Basic Data Types

2.2- Numbers الاعداد

1. Introduction to numbers (integers and floats)
2. Basic arithmetic operations
3. Order of operations (PEMDAS)
4. Type conversion (int to float and vice versa)
5. Common number functions (abs, round, pow, etc.)
6. Exercises with solutions

Learning Objectives

By the end of this lesson, you will be able to:

- Differentiate between integers and floats
- Perform basic arithmetic operations
- Use Python as a calculator
- Understand order of operations (PEMDAS)
- Convert between number types
- Use common math functions

In Python, numbers are a fundamental data type used to represent numeric values. Python supports various types of numbers, including integers, floating-point numbers, and complex numbers.

1. Types of Numbers in Python انواع الاعداد في بايثون

Two main types of numbers:

python

```python
# Integers - whole numbers (positive or negative)
age = 25
students_count = -30
year = 2024

# Floats - decimal numbers
price = 19.99
temperature = -5.5
pi = 3.14159
```

```python
print(type(age))        # <class 'int'>
print(type(price))      # <class 'float'>
```

2. Basic Arithmetic Operations

```python
# Addition
result1 = 10 + 5
print("10 + 5 =", result1)

# Subtraction
result2 = 10 - 5
print("10 - 5 =", result2)

# Multiplication
result3 = 10 * 5
print("10 × 5 =", result3)

# Division (always returns a float)
result4 = 10 / 5
print("10 ÷ 5 =", result4)

# Integer Division (rounds down to whole number)
result5 = 10 // 3
print("10 // 3 =", result5)  # 3 (not 3.333)

# Modulus (remainder after division)
result6 = 10 % 3
print("10 % 3 =", result6)   # 1 (remainder)

# Exponentiation (power)
result7 = 2 ** 3
print("2³ =", result7)       # 8
```

3. Order of Operations (PEMDAS)

PEMDAS Rule: Parentheses → Exponents → Multiplication/Division → Addition/Subtraction

```python
# Example 1: Without parentheses
result1 = 5 + 3 * 2      # 5 + 6 = 11
print("5 + 3 × 2 =", result1)
```

```python
# Example 2: With parentheses
result2 = (5 + 3) * 2    # 8 × 2 = 16
print("(5 + 3) × 2 =", result2)

# Example 3: Complex calculation
result3 = 10 + 2 ** 3 * 2 - 5 // 2
# Step-by-step: 10 + (8 × 2) - (5 // 2)
#              10 + 16 - 2 = 24
print("10 + 2³ × 2 - 5 // 2 =", result3)
```

4. Working with Variables استخدام المتغيرات

python

```python
# Using variables in calculations
x = 15
y = 4

sum_result = x + y
difference = x - y
product = x * y
quotient = x / y

print(f"{x} + {y} = {sum_result}")
print(f"{x} - {y} = {difference}")
print(f"{x} × {y} = {product}")
print(f"{x} ÷ {y} = {quotient}")

# Updating variables with arithmetic
score = 100
score = score + 10      # Add 10 to current score
score += 5              # Shortcut: same as score = score + 5
score *= 2              # Multiply score by 2
print("Final score:", score)
```

5. Type Conversion

python

```python
# Converting between int and float
number_int = 10
number_float = float(number_int)    # Convert to float
print(number_float)                 # 10.0
```

```python
decimal_num = 7.8
whole_num = int(decimal_num)        # Convert to int (truncates decimal)
print(whole_num)                    # 7

# Input from user (always comes as string)
age_input = "25"
age_number = int(age_input)         # Convert string to int
print(age_number + 5)               # 30

# Rounding numbers
pi = 3.14159
rounded_pi = round(pi, 2)           # Round to 2 decimal places
print(rounded_pi)                   # 3.14

large_num = 123.4567
rounded_large = round(large_num)    # Round to nearest whole number
print(rounded_large)                # 123
```

6. Common Math Functions

```python
# Using built-in functions
numbers = [4, 2, 9, 5, 1]

print("Maximum:", max(numbers))     # 9
print("Minimum:", min(numbers))     # 1
print("Absolute value:", abs(-10))  # 10
print("Power:", pow(2, 3))          # 8 (same as 2**3)

# Using the math module for advanced operations
import math

print("Square root:", math.sqrt(25))        # 5.0
print("Ceiling:", math.ceil(4.2))           # 5 (rounds up)
print("Floor:", math.floor(4.9))            # 4 (rounds down)
print("Pi constant:", math.pi)              # 3.141592653589793
```

7. Special Number Values

```python
# Infinity
positive_inf = float('inf')
```

```
negative_inf = float('-inf')
print("Positive infinity:", positive_inf)
print("Negative infinity:", negative_inf)

# Not a Number (NaN) - results from invalid operations
import math
result = math.sqrt(-1)  # Can't take square root of negative
print("NaN result:", result)  # nan

# Checking for special values
print("Is infinity?", math.isinf(positive_inf))
print("Is NaN?", math.isnan(result))
```