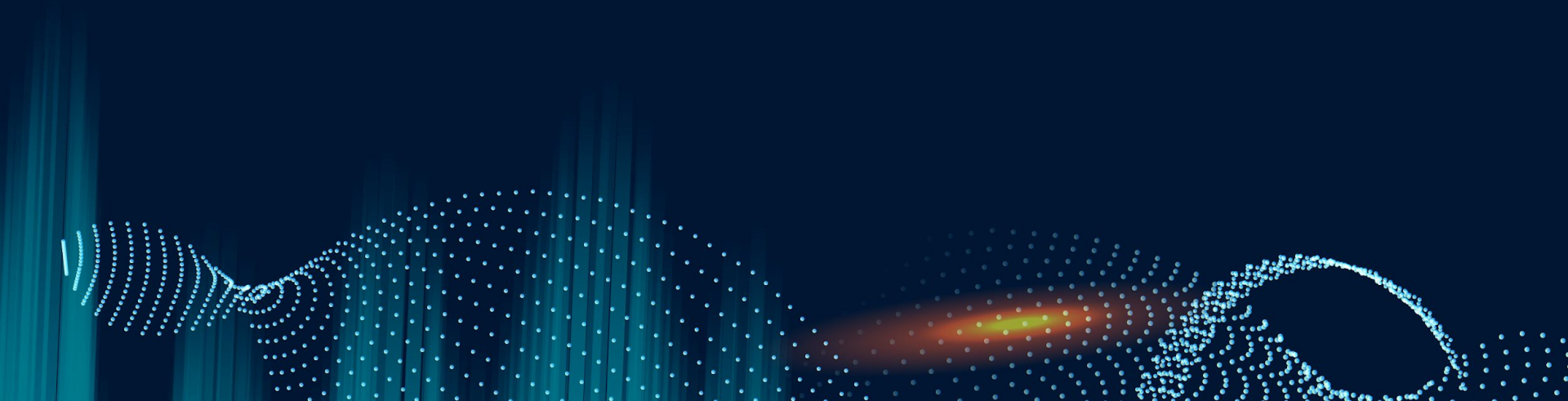
Familiarizing with AI

Session 3 : Python Fundamentals



Python Fundamentals

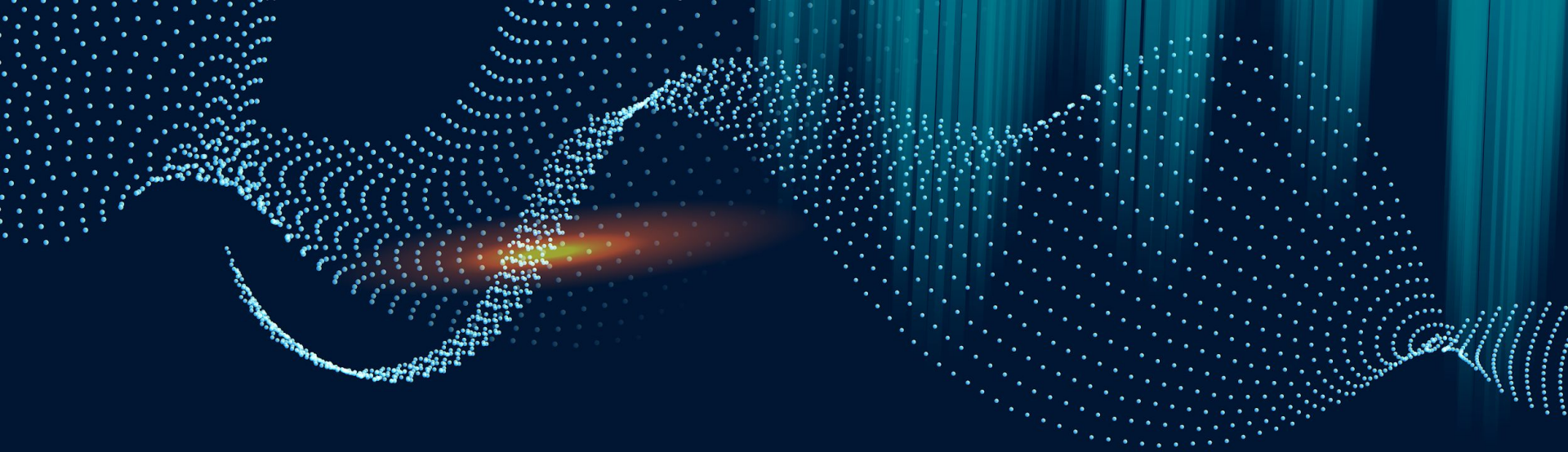
01 Introduction to Python

02 Basic Python Syntax

03 Operators

04 Data TYpes & Variables

05 Control Flow

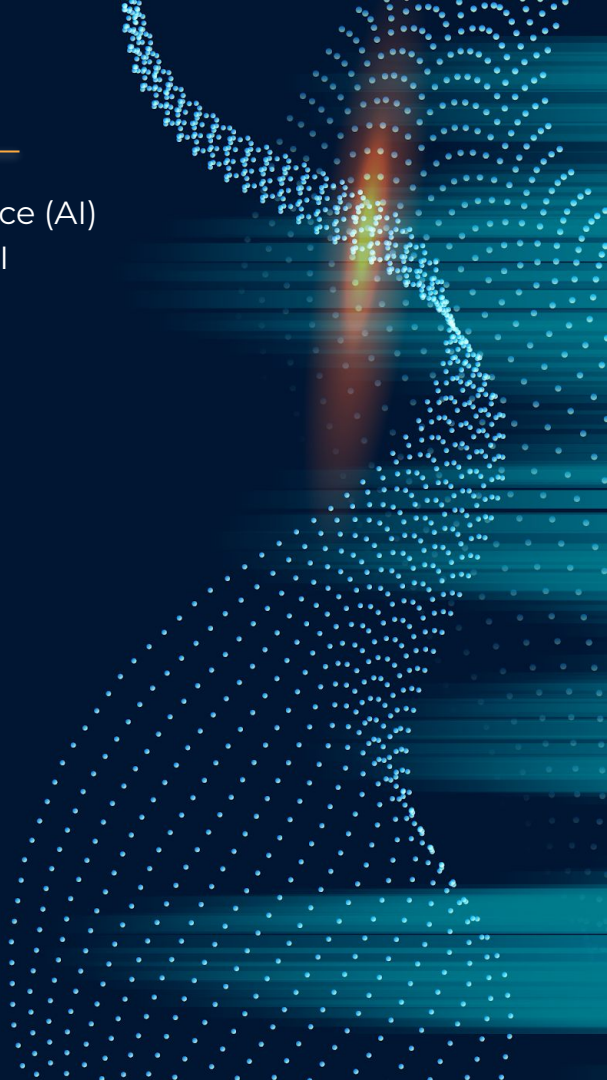


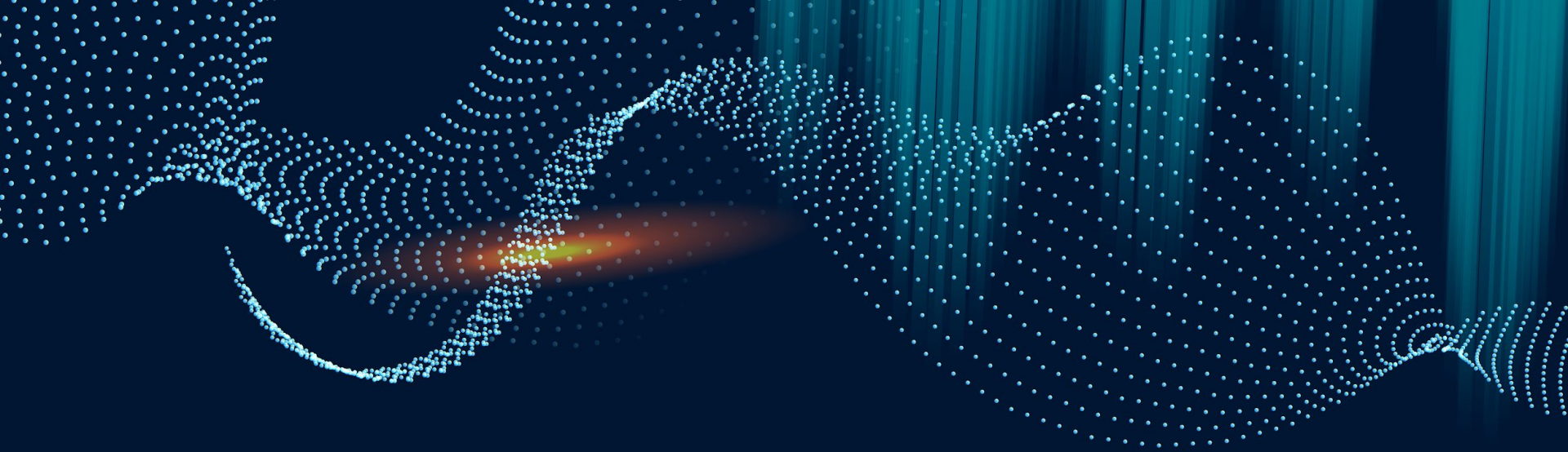
01 Introduction to Python

Introduction to Python

