Familiarizing with AI

Session 3 : Python Fundamentals

Python Fundamentals

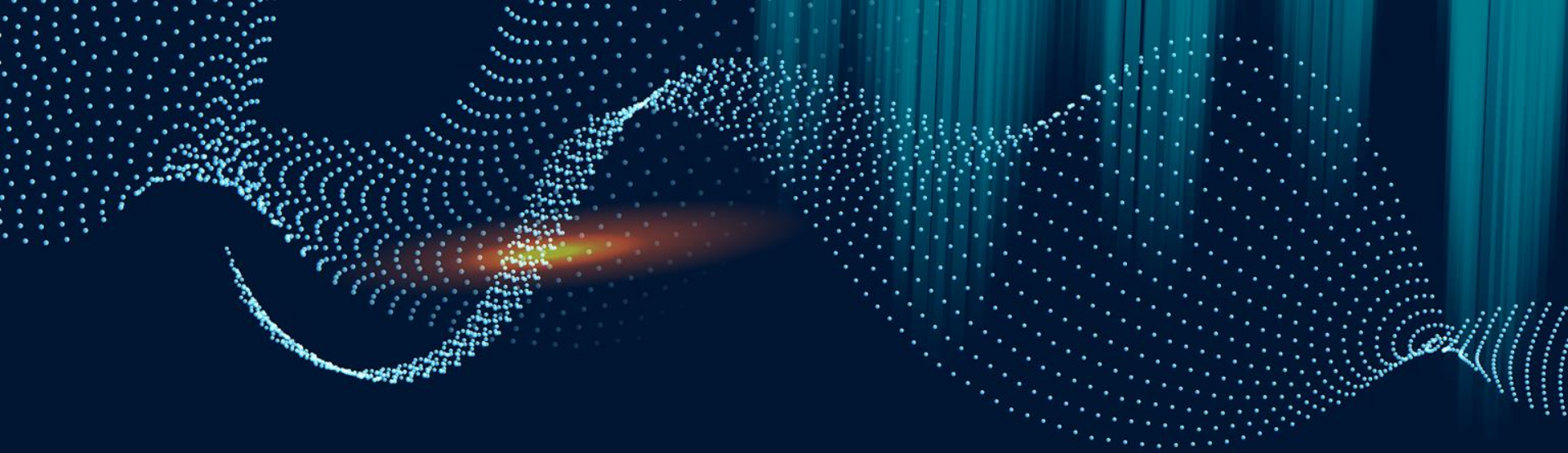
01 Introduction to Python

02 Python Syntax & Structure

03 Data Types & Variables

04 Control Flow (Loops and Conditionals)

05 Functions

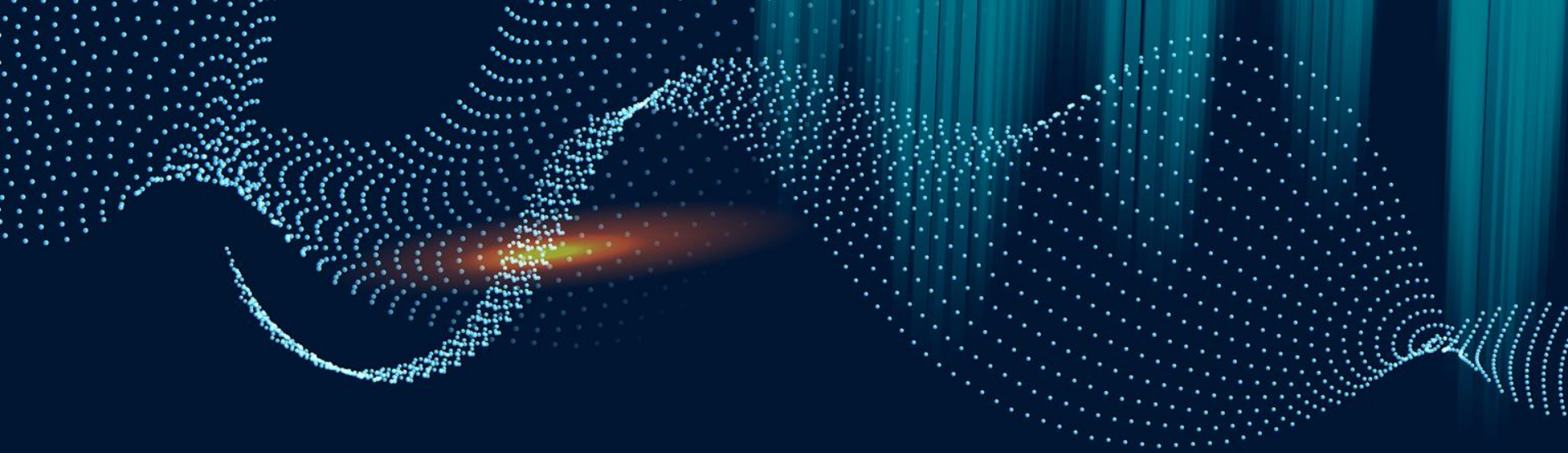


01 Introduction to Python

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

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

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
```

Basic Python Syntax

Operators

Python supports various operators for performing operations on variables and values. These include:

Arithmetic Operators
+, -, *, /, //, %, **

