Learn Python Programming from Scratch

Topic: Boolean Values in Python

1. What are Boolean Values?

Boolean values are a fundamental data type in Python that represent truth values. They can only have one of two values:

- `True` - represents a true condition
- `False` - represents a false condition

Boolean values are named after George Boole, a mathematician who developed Boolean algebra. They are essential for:

- Making decisions in programs (if statements)
- Controlling program flow (while loops)
- Representing binary states (on/off, yes/no, exists/doesn't exist)
- Logical operations and comparisons

2. Why Boolean Values are Important

Boolean values enable your programs to:

- Make decisions based on conditions
- Control the execution flow of your code
- Represent binary states clearly and efficiently
- Work with logical operations
- Validate data and user input

3. Boolean Literals

In Python, boolean values are written as:

- `True` (with capital T)
- `False` (with capital F)

Note: These are **keywords** in Python, so they must be capitalized exactly as shown.

4. Creating and Using Boolean Values

In [1]:
```
# Creating and Using Boolean Values

print("=== BASIC BOOLEAN VALUES ===")

# Direct assignment of boolean literals
is_sunny = True
is_raining = False
```

```python
game_over = True
user_logged_in = False

print(f"is_sunny = {is_sunny}")
print(f"is_raining = {is_raining}")
print(f"game_over = {game_over}")
print(f"user_logged_in = {user_logged_in}")

# Check the type of boolean values
print(f"\nType of True: {type(True)}")
print(f"Type of False: {type(False)}")
print(f"Type of is_sunny: {type(is_sunny)}")

print("\n" + "="*50)

# Boolean values from comparison operations
print("\n=== BOOLEANS FROM COMPARISONS ===")

age = 25
temperature = 30
score = 85

is_adult = age >= 18
is_hot = temperature > 25
passed_exam = score >= 70
is_perfect_score = score == 100

print(f"age = {age}")
print(f"is_adult (age >= 18): {is_adult}")
print(f"temperature = {temperature}")
print(f"is_hot (temp > 25): {is_hot}")
print(f"score = {score}")
print(f"passed_exam (score >= 70): {passed_exam}")
print(f"is_perfect_score (score == 100): {is_perfect_score}")

print("\n" + "="*50)

# Boolean values in expressions
print("\n=== BOOLEAN EXPRESSIONS ===")

x = 10
y = 5

print(f"x = {x}, y = {y}")
print(f"x > y: {x > y}")
print(f"x == y: {x == y}")
print(f"x != y: {x != y}")
print(f"x <= y: {x <= y}")

# String comparisons
name1 = "Alice"
name2 = "Bob"
name3 = "Alice"

print(f"\nString comparisons:")
print(f"'{name1}' == '{name2}': {name1 == name2}")
print(f"'{name1}' == '{name3}': {name1 == name3}")
print(f"'{name1}' < '{name2}': {name1 < name2}")  # Alphabetical order
```

```
=== BASIC BOOLEAN VALUES ===
is_sunny = True
is_raining = False
game_over = True
user_logged_in = False

Type of True: <class 'bool'>
Type of False: <class 'bool'>
Type of is_sunny: <class 'bool'>

==================================================

=== BOOLEANS FROM COMPARISONS ===
age = 25
is_adult (age >= 18): True
temperature = 30
is_hot (temp > 25): True
score = 85
passed_exam (score >= 70): True
is_perfect_score (score == 100): False

==================================================

=== BOOLEAN EXPRESSIONS ===
x = 10, y = 5
x > y: True
x == y: False
x != y: True
x <= y: False

String comparisons:
'Alice' == 'Bob': False
'Alice' == 'Alice': True
'Alice' < 'Bob': True
```

5. Truthy and Falsy Values

In Python, all values have an inherent boolean "truthiness". When used in a boolean context, some values are considered "truthy" (equivalent to True) and others are "falsy" (equivalent to False).

