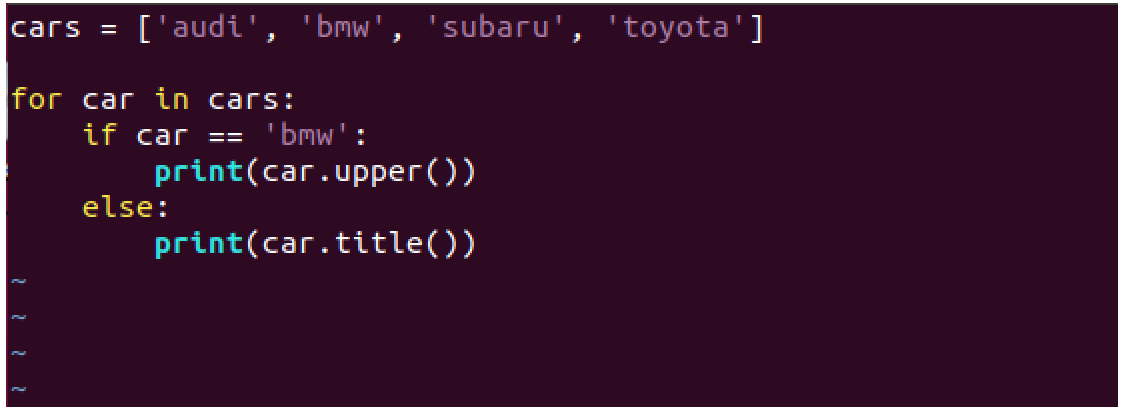
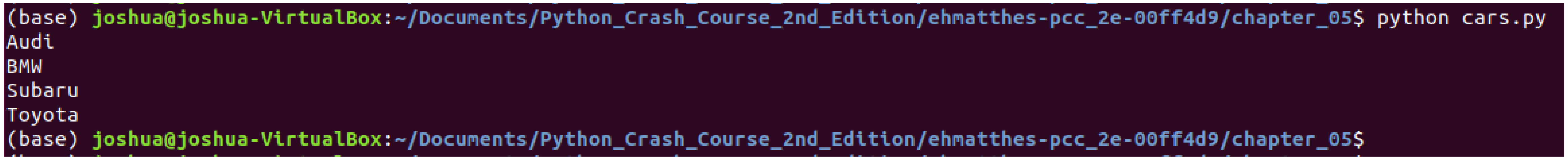
If Statements

A Simple Example



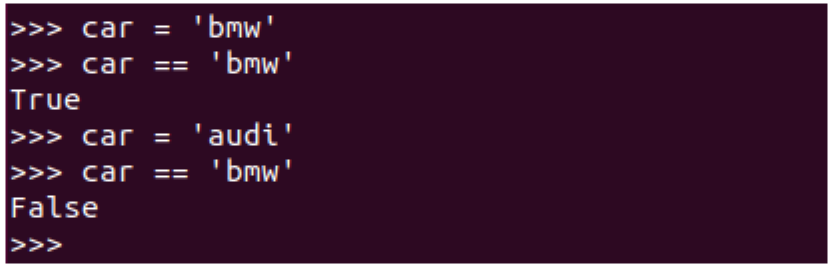
```
cars = ['audi', 'bmw', 'subaru', 'toyota']

for car in cars:
    if car == 'bmw':
        print(car.upper())
    else:
        print(car.title())
~
~
~
~
```



```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$ python cars.py
Audi
BMW
Subaru
Toyota
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$
```

