# Day 3 - Control Flow and Logical Operators 🚦

## Overview

In this session, we explore control flow and logical operators in Python. We cover essential concepts such as control flow with `if`/`else` statements, conditional operators, the modulo operator, nested `if` statements, `elif` statements, multiple `if` statements in succession, and logical operators. The main focus is on building a text-based adventure game called "Treasure Island," which involves making choices and navigating through different scenarios.

## Project: Treasure Island 🏴‍☠️

![day03](https://user-images.githubusercontent.com/98851253/154304693-a3aa3a5a-e8f3-46b8-bcea-6f0884aad99c.gif)

GIF credit: [Phillipai](https://github.com/phillipai)

### Description

The Treasure Island project challenges you to embark on an adventure to find hidden treasure. In this text-based game, you will make choices that determine your fate. You'll need to choose your path wisely as you encounter various challenges and obstacles. The game's outcome depends on your decisions, leading to either success or failure.

### How it Works

Here's a step-by-step breakdown of how the Treasure Island game works:

1. \*\*Introduction\*\*: The program starts by welcoming you to Treasure Island and informing you of your mission to find the treasure.

2. \*\*Decision-Making\*\*: You are presented with choices that include directions (left or right) and actions (swim or wait). Your choices at each step affect the game's outcome.

3. \*\*Outcome\*\*: Based on your choices, the game will lead you to different scenarios, including doors of various colors. Each choice has consequences that can result in either winning the game or encountering a game over scenario.

4. \*\*Conclusion\*\*: The game concludes with a message indicating whether you've successfully found the treasure or if your adventure has come to an end.

This project provides an engaging way to practice control flow and logical operators in Python while having fun on a treasure hunt! Enjoy playing the Treasure Island game and making decisions that shape your destiny! 🏴‍☠️

# Print the initial game scene (ASCII art)

print('''

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

| | | |

\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.=""\_;=.\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_

| | ,-"\_,="" `"=.| |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_"=.\_o`"-.\_ `"=.\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| `"=.\_o`"=.\_ \_`"=.\_ |

\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:=.\_o "=.\_."\_.-="'"=.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_

| | \_\_.--" , ; `"=.\_o." ,-"""-.\_ ". |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_.\_" ,. .` ` `` , `"-.\_"-.\_ ". '\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| |o`"=.\_` , "` `; .". , "-.\_"-.\_; ; |

\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_| ;`-.o`"=.\_; ." ` '`."\` . "-.\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_

| | |o; `"-.o`"=.\_`` '` " ,\_\_.--o; |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_| ; (#) `-.o `"=.`\_.--"\_o.-; ;\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_|o;.\_ " `".o|o\_.--" ;o;\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_

/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_"=.\_o--.\_ ; | ; ; ;/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_

\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_"=.\_o--.\_ ;o|o; \_.\_;o;\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_

/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_"=.\_o.\_; | ;\_.--"o.--"\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_

\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_"=.o|o\_.--""\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_

/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_ /

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

''')

# Print game introduction

print("Welcome to Treasure Island.")

print("Your mission is to find the treasure.")

# Prompt the player to choose a direction

direction = input("You're at a crossroad. Do you want to go left or right? ")

if direction.lower() == 'left':

# If the player chooses to go left, ask them to swim or wait

swim\_or\_wait = input("You come to a lake. Do you want to swim across or wait for a boat? ")

if swim\_or\_wait.lower() == 'wait':

# If the player chooses to wait, ask them to choose a door

door = input("You arrive at an island with three doors - red, blue, and yellow. Which door do you choose? ")

if door.lower() == 'yellow':

# If the player chooses the yellow door, they win

print("Congratulations! You have found the treasure. You win!")

elif door.lower() == 'red':

# If the player chooses the red door, they are burned by fire and lose

print("Burned by fire. Game Over.")

elif door.lower() == 'blue':

# If the player chooses the blue door, they are eaten by beasts and lose

print("Eaten by beasts. Game Over.")

else:

# If the player chooses any other door, they lose

print("Game Over.")

else:

# If the player chooses to swim, they are attacked by trout and lose

print("Attacked by trout. Game Over.")

else:

# If the player chooses to go right, they fall into a hole and lose

print("Fall into a hole. Game Over.")

