food = "ice cream"
money = 2000

In this example, "food" stores the string "ice cream", while "money" stores the integer 2000.

Variables can be updated throughout the program. For example:

python
age = 30
age = age + 1print(age)

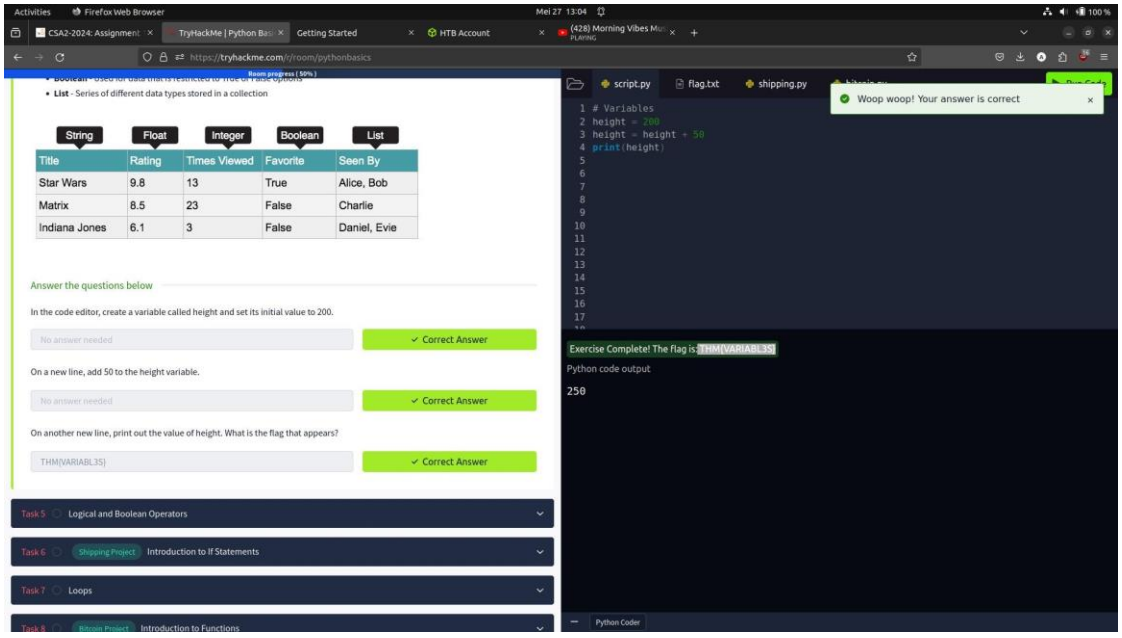
Here, the variable "age" is initially set to 30 and then incremented by 1, resulting in a final value of 31.

Data Types

Data types in Python specify the kind of data a variable can hold. Common data types include:

String: Used for text, consisting of characters or symbols.
Integer: Represents whole numbers.
Float: Represents numbers with decimal points or fractions.
Boolean: Represents data restricted to True or False values.
List: Represents a collection of different data types in a series.

Understanding data types is crucial for effective data manipulation and programming logic.



If Statements

If statements in Python enable programs to make decisions based on conditions. They allow the program to choose a specific action depending on whether a condition is true or false.

For example:

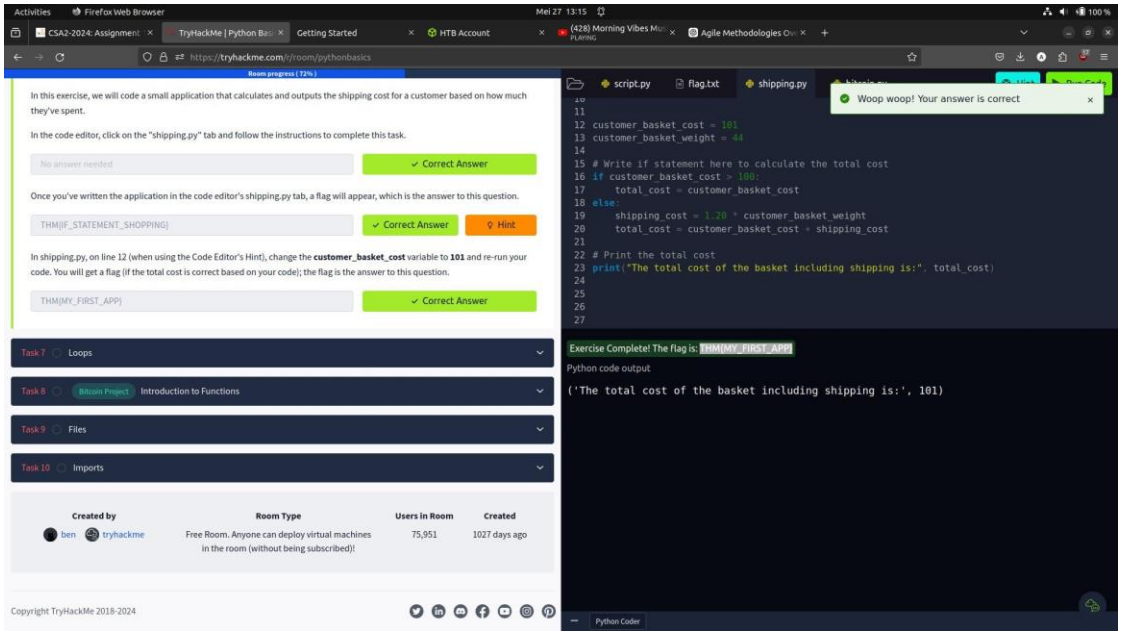
python
if age < 17:
print('You are NOT old enough to drive')else:

```
print('You are old enough to drive')
```

In this example, if the variable "age" is less than 17, the program outputs "You are NOT old enough to drive"; otherwise, it outputs "You are old enough to drive".

