OPERATORS

What are Operators?

Operators are special symbols in Python that perform operations on values and variables[1]. Think of them as the action words of programming. For example, the `+` operator performs addition.

Python
```
# The '+' operator adds two numbers
result = 10 + 5
print(result) # Output: 15 [cite: 14]
```

Types of Operators

Python has several types of operators for different tasks.

1. Arithmetic Operators

These are used for standard mathematical calculations.

- `+` (Addition)
- `-` (Subtraction)
- `*` (Multiplication)
- `/` (Division): **Note:** This always results in a floating-point number (e.g., `10 / 2` is `5.0`).
- `%` (Modulus): Gives the remainder of a division.
- `**` (Exponentiation): Raises a number to a power.
- `//` (Floor Division): Divides and rounds down to the nearest whole number.

Python
```
x = 10
y = 3

print(x + y)  # Output: 13 [cite: 47]
print(x / y)  # Output: 3.333... [cite: 50]
print(x % y)  # Output: 1 [cite: 51]
print(x // y) # Output: 3 [cite: 53]
```

2. Comparison Operators

These compare two values and return a Boolean result (`True` or `False`).

- `==` (Equal to)
- `!=` (Not equal to)
- `>` (Greater than)
- `<` (Less than)
- `>=` (Greater than or equal to)
- `<=` (Less than or equal to)

Python
```
x = 5
y = 10

print(x == y) # Output: False [cite: 87]
print(x < y)  # Output: True [cite: 90]
```

3. Logical Operators

These are used to combine conditional statements.

- `and`: Returns `True` if both statements are true.
- `or`: Returns `True` if at least one statement is true.
- `not`: Reverses the result, returning `False` if the result is true, and vice versa.

Python
```
x = 5
y = 10
z = 15

print(x < y and y < z) # Output: True, because both are true [cite: 109, 112]
print(not(x > y))      # Output: True, because x > y is False [cite: 111, 114]
```

4. Assignment Operators

These are used to assign values to variables. They often provide a shorthand way to perform an operation and assign the result.

Python
```
x = 10
x += 5  # This is the same as x = x + 5 [cite: 140]
print(x) # Output: 15

y = 20
y -= 2 # This is the same as y = y - 2 [cite: 140]
print(y) # Output: 18
```

5. Membership Operators

These test if a sequence is present in an object, like a string or a list.

- `in`: Returns `True` if a value is found in the sequence.
- `not in`: Returns `True` if a value is **not** found in the sequence.

Python
```
text = "hello world"

print("world" in text)     # Output: True [cite: 176]
print("python" not in text) # Output: True [cite: 177]
```

6. Identity Operators

These compare the memory locations of two objects to see if they are the exact same object.

- `is`: Returns `True` if both variables point to the same object.
- `is not`: Returns `True` if they point to different objects.

Python
```
x = [1, 2, 3]
y = x        # y now points to the same object as x
z = [1, 2, 3] # z is a new object with the same content

print(x is y) # Output: True
print(x is z) # Output: False
```

Order of Operations (Precedence)

Python follows the standard mathematical order of operations, commonly remembered by the acronym **PEMDAS** (or BODMAS)[19]. The order of precedence from highest to lowest is:

1. **P**arentheses `()`.

2. **E**xponents **.
3. **M**ultiplication *, **D**ivision /, Floor Division //, Modulus % (evaluated left-to-right).
4. **A**ddition + and **S**ubtraction - (evaluated left-to-right).
5. Comparisons, Identity, and Membership operators (>, is, in, etc.).
6. Logical operators (not, then and, then or).

Python
```
# (2 + 3) is 5. Then 5 ** 2 is 25. Then 25 * 4 is 100.
result = (2 + 3) ** 2 * 4
print(result) # Output: 100
```