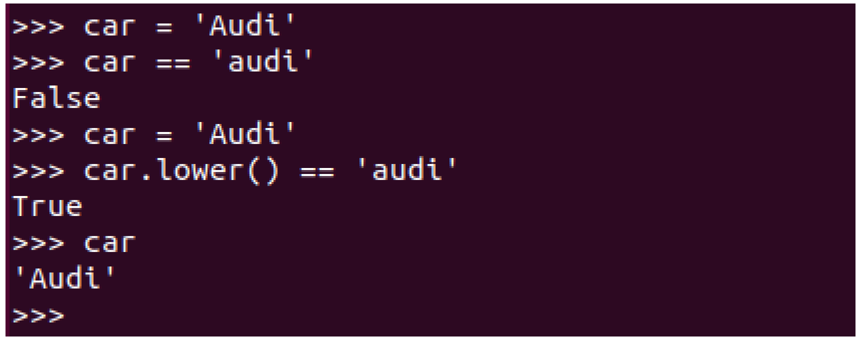
Conditional Tests

Checking for Equality



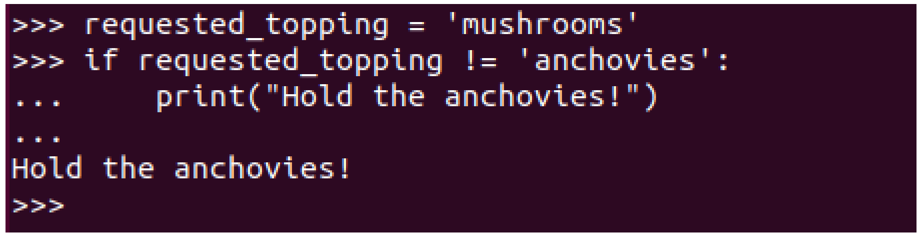
```
>>> car = 'bmw'
>>> car == 'bmw'
True
>>> car = 'audi'
>>> car == 'bmw'
False
>>>
```

Ignoring Case When Checking for Equality



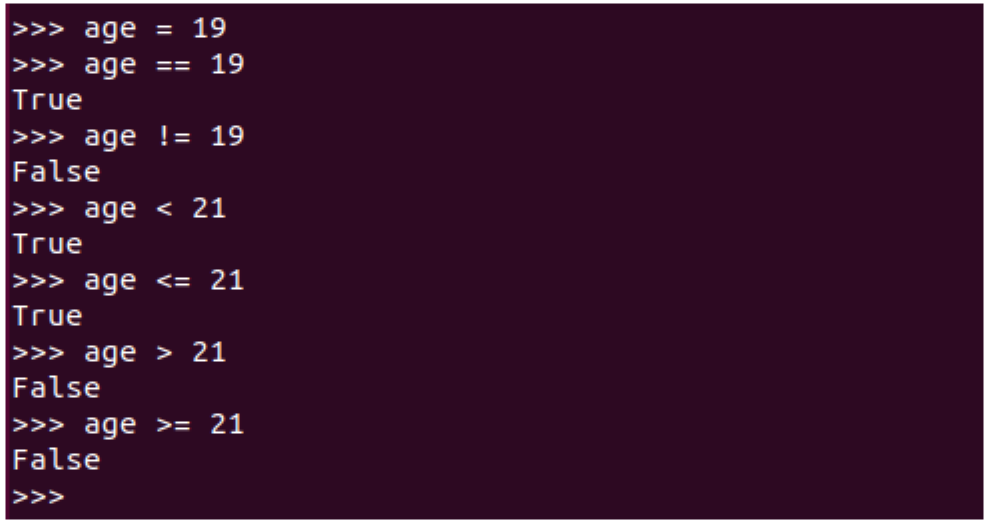
```
>>> car = 'Audi'
>>> car == 'audi'
False
>>> car = 'Audi'
>>> car.lower() == 'audi'
True
>>> car
'Audi'
>>>
```

Checking for Inequality



```
>>> requested_topping = 'mushrooms'
>>> if requested_topping != 'anchovies':
...     print("Hold the anchovies!")
...
Hold the anchovies!
>>>
```

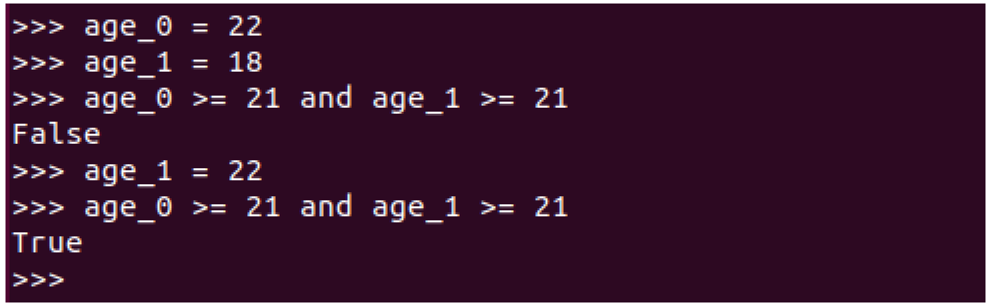
Numerical Comparisons



```
>>> age = 19
>>> age == 19
True
>>> age != 19
False
>>> age < 21
True
>>> age <= 21
True
>>> age > 21
False
>>> age >= 21
False
>>>
```

