Week 4 Asynchronous Materials Guide

Math Operations and Operators

This Week's Topics Overview

Building on Week 3's string operations, you'll now learn to:

- Use all arithmetic operators: +, -, *, /, //, %, **
- Understand operator precedence (order of operations)
- Use comparison operators: ==, !=, <, >, <=, >=
- Work with assignment operators: =, +=, -=, *=, /=
- Create and evaluate boolean expressions
- Combine operators in complex expressions

Start Here: Video Tutorial

Python Operators Tutorial

Python Tutorial for Beginners 4: Operators and Expressions

- **Duration:** 11 minutes
- **What you will learn:** All types of operators and how to use them
- **Why this helps:** Visual examples of operator precedence and comparison
- **Study approach:** Pause and try each example yourself

Arithmetic Operators

W3Schools Python Operators

https://www.w3schools.com/python/python_operators.asp

- Complete guide to all operator types
- Examples you can test immediately

- Clear explanation of each operator's purpose

LearnPython.org Basic Operators

https://www.learnpython.org/en/Basic_Operators

- Interactive tutorial with exercises
- Covers arithmetic and comparison operators
- Complete the exercise at the end

Arithmetic Operators Reference

Operator	Name	Example	Result	Notes
+	Addition	5 + 3	8	Also concatenates strings
-	Subtraction	5 - 3	2	Also negation: -5
*	Multiplication	5 * 3	15	Also repeats strings
/	Division	5 / 3	1.6666...	Always returns float
//	Floor Division	5 // 3	1	Rounds down to whole number
%	Modulo	5 % 3	2	Remainder after division
**	Exponentiation	5 ** 3	125	5 to the power of 3

Arithmetic Examples

```
# Basic arithmetic
a = 10
b = 3

print(f"a + b = {a + b}")    # 13
print(f"a - b = {a - b}")    # 7
print(f"a * b = {a * b}")    # 30
print(f"a / b = {a / b}")    # 3.3333...
print(f"a // b = {a // b}")  # 3 (floor division)
print(f"a % b = {a % b}")    # 1 (remainder)
print(f"a ** b = {a ** b}")  # 1000 (10^3)
```

```python
# Practical examples
price = 19.99
quantity = 3
total = price * quantity
print(f"Total: ${total}")

# Checking if number is even or odd
number = 17
if number % 2 == 0:
    print(f"{number} is even")
else:
    print(f"{number} is odd")

# Converting minutes to hours and minutes
total_minutes = 125
hours = total_minutes // 60  # 2
minutes = total_minutes % 60  # 5
print(f"{total_minutes} minutes = {hours} hours and {minutes} minutes")
```

Comparison Operators

Real Python Comparison Operators

https://realpython.com/python-operators-expressions/#comparison-operators

- Detailed explanation of each comparison operator
- Examples with different data types
- Understanding when comparisons return True or False

Comparison Operators Reference

Operator	Name	Example	Result	Notes
==	Equal to	5 == 5	True	Checks if values are the same
!=	Not equal to	5 != 3	True	Checks if values are different
<	Less than	3 < 5	True	Strict inequality
>	Greater than	5 > 3	True	Strict inequality

<=	Less than or equal	3 <= 5	True	Includes equality
>=	Greater than or equal	5 >= 5	True	Includes equality

Comparison Examples

```
# Basic comparisons
age = 20
voting_age = 18

print(f"age == voting_age: {age == voting_age}")  # False
print(f"age >= voting_age: {age >= voting_age}")  # True
print(f"age != voting_age: {age != voting_age}")  # True

# Comparing strings
name1 = "Alice"
name2 = "alice"
print(f"name1 == name2: {name1 == name2}")  # False (case sensitive)
print(f"name1.lower() == name2: {name1.lower() == name2}")  # True

# Grade checking
score = 85
print(f"A grade (>=90): {score >= 90}")  # False
print(f"B grade (>=80): {score >= 80}")  # True
print(f"Passing (>=60): {score >= 60}")  # True

# Price comparison
original_price = 29.99
sale_price = 19.99
savings = original_price - sale_price
print(f"On sale: {sale_price < original_price}")  # True
print(f"Good deal (>$5 off): {savings > 5}")  # True
```

Assignment Operators

W3Schools Assignment Operators

https://www.w3schools.com/python/python_operators.asp

- Scroll to "Assignment Operators" section
- Shows shorthand ways to modify variables
- Useful for counters and accumulating values

Assignment Operators Reference

Operator	Example	Equivalent To	Description
=	x = 5	x = 5	Assign value
+=	x += 3	x = x + 3	Add and assign
-=	x -= 3	x = x - 3	Subtract and assign
*=	x *= 3	x = x * 3	Multiply and assign
/=	x /= 3	x = x / 3	Divide and assign
%=	x %= 3	x = x % 3	Modulo and assign
**=	x **= 3	x = x ** 3	Exponent and assign

Assignment Operator Examples

```python
# Basic assignment
score = 0
print(f"Starting score: {score}")

# Adding points (shorthand)
score += 10  # Same as: score = score + 10
print(f"After bonus: {score}")

score += 25  # Add more points
print(f"After level complete: {score}")

# Other operations
health = 100
health -= 15  # Take damage
print(f"Health after damage: {health}")

multiplier = 2
multiplier *= 3  # Triple the multiplier
print(f"New multiplier: {multiplier}")

# Practical example: Shopping cart
cart_total = 0.0
cart_total += 19.99  # Add first item
cart_total += 25.50  # Add second item
cart_total += 12.99  # Add third item
print(f"Cart total: ${cart_total}")

# Apply discount
cart_total *= 0.9  # 10% discount
print(f"After 10% discount: ${cart_total:.2f}")

# Counter example
page_views = 0
page_views += 1  # Someone visits
page_views += 1  # Another visit
page_views += 1  # Another visit
print(f"Total page views: {page_views}")
```

Understanding Operator Precedence - VERY IMPORTANT

Just like in math class, Python follows a specific order when evaluating expressions with multiple operators. This is called **operator precedence**.