Key components of an if statement include:

The **if** keyword, indicating the start of the statement, followed by the condition.
The statement within the if block executes only if the condition is true.
Optionally, an **else** block can be included to handle conditions not met by the if statement.
The **colon** (:) marks the end of the if statement.
Indentation signifies the code block associated with the if statement.



Introduction to Functions

As programs become more intricate, code repetition becomes a concern. Functions address this issue by encapsulating blocks of code that can be reused at different points in a program.

A function could perform calculations like determining the distance between two map points or output formatted text based on conditions, effectively eliminating redundant code.

python
```
def sayHello(name):
    print("Hello " + name + "! Nice to meet you.")

sayHello("ben") # Output: Hello Ben! Nice to meet you
```

Key components of a function:

The **def** keyword initiates the function declaration, followed by a user-defined name (in this case, sayHello).
Parentheses () enclose parameters, allowing data to be passed into the function. Here, it's a name.
A colon : denotes the end of the function header.
Indentation signifies the function's code block, similar to if statements.

Functions can also return results:

```
python
def calcCost(item):
    if(item == "sweets"):
        return 3.99
    elif (item == "oranges"):
        return 1.99
    else:
        return 0.99

spent = 10
spent = spent + calcCost("sweets")print("You have spent:" + str(spent))
```

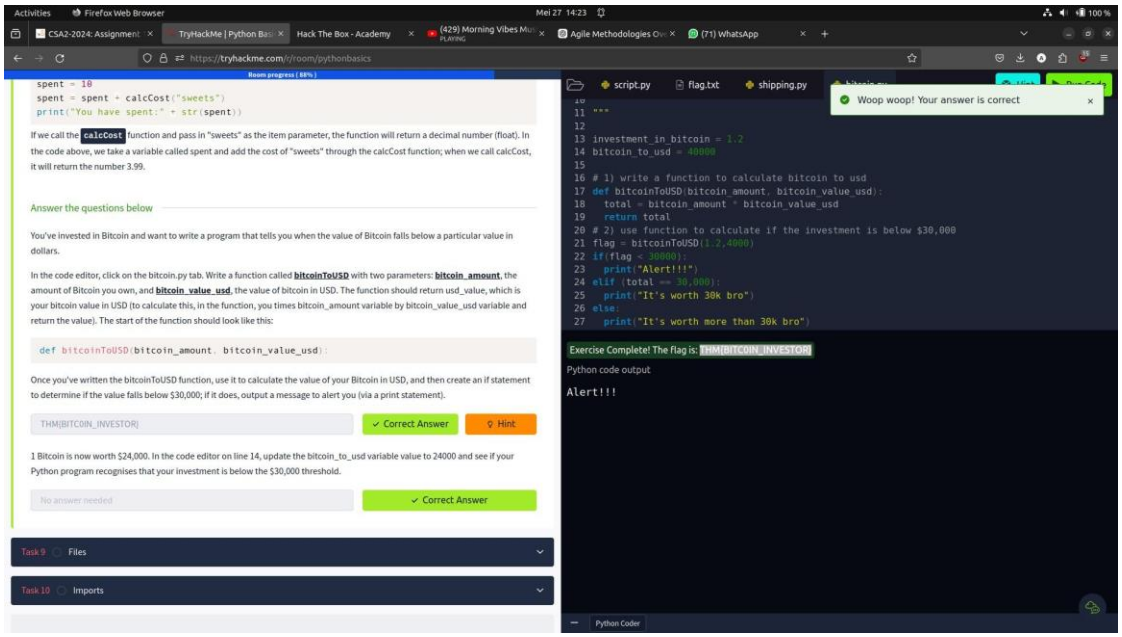
Here, the function **calcCost** returns the cost of an item. The returned value can be used in computations, as shown when adding the cost of "sweets" to the variable **spent**.

Implementing bitcoinToUSD Function

To create a function named **bitcoinToUSD**, define it with two parameters: **bitcoin_amount** (the amount of Bitcoin owned) and **bitcoin_value_usd** (the Bitcoin value in USD). The function should return **usd_value** (calculated by multiplying **bitcoin_amount** by **bitcoin_value_usd**). The function's start should resemble:

```
python
def bitcoinToUSD(bitcoin_amount, bitcoin_value_usd):
```

John-mbithi-mutave



Reading and Writing Files in Python

In Python, reading from and writing to files is a common task, particularly in cybersecurity where scripts may import or export data for analysis or storage.

To read from a file:

python
f = open("file_name", "r")print(f.read())

The **open()** function opens the file, with "r" indicating we're reading its contents. The **read()** method reads the file's contents.

You can also iterate over each line using **readlines()** method:

python
for line in f.readlines():
    print(line)

To write to a file:

python
f = open("demofile1.txt", "a") # Append to an existing file
f.write("The file will include more text..")
f.close()

f = open("demofile2.txt", "w") # Creating and writing to a new file
f.write("demofile2 file created, with this content in!")
f.close()

Use "a" to append to an existing file and "w" to create and write to a new file. After writing, close the file using **close()** to prevent further writing.

John-mbithi-mutave