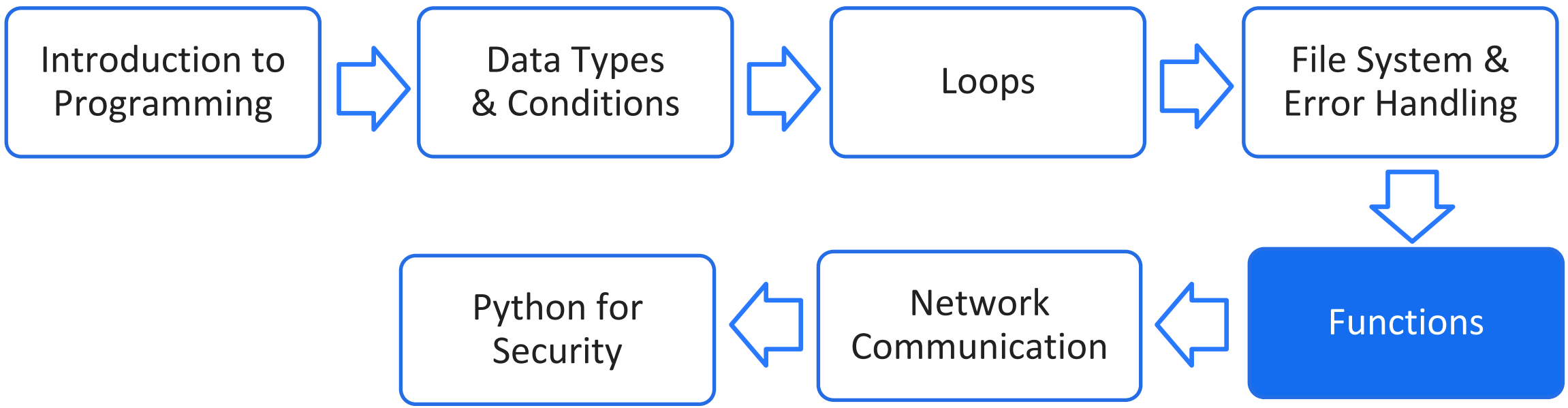
Cybersecurity Professional Program

Functions

Introduction to Python for Security

Course Path


Introduction to Programming
Data Types & Conditions
Loops
File System & Error Handling
Functions
Network Communication
Python for Security

Objectives

This lesson is an introduction to Python functions and code handling. The concepts of recursion and object-oriented programming are also introduced and demonstrated.

- Introduction to Functions
- Code Handling
- Recursion
- Object-Oriented Programming

Functions

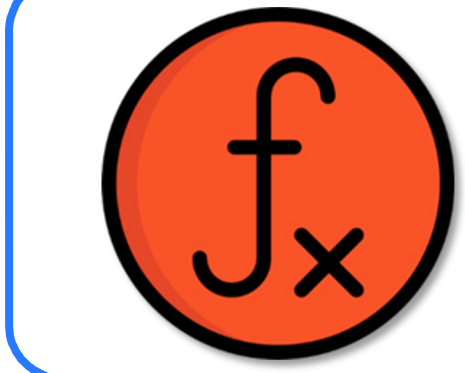
Introduction to Functions

Why Functions Are Needed

- Functions are necessary to prevent code from becoming difficult to understand.
- They save the time and effort of having to write the same functional code in different parts of a program.
- They also make code maintenance more efficient.

```
from random import randrange

dice = {1:0, 2:0, 3:0, 4:0, 5:0, 6:0}

dice_roll = int(input("Times to roll the dice: "))
print("Rolling the dice")

for i in range(dice_roll):
    dice_roll_res = randrange(1,7)
    dice[dice_roll_res] += 1

print(dice)
```

What Are Functions?



- Blocks of code with reusable logic
- Define functionality in the program.
- Separate programs into small, readable, and manageable sections.

> Functions are not parts of other elements in a script.
>
> Methods are functions that are parts of other elements in a script.

Function Declaration

A function must be defined first with the ***def*** keyword.

After ***def***, the name of the function is provided.

