UNIT I: Introduction

Subtopic 2: Basic Syntax and Data Types

Understanding the Python Syntax and Structure

2.1 Python Syntax Basics:

Python's syntax is designed to be readable and straightforward. Here are some key points:

1. **Comments:** Use the `#` symbol to add comments.

```
# This is a comment
print("Hello, World!")  # This prints a message
```

2. **Indentation:** Python uses indentation to define blocks of code. Each indentation level is typically four spaces.

```
if True:
    print("This is indented")
    if False:
        print("This won't print")
```

3. **Variables:** Variables do not need explicit declaration and can be assigned any value.

```
x = 5
y = "Hello"
z = 3.14
```

Data Types in Python

Python has several built-in data types. Here are the most common ones:

2.2 Integers (int):

- Whole numbers, positive or negative, without decimals. `python  a = 10  b = -5`

Example:

```
a = 10
b = -5
print(type(a))  # Output: <class 'int'>
print(a + b)    # Output: 5


# Example for Integers
a = 10
b = -5
print(type(a))  # Output: <class 'int'>
```

```
print(a + b)    # Output: 5
```

2.3 Floats (float):

- Numbers with decimals. `python  c = 3.14  d = -0.001`

Example:

```
c = 3.14
d = -0.001
print(type(c))  # Output: <class 'float'>
print(c + d)    # Output: 3.139


# Example for Floats
c = 3.14
d = -0.001
print(type(c))  # Output: <class 'float'>
print(c + d)    # Output: 3.139
```

2.4 Strings (str):

- A sequence of characters enclosed in quotes, either single (`'`) or double (`"`). `python  e = "Hello"  f = 'World'`

Example:

```
e = "Hello"
f = 'World'
print(type(e))    # Output: <class 'str'>
print(e + " " + f)  # Output: Hello World


# Example for Strings
e = "Hello"
f = 'World'
print(type(e))    # Output: <class 'str'>
print(e + " " + f)  # Output: Hello World
```

2.5 Booleans (bool):

- Represents `True` or `False` values. `python  g = True  h = False`

Example:

```
g = True
h = False
```

```
print(type(g))    # Output: <class 'bool'>
print(g and h)    # Output: False
print(g or h)     # Output: True


# Example for Booleans
g = True
h = False
print(type(g))    # Output: <class 'bool'>
print(g and h)    # Output: False
print(g or h)     # Output: True
```

Working with Different Data Types

2.6 Type Conversion:

- You can convert between data types using functions like `int()`, `float()`, `str()`, and `bool()`. `python  i = "123"  j = int(i)  k = float(i)`

Example:

```
i = "123"
j = int(i)
k = float(i)
print(type(j))  # Output: <class 'int'>
print(type(k))  # Output: <class 'float'>
print(j, k)     # Output: 123 123.0


# Example for Type Conversion
i = "123"
j = int(i)
k = float(i)
print(type(j))  # Output: <class 'int'>
print(type(k))  # Output: <class 'float'>
print(j, k)     # Output: 123 123.0
```

2.7 Combining Different Data Types:

- Use appropriate operations and functions to combine and manipulate different data types. `python  l = 5  m = " apples"  n = str(l) + m`

Example:

```
l = 5
m = " apples"
n = str(l) + m
print(n)  # Output: 5 apples
```

```python
# Example for Combining Different Data Types
l = 5
m = " apples"
n = str(l) + m
print(n)  # Output: 5 apples
```

Summary

Understanding Python's basic syntax and data types is essential for writing effective programs. Python's simplicity in syntax and its flexibility in handling different data types make it a powerful tool for beginners and experienced programmers alike. By mastering these fundamentals, you will be well-equipped to tackle more complex programming tasks and applications in engineering.