Remember PEMDAS from Math Class?

Parentheses, **E**xponents, **M**ultiplication and **D**ivision, **A**ddition and **S**ubtraction

Python follows a similar order!

Operator Precedence (Highest to Lowest)

Precedence	Operators	Description
1 (Highest)	()	Parentheses
2	**	Exponentiation
3	*, /, //, %	Multiplication, Division, Floor Division, Modulo
4	+, -	Addition, Subtraction
5	<, <=, >, >=, ==, !=	Comparison operators

6 (Lowest)	=, +=, -=, etc.	Assignment operators

Precedence Examples

```python
# Without parentheses - follows precedence rules
result1 = 2 + 3 * 4  # Multiplication first: 2 + 12 = 14
print(f"2 + 3 * 4 = {result1}")

# With parentheses - overrides precedence
result2 = (2 + 3) * 4  # Addition first: 5 * 4 = 20
print(f"(2 + 3) * 4 = {result2}")

# More complex example
result3 = 10 + 2 ** 3 * 4 / 2  # 10 + 8 * 4 / 2 = 10 + 32 / 2 = 10 + 16 = 26
print(f"10 + 2 ** 3 * 4 / 2 = {result3}")

# Step by step breakdown:
# 1. 2 ** 3 = 8      (exponentiation first)
# 2. 8 * 4 = 32      (multiplication)
# 3. 32 / 2 = 16     (division)
# 4. 10 + 16 = 26    (addition last)

# Using parentheses for clarity
result4 = 10 + ((2 ** 3) * 4) / 2  # Same result, but clearer
print(f"With parentheses: {result4}")

# Comparison with arithmetic
age = 25
is_adult = age >= 18  # Comparison happens after variable lookup
print(f"Is adult: {is_adult}")

# Complex expression
x = 5
y = 3
z = 2
complex_result = x + y * z ** 2 > 10  # 5 + 3 * 4 > 10 = 17 > 10 = True
print(f"Complex expression result: {complex_result}")
# Step by step: z**2=4, y*4=12, x+12=17, 17>10=True
```

When in Doubt, Use Parentheses!

If you're not sure about precedence, use parentheses to make your intentions clear. It's better to be explicit than to rely on precedence rules that others might not remember.

```
# Less clear
result = a + b * c / d

# More clear
result = a + ((b * c) / d)
```

Boolean Expressions

Python Boolean Logic

https://realpython.com/python-boolean/

- Understanding True and False values
- How comparisons create boolean results
- Boolean operators: and, or, not (next week's topic)

Boolean Expression Examples

```
# Simple boolean expressions
age = 22
has_license = True
gpa = 3.7

# Single comparisons
can_vote = age >= 18
print(f"Can vote: {can_vote}")

is_honor_student = gpa >= 3.5
print(f"Honor student: {is_honor_student}")

is_teenager = 13 <= age <= 19  # Chained comparison (Python specialty!)
print(f"Is teenager: {is_teenager}")

# Using boolean variables
can_drive = has_license and age >= 16
print(f"Can drive: {can_drive}")
```

```python
# Complex expressions for real-world logic
price = 29.99
is_member = True
order_amount = 50.00

# Discount eligibility
gets_discount = is_member or order_amount >= 25
print(f"Gets discount: {gets_discount}")

# Final price calculation
if gets_discount:
    final_price = price * 0.9  # 10% discount
else:
    final_price = price

print(f"Final price: ${final_price:.2f}")

# Grade calculation
test_score = 87
assignment_score = 92
participation = 95

average = (test_score + assignment_score + participation) / 3
letter_grade = "A" if average >= 90 else "B" if average >= 80 else "C"
print(f"Average: {average:.1f}, Grade: {letter_grade}")
```

Quick Reference

Operator Cheat Sheet

```
# Arithmetic operators
+   # Addition
-   # Subtraction
*   # Multiplication
/   # Division (always returns float)
//  # Floor division (rounds down)
%   # Modulo (remainder)
**  # Exponentiation (power)

# Comparison operators
```

```
==   # Equal to
!=   # Not equal to
<    # Less than
>    # Greater than
<=   # Less than or equal
>=   # Greater than or equal

# Assignment operators
=    # Assign
+=   # Add and assign
-=   # Subtract and assign
*=   # Multiply and assign
/=   # Divide and assign
```

Common Patterns

```
# Counter
count = 0
count += 1  # Increment by 1

# Accumulator
total = 0
total += value  # Add to running total

# Even/odd check
is_even = number % 2 == 0

# Range checking
is_valid_grade = 0 <= grade <= 100

# Price calculation
subtotal = quantity * price
tax = subtotal * tax_rate
total = subtotal + tax

# Discount application
discounted_price = original_price * (1 - discount_rate)
```

Precedence Reminder

```
# Order of operations (PEMDAS-like):
# 1. Parentheses ()
# 2. Exponentiation **
# 3. Multiplication *, Division /, Floor //, Modulo %
# 4. Addition +, Subtraction -
# 5. Comparisons <, <=, >, >=, ==, !=
# 6. Assignment =, +=, -=, etc.

# Examples:
2 + 3 * 4        # = 14 (not 20)
(2 + 3) * 4      # = 20
2 ** 3 * 4       # = 32 (8 * 4)
2 ** (3 * 4)     # = 4096 (2^12)
```