The last part is parentheses ***()*** for optional parameters and a colon ***:*** to define the code block.

```
def Print_Multiple(message, times):
    for i in range(times):
        print(message)

Print_Multiple("Hello", 2)
Print_Multiple("Hi!")
```

Invoking a Function

- A defined function will not yet be executed by Python.
- It must first be invoked before it can be used.
- Functions can also accept parameters.

```
def greetings():
        name = input("Enter your name: ")
        print("Greetings, {}!".format(name))

greetings()
```

Default Values

- Function parameters can include default values.
- If no value is passed, the default value will be used.
- If a value is passed, the default value will be overridden.

```
def print_my_mood(mood="happy"):
    print(mood)

print_my_mood()
print_my_mood("sad")


==========================================================
"C:\Users\johnd\PycharmProjects\Python\venv\Scripts\python.exe"
"C:/Users/johnd/PycharmProjects/Python/PY-05/Default Values.py"
happy
sad

Process finished with exit code 0
```

Returning Values

- Functions can pass data at the end of the execution.
- The ***return*** keyword is used to pass the values.
- ***return*** also terminates the execution of a function.

```
def Calculation(x, y):
        answer = x + y
        return answer

Result = Calculation(2, 4)
```

Lab PY-05-L1

Calculator

15–25 Min.

Mission

Create a function to handle calculation operations.

Steps

- Declare input variables.
- Define basic calculation functions.
- Define the main function.
- Write the function invocation logic.

Environment & Tools

- Windows/Linux
- Python 3
- PyCharm

Related Files

- Lab document

Returning Data

- A function can return multiple values, a list, or a dictionary.
- Each type of returned information is placed in separate parentheses.
- Commas are used to separate the values.

```
def return_multiple():
    first_name = "John"
    last_name = "Doe"
    return first_name, last_name

print(return_multiple())
```

```
==========================================================================
('John', 'Doe')

Process finished with exit code 0
```

Complete Function Example

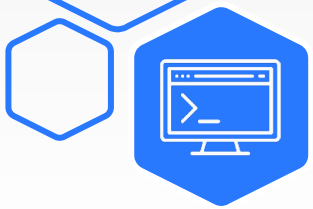
def defines a function.

The ***function name*** with parentheses will set the parameters.

return ends the execution and returns the result.

```
from random import randrange

def main():
    dice = {1:0, 2:0, 3:0, 4:0, 5:0, 6:0}
    dice_roll_res(dice)

def dice_rolling():
    rolls = int(input("Times to roll the dice: "))
    print("Rolling the dice")
    return rolls

def dice_roll_res(dice):
    for i in range(dice_rolling()):
        roll_res = randrange(1,7)
        dice[roll_res] += 1
    print(dice)

main()
```

None



None is not a value and refers to empty data.

It cannot be used in expressions but can be inserted in a variable or compared with them.

Every function by default returns a ***None*** value.

```
def first_function():
    return

def second_function():
    return None

def third_function():
    x = 5

print(first_function())
print(second_function())
print(third_function())
```

```
==========================================================================
None
None
None

Process finished with exit code 0
```

Lab PY-05-L2

Returning Lists

15–25 Min.

Mission

Create multiple functions that return different list types and check their types.

Steps

- Define functions.
- Return different data structures.
- Print outputs.

Environment & Tools

- Windows/Linux
- Python 3
- PyCharm

Related Files

- Lab document

Functions

Code Handling

Scope

Scope is what defines the visibility of a variable.

A variable in a function is accessible only in that function.

There are four types of scope: local, enclosed, global, and built-in.

