BTEC Level 3 Computing

Unit 1 - Principles of Computer Science

Control Structures


Selecting, applying, using and interpreting common programming control structures to analyse and improve the effectiveness of code.

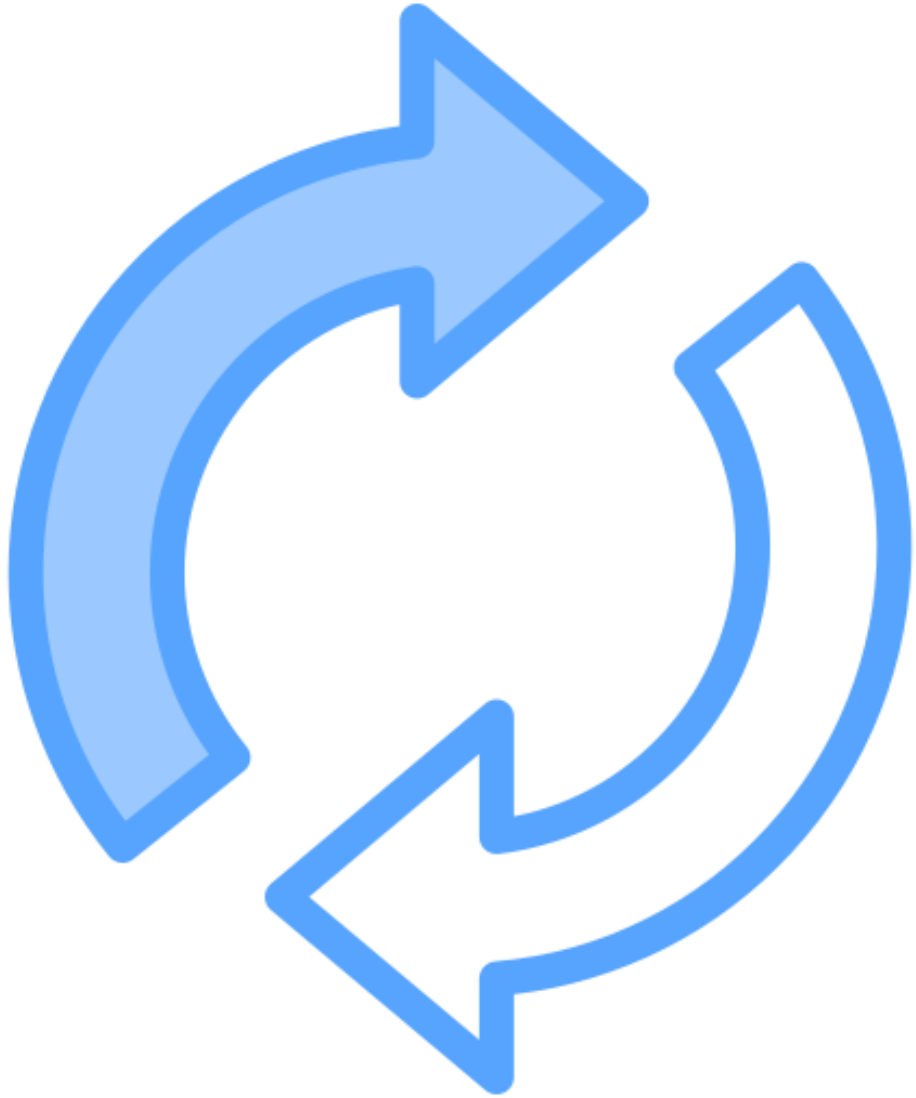
Control Structures

The Main Control Sturucture Categories

- Loops.
- Branches.
- Function/Procedure Calls.

Loops

- They provide a way to automate repetitive tasks and avoid writing the same code multiple times.
- Instead of writing the same instructions over and over, you put them inside a loop, and the loop handles the repetition for you.
- Repeat.
- For.
- While.
- Break (used to stop the loop).