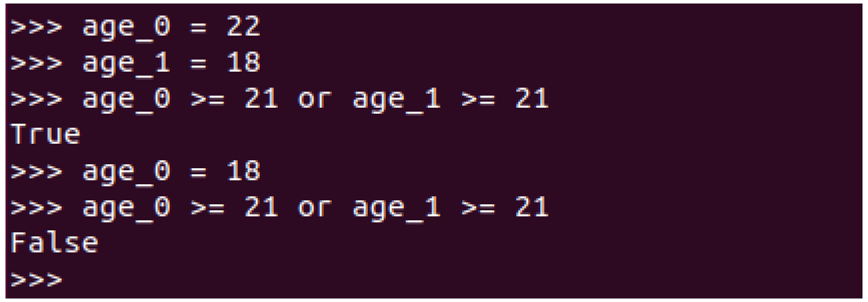
Checking Multiple Conditions

- Using `and` to Check Multiple Conditions



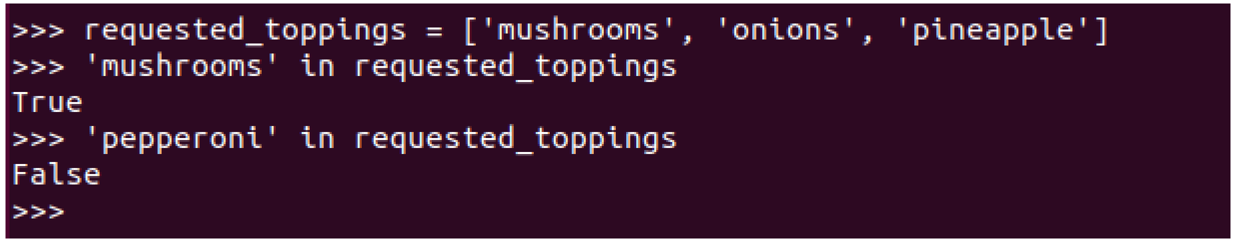
```
>>> age_0 = 22
>>> age_1 = 18
>>> age_0 >= 21 and age_1 >= 21
False
>>> age_1 = 22
>>> age_0 >= 21 and age_1 >= 21
True
>>>
```

- Using `or` to Check Multiple Conditions



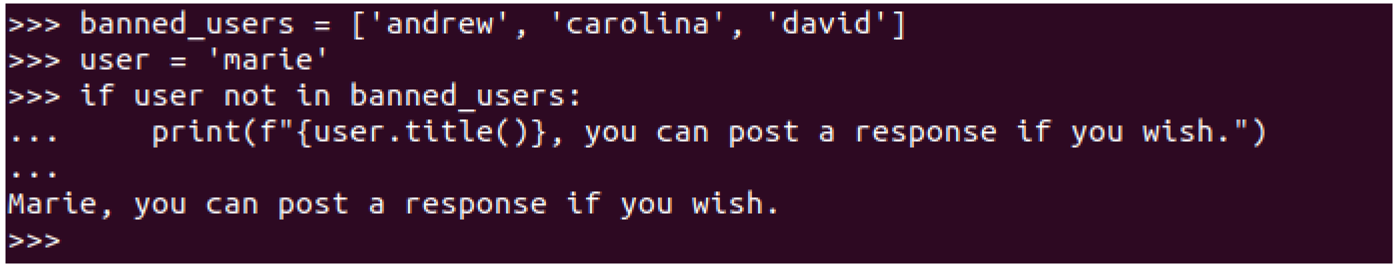
```
>>> age_0 = 22
>>> age_1 = 18
>>> age_0 >= 21 or age_1 >= 21
True
>>> age_0 = 18
>>> age_0 >= 21 or age_1 >= 21
False
>>>
```

Checking Whether a Value Is in a List



```
>>> requested_toppings = ['mushrooms', 'onions', 'pineapple']
>>> 'mushrooms' in requested_toppings
True
>>> 'pepperoni' in requested_toppings
False
>>>
```

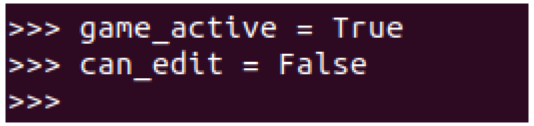
Checking Whether a Value Is Not in a List



```
>>> banned_users = ['andrew', 'carolina', 'david']
>>> user = 'marie'
>>> if user not in banned_users:
...     print(f"{user.title()}, you can post a response if you wish.")
...
Marie, you can post a response if you wish.
>>>
```

Boolean Expressions

- A *Boolean expression* is just another name for a conditional test.
- A *Boolean value* is either `True` or `False`, just like the value of a conditional expression after it has been evaluated.

