Class Inheritance and Slicing:

Class Inheritance:
Process of inheriting behaviour and appearance from an existing class is known as Class Inheritance. Both appearance (attributes) and behaviour (methods) can be inherited.

```
class Fish:
        def __init__(self):
```

In order for the fish class to inherit another class (say, Animal), all we have to do is add the name of the class in a set of parentheses (the name of the class it's inheriting from) and add super() in order to inherit everything of the parent class (the attributes and methods).

```
class Fish(Animal):
        def __init__(self):
        super().__init__()
```

E.g.:

```
Class Animal:
        def __init__(self):
                self.num_eyes = 2
        def breathe(self):
                print("Inhale, Exhale.")

class Fish(Animal):
        def __init__(self):
                super().__init__()
        def breathe(self):
                super().breathe()
                print("Doing this under water")
        def swim(self):
                print("Moving in water.")
```

Slicing:

```
piano_keys  = ["a", "b", "c", "d", "e", "f", "g"]
```

```
| a |  b | c  | d | e | f  |

0   1    2    3   4   5    6
```

```
piano_keys[2:5]
--- ["c", "d", "e"]

piano_keys[2:]
--- ["c", "d", "e", "f", "g"]

piano_keys[:5]
---  ["a", "b", "c", "d", "e"]

piano_keys[2:5:2]
---["c", "e"]
```

piano_keys[::2]   (go from beginning to the end & skip every 2nd item)
-- [“a”, “c”, “e”, “g”]

piano_keys[::-1]   (reverse the list)
--- [“f”, “e”, “d”, “c”, “b”, “a”]

NOTE: We can use the same method of slicing to work with our tuples.