```python
a = 5 + 3   # Addition
b = 10 // 2 # Floor division
c = 3 ** 2  # Exponentiation
```

Comparison Operators
==, !=, >, <, >=, <=

```python
is_equal = 5 == 5   # True
is_greater = 5 > 3  # True
```

Comparison Operators
and, or, not

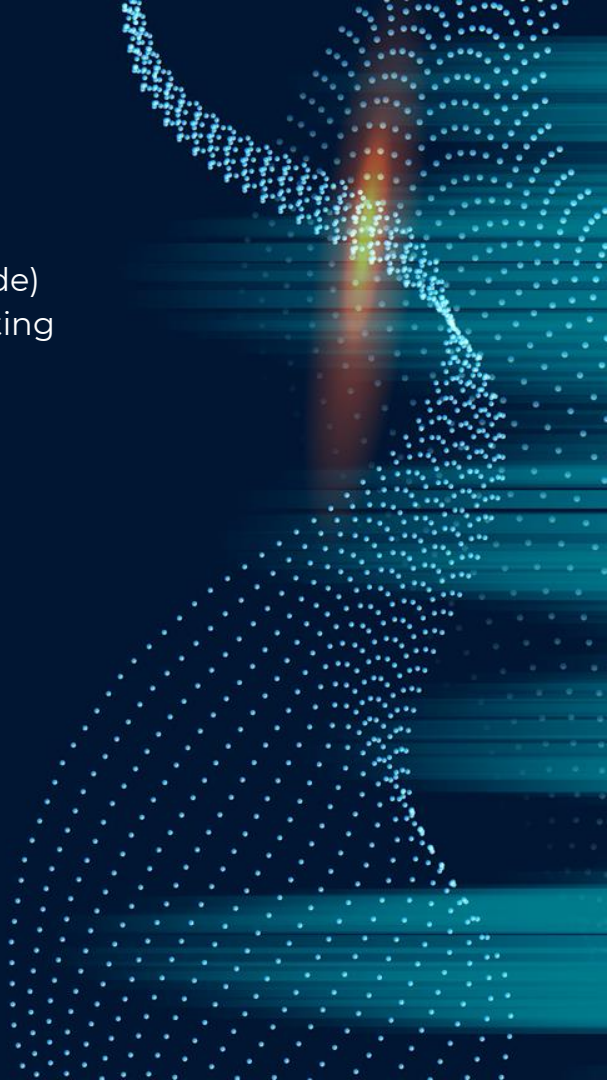
```python
result = True and False  # False
```

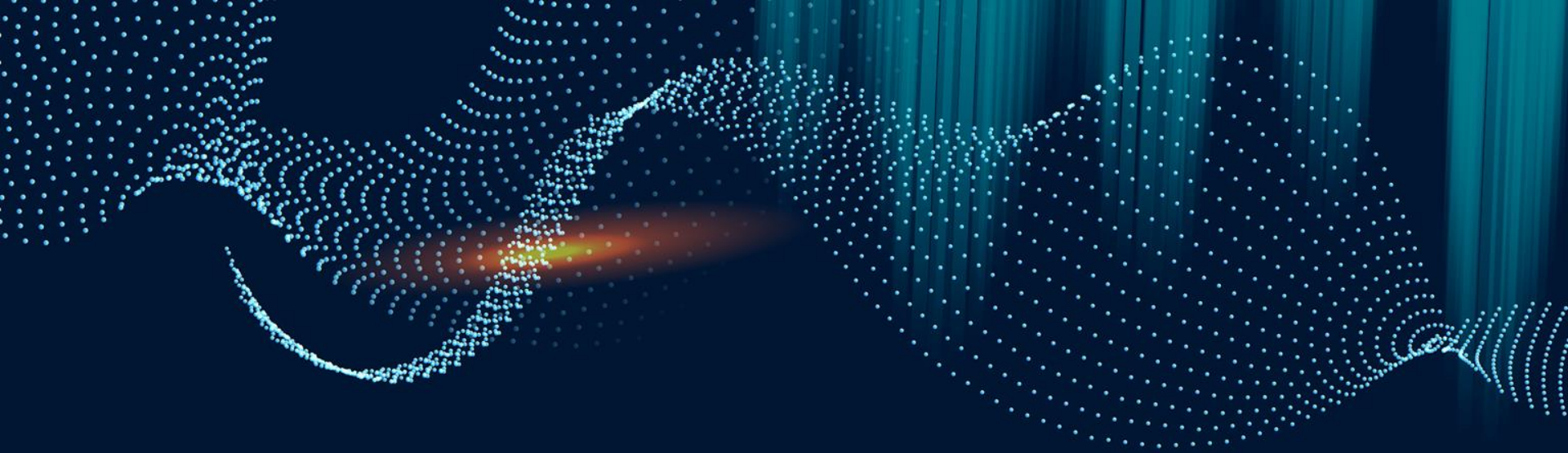
Basic Python Syntax

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))
```