Python has become one of the most popular languages for artificial intelligence (AI) due to its simplicity, versatility, and a rich ecosystem of libraries that support AI development.

- Ease of Learning and Use
- Extensive Libraries and Frameworks
- Flexibility
- Strong Community and Documentation
- Integration with Other Languages





02 Basic Python Syntax

Basic Python Syntax

Case Sensitivity

Python is case-sensitive. This means that the names `Name`, `name`, and `NAME` are three completely different identifiers.

```
value = 10
Value = 20
print(value)  # 10
print(Value)  # 20
```

Basic Python Syntax

Comments

Comments in Python start with a # symbol. They are used to add notes or explanations within your code but are ignored when the program runs.

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

For multi-line comments, or documentation, we use triple quotes

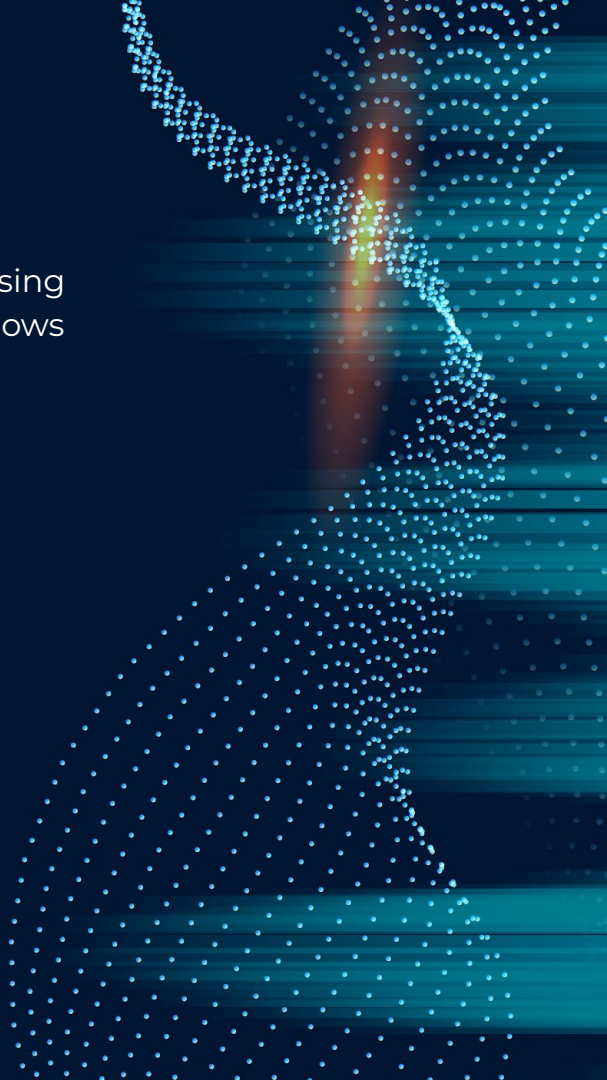
```
"""
This is a longer comment
spanning multiple lines
"""
```

