Learning Objectives - Mutability

- **Define the term mutable**
- **Construct a function to modify instance variables (attributes)**

Mutability and Functions

Mutability

Objects are mutable, which means that objects (specifically their attributes) can change value. Think of a video game; the main character in the game is constantly changing. It could be their position on the screen, the score, their health, the items in their inventory, etc. Imagine a simple class called `Player`. A newly instantiated `Player` object has a `health` of 100, a `score` of 0, and starts on `level` 1. This object can lose health, increase their score, and advance levels.

```
class Player:
  """Simple player class"""
  def __init__(self, health=100, score=0, level=1):
    self.health = health
    self.score = score
    self.level = level

player1 = Player()
print(f"This player has {player1.health} health, a score of
        {player1.score}, and is on level {player1.level}.")
player1.health -= 10
player1.score += 25
player1.level += 1
print(f"This player has {player1.health} health, a score of
        {player1.score}, and is on level {player1.level}.")
```

challenge

Try these variations:

- Change the health of `player1` to 0?
- Print the status of `player1` once their `health` is 0?

▼ **One Possible Solution**

```
print(f"This player is dead. They died on level
        {player1.level} with a score of
        {player1.score}.")
```

Functions and Objects

One of the benefits of functions is code reusability. The example above has a repetition of the `print` statement. This is a good opportunity to use a function.

```
class Player:
  """Simple player class"""
  def __init__(self, health=100, score=0, level=1):
    self.health = health
    self.score = score
    self.level = level

def print_player(p):
  """Print the status of a player"""
  print(f"This player has {p.health} health, a score of
        {p.score}, and is on level {p.level}.")

player1 = Player()
print_player(player1)
player1.health -= 10
player1.score += 25
player1.level += 1
print_player(player1)
```

Using a function to print the status of`player1` may not seem like it was worth the effort to change the code. But when these functions become more complex, The efficiency becomes clear.

```
class Player:
  """Simple player class"""
  def __init__(self, health=100, score=0, level=1):
    self.health = health
    self.score = score
    self.level = level

def print_player(p):
  """Print the status of a player"""
  if p.health <= 0:
    print(f"This player is dead. They died on level {p.level}
        with a score of {p.score}.")
  else:
    print(f"This player has {p.health} health, a score of
        {p.score}, and is on level {p.level}.")

player1 = Player()
print_player(player1)
player1.health = 0
player1.score += 25
player1.level += 1
print_player(player1)
```

challenge

Can you:

- Create a function to change a player's health?
  - ▼ **One possible solution**

```
def change_health(p, amount):
  """Change a player's health"""
  p.health += amount
```

- Create a function to change a player's level?
  - ▼ **One possible solution**

```
def change_level(p):
  """Change a player's level"""
  p.level += 1
```

Changing Objects with Functions Formative Assessment 1

Changing Objects with Functions Formative Assessment 2