Conditional statements and operators

INTRODUCTION TO PYTHON FOR DEVELOPERS

Jasmin Ludolf
Senior Data Science Content Developer

Booleans

```python
# Boolean variable
the_truth = True
print(the_truth)
```

```
True
```

- Used to make comparisons

Operators

- Comparison operators
  - Symbols or combinations of symbols
  - Used to compare values
- Check if two things are equal
  - `==`

Checking for equality

```
# Compare if 2 is equal to 3
2 == 3
```

```
False
```

```
# Check that 2 is not equal to 3
2 != 3
```

```
True
```

- Common use-case: checking login details

Numeric comparison operators

```
# Is 5 less than 7?
5 < 7
```

```
True
```

```
# Is 5 greater than 7?
5 > 7
```

```
False
```

```
# Is 5 less than or equal to 7?
5 <= 7
```

```
True
```

```
# Is 5 greater or equal to 7?
5 >= 7
```

```
False
```

Other comparisons

```python
# Is James greater than Brian
"James" > "Brian"
```

```
True
```

- Strings are evaluated in alphabetical order

Conditional statements

- `if` condition is met, then perform action, otherwise skip

```
# Check pasta quantities
required_quantity = 500
pasta_quantity = 200

# Compare pasta quantities
if pasta_quantity >= required_quantity
```

Conditional statements

- `if` condition is met, then perform action, otherwise skip

```python
# Check pasta quantities
required_quantity = 500
pasta_quantity = 200

# Compare pasta quantities
if pasta_quantity >= required_quantity:
```

Conditional statements

- `if` condition is met, then perform action, otherwise skip

```
# Check pasta quantities
required_quantity = 500
pasta_quantity = 200

# Compare pasta quantities
if pasta_quantity >= required_quantity:
    print("You have enough pasta!")
```

Indentation

```python
# Check pasta quantities
required_quantity = 500
pasta_quantity = 200

# Compare pasta quantities
if pasta_quantity >= required_quantity:
print("You have enough pasta!") # This line is not indented
```

```
    print("You have enough pasta!")
    ^
IndentationError: expected an indented block
```

Elif statement

```python
# Check pasta quantities
required_quantity = 500
pasta_quantity = 200
# Compare pasta quantities
if pasta_quantity >= required_quantity:
    print("You have enough pasta!")
elif pasta_quantity >= 300:
    print("Nearly enough pasta. Try a smaller portion.")
```

- Can use as many `elif` keywords as we like!

Else statement

```python
# Check pasta quantities
required_quantity = 500
pasta_quantity = 200
# Compare pasta quantities
if pasta_quantity >= required_quantity:
    print("You have enough pasta!")
elif pasta_quantity >= 300:
    print("Nearly enough pasta. Try a smaller portion.")
# Otherwise...
else:
    print("Not enough pasta.")
```

```
Not enough pasta.
```

Comparison operators cheat sheet

Operator	Function
`==`	Equal to
`!=`	Not equal to
`>`	More than
`>=`	More than or equal to
`<`	Less than
`<=`	Less than or equal to

Keyword	Function	Use
`if`	If condition is met	First in the workflow
`elif`	Else check if condition is met	After `if`
`else`	Else perform this action	After `elif`

Let's practice!

INTRODUCTION TO PYTHON FOR DEVELOPERS

For loops

INTRODUCTION TO PYTHON FOR DEVELOPERS

Jasmin Ludolf
Senior Data Science Content Developer

Individual comparisons

```python
# Ingredient quantities
quantities = [500, 400, 15, 20, 30, 5]
```

```python
# Validate values
quantities[0] < 10
```

```
False
```

```python
quantities[1] < 10
```

```
False
```

For loop syntax

```python
for value in sequence:
    action
```

- `for` each `value` in `sequence`, perform this `action`
  - `action` is indented because of the colon in the previous line
- `sequence` = iterable e.g., list, dictionary, etc.
- `value` = iterator, i.e., the index
  - Placeholder (can give it any name), `i` is common

Print individual values

```python
# Ingredients list
ingredients = ["pasta", "tomatoes", "garlic", "basil", "olive oil", "salt"]

# Loop through and print each ingredient
for ingredient in ingredients:
    print(ingredient)
```

```
pasta
tomatoes
garlic
basil
olive oil
salt
```

Conditional statements in for loops

```python
quantities = [1000, 800, 40, 30, 30, 15]


for qty in quantities:
```

Conditional statements in for loops

```
quantities = [1000, 800, 40, 30, 30, 15]

for qty in quantities:
    # Check if quantity is more than 500
    if qty > 500:
      print("Plenty in stock")
    elif qty >= 100:
        print("Enough for a small portion")
    else:
        print("Nearly out!")
```

Conditional statements in for loops

```
Plenty in stock
Enough for small
Nearly out!
Nearly out!
Nearly out!
```

Looping through strings

```python
ingredient_name = "pasta"
# Loop through each character
for letter in ingredient_name:
    print(letter)
```

```
p
a
s
t
a
```

- Useful for validating text, checking for special characters

Looping through dictionaries

```python
ingredients = {"pasta": 500, "tomatoes": 400, "garlic": 30}

# Loop through keys and values
for item, qty in ingredients.items():
    print(item, ":", qty, "grams")
```

```
pasta : 500 grams
tomatoes : 400 grams
garlic : 30 grams
```

- `item` = key (ingredient name)
- `qty` = value (quantity)

Looping through dictionaries

```python
ingredients = {"pasta": 500, "tomatoes": 400, "garlic": 30}
factor = 2
# Calculate scaled quantities
for item, qty in ingredients.items():
    scaled_qty = qty * factor
    print(item, ":", scaled_qty, "grams")
```

```
pasta : 1000 grams
tomatoes : 800 grams
garlic : 60 grams
```

Looping through dictionaries

```
# Loop through keys only
for item in ingredients.keys():
    print(item)
```

```
pasta
tomatoes
garlic
```

```
# Loop through values only
for qty in ingredients.values():
    print(qty, "grams")
```

```
500 grams
400 grams
30 grams
```

Range

```
range(start, end + 1)
```

- `start` = starting number
- `end` = ending number
- Used to generate or modify values

```
for i in range(1, 6):
    print(i)
```

```
1
2
3
4
5
```

Let's practice!

INTRODUCTION TO PYTHON FOR DEVELOPERS

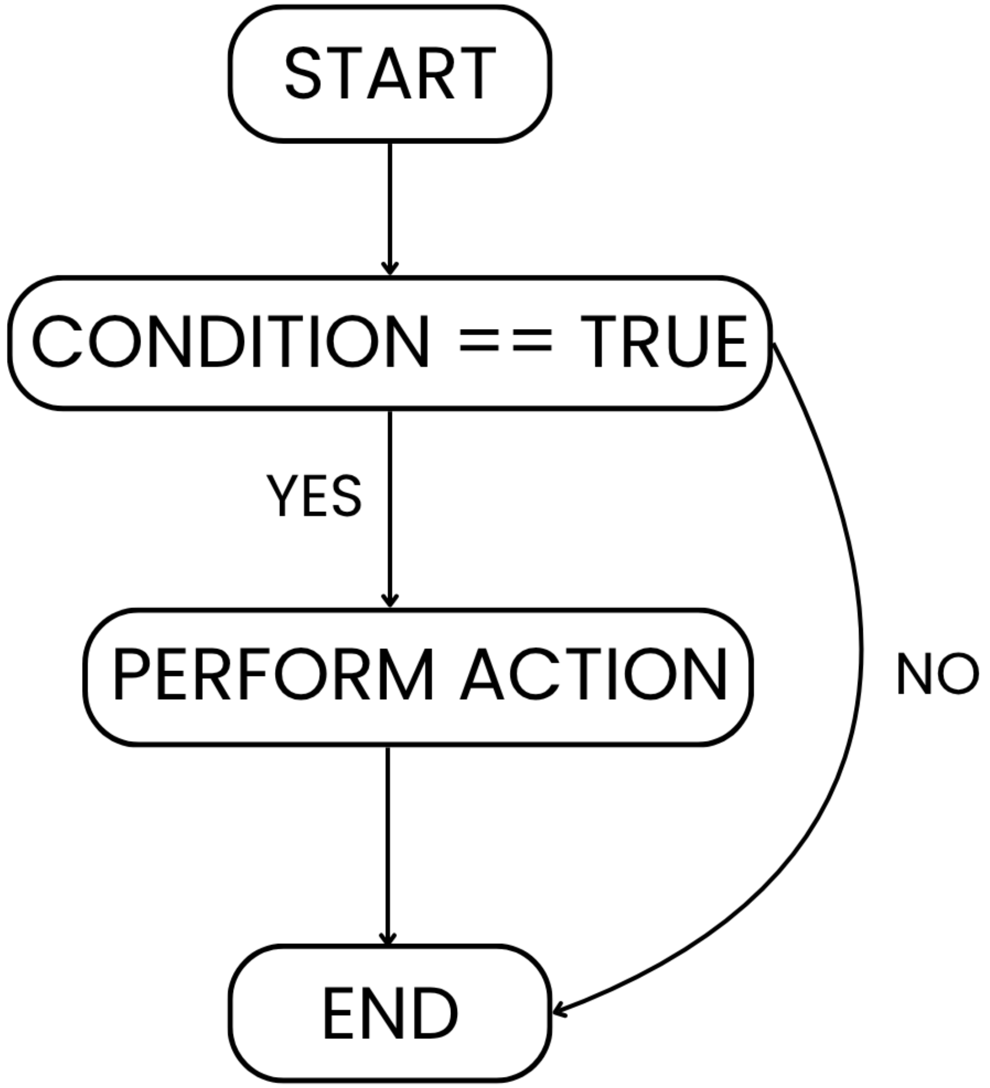
While loops

INTRODUCTION TO PYTHON FOR DEVELOPERS

Jasmin Ludolf
Senior Data Science Content Developer

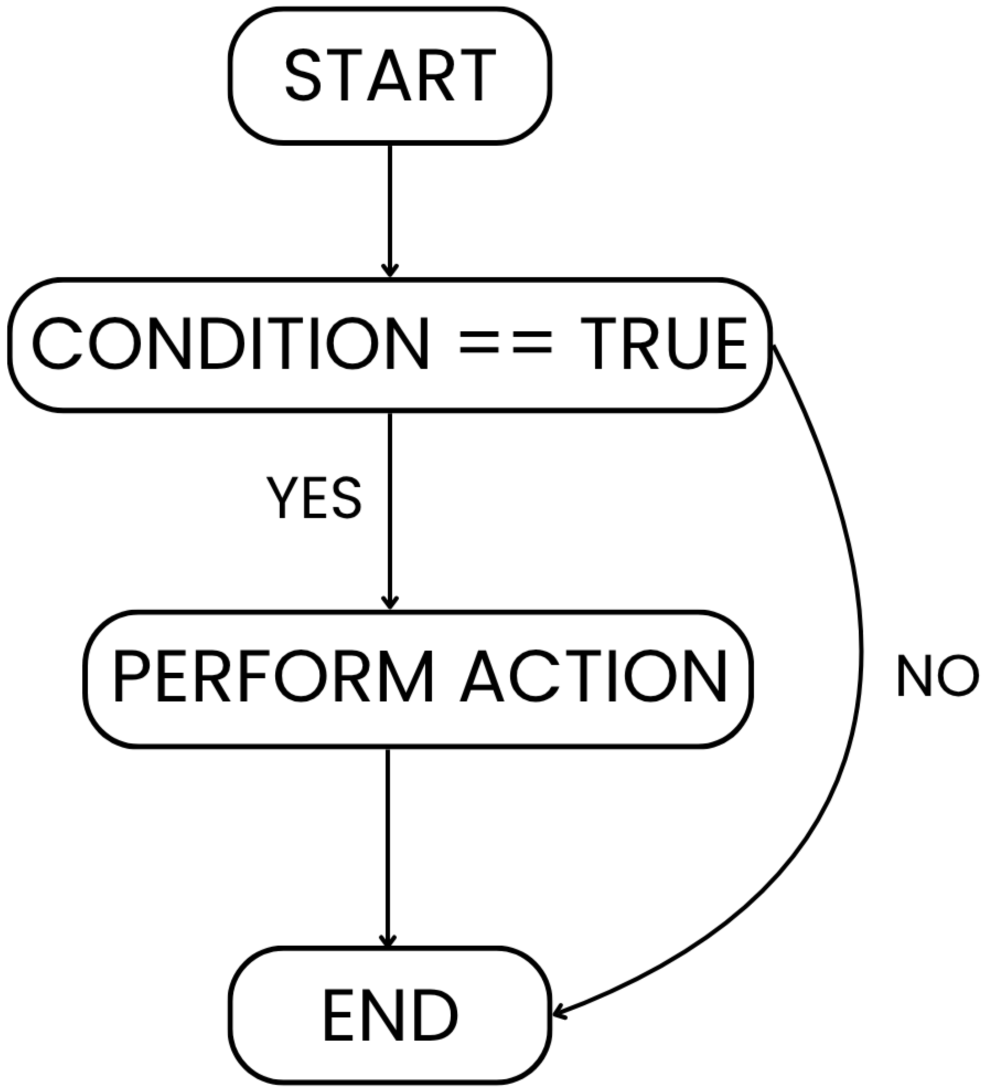
If statement

If statement

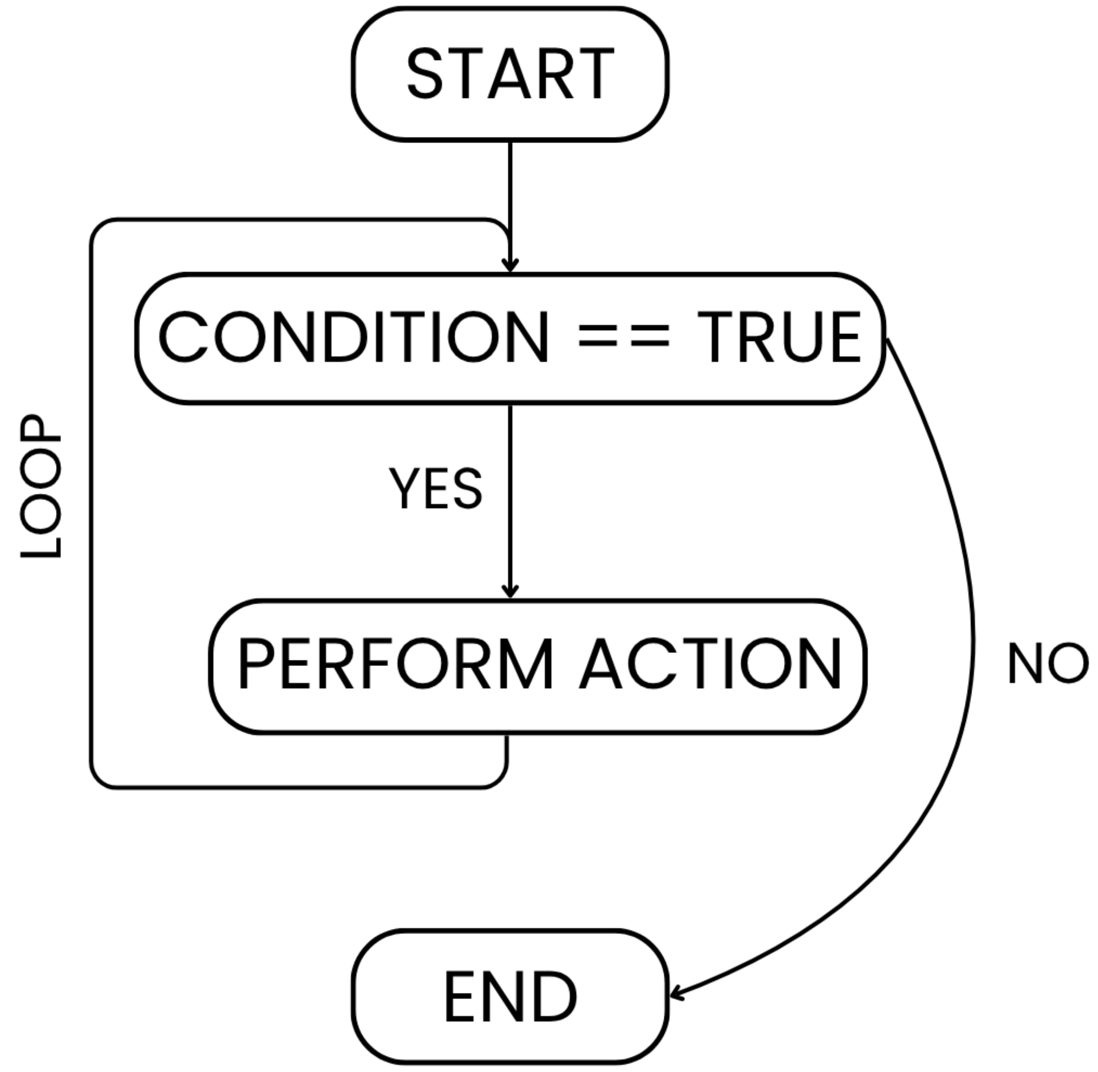


If statement versus while loop

If statement



While loop



While loop

```
while condition:
    action
```

- Any continuous task
    - Accelerate `while` a button is pressed

[1] https://unsplash.com/@joaoscferrao

While loop

```python
ingredients_to_add = 5
items_added = 0
```

```python
# Keep adding while we have items left
while items_added < ingredients_to_add:
    items_added += 1
    remaining = ingredients_to_add - items_added
    print(remaining, "ingredients left to add")
```

Output

```
4 ingredients left to add
3 ingredients left to add
2 ingredients left to add
1 ingredients left to add
0 ingredients left to add
```

- Loop exits when `items_added` equals `ingredients_to_add`

A word of caution

- `while` runs continually while the condition is met

```
ingredients_to_add = 5
items_added = 0

while items_added < ingredients_to_add:
    remaining = ingredients_to_add - items_added
    print(remaining, "ingredients left")
```

Running forever

```python
ingredients_to_add = 5
items_added = 0

# INFINITE LOOP - never exits!
while items_added < ingredients_to_add:
    remaining = ingredients_to_add - items_added
    print(remaining, "ingredients left")
    # Forgot to increment items_added!
```

- Condition never becomes False
- Loop runs forever, program freezes
- Common developer mistake

Breaking a loop

```
while items_added < ingredients_to_add:
    remaining = ingredients_to_add - items_added
    print(remaining, "ingredients left")

    # Terminate the loop
    break
```

- `break` can also be used in `for` loops
- If the code is already running: Control + C / Command + C

Conditional statements within while loops

```python
ingredients_to_add = 5
items_added = 0

while items_added < ingredients_to_add:
    items_added += 1
    remaining = ingredients_to_add - items_added
    if remaining > 3:
        print("Several ingredients remaining")
    elif remaining >= 1:
        print("Almost done!")
    else:
        print("Shopping list complete!")
```

Conditional statements output

```
Several ingredients remaining
Several ingredients remaining
Almost done!
Almost done!
Shopping list complete!
```

Let's practice!

INTRODUCTION TO PYTHON FOR DEVELOPERS

Building a workflow

INTRODUCTION TO PYTHON FOR DEVELOPERS

Jasmin Ludolf
Senior Data Science Content Developer

Complex workflows

- Loops through data structures
  - `for` , `while`
- Evaluate multiple conditions
  - `if` , `elif` , `else` , `>` , `>=` , `<` , `<=` , `==` , `!=`
- Update variables
  - `+=`
- Return outputs
  - `print()`

The "in" keyword

- `in` = check if a value is **in** a variable/data structure

```
recipe = {"pasta": 500, "tomatoes": 400,
          "garlic": 15, "basil": 20}

if "pasta" in recipe.keys():
    print(True)
else:
    print(False)
```

```
True
```

- Faster than looping through every key

The "not" keyword

- `not` = check if a condition **is not** met
- Useful for validating that something is missing

```python
pantry_items = ["flour", "sugar", "olive oil"]
# Check if "salt" is NOT in our pantry
if "salt" not in pantry_items:
    print(True)
else:
    print(False)
```

```
True
```

The "and" keyword

- `and` = check if **multiple conditions** are met
- Use when multiple requirements must be met

```python
pasta_quantity = 600
olive_oil_quantity = 30
# Check if we have enough of BOTH ingredients
if pasta_quantity >= 500 and olive_oil_quantity >= 30:
    print(True)
else:
    print(False)
```

```
True
```

The "or" keyword

- `or` = check if one (**or more**) condition is met
- Use when any of several options are acceptable

```python
pasta_quantity = 600
olive_oil_quantity = 30
# Check if we have enough of EITHER ingredient
if pasta_quantity >= 500 or olive_oil_quantity >= 30:
    print(True)
else:
    print(False)
```

```
True
```

Adding/subtracting from variables

- Combine keywords with other techniques to build complex workflows

```python
ingredients_checked = 0
for ingredient in recipe_list:
    # ingredients_checked = ingredients_checked + 1
    ingredients_checked += 1
items_to_buy = 10
for item in shopping_list:
    # items_to_buy = items_to_buy - 1
    items_to_buy -= 1
```

- `+=` adds to a variable, `-=` subtracts from it
- Other ways to update variables

Appending

- Store information that meets specific criteria in a list

```
# Create empty list to hold results
shopping_list = []

# Loop through recipe ingredients
for ingredient, qty_needed in recipe.items():
    # Check if we need to buy it
    if ingredient not in pantry:
        # Add to shopping list
        shopping_list.append(ingredient)
```

Appending

```
print(shopping_list)
```

```
['tomatoes', 'salt']
```

Let's practice!

INTRODUCTION TO PYTHON FOR DEVELOPERS

Congratulations!

INTRODUCTION TO PYTHON FOR DEVELOPERS

Jasmin Ludolf
Senior Data Science Content Developer

Chapter 1 recap

```python
# Define quantity
quantity = 10

# Single quotes
ingredient_name = 'pasta'

# Double quotes also work
ingredient_name = "pasta"
```

Chapter 2 recap

```python
ingredient_name = "Basil Leaves"

# Convert to lowercase
ingredient_name = ingredient_name.lower()
```

```python
# List of ingredients
ingredients = ["pasta", "tomatoes", "garlic",
               "basil", "olive oil", "salt"]
# Get the value at the first index
ingredients[0]
```

```python
# Creating a dictionary
recipe_dict = {"pasta": 500,
               "tomatoes": 400,
               "garlic": 15,
               "basil": 20,
               "olive oil": 30,
               "salt": 5}
```

```python
# Create a set
ingredients_set = {"pasta", "tomatoes", "pasta",
                   "basil", "garlic", "olive oil",
                   "salt"}
# Create a tuple
serving_sizes = (1, 2, 4, 6, 8)
```

Chapter 3 recap

Operator	Function
`==`	Equal to
`!=`	Not equal to
`>`	More than
`>=`	More than or equal to
`<`	Less than
`<=`	Less than or equal to

Keyword	Function	Use
`if`	If condition is met	First in the workflow
`elif`	Else check if condition is met	After `if`
`else`	Else perform this action	After `elif`

Chapter 3 recap

```python
# Ingredients list
ingredients = ["pasta", "tomatoes", "garlic", "basil", "olive oil", "salt"]

# Loop through and print each ingredient
for ingredient in ingredients:
    print(ingredient)
```

```
pasta
tomatoes
garlic
basil
olive oil
salt
```

Chapter 3 recap

```python
ingredients_to_add = 5
items_added = 0

while items_added < ingredients_to_add:
    items_added += 1
    remaining = ingredients_to_add - items_added

    if remaining > 3:
        print("Several ingredients remaining")
    elif remaining >= 1:
        print("Almost done!")
    else:
        print("Shopping list complete!")
```

Chapter 3 recap

Keyword	Function
`and`	Evaluate if multiple conditions are true
`or`	Evaluate one or more conditions are true
`in`	Evaluate if a value is in a data structure
`not`	Evaluate if a value is not in a data structure

- `list.append()`

Next steps

- Additional built-in functions
  - `zip()`
  - `enumerate()`
- Packages and modules
  - `os`
  - `time`
  - `venv`
  - `pandas`
  - `requests`
  - `numpy`

- Building custom functions

- **Intermediate Python for Developers** course on DataCamp

Congratulations!

INTRODUCTION TO PYTHON FOR DEVELOPERS