Basic Python Syntax

Indentation

Python uses indentation (spaces or tabs) to define blocks of code, instead of using braces { } like many other languages. For example, in a for loop, indentation shows which statements are part of the loop.

```python
for i in range(5):
    print(i)
```



Basic Python Syntax

Line Breaks and Continuation

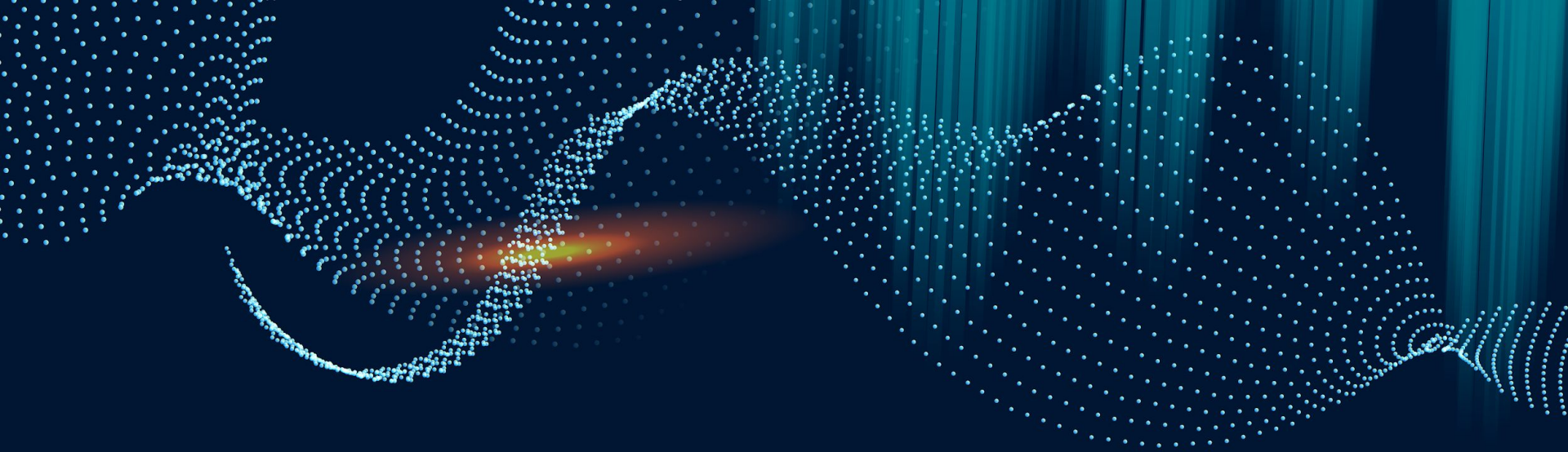
You can have multiple lines in python by using either:

using `\`; or

```
total = 1 + 2 + 3 + \
        4 + 5 + 6
print(total)
```

using `()`

```
total = (1 + 2 + 3 +
         4 + 5 + 6)
print(total)
```



03 Operators

Operators

Arithmetic Operators

Used for basic Math operations

```
a = 10
b = 3

print("Addition:", a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
print("Division:", a / b)   # normal division
print("Floor Division:", a // b) # discards remainder
print("Modulus:", a % b)    # remainder
print("Exponent:", a ** b)  # power
```

Operators

Comparison Operators

Compare values and return True or False.

```
x = 5
y = 10

print(x == y)  # Equal?
print(x != y)  # Not equal?
print(x > y)   # Greater?
print(x < y)   # Less?
print(x >= y)  # Greater or equal?
print(x <= y)  # Less or equal?
```

Operators

Logical Operators

Logical operators combine conditional statements.

```
x = True
y = False

print(x and y)  # Both must be true
print(x or y)   # At least one must be true
print(not x)    # Negation
```

Operators

Assignment Operators

Used to assign values to variables, sometimes while performing an operation.

```
num = 10
print("Start:", num)

num += 5   # same as num = num + 5
print("After += 5:", num)

num *= 2   # same as num = num * 2
print("After *= 2:", num)

num -= 3
print("After -= 3:", num)

num /= 4
print("After /= 4:", num)
```

Operators

Membership Operators

Check if a value is inside a sequence, like a list or string.

```
fruits = ["apple", "banana", "cherry"]

print("apple" in fruits)     # True
print("grape" not in fruits) # True
```

Operators

Identity Operators

Check whether two variables point to the same object in memory.