Repeat Loop

- This is not an actual loop.
- This is what the loop does.
- This can be created using a For Loop or a While Loop.

For Loop

Allows you to repeatedly execute a block of code a specific number of times or for each item in a collection (like a list, tuple, string, or dictionary).

It is useful when you know in advance how many times you want to iterate or when you want to process each element of a sequence.

For Loop Example Python

```
# For Loop Example

text_to_show = "RonsTechHub is amazing......"

counter = 0
for i in range (0, 100):
    print("Count:", counter, text_to_show)
    counter = counter+1
```

While Loop

A control flow statement that allows you to execute a block of code as long as a certain condition remains true.

Unlike a for loop, which iterates over a sequence a fixed number of times, a while loop continues to run until its condition becomes false.

For example, as long as there is power to this device do this.

Keep counting until you get to 10,000.

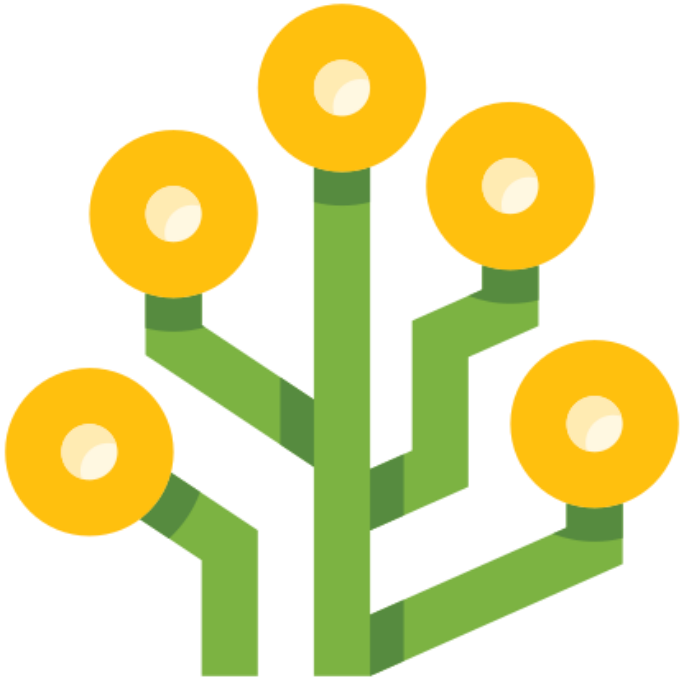
While Loop Example

```
count = 0
while count < 10000:
    print("Count:",count)
    count += 1  # Important: Update the condition!
```

Branches

- Branching lets your code make decisions. It uses conditions (true/false) to choose which code to run.

- If.
- Else.
- ElseIf (Elif).
- Then (replacable with else or elif).



If Statement

An if statement is like asking a question: "Is this condition true?"

If the answer is "yes" (true), the computer runs a specific block of code.

If the answer is "no" (false), the computer skips that code. It's how your program makes simple decisions.

Else If or Elif Statement

The purpose of else if (often written as elif in Python, elseif in some other languages) is to provide a way to check *multiple* conditions in sequence.

It's an extension of the basic if statement.

Check one thing, then use this to check another.

Else Statement

The else keyword in programming provides a way to define a block of code that should be executed when a specific condition is *not* met.

It's always used in conjunction with a conditional statement, most commonly an if statement (and sometimes with loops as we'll see).

This is a good way to catch errors, for example, the "if" might not be true and the "elif" might not be true.

To prevent your code from crashing, you can use else to catch all other possibilities.

