CptS 111 — Lab #7

Counting `for`-loops, Lists, and the `range()` Function

Learning Objectives:

- Write counting `for`-loops
- Use the `range()` function with integer arguments
- Use the `len()` function with lists and the `range()` function

Prerequisites:

- Ability to use functions and conditionals
- Know basics of counting `for`-loops
- Know basics of the `range()` function

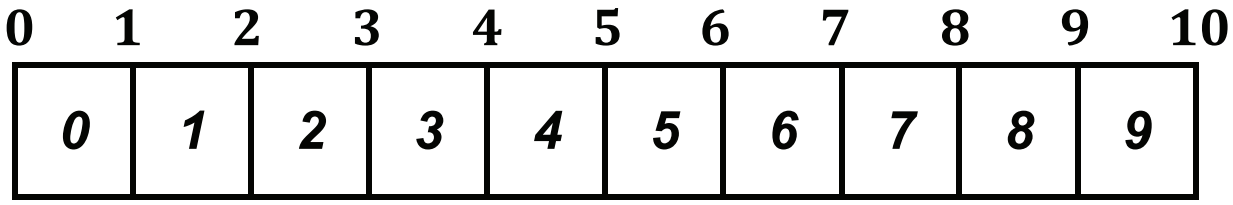
Task 1: The `range()` function and counting for-loops

We can think of `range()` as a function that generates a list of integers.[1] It actually generates the list elements one at a time, but we can force it to generate all the integers at the same time using the `list()` function.[2]

Recall that `range()` can be passed one, two, or three integers or expressions that evaluate to integers. With one argument, `range()` will generate integers starting at zero and stopping at one less than the argument. For example, `list(range(5))` returns the list:

```
[0, 1, 2, 3, 4]
```

The number of elements in this list corresponds to the argument `5` which is the stop value. **The numbers produced by `range()` never include the stop value!** It's important to remember that `range(n)` produces `n` integers from `0` to `n-1`. In the figure below, the integers on top represent stop values, and the integers in the boxes are the values generated by `range()` for a given stop value, e.g., when you pass `range()` a stop value of `5`, it generates the 5 integers shown in the boxes before the `5` on top.



We'll continue our study of the `range()` function in an IDLE Shell window. Read the comments below, and type in all the statements in boldface font. As mentioned in an earlier lab, you can use Alt-p (Windows) or Control-p (macOS) to scroll back to previous commands and, thus, minimize the amount you need to type.