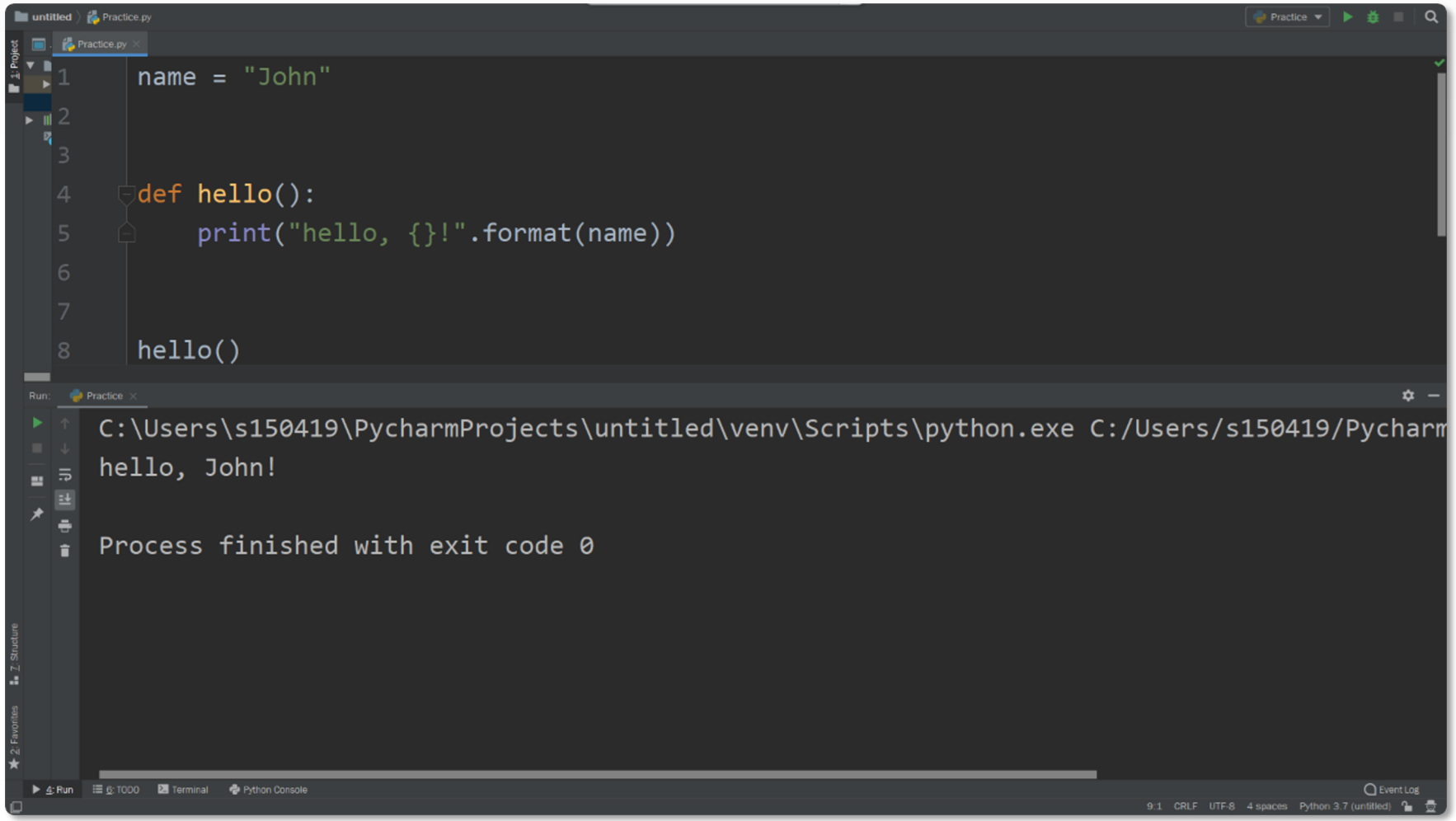
```
def func():
    txt = "hi"
    print(txt)

func()
print(txt)
==========================================================
hi
Traceback (most recent call last):
  File "C:/Users/asafr/PycharmProjects/Michigan Python/PY-05/Scope.py",
line 6, in <module>
    print(txt)
NameError: name 'txt' is not defined

Process finished with exit code 1
```

Code Handling

Global Scope

Global scope is the main scope of a program.

Variables defined in a global scope will be recognized in all other scopes as well.