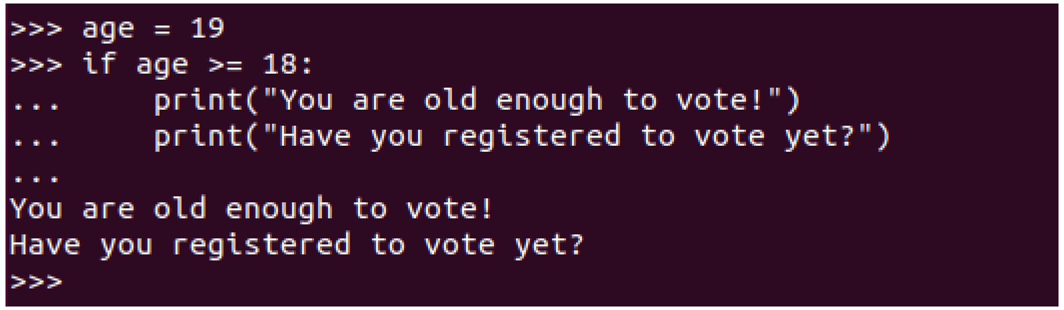


```
>>> game_active = True
>>> can_edit = False
>>>
```

if Statements

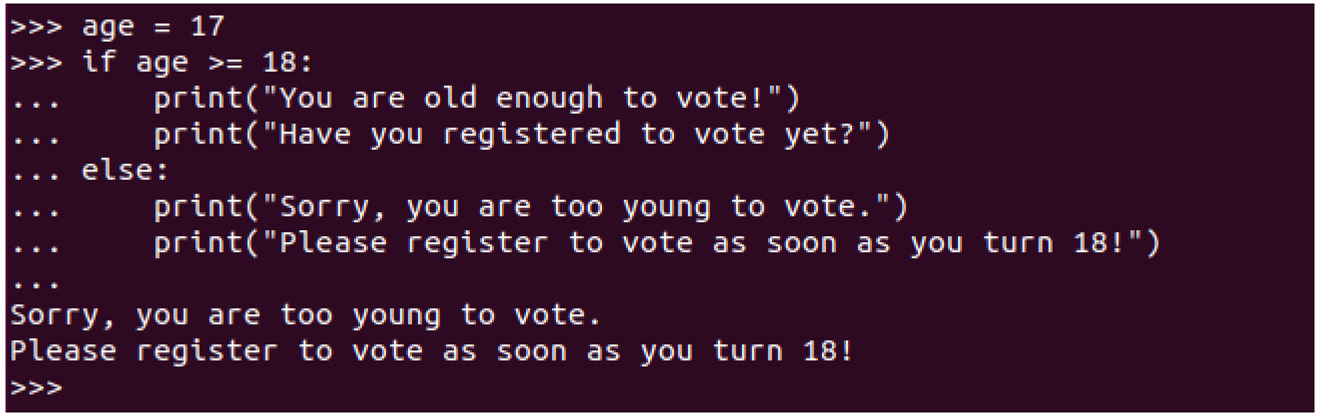
Simple if Statements

- `if conditional_test:`
  `    do something`



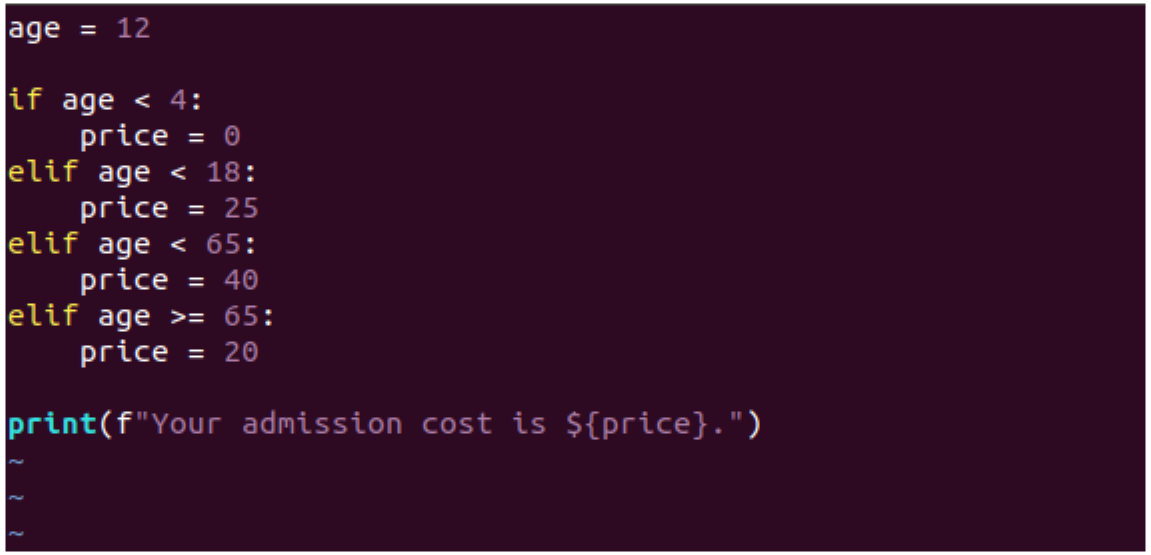
```
>>> age = 19
>>> if age >= 18:
...     print("You are old enough to vote!")
...     print("Have you registered to vote yet?")
...
You are old enough to vote!
Have you registered to vote yet?
>>>
```

