
# Working With Lists (Chapter 4)

In [4]:
```
# Looping through an entire list
fruits=['apple','banana','pear']
for fruit in fruits:
    print(fruit)

# the loop variable does not have to be related to the list name
for x in fruits:
    print(x) # but this isn't very easy to read

# TODO Print a list of your favorite songs
songs = ['cruel summer', '16', '7 summers']
for song in songs:
    print(song)
```

```
apple
banana
pear
apple
banana
pear
cruel summer
16
7 summers
```

In [7]:
```
# you can put as much code as you want in a loop
fruits=['apple','banana','pear']
for fruit in fruits:
    print(f"That looks like one tasty {fruit}!")
    print('\tmunch, munch, munch') # an extra blank line

# TODO write your own multi-line for loop using your song list
songs = ['cruel summer', '16', '7 summers']
for song in songs:
    print(song)
    print('\trocks!')
```

```
That looks like one tasty apple!
        munch, munch, munch
That looks like one tasty banana!
        munch, munch, munch
That looks like one tasty pear!
        munch, munch, munch
cruel summer
        rocks!
16
        rocks!
7 summers
        rocks!
```

In [8]:
```python
# do add code after the loop, simply do not indent it
fruits=['apple','banana','pear']
for fruit in fruits:
    print(f"That looks like one tasty {fruit}!")
    print("\tmunch, munch, munch")
print("Now I'm full!") # only runs once
```

```
That looks like one tasty apple!
        munch, munch, munch
That looks like one tasty banana!
        munch, munch, munch
That looks like one tasty pear!
        munch, munch, munch
Now I'm full!
```

In [10]:
```python
# Be careful with your identation!
# TODO fix each of the indentation errors below

for fruit in fruits:
    print(fruit)

    print("I like fruit!")
```

```
apple
I like fruit!
banana
I like fruit!
pear
I like fruit!
```

## Making Numerical Lists

In [13]:
```python
# the range() function generates a list of numbers
for value in range(5):
    print(value) # 0..4, DOES NOT INCLUDE 5!

print("\n")

# you can specify the start and end points
for value in range(2,10):
    print(value)

print("\n")

# you can also specify the step size
```

```
for value in range(2,10,2):
    print(value)

# TODO
# print a countdown from 10 to 0 and then print "Blast off!"
print("\n")
for value in range(10,-1,-1):
    print(value)
print('Blast off!')
# count to 100 by 10's
print("\n")
for value in range(0,100,10):
    print(value)
```

```
0
1
2
3
4


2
3
4
5
6
7
8
9


2
4
6
8


10
9
8
7
6
5
4
3
2
1
0
Blast off!


0
10
20
30
40
50
60
70
80
90
```

`range()` is a generator, you can use the `list()` function to convert it into a typical list variable

In [14]:
```
values = range(10)
print(values) # probably not what you want

values = list(range(10)) # convert range output into a list
print(values)
```

```
range(0, 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [17]:
```
# built-in functions provide basic statistics for numeric lists
values = [-100,10,4,56,-10]

print(min(values))

print(max(values))

print(sum(values))

## TODO compute the sum of the squares from 1-10 (1+4+9+...+100)
squares = [1,4,9,16,25,36,49,64,81,100]
print(sum(squares))
```

```
-100
56
-40
385
```

In [21]:
```
# list comprehensions can simplify your code
positive_values = [abs(x) for x in [-10,-4,5,8,-6]]
print(positive_values)

# TODO write a list comprehension to compute the sum of the squares from 1-10
sum_squares = [pow(y,2) for y in [1,2,3,4,5,6,7,8,9,10]]
print(sum(sum_squares))
```

```
[10, 4, 5, 8, 6]
385
```

## Working with part of a list

In [22]:
```
# slicing a list
students = ["alice","bob","charlie","dan","edward","frank"]
print(students[1:3]) # does not include 3 (like range)

print(students[3:5]) # does include 3 (again, just like range)

# start at the begining
print(students[:4])

# go to the end
print(students[3:])

# count back from the end
print(students[-2:])
```

```
['bob', 'charlie']
['dan', 'edward']
['alice', 'bob', 'charlie', 'dan']
['dan', 'edward', 'frank']
['edward', 'frank']
```

In [23]:
```
# loops work on slices too
students = ["alice","bob","charlie","dan","edward","frank"]

print("The first students are:")
for student in students[:3]:
    print(f"\t{student}")
print("The last students are:")
```

```
for student in students[3:]:
    print(f"\t{student}")
```

```
The first students are:
        alice
        bob
        charlie
The last students are:
        dan
        edward
        frank
```

## Copying lists

Be careful! Making a copy is not as simple as setting a new variable equal to the old variable

In [24]:
```
# to copy a list make a slice that includes everything
fruits=['apple','banana','pear']

fruit_copy = fruits[:]
fruit_copy.append("strawberry")
print(f"Fruits: {fruits}")
print(f"Copy of Fruits: {fruit_copy}")

print("\n")