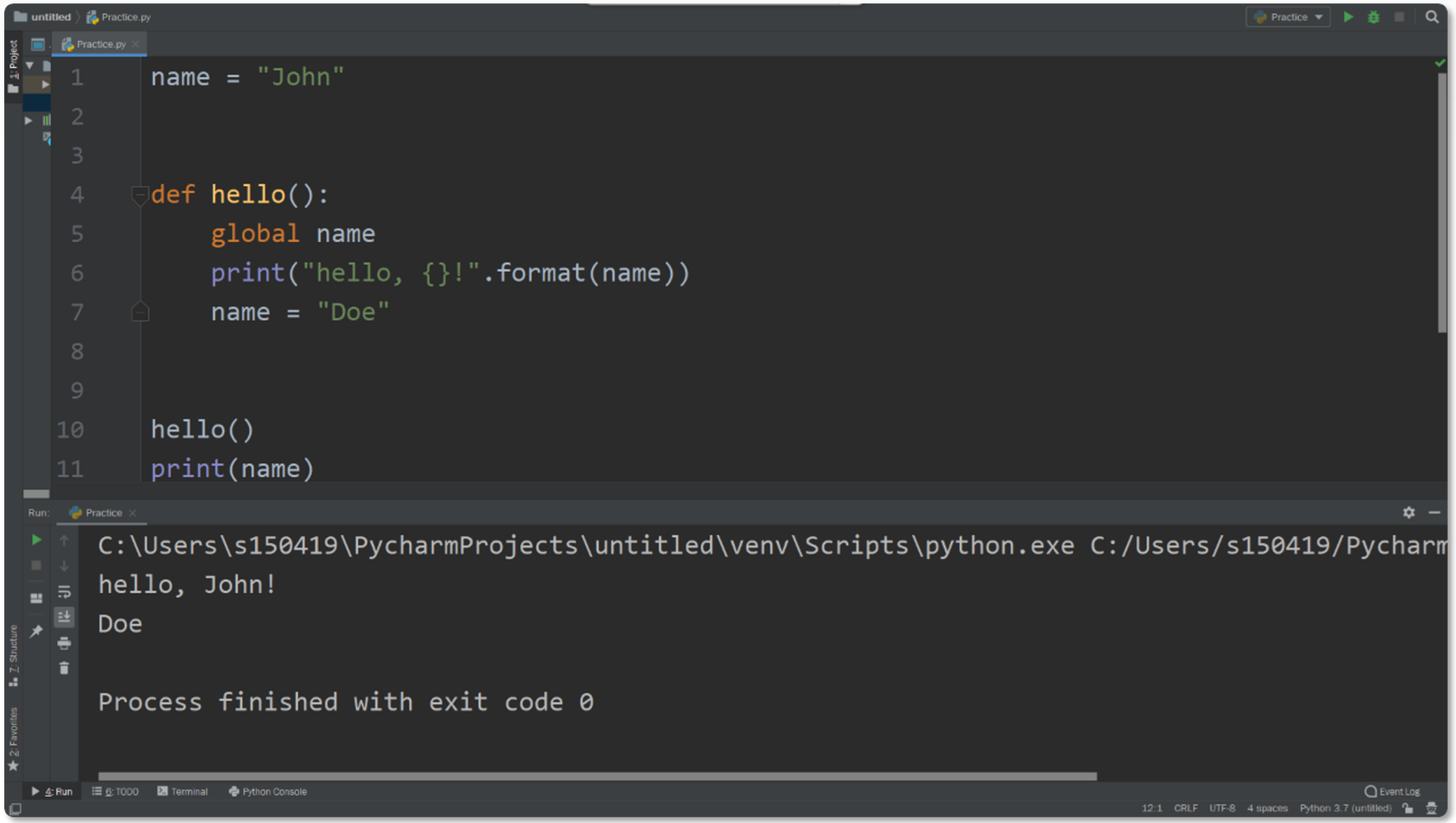


Code Handling

Global Keyword

The ***global*** keyword is used to modify a global variable within a function in which a variable of the same name exist.

