

# Day\_9

May 10, 2024

## 1 Day 9

### 1.1 Revising Day 1 to Day 8

#### Introduction to Python and Installation

Python is a high-level programming language known for its simplicity and readability. To install Python, download the installer from [python.org](https://python.org) and follow the installation instructions. Basic syntax involves using indentation to denote blocks of code and writing statements line by line. Variables are used to store data, and data types include integers, floats, strings, booleans, etc. Basic input/output operations involve functions like `input()` and `print()`.

*Example:*

```
# Basic syntax
print("Hello, world!") # Example of a print statement

# Variables and data types
x = 10
y = "Python"
print("x is", x, "and y is", y)

# Basic input/output
name = input("Enter your name: ")
print("Hello,", name)
```

#### Control Flow

Conditional statements like `if`, `elif`, and `else` are used for decision-making in code. Loops, including `for` and `while` loops, are used for repetitive tasks. `Break` and `continue` statements alter the flow of loop execution.

*Example:*

```
# Conditional statements
num = 10
if num > 0:
    print("Positive")
elif num < 0:
    print("Negative")
else:
    print("Zero")
```

```

# Loops
for i in range(5):
    print(i)

# Break and continue statements
for i in range(10):
    if i == 5:
        break
    print(i)

```

## Functions and Modules

Functions are blocks of reusable code, defined using the `def` keyword. Arguments can be passed to functions to provide input, and functions can return values using the `return` statement. Modules are Python files containing definitions and statements, which can be imported into other files.

*Example:*

```

# Writing functions
def greet(name):
    return "Hello, " + name

```

```
print(greet("Alice"))
```

```

# Importing modules
import math
print(math.sqrt(16))

```

## Lists

Lists are ordered collections of items, defined using square brackets. Elements in a list can be accessed using indexing and slicing. List methods like `append()`, `pop()`, and `remove()` manipulate lists efficiently. List comprehensions provide a concise way to create lists based on existing lists.

*Example:*

```

# Creating lists
numbers = [1, 2, 3, 4, 5]

# Accessing elements
print(numbers[0]) # Output: 1

# List methods
numbers.append(6)
print(numbers) # Output: [1, 2, 3, 4, 5, 6]

# List comprehensions
squares = [x**2 for x in range(1, 6)]
print(squares) # Output: [1, 4, 9, 16, 25]

```

## Dictionaries and Sets

# Day 9

## Revising Day 1 to Day 7

### Introduction to Python and Installation

Python is a high-level programming language known for its simplicity and readability. To install Python, download the installer from [python.org](https://python.org) and follow the installation instructions. Basic syntax involves using indentation to denote blocks of code and writing statements line by line. Variables are used to store data, and data types include integers, floats, strings, booleans, etc. Basic input/output operations involve functions like `input()` and `print()`.

*Example:*

```
# Basic syntax
print("Hello, world!") # Example of a print statement

# Variables and data types
x = 10
y = "Python"
print("x is", x, "and y is", y)

# Basic input/output
name = input("Enter your name: ")
print("Hello,", name)
```

### Control Flow

Conditional statements like `if`, `elif`, and `else` are used for decision-making in code. Loops, including `for` and `while` loops, are used for repetitive tasks. `break` and `continue` statements alter the flow of loop execution.

*Example:*

```
# Conditional statements
num = 10
if num > 0:
    print("Positive")
elif num < 0:
    print("Negative")
else:
    print("Zero")

# Loops
for i in range(5):
    print(i)
```

Dictionaries are unordered collections of key-value pairs, defined using curly braces. Sets are unordered collections of unique elements, also defined using curly braces. Dictionary and set elements can be accessed using keys and methods like `keys()`, `values()`, and `add()`.

*Example:*

```
# Creating dictionaries
person = {"name": "Alice", "age": 30}

# Accessing elements
print(person["name"]) # Output: Alice

# Dictionary methods
person["city"] = "New York"
print(person) # Output: {'name': 'Alice', 'age': 30, 'city': 'New York'}

# Creating sets
fruits = {"apple", "banana", "cherry"}

# Set methods
fruits.add("orange")
print(fruits) # Output: {'banana', 'cherry', 'apple', 'orange'}
```

## File Handling

File handling involves opening, reading from, and writing to files. Files are opened using the `open()` function and closed using the `close()` method. Different file modes like `'r'`, `'w'`, and `'a'` determine the file's behavior.

*Example:*

```
# Writing to a file
with open("example.txt", "w") as file:
    file.write("Hello, world!")

# Reading from a file
with open("example.txt", "r") as file:
    content = file.read()
    print(content)
```

## Classes and Objects

Classes are blueprints for creating objects, containing attributes and methods. Objects are instances of classes, representing specific entities in code. Class attributes are variables shared by all instances, while methods are functions defined within a class.

*Example:*

```
# Defining a class
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```