if-else Statements

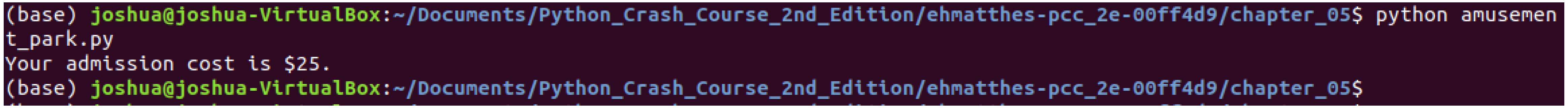


```
>>> age = 17
>>> if age >= 18:
...     print("You are old enough to vote!")
...     print("Have you registered to vote yet?")
... else:
...     print("Sorry, you are too young to vote.")
...     print("Please register to vote as soon as you turn 18!")
...
Sorry, you are too young to vote.
Please register to vote as soon as you turn 18!
>>>
```

The if-elif-else Chain

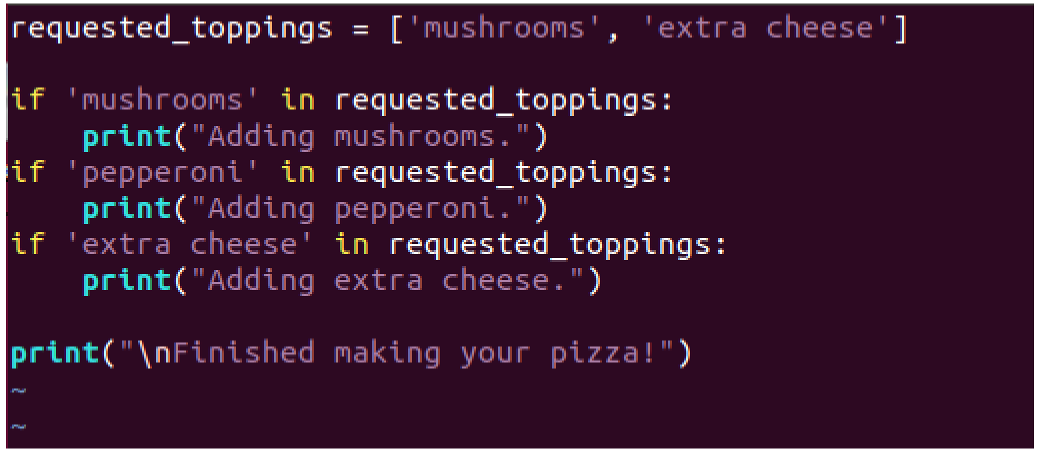


```
age = 12

if age < 4:
    price = 0
elif age < 18:
    price = 25
elif age < 65:
    price = 40
elif age >= 65:
    price = 20

print(f"Your admission cost is ${price}.")
~
~
~
```

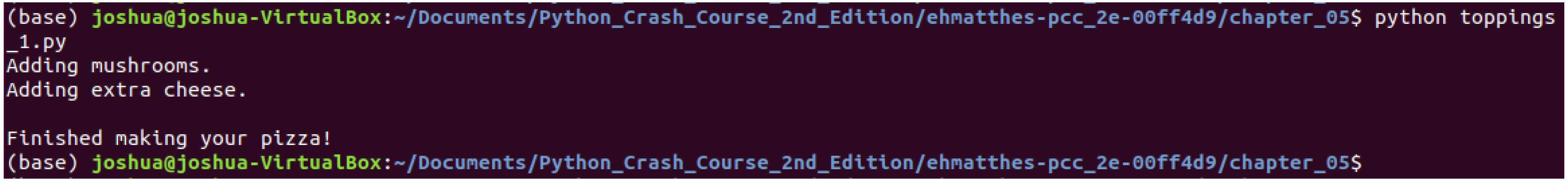


```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$ python amusemen
t_park.py
Your admission cost is $25.
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$
```