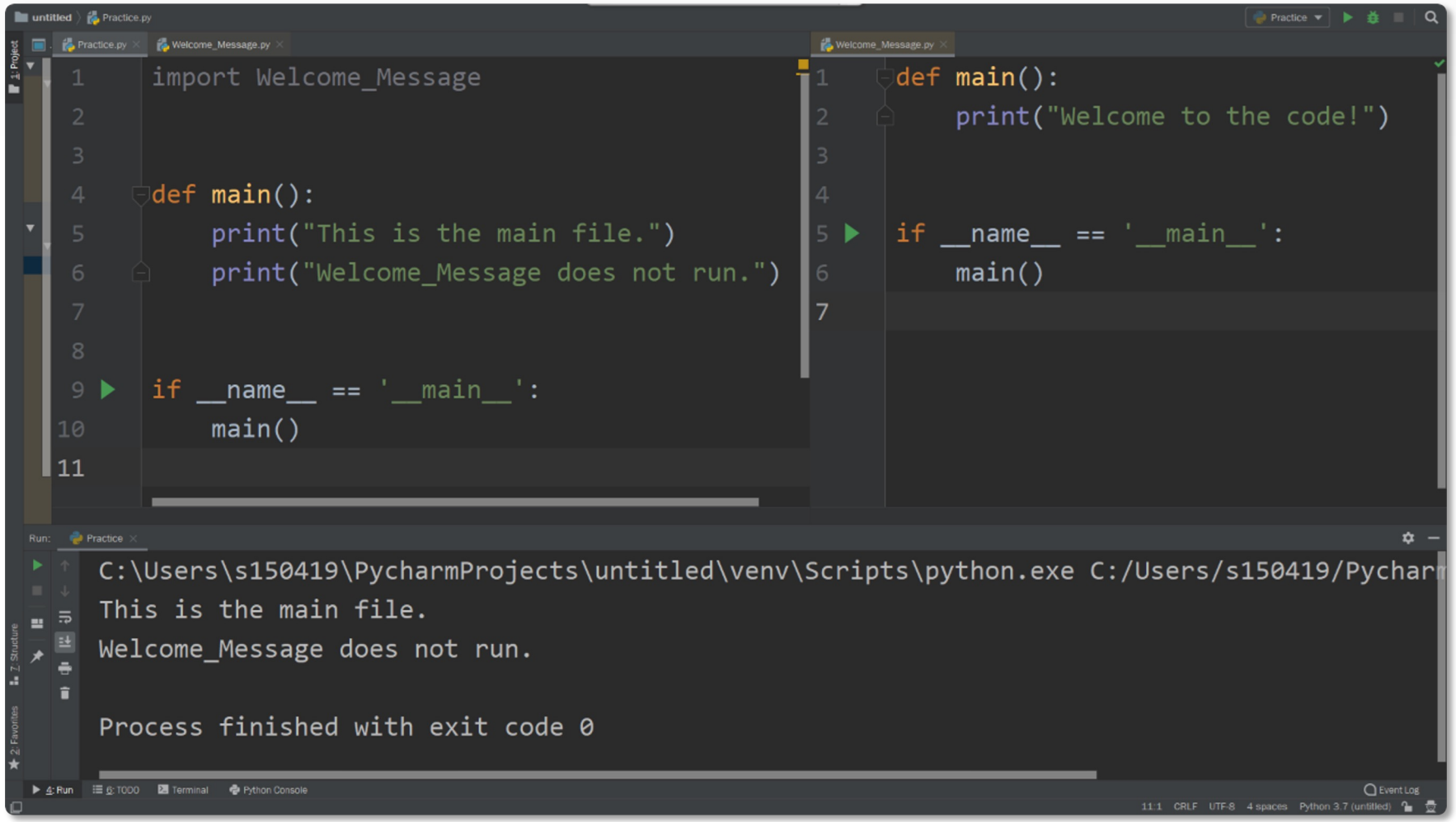
It instructs Python to use the variable from the global scope.



Code Handling

__name__ Variable

- Every Python file has the ***__name__*** variable.
- It returns a different result depending on the execution state of a file.
- The variable can be used to check if a file was imported.



Proper Code Management

- ***import*** should be at the beginning of a program.
- **Global variables** should follow imports.
- ***__name__ == '__main__'*** should be written at the end of the file.

```
import random

number = random.randint(0, 5)

def main():
    for i in range(number):
        print("Hello")

if __name__ == '__main__':
    main()
```

```
==========================================================================
Hello
Hello
Hello

Process finished with exit code 0
```

Code Handling

Lab PY-05-L3

Scope Behavior

10–20 Min.

Mission

Create global variables and try to make changes to them within a function.

Steps

- Declare global variables.
- Create a function to manipulate the variables.
- Examine the results.

Environment & Tools

- Windows/Linux
- Python 3
- PyCharm

Related Files

- Lab document

Code Handling

Lab PY-05-L4

Main Identification

15–25 Min.

Mission

> Prepare a program that executes only if its main file is run directly.

Steps

- Create a file to be imported.
- Create a function with a print message.
- Create a file import operation.

Environment & Tools

- Windows/Linux
- Python 3
- PyCharm

Related Files

- Lab document

Code Handling

Lab PY-05-L5

Bullseye

15–25 Min.

Mission

Create a game in which a user guesses a random number and receives the correct answer in return.

Steps

- Generate a random number.
- Check if the user guessed the correct random number.

Environment & Tools

- Windows, Linux, or macOS
- Python 3
- PyCharm

Related Files

- Lab document

Code Handling

TDX Arena Challenge

Troll

Mission

> Find the troll's login information to set the divine memes free!

Steps

- Log into TDX Arena and search for the challenge named **Troll**.
- Reverse the legacy troll code.
- Find the username.
- Find the passwords.

Code Handling

TDX Arena Challenge

Blocks

Mission

> Try to decrypt a message using the Python script you will find.