# THIS DOES NOT WORK
bad_copy = fruits
bad_copy.append("watermelon")
print(f"Fruits: {fruits} ????")
print(f"Copy of Fruits: {bad_copy}")

# Helpful Hint: Think of variables as "labels" not "storage boxes"
```

```
Fruits: ['apple', 'banana', 'pear']
Copy of Fruits: ['apple', 'banana', 'pear', 'strawberry']


Fruits: ['apple', 'banana', 'pear', 'watermelon'] ????
Copy of Fruits: ['apple', 'banana', 'pear', 'watermelon']
```

## Using `range()` as an index variable

Sometimes you want to work with multpile lists at the same time or only certain items in a list. The `range()` function can be useful in both situations.

In [25]:
```
students = ["alice","bob","charlie","dan","edward","frank"]
foods =    ["apples","bananas","celery","danishes","eggplant","frozen yogurt"]

for i in range(6):
    print(f"{students[i]} likes {foods[i]}")

print("\n")

print("a selection of foods:")
```

```
for i in range(1,6,2):
    print(foods[i])
```

```
alice likes apples
bob likes bananas
charlie likes celery
dan likes danishes
edward likes eggplant
frank likes frozen yogurt


a selection of foods:
bananas
danishes
frozen yogurt
```

## Tuples

Tuples are like lists but they cannot change. You can think of a tuple as a simpler version of a list. Use them when you need constant values.

In [26]:
```
fruit_list = ['apple','banana','pear']
fruit_tuple =  ('apple','banana','pear')

# Indexing works the same for lists and tuples
print(f"I want an {fruit_list[0]}")
print(f"I want an {fruit_tuple[0]}")

# loops and slices work the same for lists and tuples
for fruit in fruit_list[1:]:
    print(fruit)
for fruit in fruit_tuple[1:]:
    print(fruit)

# But you cannot change a tuple
fruit_list.append('cherry') # ok
fruit_tuple.append('cherry') # error!
```

```
I want an apple
I want an apple
banana
pear
banana
pear
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[26], line 16
     14 # But you cannot change a tuple
     15 fruit_list.append('cherry') # ok
---> 16 fruit_tuple.append('cherry')

AttributeError: 'tuple' object has no attribute 'append'
```

---

# Homework Problems

**4-1. Pizzas:** Think of at least three kinds of your favorit pizza. Store these pizza names in a list, and then use a `for` loop to print the name of each pizza.

- Modify your `for` loop to print a sentence using the name of the pizza instead of printing just the name of the pizza. For each pizza you should have one line of output containing a simple statement like *I like pepperoni pizza*.
- Add a line at the end of your program, outside the `for` loop, that states how much you like pizza. The output should consist of three or more lines about the kinds of pizza you like and then an additional sentence, such as *I really love pizza!*

In [29]:
```
pizza = ['cheese', 'pepperoni', 'veggie']
for flavor in pizza:
    print(f"I like {flavor} pizza.")
print("I really love pizza!")
```

```
I like cheese pizza.
I like pepperoni pizza.
I like veggie pizza.
I really love pizza!
```

**4-2. Animals:** Think of at least three different animals that have a common characteristic. Store the names of these animals in a list, and then use a `for` loop to print out the name of each animal.

- Modify your program to print a statement about each animal, such as *A dog would make a great pet.*
- Add a line at the end of your program stating what these animals have in common. You could print a sentence such as *Any of these animals would make a great pet!*

In [34]:
```
animals = ['crab', 'whale', 'shark']
for animal in animals:
    print(f"A {animal} would not make a good pet.")
print("All of these animals live in the ocean.")
```

```
A crab would not make a good pet.
A whale would not make a good pet.
A shark would not make a good pet.
All of these animals live in the ocean.
```

**4-3 Counting to Twenty:** Use a `for` loop to print the numbers from 1 to 20, inclusive

In [36]:
```
for value in range(1,21):
    print(value)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

**4-4. One Hundred:** Make a list of the numbers from one to one hundred, and then use a `for` loop to print the numbers

In [40]:
```
nums = range(1,101)
for numbers in nums:
    print(numbers)
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

```
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

**4-5. Summing a Million:** Make a list of the numbers from one to one million, and then use `min()` and `max()` to make sure your list actually starts at one and ends at one million. Also use the `sum()` function to see how quickly Python can add numbers.

In [48]:
```
nmbers = list(range(1, 1000001))
print(min(nmbers))
print(max(nmbers))
print(sum(nmbers))
```

```
1
1000000
500000500000
```

**4-6. Odd Numbers**: Use the third argument of the `range()` function to make a list of the odd numbers from 10 to 20. Use a `for` loop to print each number.

In [49]:
```
odd = list(range(11,20,2))
for num in odd:
    print(num)
```

```
11
13
15
17
19
```

**4-7. Threes:** Make a list of the multiples of 3 from 3 to 30. Use a `for` loop to print the numbers in your list.

In [50]:
```
multis = list(range(3,31,3))
for nums in multis:
    print(nums)
```

```
3
6
9
12
15
18
21
24
27
30
```

**4-8. Cubes:** A number raised to the third power is called a *cube*. For example, the cube of 2 is written as 2**3 in Python. Make a list of the first 10 cubes (that is, the cube of each integer from 1 through 10), and use a `for` loop to print out the value of each cube.

