Functional Programming:

- Focuses on the evaluation of mathematical functions and avoids changing state and mutable data.

- Emphasizes immutability, meaning that once a value is assigned, it cannot be changed.

- Uses higher-order functions, which can take other functions as arguments or return them as results.

- Supports recursion as a fundamental control structure.

- Examples of functional programming languages include Haskell, Lisp, and Erlang.

A functional language (ideally) allows you to write a mathematical function, i.e. a function that takes n arguments and returns a value. If the program is executed, this function is logically evaluated as needed.1

A procedural language, on the other hand, performs a series of sequential steps. (There's a way of transforming sequential logic into functional logic called continuation passing style.)

As a consequence, a purely functional program always yields the same value for an input, and the order of evaluation is not well-defined; which means that uncertain values like user input or random values are hard to model in purely functional languages.

1 As everything else in this answer, that’s a generalisation. This property, evaluating a computation when its result is needed rather than sequentially where it’s called, is known as “laziness”. Not all functional languages are actually universally lazy, nor is laziness restricted to functional programming. Rather, the description given here provides a “mental framework” to think about different programming styles that are not distinct and opposite categories but rather fluid ideas.

Procedural Programming:

- Organizes the program into a series of procedures or functions that are executed sequentially.

- Uses variables to store and manipulate data.

- Follows a top-down approach where the program is divided into smaller tasks or procedures.

- Focuses on solving problems by breaking them down into smaller sub-tasks.

- Examples of procedural programming languages include C, Pascal, and Fortran.

Object-Oriented Programming (OOP):

- Organizes the program around objects that encapsulate data and behavior together.

- Encourages the use of classes to define objects and their properties (attributes) and behaviors (methods).

- Supports concepts like inheritance, polymorphism, and encapsulation for code reuse and modularity.

- Allows for modeling real-world entities in software development.

- Examples of object-oriented programming languages include Java, C++, Python.

As the name suggests, Object-Oriented Programming or OOPs refers to languages that use objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

OOPs Concepts:

Class

Objects

Data Abstraction

Encapsulation

Inheritance

Polymorphism

Dynamic Binding

Message Passing

Comparison:

1. Data Handling: Functional programming avoids mutable data while procedural programming uses variables to store and manipulate data. Object-oriented programming encapsulates data within objects.

2. Control Flow: Functional programming relies heavily on recursion for control flow. Procedural programming uses loops and conditional statements. Object-oriented programming uses method calls on objects to control flow.

3. State Management: Functional programming emphasizes immutability, avoiding changing state altogether. Procedural programming allows for changing state using variables. Object-oriented programming manages state within objects.

4. Code Organization: Functional programming focuses on composing functions to solve problems. Procedural programming organizes code into procedures or functions. Object-oriented programming organizes code into objects and classes.

5. Code Reusability: Functional programming promotes code reusability through higher-order functions. Procedural programming allows for code reuse through modularization. Object-oriented programming enables code reuse through inheritance and polymorphism.

6. Paradigm Focus: Functional programming focuses on the evaluation of mathematical functions. Procedural programming focuses on procedures or functions as the main building blocks. Object-oriented programming focuses on objects and their interactions.

It's important to note that these paradigms are not mutually exclusive, and many modern languages support multiple paradigms or a combination of them.