# Day 4 - Randomisation and Python Lists 🎲

## Overview

In this session, we dive into the concept of randomization and working with Python lists. We explore the use of the random module for generating random choices and gain a better understanding of lists, including appending items and handling index errors. The primary focus is on creating a classic game of Rock, Paper, Scissors.

## Project: Rock Paper Scissors ✊📄✂️

![day04](https://user-images.githubusercontent.com/98851253/154310127-00f4adf7-fac0-40c0-a374-a49ac22292d4.gif)

GIF credit: [Phillipai](https://github.com/phillipai)

### Description

The Rock Paper Scissors project brings the classic hand game to life in Python. Users play against the computer by choosing Rock, Paper, or Scissors, and the computer generates its choice randomly. The program then determines the winner based on the rules of the game. It's a simple yet engaging project that demonstrates randomization and list manipulation in Python.

### How it Works

Here's a high-level overview of how the Rock Paper Scissors game works:

1. \*\*ASCII Art\*\*: The program starts by defining ASCII art representations for Rock, Paper, and Scissors.

2. \*\*User Input\*\*: The user is prompted to choose Rock (0), Paper (1), or Scissors (2).

3. \*\*Computer's Choice\*\*: The computer generates a random choice (0, 1, or 2) using the random module.

4. \*\*Gameplay\*\*: The program compares the user's and computer's choices to determine the winner based on the Rock, Paper, Scissors rules.

5. \*\*Outcome\*\*: The game announces the winner or declares a draw.

This project provides an excellent opportunity to explore randomization, lists, and conditional statements while having fun playing the classic Rock Paper Scissors game! ✊📄✂️

Code:

# Define the rock, paper, and scissors ASCII art

rock = '''

\_\_\_\_\_\_\_

---' \_\_\_\_)

(\_\_\_\_\_)

(\_\_\_\_\_)

(\_\_\_\_)

---.\_\_(\_\_\_)

'''

paper = '''

\_\_\_\_\_\_\_

---' \_\_\_\_)\_\_\_\_

\_\_\_\_\_\_)

\_\_\_\_\_\_\_)

\_\_\_\_\_\_\_)

---.\_\_\_\_\_\_\_\_\_\_)

'''

scissors = '''

\_\_\_\_\_\_\_

---' \_\_\_\_)\_\_\_\_

\_\_\_\_\_\_)

\_\_\_\_\_\_\_\_\_\_)

(\_\_\_\_)

---.\_\_(\_\_\_)

'''

# Import the random module for generating computer's choice

import random

# Get the user's choice as an integer (0 for Rock, 1 for Paper, 2 for Scissors)

user\_choice = int(input("What do you choose? 0 for Rock, 1 for Paper, 2 for Scissors \n"))

# Generate a random choice for the computer (0 for Rock, 1 for Paper, 2 for Scissors)

computer\_choice = random.randint(0, 2)

# Create a list of game options for printing

game\_list = [rock, paper, scissors]

# Check if the user's input is valid (0, 1, or 2)

if user\_choice < 0 or user\_choice >= 3:

print("You entered an invalid number. You lose!")

else:

# Print the user's and computer's choices using the ASCII art

print("You chose:")

print(game\_list[user\_choice], "\n")

print("Computer chose:")

print(game\_list[computer\_choice])

# Determine the winner of the game

if user\_choice == computer\_choice:

print("It's a draw!")

elif (

(user\_choice == 0 and computer\_choice == 2)

or (user\_choice == 1 and computer\_choice == 0)

or (user\_choice == 2 and computer\_choice == 1)

):

print("You win!")

else:

print("You lose!")

# Day 5 - Python Loops 🔄

## Overview

In this session, we explore the concept of Python loops, specifically the `for` loop. We'll learn how to use `for` loops with lists, utilize the `range()` function, and work with the random module. The primary focus is on creating a password generator program that generates random passwords based on user input.

## Project: Password Generator 🔐

![day05](https://user-images.githubusercontent.com/98851253/154311198-83cc6a60-6a57-4e21-bb01-6b54593def0e.gif)

GIF credit: [Phillipai](https://github.com/phillipai)

### Description

The Password Generator project enables you to create strong and random passwords to enhance your online security. This Python program prompts you to specify the number of letters, symbols, and numbers you want in your password. It then generates a unique password that includes a combination of these elements.

