

## 03: Control Structures

## Objectives

In this lab, you will learn about control structures and how they can be useful in nearly any type of program. You will create an automated rock-paper-scissors program using these control structures, which can be used later in the semester for machine learning. You will:

- ☐ Learn about **if**, **for**, and **while** control structures.
- ☐ Create a program to run rock-paper-scissors games with user input.
- ☐ Store the results of each game in an array and calculate result statistics.
- ☐ Utilize functions to simplify programs.

## Background

Control structures are used to make decisions in a program. They are essential for creating robust programs that can execute different code depending on a variable's value. Control structures perform certain tasks based on a precondition. A precondition is the value of a variable which is used in the control structure, which determines how the control structure operates. The most simple example of a precondition is a Boolean in an **if** control structure. If the precondition is initially true, the **if** control structure will execute, otherwise, it will be skipped over entirely. For example, algorithm 1 will print out **True** since the precondition for the **if** control structure is met.

Algorithm 1: Preconditions in an <b>if</b> statement
<pre>1 precondition ← True; 2 a ← False; 3 if precondition == True then 4     a = True; 5 end 6 Print(a);</pre>

## Logical Operators

Logical operators are used to determine when a control structure should execute the code it contains. All logical operators convert to Boolean values (true or false). A logical operator can take on the following forms:

Additionally, multiple logical operators can be used together with **AND** and **OR** to allow preconditions to interact in a control structure (ie. if this AND that).

Symbol	Name
==	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
!= ( $\sim$ = for MATLAB users)	Not equal to

Table 1: Basic logical operators used in control statements. Keep in mind that the syntax varies with language. You should always check your language's syntax.

## if Statements

The **if** statement is the simplest control structure. If the Boolean value is true for an **if** statement, then the code within it will execute, otherwise it will be entirely skipped over. An **else if** statement can be used to allow a single **if** statement to handle many different criteria. Finally, **else** control structure can be used in conjunction with an **if** statement to account for scenarios when the precondition does not meet the criteria for the **if** or any other **else if** statement to execute. Algorithm 2 shows an example of the different types of **if** statements, and would result in an output of **b**.

### Algorithm 2: if, else if, and else statements

```

1 precondition ← b;
2 if precondition == a then
3   | Print(a);
4 else if precondition == b then
5   | Print(b);
6 else
7   | Print(c);
8 end

```

## for Loops

**for** loops are used to perform iterative operations a known number of times. A **for** loop's iteration can be instantly stopped by including a **break** statement. Additionally, a **continue** statement can be used to prevent the execution of any further code below it, while still allowing the **for** loop to continue iterating.

## while Loops

**while** loops execute their code as long as as their precondition is met. The **while** control structure is can be used to loop through code an initially unknown number of times. **while**

loops do not have built in iteration like a **for** loop, so it is easy to get 'trapped' in the while loop, as one might never meet the stopping criterion ([https://en.wikipedia.org/wiki/Infinite\\_loop](https://en.wikipedia.org/wiki/Infinite_loop)). To avoid infinite loops, one should always include a stopping criterion.

**Algorithm 3:** Using a **while** loop to approximate a number

```

1 oldguess ← 1;
2 newguess ← 100;
3 while |oldguess − newguess| / oldguess > 0.01 do
4   | oldguess ← newguess;
5   | newguess ← UpdateGuess(oldguess);
6 end

```

## Functions

Functions are a convenient way of compartmentalizing programs into self-contained blocks of code. They can take no inputs, or as many inputs as is desired, and return as many outputs as required. Algorithm 4 shows an example of a function which takes two inputs and multiplies them by 100 and 50, returning the results as two outputs.

**Algorithm 4:** A function that takes inputs **a** and **b** and returns outputs **x** and **y**

```

1 Function ExampleFunction(a, b)
2   | x = 100 · a;
3   | y = 50 · b;
4   | return x, y;

```

## Chat GPT!

ChatGPT is an advanced language model powered by OpenAI. It is a cutting-edge AI technology that has been trained on a massive amount of text data, enabling it to understand and generate human-like text responses. This remarkable language model is designed to engage in conversations with users, providing intelligent and contextually relevant answers to various questions.

