```python
# class code, 3/8/2020

class character:

    def __init__(self, hp, level):
        self.hp = hp
        self.level = level

    def universal_intro(self):
        print(f"I am {self.__class__.__name__}, level {self.level}.")

    def intro(self):
        print(f"I am character, level {self.level}.")

class hero(character):
    def __init__(self, hp, level, item):
        super().__init__(hp, level)
        self.item = item

    def intro(self):
        print(f"I am hero, I have {self.item}.")

class monster(character):

    # no need for super() if we do not initialize the subclass
    # that is, we are not using def __init__() method here

    def intro(self):
        print(f"I am monster, level {self.level}.")

class npc(character):
    def intro(self):
        print(f"I am npc, level {self.level}.")

h = hero(10, 1, "sword")
m = monster(5, 2)
n = npc(1, 9)

chars = [h, m, n]
print()
print("Using subclass methods: polymorphism")
print("Use help(subclass_name) to see MRO")
print("to confirm that child method comes first")
print("Child's method with same name overrides the parent's method")
```

```
for i in chars:
    i.intro()

print("Using the parent class method; this is NOT polymorphism")
for i in chars:
    i.universal_intro()

print()
print("Using isinstance()...")
print(isinstance(h, hero))
print(isinstance(h, character))
print(isinstance(m, hero))

print()
print("Using issubclass()...")
print(issubclass(hero, character))
print(issubclass(character, hero))
print(issubclass(hero, monster))
print(issubclass(hero, hero))

print("To see MRO use help(): ")
help(hero)
```