03 Data Types and Variables

Data Types and Variables

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.

```python
x = 10        # Integer
y = 3.14      # Float
name = "Bob"  # String
```

Data Types

- **Integers**: Whole numbers
- **Floats**: Decimal Numbers
- **Strings**: Text Data
- **Booleans**: True or False

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}
```

Data Types and Variables

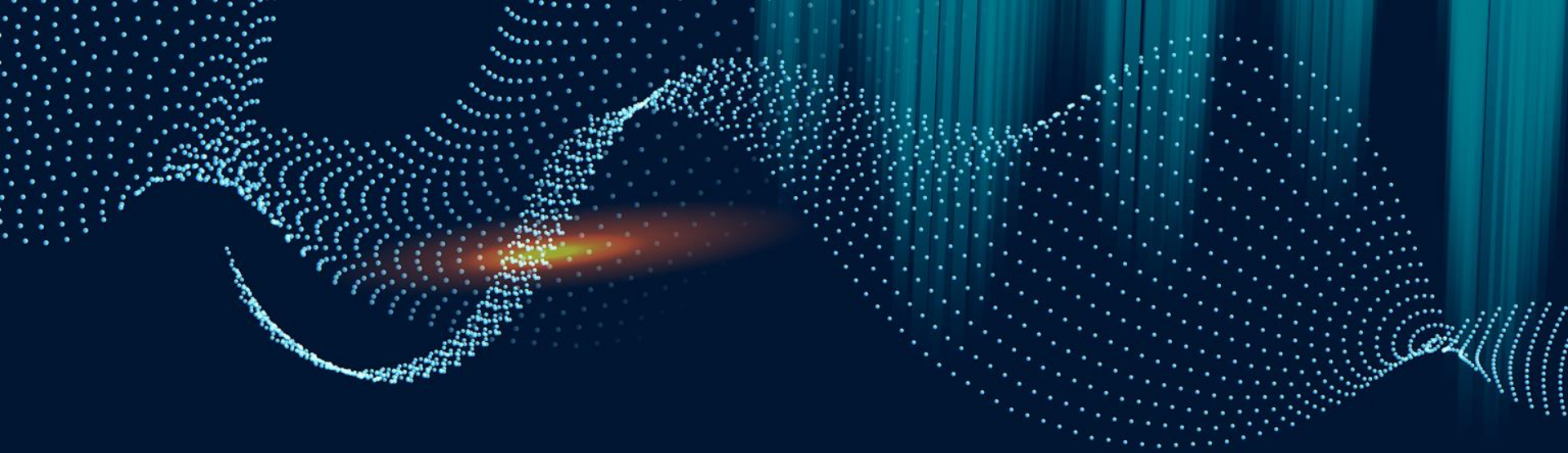
Object-Oriented Programming (OOP)

Python is an object-oriented language, allowing the creation of classes and objects.

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

    def greet(self):
        print(f"Hi, I'm {self.name}.")

p = Person("Alice", 30)
p.greet()
```