Testing Multiple Conditions



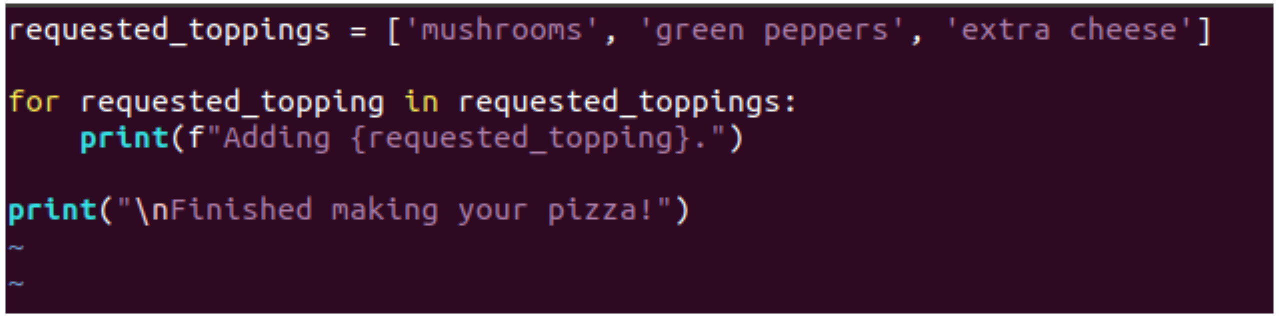
```
requested_toppings = ['mushrooms', 'extra cheese']

if 'mushrooms' in requested_toppings:
    print("Adding mushrooms.")
if 'pepperoni' in requested_toppings:
    print("Adding pepperoni.")
if 'extra cheese' in requested_toppings:
    print("Adding extra cheese.")

print("\nFinished making your pizza!")
~
~
```



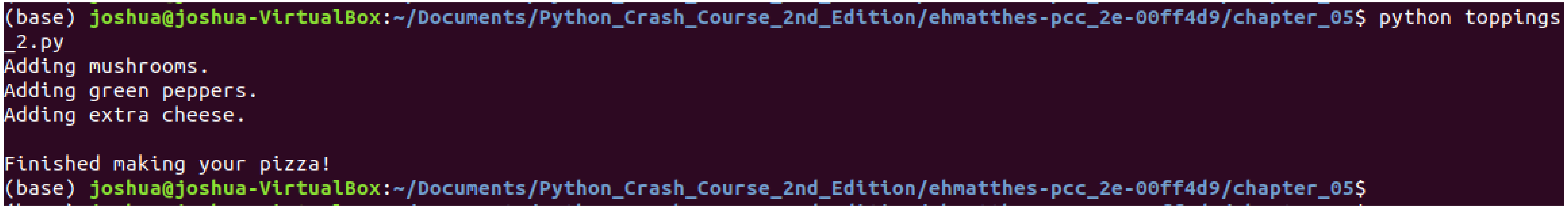
```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$ python toppings
_1.py
Adding mushrooms.
Adding extra cheese.

Finished making your pizza!
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$
```

Using if Statements with Lists

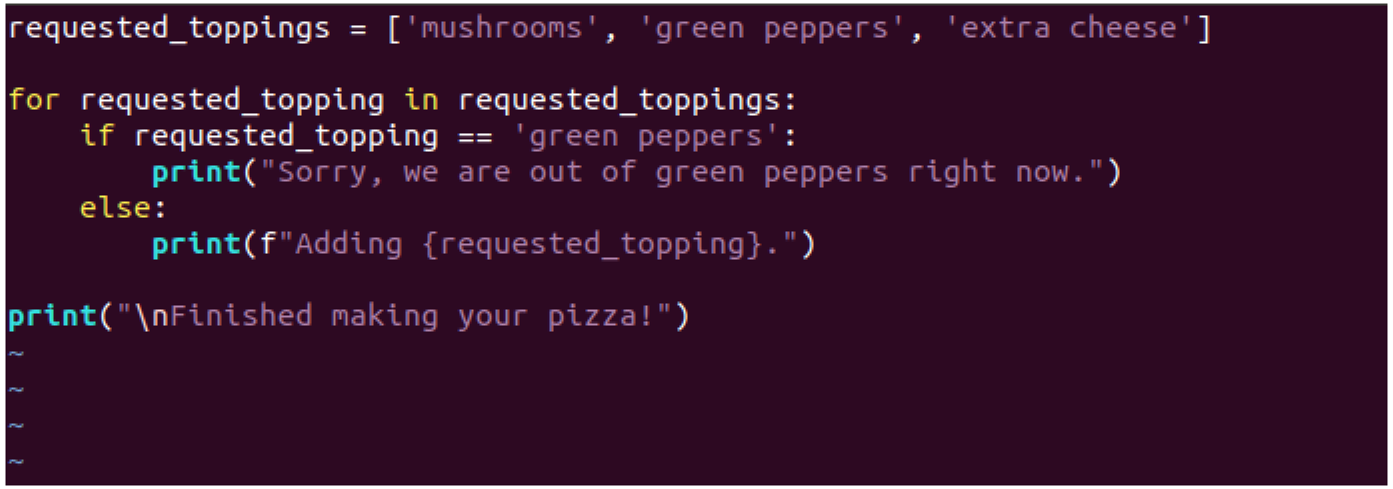
Checking for Special Items



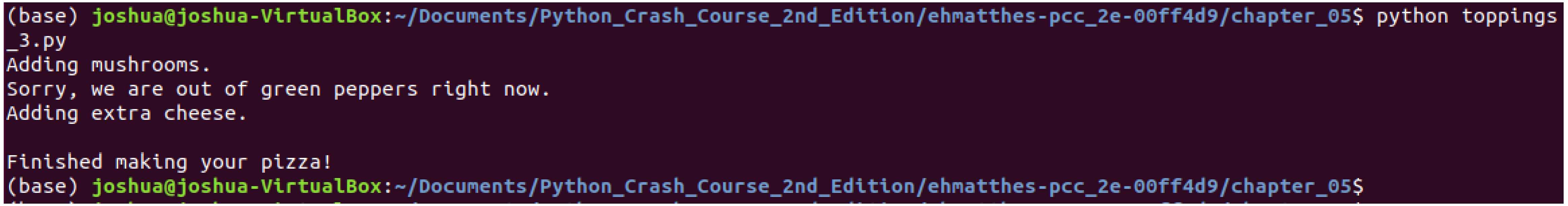
```
requested_toppings = ['mushrooms', 'green peppers', 'extra cheese']

for requested_topping in requested_toppings:
    print(f"Adding {requested_topping}.")

print("\nFinished making your pizza!")
~
~
```



```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$ python toppings
_2.py
Adding mushrooms.
Adding green peppers.
Adding extra cheese.

Finished making your pizza!
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$
```

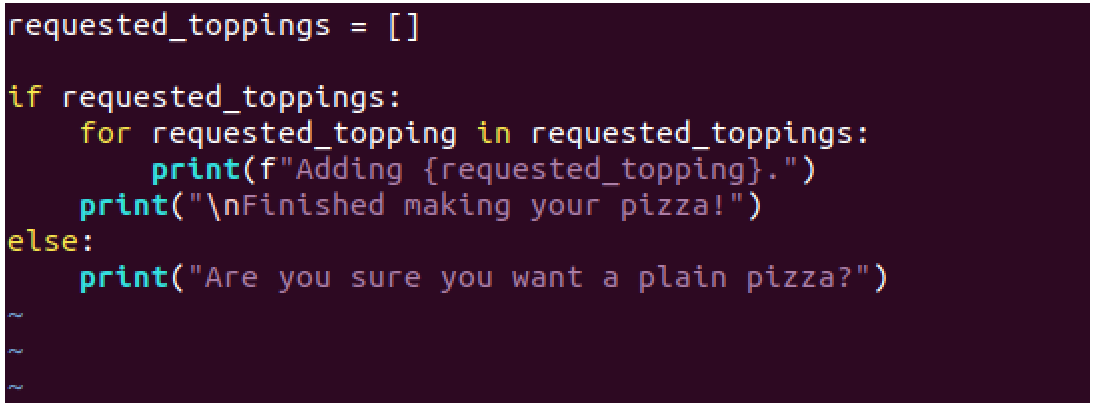


```
requested_toppings = ['mushrooms', 'green peppers', 'extra cheese']

for requested_topping in requested_toppings:
    if requested_topping == 'green peppers':
        print("Sorry, we are out of green peppers right now.")
    else:
        print(f"Adding {requested_topping}.")

print("\nFinished making your pizza!")
~
~
~
~
```

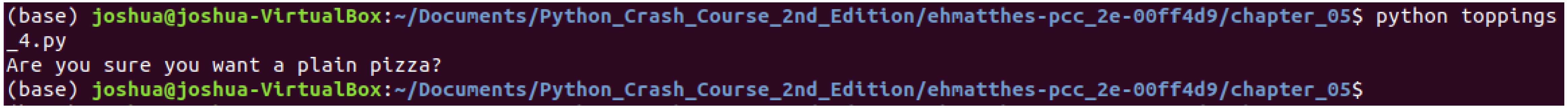


```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$ python toppings
_3.py
Adding mushrooms.
Sorry, we are out of green peppers right now.
Adding extra cheese.

Finished making your pizza!
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$
```

Checking That a List Is Not Empty

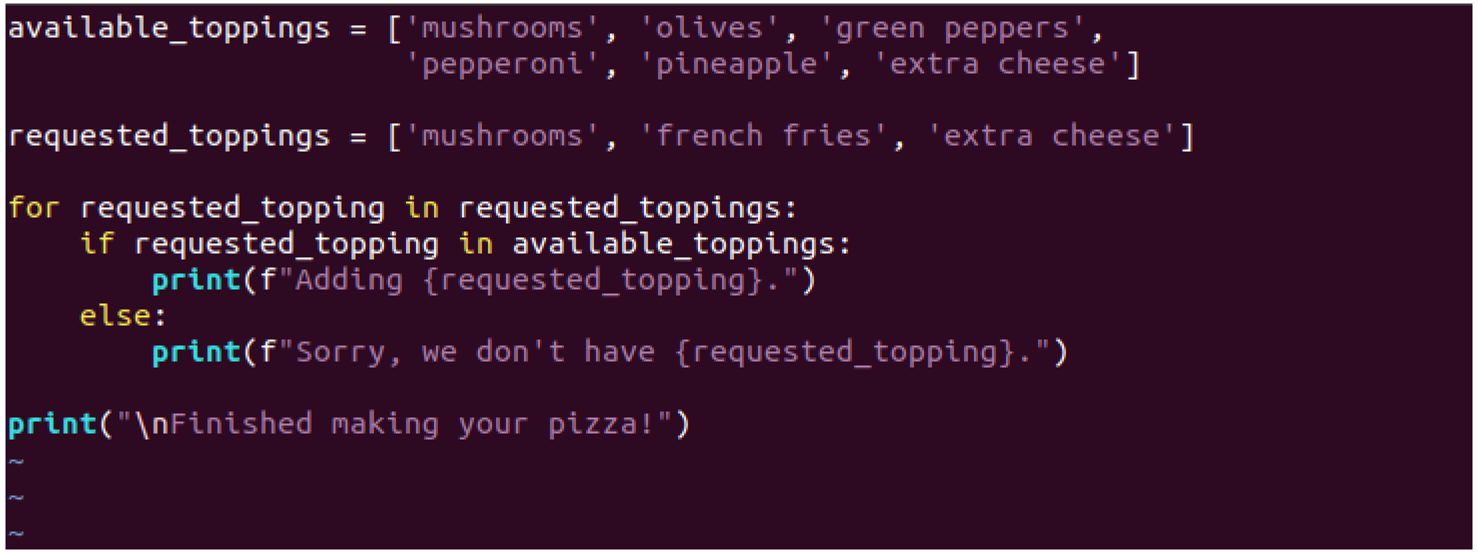


```
requested_toppings = []

if requested_toppings:
    for requested_topping in requested_toppings:
        print(f"Adding {requested_topping}.")
    print("\nFinished making your pizza!")
else:
    print("Are you sure you want a plain pizza?")
~
~
~
```

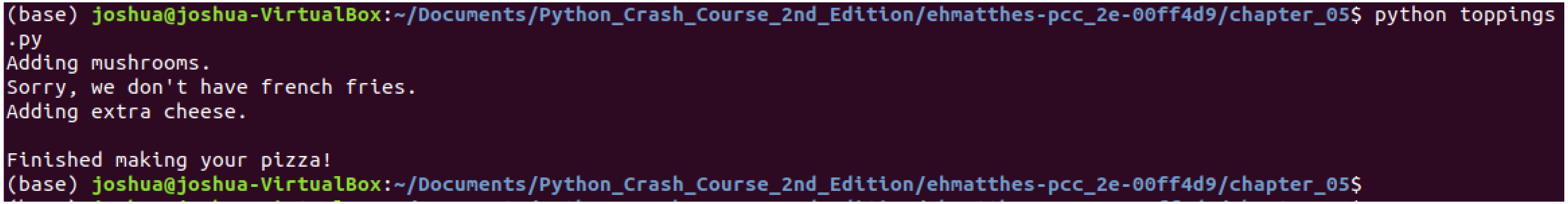


```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$ python toppings
_4.py
Are you sure you want a plain pizza?
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$
```

Using Multiple Lists



```
available_toppings = ['mushrooms', 'olives', 'green peppers',
                      'pepperoni', 'pineapple', 'extra cheese']

requested_toppings = ['mushrooms', 'french fries', 'extra cheese']

for requested_topping in requested_toppings:
    if requested_topping in available_toppings:
        print(f"Adding {requested_topping}.")
    else:
        print(f"Sorry, we don't have {requested_topping}.")

print("\nFinished making your pizza!")
~
~
~
```



```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$ python toppings
.py
Adding mushrooms.
Sorry, we don't have french fries.
Adding extra cheese.

Finished making your pizza!
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_05$
```

Styling Your if Statements

- The only recommendation PEP 8 provides for styling conditional tests is to use a single space around comparison operators, such as `==`, `>=`, `<=`.
- For example:

```
if age < 4:
```

  is better than:

```
if age<4:
```

- Such spacing does not affect the way Python interprets your code; it just makes your code easier for you and others to read.