ICT @ Mzuni

Programming with Python

by: Chimango Nyasulu, PhD

ICT @ Mzuni

Outline

- Conditions
- Boolean Logic
- Logic operators
- Ranges
- Control Statements (If-else, Loops, Short circuit evaluation)

ICT @ Mzuni

Conditions

- Conditions in Python are expressions that evaluate to either True or False.
- They are used to control the flow of a program by making decisions based on whether a specific condition is met.
- Conditions are fundamental to implementing logic in programs and are often used with control structures like ***if***, ***elif***, and ***else***.

ICT @ Mzuni

Conditions

Key Characteristics of Conditions

1. **Boolean Evaluation**

- Conditions result in a Boolean value (True or False), which determines the path a program will take.

2. **Comparison Operators**

- Conditions are typically created using comparison operators:

  == (Equality): Checks if two values are equal.

  != (Inequality): Checks if two values are not equal.

  > (Greater than), < (Less than), >= (Greater than or equal to), <= (Less than or equal to).

ICT @ Mzuni

Conditions

Key Characteristics of Conditions

3. **Logical Expressions**

- Conditions can be combined using logical operators (and, or, not) to form more complex evaluations.

4. **Use in Control Structures**

- Conditions are commonly used in if-else statements, loops, and other control structures to execute specific blocks of code based on the outcome of the condition.

ICT @ Mzuni

Conditions

```python
# Check if a number is positive
x = 10
if x > 0:
    print("x is positive")
else:
    print("x is not positive")
```

```python
# Check if a number is between 10 and 20
num = 15
if num > 10 and num < 20:
    print("Number is between 10 and 20")
else:
    print("Number is outside the range")
```

Boolean Logic and Logic Operators

- Boolean logic in Python refers to the use of Boolean values (True and False) and operations that evaluate to Boolean results.

- It is a fundamental concept in programming, used to make decisions, control program flow, and perform logical evaluations.

Key Characteristics of Boolean Logic

1. Boolean Values

- The two Boolean values in Python are True and False.
- These represent truth (True) or falsehood (False) in logical expressions.

Boolean Logic and Logic Operators

Key Characteristics of Boolean Logic

2. Logical Operators

- Boolean logic is implemented using logical operators

  - and: Returns True if both operands are True.

  - or: Returns True if at least one operand is True.

  - not: Inverts the Boolean value (e.g., not True becomes False).

3. Comparison Expressions

- Conditions such as equality (==), inequality (!=), greater than (>), less than (<), etc., evaluate to Boolean values.

4. Control Flow

- Boolean logic is used in control structures like if, elif, else, and loops to determine which code blocks to execute.

ICT @ Mzuni

Boolean Logic and Logic Operators

```python
age = 20
has_license = True


# Check if someone can drive
can_drive = age >= 18 and has_license
print(can_drive)  # Output: True
```

```python
temperature = 30
is_hot = temperature > 25


if is_hot:
    print("It's a hot day!")
else:
    print("It's a cool day.")
```

ICT @ Mzuni

Ranges

- In Python, the range() function is a built-in function used to generate a sequence of numbers.
- It is commonly used in loops and other scenarios where a sequence of numbers is required.
- The range() function is memory-efficient because it generates the numbers on-the-fly rather than storing them all in memory at once.
- A range object is immutable, meaning its values cannot be changed after creation.
- Often used in for loops to iterate over a sequence of numbers.

ICT @ Mzuni

Ranges

Syntax

range(start, stop, step)

start: (Optional) The starting value of the sequence (default is 0).

stop: The end value of the sequence (this value is not included in the sequence).

step: (Optional) The difference between each number in the sequence (default is 1)

```
1  # Generate numbers from 0 to 4
2  for i in range(5):
3      print(i)
4  # Output: 0, 1, 2, 3, 4
```

Control Statements

- Control statements in Python are used to manage the flow of execution in a program.

- They allow developers to make decisions, repeat tasks, and control how code is executed based on specific conditions.

- These statements are essential for creating dynamic and responsive programs.

Types of Control Statements

1. Conditional Statements

- Conditional statements allow the program to execute different blocks of code based on whether a condition evaluates to True or False.

ICT @ Mzuni

Control Statements

- `if` Statement : Executes a block of code if a condition is `True` .

```python
age = 20
if age >= 18:
    print("You are an adult.")
```

- `if-else` Statement : Executes one block of code if the condition is `True` , and another if it is `False` .

```python
temperature = 30
if temperature > 25:
    print("It's a hot day!")
else:
    print("It's a cool day.")
```

ICT @ Mzuni

Control Statements

- `if-elif-else` Statement : Handles multiple conditions.

```python
score = 85
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
else:
    grade = "C"
print(f"Your grade is: {grade}")
```

Control Statements

Types of Control Statements

2. Loops

- Loops allow you to execute a block of code repeatedly until a certain condition is met.

- `for` Loop : Iterates over a sequence (e.g., list, range, string).

```python
for i in range(5):
    print(i)
# Output: 0, 1, 2, 3, 4
```

ICT @ Mzuni

Control Statements

- `while` Loop : Repeats a block of code as long as a condition is `True`.

```python
count = 0
while count < 5:
    print(count)
    count += 1
# Output: 0, 1, 2, 3, 4
```

ICT @ Mzuni

Control Statements

Types of Control Statements

3. Short-Circuit Evaluation

- Short-circuit evaluation occurs when the result of a logical expression can be determined without evaluating all parts of the expression.

- `and` Operator : If the first condition is `False`, the second condition is not evaluated.

```python
x = 5
y = 10
if x > 10 and y > 5:
    print("Both conditions are True")
# Output: No output (first condition is False, so the second is skipped)
```

Control Statements

Types of Control Statements

3. Short-Circuit Evaluation

- `or` Operator : If the first condition is `True`, the second condition is not evaluated.

```python
if x < 10 or y > 15:
    print("At least one condition is True")
# Output: At least one condition is True
```