04 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

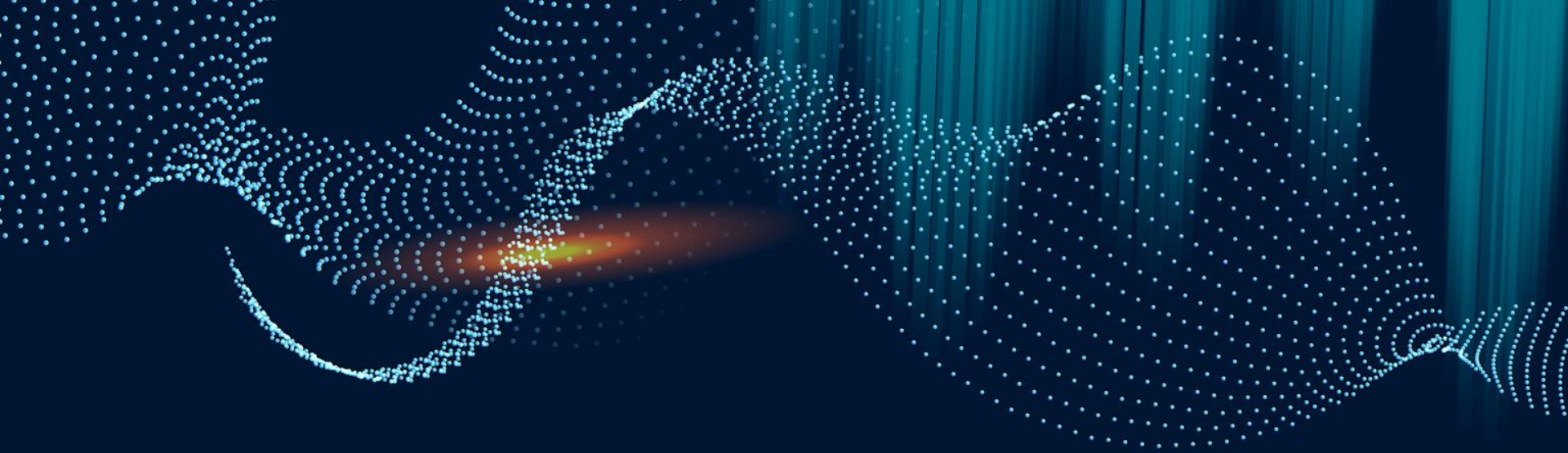
```
i = 0
while i < 5:
    print(i)
    i += 1
```

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.")
```



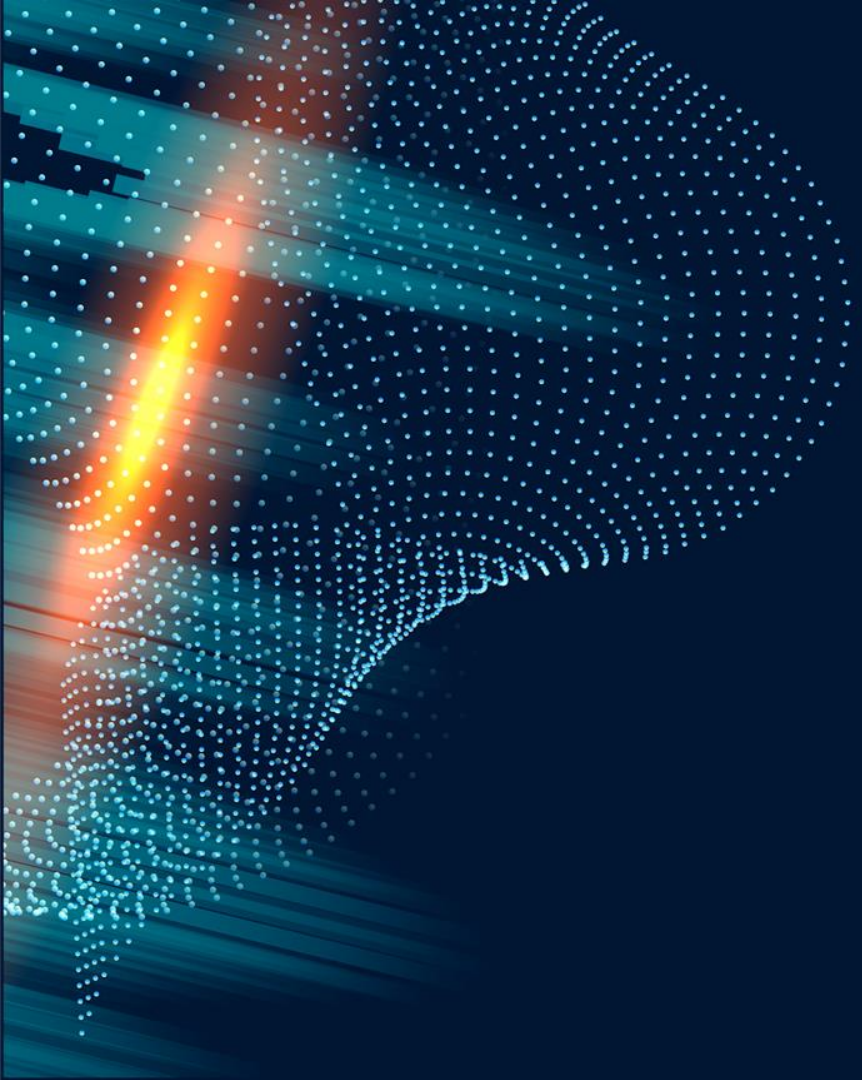
05 Functions

Functions

Functions in Python are defined using the def keyword. Functions can take arguments and return values.

```
def greet(name):
    return "Hello, " + name

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



Q&A