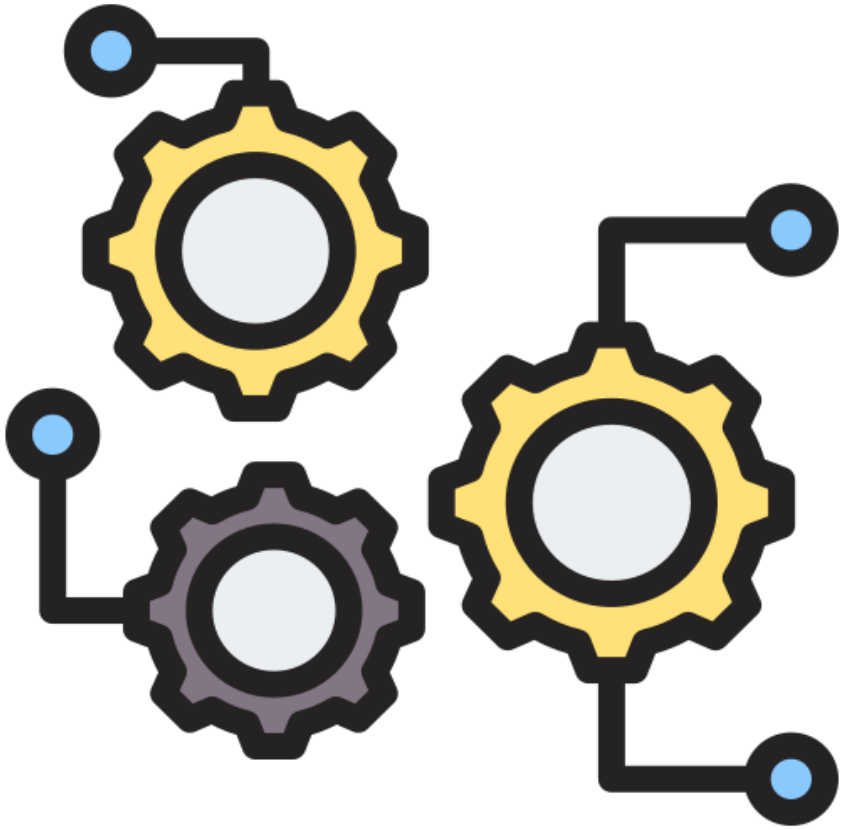
If, Else and Elif Statement Example

- ```
  age = 17

  if age == 17:
      print("You are NOT allowed to drive.")
  elif age == 18:
      print("You are allowed to drive")
  else:
      print("There was an error, please try again")
  ```
  ```

Functions

- Functions are like mini-programs within your main program.
- They are reusable blocks of code that perform a specific task.
- Think of them as tools in a toolbox: each tool has a particular job, and you can use them whenever you need to perform that job, without having to recreate the tool every time.
- Defining Functions.
- Declaring Arguments.
- Calling Functions.



Defining A Function

This simply means to create a function.

For Python3, the "def" keyword is used to start a function creationg.

This "def" cannot be used in your program, it is a built in name in Python.

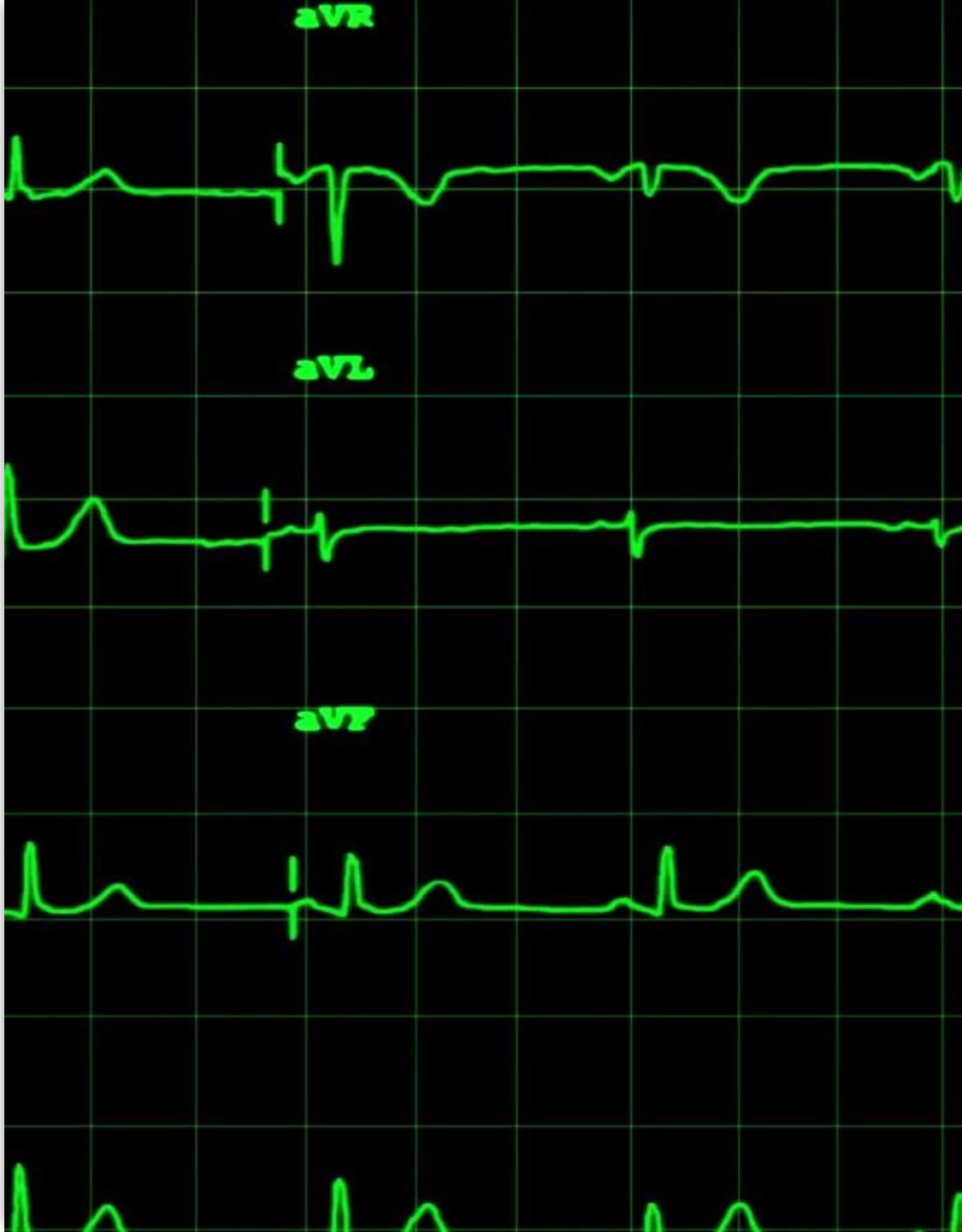
After you define, you need to then call the function.

Defining A Function Example

```
def SpeakTheTruth():
    print("RonsTechHub is amazing....")
```

Declaring Arguments

- This tells the function what is needs to do its work.
- For example, a function needs two numbers as an input, then it will multiply them.
- The function is still not able to be run, we will then need to call the function.



Declaring Arguments Example

```
def multiply_integers(num1, num2):
    product = num1 * num2                # Perform the multiplication
    return product                       # Return the result
```

Calling Functions

- This is where we tell the program to use the function we made earlier.

```
def multiply_integers(num1, num2):
    product = num1 * num2                # Perform the multiplication
    print (product)                       # Return the result

multiply_integers(10, 51)
```

Functions vs Procedures

- Functions Return a value to the user.

- Procedures do not return a value to the user.

Function Example

```
def multiply_integers(num1, num2):
    product = num1 * num2                  # Perform the multiplication
    print (product)                         # Return the result

multiply_integers(10, 51)
```

Procedure Example

```
def multiply_integers(num1, num2):
    product = num1 * num2                  # Perform the multiplication
    return (product)                         # Return the result

multiply_integers(10, 51)
```

Next Time

Data Structures