In this coding class, we will explore the capabilities of ChatGPT, learn how to integrate it into our projects, and leverage its natural language processing abilities to enhance our applications. We will witness how ChatGPT can generate human-like responses and have engaging conversations with users. This exposure will provide a hands-on experience in working with cutting-edge AI technologies and demonstrate the potential for incorporating AI-driven solutions into real-world programming scenarios.

Throughout the class, we will also discuss the ethical considerations surrounding AI technology and the responsible use of language models like ChatGPT. As we delve into the exciting world of natural language processing, we will learn how to make the most of this powerful tool while respecting privacy, security, and societal impact.

By the end of this coding class, you will have gained valuable insights into the capabilities of ChatGPT, developed practical skills in working with advanced language models, and learned to harness the power of AI to enhance your coding projects. Whether you are interested in building intelligent chatbots, automating text-based tasks, or exploring new horizons in natural language understanding, ChatGPT will be an invaluable addition to your programming toolkit.

## ChatGPT

### Did you use ChatGPT to solve Lab 01 and Lab 02?

Let's try couple of problem from Lab 02 to solve using chat GPT (Figure 1, 2, and 3)

## Lab Exercises: Rock-Paper-Scissors

Create a script using some or all of the above control structures to play rock-paper-scissors with user input. The user can input **r** for rock, **p** for paper, or **s** for scissors. A case should be implemented in case the player inputs any other variable, giving an unknown result. The script should store the results of each game in an array which can have 3 character types for any element: **c** (computer wins) **p** (player wins) and **d** (draw). Any unknown results should not be stored. Additionally, the script should store the player and computer's choices in separate arrays for every successful game. Both the total number of wins and win percentages for the player and computer should both be calculated and printed, along with the number and percentage of games that ended in a draw. Finally, the total number of successfully completed games should be output, along with the percentage of times both the player and computer chose rock, paper, and scissors.

Your code must ask the player whether they wish to continue playing after every game, taking a **y/n** character input. An **n** input should stop the code, while inputting **y** should cause another game to be played. A **maxGames** criterion should also be used to stop the code, regardless of user input. (Note: MATLAB users may find the **switch** statement particularly useful in this lab, while Python users can use dictionaries to accomplish an equivalent concept.)

SH

A = [1 2 3 4 5]; Write a python code to perform element -wise arithmetic on a single array. Calculate  $5*A-2$



Sure! Here's the Python code to perform element-wise arithmetic on the array A and calculate  $5*A-2$ :

python

Copy code

```
import numpy as np

# Define the array A
A = np.array([1, 2, 3, 4, 5])

# Perform element-wise arithmetic
result = 5 * A - 2

# Print the result
print(result)
```

Output:

css

Copy code

```
[ 3  8 13 18 23]
```

Figure 1: Python code from Chat GPT for the Element-wise arithmetic on a single array.

SH

A = [1 2 3 4 5]; Write a matlab code to perform element -wise arithmetic on a single array. Calculate  $5*A-2$



Sure! Here's the MATLAB code to perform element-wise arithmetic on the array A and calculate  $5*A-2$ :

matlab

Copy code

```
% Define the array A
A = [1, 2, 3, 4, 5];

% Perform element-wise arithmetic
result = 5 * A - 2;

% Display the result
disp(result);
```

Output:

Copy code

```
3      8     13     18     23
```

Figure 2: Matlab code from Chat GPT for the Element-wise arithmetic on a single array.

SH

Create a function `fibonacci(N)` that returns an array of length `N` containing the First `N` terms of the Fibonacci sequence. Output a Fibonacci sequence with 10 terms.



Sure! Here's the Python code for the `fibonacci(N)` function that generates the first `N` terms of the Fibonacci sequence:

python

Copy code

```
def fibonacci(N):  
    sequence = [0, 1] # Initialize the sequence with the first two terms  
  
    # Generate the Fibonacci sequence  
    while len(sequence) < N:  
        next_term = sequence[-1] + sequence[-2]  
        sequence.append(next_term)  
  
    return sequence  
  
# Generate the Fibonacci sequence with 10 terms  
fib_sequence = fibonacci(10)  
  
# Print the Fibonacci sequence  
print("Fibonacci sequence with 10 terms:", fib_sequence)
```

Output:

csharp

Copy code

```
Fibonacci sequence with 10 terms: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

Figure 3: Python code from Chat GPT for the Fibonacci function.