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

    def __str__(self):
        return f'Person = ( name - {self.name} , age - {self.age})'

    def __gt__(self, other):
        return self.age > other.age

    def __lt__(self, other):
        return self.age < other.age

p1 = Person('mohamed', 21)
p2 = Person('Ali', 23)
print(p1)
print(p1.name)
print(p1.age)

print(p1 > p2)
```

Explanation:

1. **Initialization (`__init__` method)**: The `__init__` method initializes the attributes `name` and `age` for instances of the `Person` class.
2. **String Representation (`__str__` method)**: The `__str__` method returns a formatted string representing the `Person` instance.
3. **Comparison Methods (`__gt__` and `__lt__` methods)**: These methods define how to compare two `Person` instances based on their `age` attribute. `__gt__` (greater than) and `__lt__` (less than) are implemented to compare the `age` attribute.
4. **Usage**: Two instances, `p1` and `p2`, are created and compared. The output shows the string representation of `p1`, its attributes, and the comparison result. The comparison is based on the `age` attribute, so `p1 > p2` will return `False` because 21 is less than 23.

File 2: Dataclass Implementation

```python
from dataclasses import dataclass

@dataclass(order=True)
class Person:
    name: str
    age: int

    def __str__(self) -> str:
        return self.name

p1 = Person(21, 'mohamed')
p2 = Person(23, 'Ali')
```

```
print(p1 > p2)
```

Explanation:

1. **Dataclass Decorator (`@dataclass(order=True)`)**: The `@dataclass` decorator automatically generates the `__init__`, `__repr__`, `__eq__`, `__lt__`, `__le__`, `__gt__`, and `__ge__` methods. The `order=True` parameter allows instances to be compared based on the order of fields.
2. **String Representation (`__str__` method)**: This method is overridden to return only the `name` of the `Person` instance.
3. **Comparison**: Instances `p1` and `p2` are compared. The comparison is based on the order of fields. Since `age` is the second field in this case, instances are compared based on `age`. Thus, `p1 > p2` will return `False` because 21 is less than 23.

Arabic Explanation

الملف الأول: تنفيذ الفئة المخصصة

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

    def __str__(self):   # for edit format string
        return f'Person = ( name - {self.name} , age - {self.age})'

    def __gt__(self, other):
        return self.age > other.age

    def __lt__(self, other):
        return self.age < other.age

p1 = Person('mohamed', 21)
p2 = Person('Ali', 23)
print(p1)
print(p1.name)
print(p1.age)

print(p1 > p2)
```

التفسير:

1. **تعيين الخصائص (`__init__` method)**: يقوم الدالة `__init__` بتعيين خصائص `name` و `age` لكائنات الفئة `Person`.
2. **تمثيل النص (`__str__` method)**: تقوم الدالة `__str__` بإرجاع نص ممثل لكائن الفئة `Person`.
3. **طرق المقارنة (`__gt__` و `__lt__` methods)**: هذه الطرق تحدد كيفية مقارنة كائنات `Person` بناءً على خاصية `age`. يتم تنفيذها لمقارنة خاصية `age` (أصغر من) `__lt__` و (أكبر من) `__gt__`.
4. **الاستخدام**: يتم إنشاء كائنين، `p1` و `p2`، ويتم مقارنتهما. المخرجات تظهر النص الممثل لـ `p1` المقارنة. المقارنة وخصائصه ونتيجة المقارنة. لأن 21 أصغر من 23 ستعيد `p1 > p2` `False`، لذا تعتمد على خاصية `age`.

الملف الثاني: تنفيذ باستخدام الفئة البيانات

```
from dataclasses import dataclass

@dataclass(order=True)  # add this line for Decorator  and add order for < or >
or = or <= or etc ....
class Person:
    name: str
    age: int

    def __str__(self) -> str:  # for edit format string
        return self.name

p1 = Person(21, 'mohamed')
p2 = Person(23, 'Ali')

print(p1 > p2)
```

التفسير:

1. **الديكورتر** `@dataclass(order=True)` : الديكورتر `@dataclass` يقوم تلقائيًا بإنشاء دوال `__init__`، `__repr__`، `__eq__`، `__lt__`، `__le__`، `__gt__`، و `__ge__`. المعامل `order=True` يسمح بمقارنة الكائنات بناءً على ترتيب الخصائص.
2. **تمثيل النص (`__str__` method)**: تم تجاوز هذه الطريقة لإرجاع فقط اسم الكائن `Person`.
3. **المقارنة**: يتم مقارنة الكائنين `p1` و `p2`. نظرًا لأن خاصية `age` هي الثانية في هذه الحالة، يتم المقارنة تعتمد على ترتيب الخصائص. لذا `p1 > p2` ستعيد `False` لأن 21 أصغر من 23.