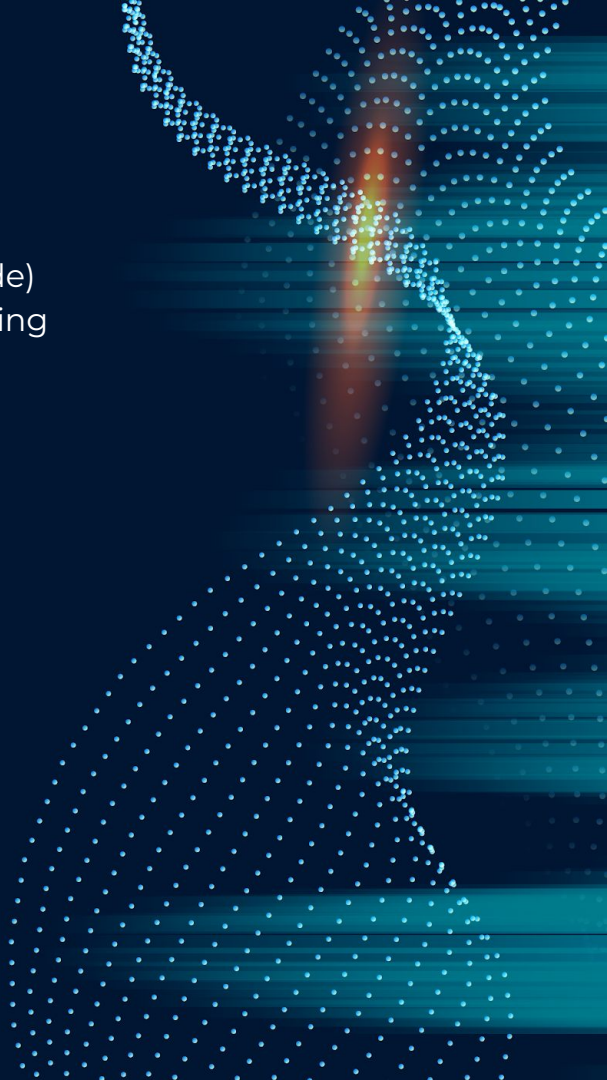
```
a = [1, 2, 3]
b = [1, 2, 3]
c = a

print(a is b)   # False (different objects with same contents)
print(a is c)   # True (same object)
print(a == b)   # True (contents are equal)
```

