CONTROL
STATEMENTS

OUTLINE

- Introduction
- If statement
- Else statement
- Elif statement
- For statement
- While statement

INTRODUCTION

Control Statements help out in adding controls to our codes. It's all about decision making, just the way it applies in our normal daily life experiences. Like doing something based on an outcome of a particular condition. Actually, most of the decisions we make in life depends on conditional outcomes.

In a nut shell, the way you make decisions in life is the same way you can also make decisions in programming. It's nothing strange.

THE "and" AND "or" EXPRESSION

The "and" and "or" expression:: the "and" expression is used to check if two conditions are true in a decision making process while the "or" expression checks if either of the condition is true. If one the condition is false, the "or" expression still carries out its statements but if both conditions are false, the "or" expression cannot execute its block of code.

IF
STATEMENT

The If statement consist of a Boolean expression which is followed by an executable statement or what is referred to as “block of code or code block”.

What it does is that it tests the condition or Boolean expression and on the returns of a true Boolean value executes the code block.

Example

```
var = “chima”
If(var==“chima”):
        Print(“Yes”)
```

ELSE STATEMENT

The If-else statement also does same thing as the if statement. The major difference is that If-else statement has a code to execute when the Boolean expression returns a false value.

EXAMPLE

```
var = “chima”
If(var==“chima”):
        print(“Yes”)
else:
        print(“No”)
```

ELIF STATEMENT

The If-elif statement executes one code block from multiple statements. You have this at your disposal for usage when you have options to choose from.

In a nut shell, when there are multiple choices, the if-elif statement becomes handy for you as a programmer.

EXAMPLE

```
var = “chima”
If(var==“chima”):
        Print(“Yes”)
elif(var==“okeke”):
        Print (“Maybe”)
else:
        print(“No”)
```

LOOP
CONTROL

There are times you may feel like doing something repeatedly, instead of one after the other. Now that's what a loop helps us to achieve. It allows us to execute an item multiple times.

NB: There are two types of loops in Python namely, the for and while.

FOR LOOP

Python's for statement iterates over the items of any sequence (a list or a string), in the order that they appear in the sequence.

EXAMPLE

```
words = ['cat', 'window', 'state']
for w in words:
        print(w)
```

More on **FOR** LOOP

For loops can iterate over a sequence of numbers using the "range‟ function.

EXAMPLE

```
for w in range(10):
    print(w)

for i in range(4,10):
    print(i)
```

WHILE
LOOP

A while loop checks a condition to be true and then executes a block of code. So long the condition remains true, the loop will continue until the condition is satisfied and a false value is return.

EXAMPLE

```
# Prints out 0,1,2,3,4,5,6
c = 0
while c < 7:
        print(c)
        c += 1  # This is the same as count = count + 1
```

CONTINUE
STATEMENT

What a continue statement does is that, it skips a specified condition and then continues the flow of the program.

EXAMPLE

```
# Prints out only odd numbers - 1,3,5,7,9,11,13
for x in range(15):
        # Check if x is even
        if x % 2 == 0:
                continue
        print(x)
```

BREAK
STATEMENT

The break statement is used to terminate any loop and then transfers execution

to the statement.

EXAMPLE

```
# Prints out 0,1,2,3,4,5,6,7
c = 0
while true:
        print(c)
        c+= 1
        if c >= 8:
                break
```

ASSIGNMENT

- Write a program to calculate the total and grade for a particular school subject and display to the user.
- Write a program that uses any loop of your choice to display odd and even numbers from one to 50.
- Write a program that stops a loop if any condition of your choice is satisfied but re-iterates that loop if the condition is false.