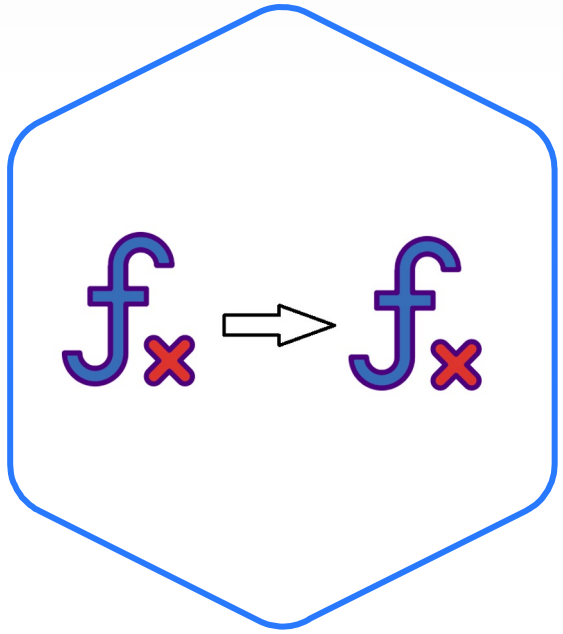
Steps

- Log into TDX Arena and search for the challenge named **Blocks**.
- Find the Python script.
- Try to understand the code.
- Decrypt the message.

Functions

Recursion

What Is Recursion?



- Functions that call themselves
- Provide a way of looping through data.
- Must be handled carefully to prevent endless execution

Recursion Implementation

- Recursion is implemented by having a function call itself repeatedly.
- It should be handled carefully to prevent endless loops.

```
count = 0

def recur(count):
    if count == 10:
        return

    print("*" * count)
    count += 1
    recur(count)

recur(count)
```

Recursion

Lab PY-05-L6

Recursive Search

15–25 Min.

Mission

Implement a recursion to print values from a nested list.

Steps

- Create a nested list.
- Print only the numbers in the list.
- Recursively iterate over the list.

Environment & Tools

- Windows, Linux, or macOS
- Python 3
- PyCharm

Related Files

- Lab document

Recursion

Lab PY-05-L7

Directory Listing

30–40 Min.

Mission

Implement recursion with various functions of the OS library.

Steps

- Request a path from the user and print system data.
- Create a function to identify files and directories.
- Recursively iterate over the data and print results.

Environment & Tools

- Windows/Linux
- Python 3
- PyCharm

Related Files

- Lab document

Functions

Object-Oriented Programming

What Is Object-Oriented Programming?

- Objects are complex data types that contain attributes and functionalities.
- They provide the flexibility of creating custom structured data.

Class

- A class is a blueprint for an object.
- It defines an object's properties.
- When a program is run, a class is the blueprint used to create objects in the system's memory.

Defining a Class

class is followed by the name of the class and a colon.

__init__ allows the attributes of a class to be initialized for an object.

Class attributes and methods are accessed using ***self***

```
class Car:
    def __init__(self, color, window_number, price):
        self.color = color
        self.window_number = window_number
        self.price = price
```

Object Creation

Call the class by its name with a value corresponding to ***__init__***

Use ***<object name>.<attribute>*** to access object attributes.

```
class Car:
    def __init__(self, color, window_number, price):
        self.color = color
        self.window_number = window_number
        self.price = price


car1 = Car("Blue", 4, 56000)
car2 = Car("Red", 2, 230000)

print(car1.color)
print(car2.price)
```

```
C:\Users\User\PycharmProjects\PerfectStorm\venv\Scripts\python.exe C:/Users/User/PycharmProjects/PerfectStorm/Intr
Blue
230000

Process finished with exit code 0
```

Object-Oriented Programming

Lab PY-05-L8

Car Creation

20–30 Min.

Mission

Implement OOP by creating a class and various objects.

Steps

- Create a ***Car*** class that includes the ***__init__*** function.
- Create two objects from the class.
- Print a different attribute from each object.

Environment & Tools

- Windows, Linux, and macOS
- Python 3
- PyCharm

Related Files

- Lab document

Object-Oriented Programming

TDX Arena Homework

Mission

> Use TDX Arena to gain a better understanding of recursion within the context of Python.

Steps

- Sign into the **TDX Arena** platform.
- Navigate to the **Practice Arena**.
- Navigate to the **Python Programming** course.
- Select ***PY03 Looping & Math***.
- Select the ***Recursions*** lab.

Complete the homework **before** the next class.

Complete any labs or challenges you **did not finish** in class.

Recursions

Thank You

Questions?