### How it Works

Here's a high-level overview of how the Password Generator works:

1. \*\*Character Lists\*\*: The program defines lists of characters, including letters (both lowercase and uppercase), numbers, and symbols.

2. \*\*Welcome Message\*\*: It displays a welcome message to the user.

3. \*\*User Input\*\*: The user is asked to specify the number of letters, symbols, and numbers they want in their password.

4. \*\*Password Generation\*\*: The program uses `for` loops and the random module to generate random characters from the defined lists. It then appends these characters to a password list.

5. \*\*Randomization\*\*: The characters in the password list are shuffled to randomize the order.

6. \*\*Final Password\*\*: The characters in the password list are combined to create the final password.

7. \*\*Display\*\*: The generated password is displayed to the user.

This project provides an excellent opportunity to practice Python loops, the `range()` function, list manipulation, and randomization while creating a useful tool for enhancing online security with strong passwords! 🔐

Code:

#Password Generator Project

import random

letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

symbols = ['!', '#', '$', '%', '&', '(', ')', '\*', '+']

print("Welcome to the PyPassword Generator!")

nr\_letters= int(input("How many letters would you like in your password?\n"))

nr\_symbols = int(input(f"How many symbols would you like?\n"))

nr\_numbers = int(input(f"How many numbers would you like?\n"))

#Eazy Level - Order not randomised:

#e.g. 4 letter, 2 symbol, 2 number = JduE&!91

# password = ""

# for alpha in range(1, nr\_letters + 1):

# password += random.choice(letters)

# for symbl in range(1, nr\_symbols + 1):

# password += random.choice(symbols)

# for num in range(1, nr\_numbers + 1):

# password += random.choice(numbers)

# print(password)

#Hard Level - Order of characters randomised:

#e.g. 4 letter, 2 symbol, 2 number = g^2jk8&P

password\_list = []

for alpha in range(1, nr\_letters + 1):

password\_list += (random.choice(letters))

for symbl in range(1, nr\_symbols + 1):

password\_list += (random.choice(symbols))

for num in range(1, nr\_numbers + 1):

password\_list += random.choice(numbers)

random.shuffle(password\_list)

password = ""

for n in password\_list:

password += n

print(password)

# Day 6 - Python Functions & Karel 🤖

## Overview

In this session, we explore Python functions and Karel, a simple programming language used for teaching purposes. We'll learn how to define and call functions in Python, understand the importance of proper indentation, work with while loops, and solve a maze escape problem using code.

## Project: Maze Escape 🧩

![day06](https://user-images.githubusercontent.com/98851253/154312745-8abc5397-27b7-4a1d-b29c-3a1527280868.gif)

GIF credit: [Phillipai](https://github.com/phillipai)

### Description

The Maze Escape project challenges you to write code to help a robot navigate through a maze and reach its goal. In this program, we use Python functions to define actions for the robot, including turning and moving forward. By controlling the robot's movements, we aim to guide it to the exit of the maze.

### How it Works

Here's a high-level overview of how the Maze Escape code works:

1. \*\*Defining Actions\*\*: We define a Python function called `turn\_right()` to make the robot turn right. This function is used to change the robot's direction within the maze.

2. \*\*Moving Forward\*\*: The robot continuously moves forward while the path ahead is clear.

3. \*\*Escaping the Maze\*\*: The primary goal is to escape the maze. The robot follows a set of rules: if the right path is clear, it turns right and moves forward; if the front path is clear, it moves forward; otherwise, it turns left.

4. \*\*Maze Solution\*\*: By implementing these rules and using the `turn\_right()` function, the robot eventually navigates through the maze and reaches its goal.

This project offers an introduction to functions in Python and provides an interactive way to practice logical problem-solving skills by guiding a robot through a maze! 🤖🧩

Code 6:

def turn\_right():

turn\_left()

turn\_left()

turn\_left()

while front\_is\_clear():

move()

turn\_left()

while not at\_goal():

if right\_is\_clear():

turn\_right()

move()

elif front\_is\_clear():

move()

else:

turn\_left()

Code 7: