# Python-learning-Day1

1. Comments in Python

Single-line comments: Use the # symbol. Everything after # on that line is ignored by the interpreter.

# This is a single-line comment

print("Hello, World!") # This is an inline comment

Multi-line comments: Technically, Python doesn’t have multi-line comment syntax, but you can use triple quotes (''' or """) as a workaround.

"""

This is a multi-line comment

that spans several lines.

"""

print("Multi-line comments above!")

3. Keywords in Python

Definition: Reserved words that have special meaning in Python. They are part of the syntax and cannot be used as identifiers (variable names).

Examples: if, else, elif, while, for, def, return, True, False, import, class, try, except, finally, with, as, break, continue, pass, global, lambda, nonlocal, yield, etc.

if True:

print("This is a keyword example.")

Note: You can view all keywords using the keyword module:

import keyword

print(keyword.kwlist)

3.1 Identifiers in Python

Definition: Names used to identify variables, functions, classes, modules, etc.

Rules:

It can contain letters (A–Z, a–z), digits (0–9), and underscores (\_).

It cannot start with a digit.

It cannot be a Python keyword.

Case-sensitive (Variable and variable are different).

name = "John" # valid identifier

\_age = 25 # valid identifier

1st\_number = 100 # invalid identifier (cannot start with a digit)

Here's an overview of the requested Python concepts:

1. Comments in Python

Single-line comments: Use the # symbol. Everything after # on that line is ignored by the interpreter.

python

Copy

Edit

# This is a single-line comment

print("Hello, World!") # This is an inline comment

Multi-line comments: Technically, Python doesn’t have multi-line comment syntax, but you can use triple quotes (''' or """) as a workaround.

python

Copy

Edit

"""

This is a multi-line comment

that spans several lines.

"""

print("Multi-line comments above!")

2. Keywords in Python

Definition: Reserved words that have special meaning in Python. They are part of the syntax and cannot be used as identifiers (variable names).

Examples: if, else, elif, while, for, def, return, True, False, import, class, try, except, finally, with, as, break, continue, pass, global, lambda, nonlocal, yield, etc.

python

Copy

Edit

if True:

print("This is a keyword example.")

Note: You can view all keywords using the keyword module:

python

Copy

Edit

import keyword

print(keyword.kwlist)

3. Identifiers in Python

Definition: Names used to identify variables, functions, classes, modules, etc.

Rules:

Can contain letters (A–Z, a–z), digits (0–9), and underscores (\_).

Cannot start with a digit.

Cannot be a Python keyword.

Case-sensitive (Variable and variable are different).

python

Copy

Edit

name = "John" # valid identifier

\_age = 25 # valid identifier

1st\_number = 100 # invalid identifier (cannot start with a digit)

4. Operators in Python

a. Arithmetic Operators

Used for mathematical operations:

+ (Addition)

- (Subtraction)

\* (Multiplication)

/ (Division)

// (Floor Division)

% (Modulus - remainder)

\*\* (Exponentiation)

a = 10

b = 3

print(a + b) # 13

print(a - b) # 7

print(a \* b) # 30

print(a / b) # 3.333...

print(a // b) # 3

print(a % b) # 1

print(a \*\* b) # 1000

b. Comparison Operators

Used to compare values, returning True or False:

== (Equal to)

!= (Not equal to)

> (Greater than)

< (Less than)

>= (Greater than or equal to)

<= (Less than or equal to)

x = 5

y = 10

print(x == y) # False

print(x != y) # True

print(x > y) # False

print(x < y) # True

print(x >= 5) # True

print(y <= 10) # True

Here's an overview of the requested Python concepts:

1. Comments in Python

Single-line comments: Use the # symbol. Everything after # on that line is ignored by the interpreter.

python

Copy

Edit

# This is a single-line comment

print("Hello, World!") # This is an inline comment

Multi-line comments: Technically, Python doesn’t have multi-line comment syntax, but you can use triple quotes (''' or """) as a workaround.

python

Copy

Edit

"""

This is a multi-line comment

that spans several lines.

"""

print("Multi-line comments above!")

2. Keywords in Python

Definition: Reserved words that have special meaning in Python. They are part of the syntax and cannot be used as identifiers (variable names).

Examples: if, else, elif, while, for, def, return, True, False, import, class, try, except, finally, with, as, break, continue, pass, global, lambda, nonlocal, yield, etc.

python

Copy

Edit

if True:

print("This is a keyword example.")

Note: You can view all keywords using the keyword module:

python

Copy

Edit

import keyword

print(keyword.kwlist)

3. Identifiers in Python

Definition: Names used to identify variables, functions, classes, modules, etc.

Rules:

Can contain letters (A–Z, a–z), digits (0–9), and underscores (\_).

Cannot start with a digit.

Cannot be a Python keyword.

Case-sensitive (Variable and variable are different).

python

Copy

Edit

name = "John" # valid identifier

\_age = 25 # valid identifier

1st\_number = 100 # invalid identifier (cannot start with a digit)

4. Operators in Python

a. Arithmetic Operators

Used for mathematical operations:

+ (Addition)

- (Subtraction)

\* (Multiplication)

/ (Division)

// (Floor Division)

% (Modulus - remainder)

\*\* (Exponentiation)

python

Copy

Edit

a = 10

b = 3

print(a + b) # 13

print(a - b) # 7

print(a \* b) # 30

print(a / b) # 3.333...

print(a // b) # 3

print(a % b) # 1

print(a \*\* b) # 1000

b. Comparison Operators

Used to compare values, returning True or False:

== (Equal to)

!= (Not equal to)

> (Greater than)

< (Less than)

>= (Greater than or equal to)

<= (Less than or equal to)

python

Copy

Edit

x = 5

y = 10

print(x == y) # False

print(x != y) # True

print(x > y) # False

print(x < y) # True

print(x >= 5) # True

print(y <= 10) # True

c. Assignment Operators

Used to assign values to variables:

= (Simple assignment)

+= (Add and assign)

-= (Subtract and assign)

\*= (Multiply and assign)

/= (Divide and assign)

//= (Floor divide and assign)

%= (Modulus and assign)

\*\*= (Exponentiate and assign)

a = 5 # Assign 5 to a

a += 3 # Equivalent to a = a + 3 (a becomes 8)

a \*= 2 # Equivalent to a = a \* 2 (a becomes 16)

a -= 4 # Equivalent to a = a - 4 (a becomes 12)

a /= 3 # Equivalent to a = a / 3 (a becomes 4.0)

# Python-learning-Day2

1. Logical Operators

Logical operators are used to combine conditional statements:

and= Returns True if both conditions are true (5 > 3) and (4 < 6) -> True

or= Returns True if at least one condition is true (5 < 3) or (4 < 6) -> True

not= Reverses the result (negation) not(5 > 3) -> False

code(syntax):-

x = 10

y = 5

# Logical AND

print(x > 5 and y < 10) # True

# Logical OR

print(x < 5 or y < 10) # True

# Logical NOT

print(not(x == 10)) # False

2. Membership Operators

Membership operators test for membership in sequences like lists, tuples, strings, etc.

in= Returns True if a value exists in the sequence 'a' in 'apple' -> True

not in= Returns True if a value does NOT exist in the sequence 3 not in [1, 2, 4] -> True

code(syntax):-

fruits = ["apple", "banana", "cherry"]

# Membership Test

print("banana" in fruits) # True

print("grape" not in fruits) # True

3. Identity Operators

Identity operators compare the memory location of two objects (whether they are the same object in memory).

is= Returns True if both variables point to the same object a is b

is not= Returns True if both variables point to different objects a is not b

code(syntax):-

a = [1, 2, 3]

b = a

c = [1, 2, 3]

print(a is b) # True (same object)

print(a is c) # False (different objects with same content)

print(a is not c) # True

Here's a breakdown of logical operators, membership operators, identity operators, and conditional statements (if, else, elif) in Python:

1. Logical Operators

Logical operators are used to combine conditional statements:

Operator Description Example

and Returns True if both conditions are true (5 > 3) and (4 < 6) → True

or Returns True if at least one condition is true (5 < 3) or (4 < 6) → True

not Reverses the result (negation) not(5 > 3) → False

Example:

python

Copy

Edit

x = 10

y = 5

# Logical AND

print(x > 5 and y < 10) # True

# Logical OR

print(x < 5 or y < 10) # True

# Logical NOT

print(not(x == 10)) # False

2. Membership Operators

Membership operators test for membership in sequences like lists, tuples, strings, etc.

Operator Description Example

in Returns True if a value exists in the sequence 'a' in 'apple' → True

not in Returns True if a value does NOT exist in the sequence 3 not in [1, 2, 4] → True

Example:

python

Copy

Edit

fruits = ["apple", "banana", "cherry"]

# Membership Test

print("banana" in fruits) # True

print("grape" not in fruits) # True

3. Identity Operators

Identity operators compare the memory location of two objects (whether they are the same object in memory).

Operator Description Example

is Returns True if both variables point to the same object a is b

is not Returns True if both variables point to different objects a is not b

Example:

python

Copy

Edit

a = [1, 2, 3]

b = a

c = [1, 2, 3]

print(a is b) # True (same object)

print(a is c) # False (different objects with same content)

print(a is not c) # True

4. Conditional Statements (if, elif, else)

Conditional statements allow you to execute code based on specific conditions.

Syntax:-

if condition:

# Code block (executes if condition is True)

elif another\_condition:

# Executes if 'if' was False, but this is True

else:

# Executes if all previous conditions were False

code:-

age = 18

if age < 18:

print("You are a minor.")

elif age == 18:

print("You just became an adult!")

else:

print("You are an adult.")

# Python-learning-Day3

1. Nested if-else Statements

Nested if-else statements are if statements inside another if or else block. They help handle multiple conditions.

Code:-

x = 15

if x > 10:

if x < 20:

print("x is between 10 and 20")

else:

print("x is greater than or equal to 20")

else:

print("x is 10 or less")

2. String Formatting

String formatting allows you to insert variables into strings.

Using f-strings (Python 3.6+):

name = "Alice"

age = 25

print(f"My name is {name} and I am {age} years old.")

Using format() method:

print("My name is {} and I am {} years old.".format(name, age))

Using % operator (older style):

print("My name is %s and I am %d years old." % (name, age))

1. String Indexing

In Python, strings are sequences of characters, and each character has a position (index).

Positive Indexing: Starts from 0 (left to right).

Negative Indexing: Starts from -1 (right to left).

Code:-

city="Rajasthan"

city[0]