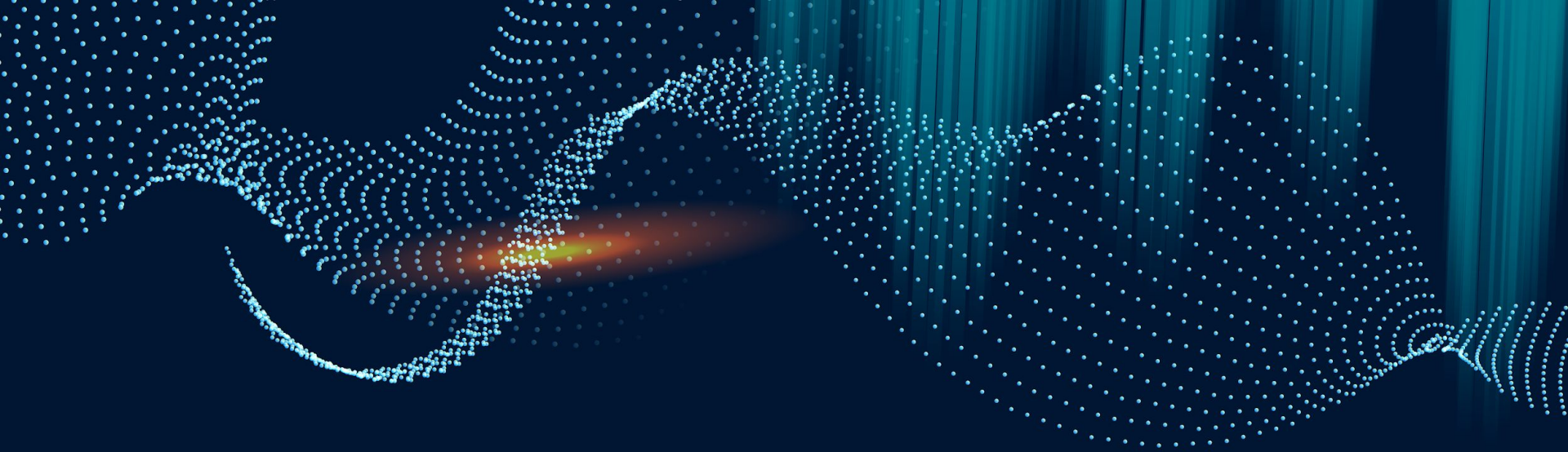
Basic Python Syntax

Modules and Packages

Python allows code to be organized into modules (files containing Python code) and packages (directories containing multiple modules). You can import existing modules or create your own.

```python
import math
print(math.sqrt(16))
```





04 Data Types & Variables

Data Types and Variables

Naming Conventions

Variable names in Python follow a few rules:

- Must start with a letter or underscore _
- Can contain letters, numbers, and underscores
- Cannot start with a number
- Cannot use reserved keywords like if, for, class

By convention, we use lowercase words with underscores: like user_name or total_price

```
2name = "John"    # ❌ invalid
class = "Math"    # ❌ invalid
```

Data Types and Variables

Dynamically Typed

Python is dynamically typed, meaning you don't need to declare variable types explicitly. You can assign a value to a variable, and Python will infer its type.

```
x = 10          # integer
print(type(x))

x = "Python"  # now a string
print(type(x))

a, b, c = 1, 2, 3
print(a, b, c)

x = y = z = 0   # all three get the same value
print(x, y, z)
```

Data Types and Variables

Basic Data Types

- **Integers**: Whole numbers
- **Floats**: Decimal Numbers
- **Strings**: Text Data
- **Booleans**: True or False
- **NoneType**: Represents 'no value' or 'empty'

Data Types and Variables

Data Structures

Python has built-in data structures that allow you to store and manage collections of data efficiently.

Lists

Ordered, mutable collections

```
my_list = [1, 2, 3]
```

Tuples

Ordered, immutable collections

```
my_tuple = (1, 2, 3)
```

Dictionaries

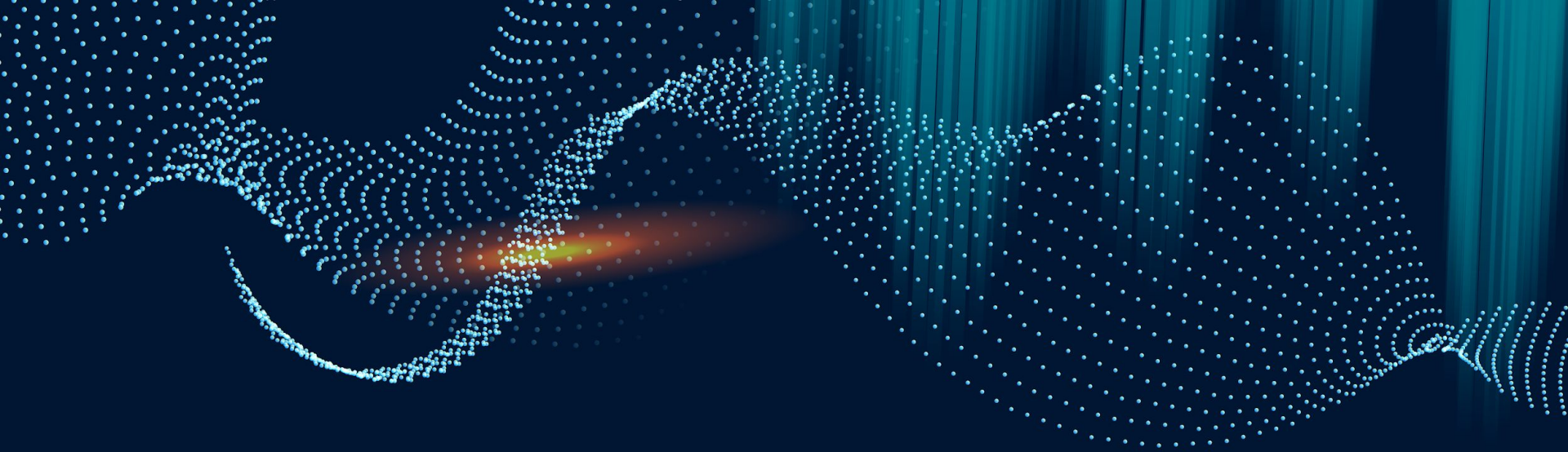
Key-value pairs, where each key must be unique

```
my_dict = {"name": "Alice", "age": 30}
```