In [2]:
```python
# Truthy and Falsy Values

print("=== FALSY VALUES (equivalent to False) ===")
print("These values are considered False in boolean context:")

# The main falsy values in Python
falsy_values = [
    False,      # Boolean False
    0,          # Zero (integer)
    0.0,        # Zero (float)
    0j,         # Zero (complex)
    "",         # Empty string
    [],         # Empty list
    (),         # Empty tuple
    {},         # Empty dictionary
    set(),      # Empty set
    None        # None value
]

for value in falsy_values:
    print(f"bool({repr(value):>12}) = {bool(value)}")

print("\n" + "="*60)

print("\n=== TRUTHY VALUES (equivalent to True) ===")
print("All other values are considered True in boolean context:")

# Examples of truthy values
truthy_values = [
    True,           # Boolean True
    1,              # Non-zero integer
    -1,             # Negative integer
    3.14,           # Non-zero float
    "hello",        # Non-empty string
    " ",            # String with space
    [1, 2, 3],      # Non-empty list
    (1, 2),         # Non-empty tuple
    {"key": "value"}, # Non-empty dictionary
    {1, 2, 3}       # Non-empty set
]

for value in truthy_values:
    print(f"bool({repr(value):>15}) = {bool(value)}")

print("\n" + "="*60)

# Practical examples using truthy/falsy values
print("\n=== PRACTICAL APPLICATIONS ===")

# Check if a string is not empty
user_input = input("Enter something (or press Enter for empty): ")
has_input = bool(user_input)
print(f"User provided input: {has_input}")

# Check if a list has items
shopping_list = ["apples", "bread"]
has_items = bool(shopping_list)
print(f"Shopping list has items: {has_items}")

# Using truthy/falsy in conditions (preview of if statements)
name = "Alice"
if name:  # Truthy check - name is not empty
    print(f"Hello, {name}!")
else:
    print("No name provided")

score = 0
if score:  # Falsy check - score is 0 (falsy)
    print(f"Your score is {score}")
else:
    print("No score recorded")
```

```
=== FALSY VALUES (equivalent to False) ===
These values are considered False in boolean context:
bool(       False) = False
bool(           0) = False
bool(         0.0) = False
bool(          0j) = False
bool(          '') = False
bool(          []) = False
bool(          ()) = False
bool(          {}) = False
bool(       set()) = False
bool(        None) = False

============================================================

=== TRUTHY VALUES (equivalent to True) ===
All other values are considered True in boolean context:
bool(           True) = True
bool(              1) = True
bool(             -1) = True
bool(           3.14) = True
bool(        'hello') = True
bool(            ' ') = True
bool(      [1, 2, 3]) = True
bool(         (1, 2)) = True
bool({'key': 'value'}) = True
bool(      {1, 2, 3}) = True

============================================================

=== PRACTICAL APPLICATIONS ===
User provided input: False
Shopping list has items: True
Hello, Alice!
No score recorded
User provided input: False
Shopping list has items: True
Hello, Alice!
No score recorded
```

6. Boolean Operations

Boolean operations allow you to combine, modify, and work with boolean values using logical operators.

