Session 4: Lists

Lists are an important part of a program. We would like to store a bunch of items in one variable.

Basics

Suppose we wanted to create a shopping list app. We would like to be able to store our user's shopping list and retrieve the items from it on demand.

This is where Python lists come in. Here is how we create a list:

```
shopping_list = ["bread", "smoked salmon", "cherry tomatoes", "cream
cheese"]
```

In Python, we can also `print` the whole list, to view all the items:

```
shopping_list = ["bread", "smoked salmon", "cherry tomatoes", "cream
cheese"]
print(shopping_list)
```

Or we can access the items one by one:

```
shopping_list = ["bread", "smoked salmon", "cherry tomatoes", "cream
cheese"]
print(shopping_list[0]) # Prints 'bread'
print(shopping_list[1]) # Prints 'smoked salmon'
print(shopping_list[2]) # Prints 'cherry tomatoes'
print(shopping_list[3]) # Prints 'cream cheese'
```

As we can see, a list is an numbered collection of items. An important thing to remember is that in Python **indexing starts at 0**. What that means is that the first item is *at* **index** 0, the second item is at index 1, etc. We access the item at index `i` (where `i` is an integer) by writing `shopping_list[i]` - the name of the list, and then the index in square brackets.

Now what would happen if we try to access the item at index 4 of our shopping list? Let's try:

```
shopping_list = ["bread", "smoked salmon", "cherry tomatoes", "cream
cheese"]
print(shopping_list[4]) # ERROR: list index out of range
```

If you run this code, you would get something called an index error. This is because there is no item at index 4. Note now, that the list has 4 items, but the highest index is 3. In general, a list with `n` items will

have a highest index of `n-1`.

We can get the *length* of the list (i.e. the number of items in it), by calling the `len` function:

```
shopping_list = ["bread", "smoked salmon", "cherry tomatoes", "cream
cheese"]
length = len(shopping_list)
print("The length of the list is: " + str(length))
```

Note that the `len` function returns an `int`, so we need to convert that to a `str` before printing.

Exercises

Easy:

- 1: Create a list of 3 cities you want to visit. Print the whole list on one line. Print the list item by item. Print the length of the list.

Medium:

- 2: Complete the program so that it prints the contents of the shopping list:

  ```
  shopping_list = ["bread", "smoked salmon", "cherry tomatoes", "cream
  cheese"]

  i = 0
  length = len(shopping_list)
  while i < length:
    # Complete this code
  ```

Hard:

- 3: Complete the program so that it prompts the user to input a number `n`, then prints `shopping_list`, item by item, `n` times in a row:

  ```
  shopping_list = ["bread", "smoked salmon", "cherry tomatoes", "cream
  cheese"]
  n = # Complete the rest of the program
  # ...
  ```

  A run of your program might look like:

  ```
  Please enter a number: 2
  bread
  smoked salmon
  ```

```
cherry tomatoes
cream cheese
bread
smoked salmon
cherry tomatoes
cream cheese
```

You can find all the solutions here.

The need for lists

We will discuss why we need lists in this section.

Imagine a world without lists. Then we have to use a different variable for each item in the shopping list, like this:

```
shopping_list_item_1 = "bread"
shopping_list_item_2 = "smoked salmon"
shopping_list_item_3 = "cherry tomatoes"
shopping_list_item_4 = "cream cheese"
```

This is certainly annoying, what's more is that there is no effective way to manipulate all items using loops. For example, if we would like to print everything in the shopping list, this is the only thing we can do...

```
print(shopping_list_item_1)
print(shopping_list_item_2)
print(shopping_list_item_3)
print(shopping_list_item_4)
```

However, with lists, we can store all items in one single variable in one single line.

```
shopping_list = ["bread", "smoked salmon", "cherry tomatoes", "cream
cheese"]
```

And we can print the whole list at once using `print(shopping_list)`. List is certainly a good thing to have in our programming language.

More list operations

We can append an element to a list - basically adding it to the end:

```
# We forgot to add mayo to the list so we can add it later like so:
shopping_list.append("mayo") # shopping_list is now [ "bread", "smoked
salmon", "cherry tomatoes", "cheddar", "mayo" ]
```

Or remove an element from the list:

```
del shopping_list[4] # shopping_list is now [ "bread", "smoked salmon",
"cherry tomatoes", "cheddar" ]
```

Alternatively, you can remove an element by value, so instead of the previous line we could have written:

```
shopping_list.remove("mayo")
```

Do note that this is more computationally expensive because the computer needs to go through the entire list to find the mayo, similarly to how you would go down a list from the start to find it.

You can also join two lists together:

```
shopping_page1 = ["bread", "smoked salmon", "cherry tomatoes"]
shopping_page2 = ["cheddar", "mayo" ]
shopping_list = shopping_page1 + shopping_page2 # shopping_list is now [
"bread", "smoked salmon", "cherry tomatoes", "cheddar", "mayo" ]
```

Note: you may have seen what Python calls lists called *arrays* in other programming languages. In most programming languages there is a difference but it is not relevant to Python or this course.

Exercises:

Expert

- 4: Make your own shopping list program! Begin by prompting a user to enter the number of things they want on their list and then ask them for each separate item. The program should output the items on the list in the same order, each on a separate line, after the user's done with their input.

  Here's a sample run:

  ```
  Welcome to the shopping list app!
  Please enter the size of your shopping list: 3
  Please enter item number 0: bread
  Please enter item number 1: smoked salmon
  Please enter item number 2: cherry tomatoes
  Your shopping list is:
  bread
  smoked salmon
  cherry tomatoes
  ```

  Hint: the next few lines might be helpful:

```
my_list = [] # This is how we create an empty list. I.e. a list with
no items
my_list.append("item1") # This is how we append to the back of our
list
my_list.append("item2") # And again
print(my_list) # Prints '["item1", "item2"]'
print(len(my_list)) # Prints '2'
```

- 5: Modify your program for slightly so that it displays the item number starting from 1 instead of 0.

```
Welcome to the shopping list app!
Please enter the size of your shopping list: 3
Please enter item number 1: bread
Please enter item number 2: smoked salmon
Please enter item number 3: cherry tomatoes
Your shopping list is:
bread
smoked salmon
cherry tomatoes
```

- 6: Do you really need to enter the number n in the beginning? Wouldn't it be more convenient for a user to tell when the shopping list *ends* instead? Make a program that does just that. Here's a sample run:

```
Welcome to the shopping list app!
Please enter item number 1: bread
Please enter item number 2: smoked salmon
Please enter item number 3: cherry tomatoes
Please enter item number 4: end
Your shopping list is:
bread
smoked salmon
cherry tomatoes
```

You can find all the solutions here.

Recap on loops

While loops

```
n = int(input("Please enter a number: "))
i = 1
while i < n:
    print(i)
    i += 1 # Increment i (i becomes bigger by 1).
```

For loops

```
n = int(input("Please enter a number: "))
for i in range(1, n): # including 1 excluding n
    print(i)
```

In general you write a `for` loop like so:

```
for i in range(a, b):
    # including a excluding b
    # this block will repeat (b - a) times
    # you will have access to the number of the current "repeat" with i
```

As you can see `for` loops take care of a lot of things for us - you don't have to define your own `i` and increment it yourself. On the other hand, that makes `for` loops less flexible - there are some cases where you might not want to have a counter or you might want to decrement it.

Loops with lists

What loops are really useful for is *iterating* over lists.

While loops

```
shopping_list = [ "bread", "smoked salmon", "cherry tomatoes", "cream
cheese" ]
n = len(shopping_list)
i = 0
while i < n:
    print("I need to buy " + shopping_list[i])
    i += 1

# This will print out:
# I need to buy bread
# I need to buy smoked salmon
# I need to buy cherry tomatoes
# I need to buy cream cheese
```

For loops

```
shopping_list = [ "bread", "smoked salmon", "cherry tomatoes", "cream
cheese" ]
n = len(shopping_list)
for i in range(0, n):
    print("I need to buy " + shopping_list[i])
```

```
# This will print out:
# I need to buy bread
# I need to buy smoked salmon
# I need to buy cherry tomatoes
# I need to buy cream cheese
```

A further simplication

In fact, we can further simplify that `for` loop, so that we don't need the variables `n` and `i` anymore.

```
shopping_list = [ "bread", "smoked salmon", "cherry tomatoes", "cream
cheese" ]
for item in shopping_list:
    print("I need to buy " + item)

# This will print out:
# I need to buy bread
# I need to buy smoked salmon
# I need to buy cherry tomatoes
# I need to buy cream cheese
```

Exercises

We will do the exercises listed in the `session5` folder. We have not covered functions yet, but you can still do the exercises. For example, if you would like to do exercise 1, you can copy the code in `session5/exercise1.py` to your local machine. Then write your code in the `def` clause.

For example,

```
def sum(xs):
    # return the sum of all the elements in the list xs
    # remove the following line
    print("Write your code here!!!!!!!")
    return 0

def test(test_case, expected):
    actual = sum(test_case)
    if actual == expected:
        print("Passed test for " + str(test_case))
    else:
        print("Didn't pass test for " + str(test_case))
        print("The result was " + str(actual) + " but it should have been "
+ str(expected))

test([], 0)
test([1, 2], 3)
test(range(10), 45)
```

Then when you run your code using Python, it will run some tests on your code automatically, and tell you whether your code is correct.