Sets

Unordered collections of unique elements

```
my_set = {1, 2, 3}
```



05 Control Flow

Control Flow

Conditions

Python uses control flow statements like if, elif, and else to execute code conditionally.

```
age = 18
if age >= 18:
    print("Adult")
elif age >= 13:
    print("Teenager")
else:
    print("Child")
```

Control Flow

Loops

Loops are used to repeat blocks of code

For loop

Iterates over a sequence like a list or string

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

While loop

Repeats as long as a condition is true

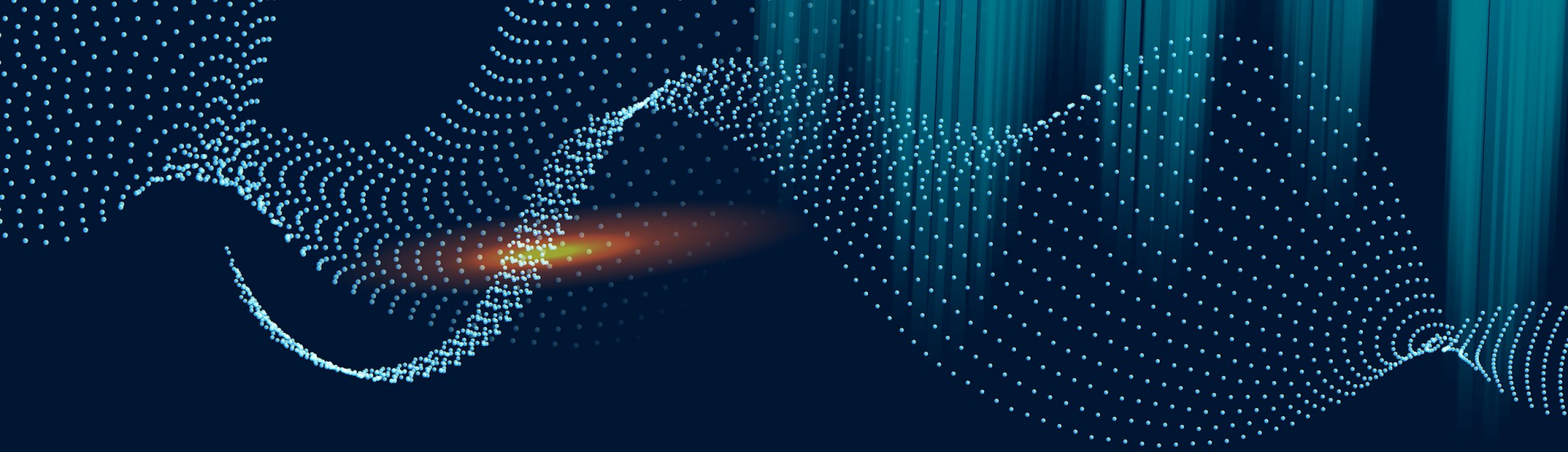
```
i = 0
while i < 5:
    print(i)
    i += 1
```

Control Flow

Exception Handling

Python uses try, except, finally, and else blocks for handling exceptions (errors) that might occur during program execution.

```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero.")
finally:
    print("This always runs.")
```



06 Functions

Functions

What are Functions?