In [3]:
```python
# Boolean Operations with Logical Operators

print("=== LOGICAL OPERATORS WITH BOOLEANS ===")

# Sample boolean variables
is_sunny = True
is_warm = True
is_weekend = False
has_money = True

print(f"is_sunny = {is_sunny}")
print(f"is_warm = {is_warm}")
print(f"is_weekend = {is_weekend}")
print(f"has_money = {has_money}")
print("-" * 50)

# AND operations
print("AND Operations (all conditions must be True):")
good_weather = is_sunny and is_warm
perfect_day = is_sunny and is_warm and is_weekend
can_go_shopping = has_money and is_weekend

print(f"good_weather (sunny AND warm): {good_weather}")
print(f"perfect_day (sunny AND warm AND weekend): {perfect_day}")
print(f"can_go_shopping (has_money AND weekend): {can_go_shopping}")

# OR operations
print("\nOR Operations (at least one condition must be True):")
nice_day = is_sunny or is_weekend
can_relax = is_weekend or (not is_sunny)
outdoor_activity = is_sunny or is_warm

print(f"nice_day (sunny OR weekend): {nice_day}")
print(f"can_relax (weekend OR not sunny): {can_relax}")
print(f"outdoor_activity (sunny OR warm): {outdoor_activity}")

# NOT operations
print("\nNOT Operations (opposite of the boolean value):")
is_cloudy = not is_sunny
is_weekday = not is_weekend
is_broke = not has_money

print(f"is_cloudy (NOT sunny): {is_cloudy}")
print(f"is_weekday (NOT weekend): {is_weekday}")
print(f"is_broke (NOT has_money): {is_broke}")

print("\n" + "="*60)

# Complex boolean expressions
print("\n=== COMPLEX BOOLEAN EXPRESSIONS ===")
age = 25
has_license = True
has_car = False
is_experienced = age >= 21

can_drive_alone = has_license and (has_car or age >= 18)
needs_supervision = not has_license or (age < 18 and not is_experienced)
eligible_for_rental = has_license and age >= 21 and has_money

print(f"age = {age}, has_license = {has_license}, has_car = {has_car}")
print(f"can_drive_alone: {can_drive_alone}")
print(f"needs_supervision: {needs_supervision}")
print(f"eligible_for_rental: {eligible_for_rental}")

print("\n" + "="*60)

# Short-circuit evaluation
print("\n=== SHORT-CIRCUIT EVALUATION ===")
print("Python evaluates boolean expressions efficiently:")

# AND short-circuit: if first is False, second is not evaluated
result1 = False and print("This won't print")  # print() is not executed
print(f"False and [expression]: {result1}")
```

```python
# OR short-circuit: if first is True, second is not evaluated
result2 = True or print("This won't print")   # print() is not executed
print(f"True or [expression]: {result2}")

# When second expression IS evaluated
result3 = True and print("This WILL print")   # print() is executed, returns None
print(f"True and [expression]: {result3}")

result4 = False or print("This WILL print")   # print() is executed, returns None
print(f"False or [expression]: {result4}")
```

```
=== LOGICAL OPERATORS WITH BOOLEANS ===
is_sunny = True
is_warm = True
is_weekend = False
has_money = True
--------------------------------------------------
AND Operations (all conditions must be True):
good_weather (sunny AND warm): True
perfect_day (sunny AND warm AND weekend): False
can_go_shopping (has_money AND weekend): False

OR Operations (at least one condition must be True):
nice_day (sunny OR weekend): True
can_relax (weekend OR not sunny): False
outdoor_activity (sunny OR warm): True

NOT Operations (opposite of the boolean value):
is_cloudy (NOT sunny): False
is_weekday (NOT weekend): True
is_broke (NOT has_money): False

============================================================

=== COMPLEX BOOLEAN EXPRESSIONS ===
age = 25, has_license = True, has_car = False
can_drive_alone: True
needs_supervision: False
eligible_for_rental: True

============================================================

=== SHORT-CIRCUIT EVALUATION ===
Python evaluates boolean expressions efficiently:
False and [expression]: False
True or [expression]: True
This WILL print
True and [expression]: None
This WILL print
False or [expression]: None
```

Key Takeaways

- **Boolean values** are either `True` or `False` (case-sensitive)
- **Comparison operations** return boolean values automatically
- **Falsy values**: `False`, `0`, `0.0`, `""`, `[]`, `()`, `{}`, `set()`, `None`

- **Truthy values**: Everything else (non-zero numbers, non-empty strings, non-empty collections)
- **Logical operators**: `and` , `or` , `not` work with boolean values
- **Short-circuit evaluation**: Python stops evaluating as soon as the result is determined
- **bool()** function converts any value to its boolean equivalent

Practice Exercises

Try these exercises to strengthen your understanding:

1. **Boolean Predicates**: Create boolean variables to represent different conditions (is_student, is_employed, etc.)
2. **Truthiness Test**: Test various values with `bool()` to see their truthiness
3. **Complex Conditions**: Write expressions combining multiple boolean conditions with `and` , `or` , `not`
4. **Validation Logic**: Create boolean expressions to validate user input or data
5. **Decision Trees**: Design logical expressions for decision-making scenarios

Common Use Cases

- **User Authentication**: `is_logged_in and has_permission`
- **Data Validation**: `not username or len(password) < 8`
- **Game Logic**: `has_key and door_is_locked`
- **Conditional Processing**: `is_weekend or is_holiday`
- **Error Checking**: `file_exists and is_readable`

Course Information

Learn Python Programming from Scratch
Author: Prakash Ukhalkar
Topic: Python Fundamentals - Boolean Values and Operations

Built with ❤️ for the Python learning community