In [54]:
```
nums = [1,8,27,64,125,216,343,512,729,1000]
for numbers in nums:
    print(numbers)
```

```
1
8
27
64
125
216
343
512
729
1000
```

**4-9. Cube Comprehension:** Use a list comprehension to generate a list of the first 10 cubes.

In [53]:
```
start = [1,2,3,4,5,6,7,8,9,10]
for num in start:
    operation = pow(num,3)
    print(operation)
```

```
1
8
27
64
125
216
343
512
729
1000
```

**4-10: Slices:** Using one of the programs you wrote above, add several lines to the end of the program that do the following:

- Print the message *The first three items in the list are:*. Then use a slice to print the first three items from that programs's list
- Print the message *Three items from the middle of the list are:*. Use a slice to print three items from the middle of the list.
- Print the message *The last three items in the list are:*. Use a slice to print the last three items in the list

In [59]:
```
start = [1,2,3,4,5,6,7,8,9,10]
new = []
for num in start:
    new.append(pow(num,3))
print(new)
print(f"The first three items in the list are {new[0:3]}")
print(f"The middle three items in the list are {new[3:6]}")
print(f"The last three items in the list are {new[6:9]}")
```

```
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
The first three items in the list are [1, 8, 27]
The middle three items in the list are [64, 125, 216]
The last three items in the list are [343, 512, 729]
```

**4-11. My Pizzas, Your Pizzas:** Start with your program from **4-1**. Make a copy o fthe list of pizzas, and call it `friend_pizzas`. Then, do the following:

- Add a new pizza to the original list.
- Add a different pizza to the list `friend_pizzas`.
- Prove that you have two separate lists. Print the message *My favorite pizzas are:*, and then use a `for` loop to print the first list. Print the message *My friend's favorite pizzas are:*, and then use a `for` loop to print the second list. Make sure each new pizza is stored in the appropriate list.

In [61]:
```
pizza = ['cheese', 'pepperoni', 'veggie', 'sausage']
friend_pizzas = ['cheese','pepperoni','veggie','pineapple']
print('My favorite pizzas are:')
for flavor in pizza:
    print(flavor)
print('\n')
print("My friend's favorite pizzas are:")
for toppings in friend_pizzas:
```

```
    print(toppings)
```

```
My favorite pizzas are:
cheese
pepperoni
veggie
sausage


My friend's favorite pizzas are:
cheese
pepperoni
veggie
pineapple
```

**4-12. Index Variables:** Make a list of your favorite 5 vacation destinations from highest to lowest. Make another list of how much each vacaction would cost.

- Use the range function to index through your list from least favorite to most favorite destination.
- Use the reverse function to print the same information
- Use the range function to index through both lists to print the cost of each vacation option.

In [72]:
```
destinations = ['Gulf Shores', 'Florida', 'Mountains', 'New York', 'California']
cost = [100,500,200,600,700]
for i in range(4,-1,-1):
    print(destinations[i])
print('\n')
destinations.reverse()
print(destinations)
destinations.reverse()
print(destinations)
print('\n')
for i in range(4,-1,-1):
    print(destinations[i])
    print(cost[i])
    print('\n')
```

```
California
New York
Mountains
Florida
Gulf Shores


['California', 'New York', 'Mountains', 'Florida', 'Gulf Shores']
['Gulf Shores', 'Florida', 'Mountains', 'New York', 'California']


California
700


New York
600


Mountains
200


Florida
500


Gulf Shores
100
```

**4-13: Buffet:** A buffet-style restaurant offers only five basic foods. Think of five simple foods and store them in a tuple.

- Use a `for` loop to print each food the restaurant offers
- The restaurant changes its menu, replacing two of the items with different foods. Add a line that rewrites the tuple, and then use a `for` loop to print each of the items on the revised menu.
- Try to modify one of the items without re-writing the tuple, and make sure that Python rejects the change

In [78]:
```
offered = ('sandwich', 'burger', 'chicken', 'fries', 'fruit')
for i in range(0,5):
    print(offered[i])
print('\n')
offered = ('sandwich', 'shrimp', 'steak', 'fries', 'fruit')
for j in range(0,5):
    print(offered[j])
offered[3] = 'turkey'
```

```
sandwich
burger
chicken
fries
fruit


sandwich
shrimp
steak
fries
fruit
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[78], line 8
      6 for j in range(0,5):
      7     print(offered[j])
----> 8 offered[3] = 'turkey'

TypeError: 'tuple' object does not support item assignment
```

**4-14. PEP 8:** Look through the original PEP 8 style guide at [https://python.org/dev/peps/pep-0003/]. You won't use much of it now, but it might be interesting to skim through it.