- Functions are reusable blocks of code.
- Defined with the “def” keyword.
- May take input (parameters).
- May produce output (return values).

```
def greet(name):
    return f"Hello, {name}!"

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

Functions

Parameters vs Arguments

- Parameters: names in function definition.
- Arguments: actual values you pass in.

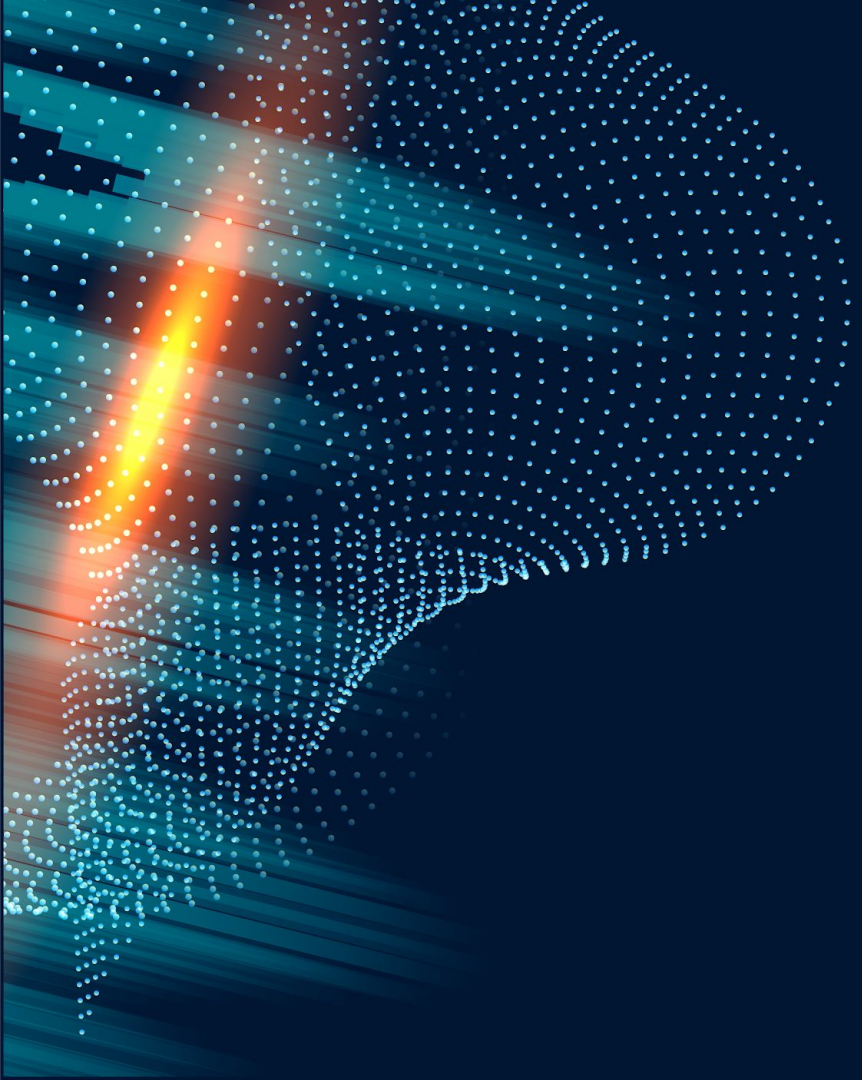
```
def greet(name):
    return f"Hello, {name}!"

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

Functions

Default Parameters

In Python, you can give parameters default values. If no argument is provided, the default will be used.

```
def greet(name="Guest"):
    return f"Hello, {name}!"

print(greet())        # Hello, Guest!
print(greet("Bob"))   # Hello, Bob!
```



Q&A