[1] Although we refer to `range()` as a function, we use it as a special function.
[2] We only use `list()` in our examples to see the values generated by `range()`.

```
>>> # We'll start with a single integer argument, i.e., the stop value.
>>> list(range(10))          # stop=10 (default: start=0, increment=1)
>>>
>>> # Next let's consider two integer arguments, start and stop values.
>>> list(range(3, 10))       # start=3, stop=10 (default: increment=1)
>>>
>>> # The three integer arguments are start, stop, and increment.
>>> list(range(3, 10, 3))    # start=3, stop=10, increment=3
>>>
>>> # What happens if the stop value is smaller than the start value?
>>> list(range(10, 0))       # start=10, stop=0 (default: increment=1)
>>>
>>> # However, we can change the increment value to a negative number!
>>> list(range(10, 0, -1))   # start=10, stop=0, increment=-1
>>>
>>> # And if we want the last value to be 0, we use a negative stop
>>> # value.
>>> list(range(10, -1, -1))
>>>
>>> # Note that if we want to change the start value, we must include
>>> # a stop value.  Similarly, if we want to change the increment
>>> # value, we must include both the start and stop values.
>>>
>>> # We can also use negative integers if we want to create a sequence
>>> # of negative values.
>>> list(range(-1, -11, -1))
>>>
>>> # However, for most applications in CptS 111, we'll use only
>>> # positive integers.  Next, answer the following questions by
>>> # trying integer arguments with list(range()).
>>>
>>> # 1. What arguments produce a list of all the even numbers between
>>> #    0 and 10 but not including 10 (note that 0 is an even number).
>>>
>>> # 2. What arguments produce a list of integers that starts at 1
>>>      and ends with 10?  Do you need 2 or 3 arguments?
>>>
>>> # Let's move on to counting for-loops.  The template for a counting
>>> # for-loop is given by:

      for i in range():
          <loop_body>

>>> # where i is the loop variable used for counting.  Let's do some
>>> # examples.
>>> for i in range(5):
        print('Go Cougs!')
>>>
>>> # In the example above, Python made 5 loops, printing Go Cougs!
```

```
>>> # each time it looped.
>>>
>>> for i in range(5):
        print(f'{i+1}. Go Cougs!')
>>>
>>> # The example above is the same as the previous one except we
>>> # actually use the loop variable i in the print() function.  We
>>> # add 1 so the list doesn't start with 0.
>>>
>>> for i in range(10, 0, -1):
        print(i)
>>>
>>> # In the example above, we again use the loop variable.  We
>>> # count down from 10 to 1 using appropriate integers as arguments.
>>>
>>> # Let's next use a counting for-loop with a list of names.
>>> names = ['Sam', 'Mohammed', 'Maria', 'Yan']
>>> len(names)
>>> for i in range(len(names)):
        print(f'Hello, {names[i]}.')
>>>
>>> # In the example above, we see that len(names) is an integer so
>>> # it's perfectly fine to use it as the argument of the range()
>>> # function.  Let's try another list.
>>> nums = []
>>> for i in range(5):
        num = float(input('Enter a decimal number: '))
        nums.append(num)

Enter a decimal number: 3.141
Enter a decimal number: 1.618
Enter a decimal number: 0.5
Enter a decimal number: 4.2
Enter a decimal number: 6.282
>>> nums
[3.141, 1.618, 0.5, 4.2, 6.282]
>>>
>>> # In the example above, we initialize an empty list, prompt the
>>> # user for a decimal number in the for-loop which adds the
>>> # number to the list.  This is repeated for each loop, and
>>> # we end up with a list of 5 decimal numbers!
```

After you've completed this task, show your work to your TA to get credit.

Task 2: Blastoff!

For this task, write a program in the IDLE Editor window with the functions described below. Don't forget to use a header in your program and docstrings in your functions—and a call to `main()`. Save your program as `lab7_t2.py` under Lab7 in your CS folder.

- **`countdown()`**: Void function with one integer parameter, `count`, that counts down from the integer value to `0` and then prints `Blastoff!` as shown in the examples below. Use a counting `for`-loop and three arguments in the `range()` function. Also, be careful where you write the `print()` command to print `Blastoff!`.
- **`main()`**: Void function that prompts the user for a positive integer as shown in the examples and then calls `countdown()` with the integer as its argument.

Examples follow.

```
Enter a positive integer: 3
3
2
1
0
Blastoff!

Enter a positive integer: 5
5
4
3
2
1
0
Blastoff!
```

Once your program is working correctly, run it for your TA to get credit.

Task 3: Creating a list and using it to print an itemized list

For this task, you'll write a program that prompts for the user's favorites (food, activities, etc.), creates a list from the input, and prints out an itemized list of favorites. The three functions you'll need to write are described below. Don't forget the header et al., and save your program as `lab7_t3.py`.

- **`get_faves()`**: This non-void function has a single integer parameter, the number of items to be obtained from the user. It initializes an empty list, uses a counting `for`-loop to prompt and obtain each item from the user as shown in the examples, appends each item to the list, and returns the list to `main()`.
- **`print_list()`**: This void function has one parameter, the list of favorites, and prints an itemized list of the favorites as shown in the examples below.
- **`main()`**: This void function has no parameters. It prompts the user for the number of items they want to enter, and it calls `get_faves()` with the integer value entered by the user as the argument. The list returned to `main()` is then used as the argument to `print_list()`.

```
Enter the number of favorites you want to list: 5
Enter a favorite food, activity, etc.: chocolate croissants
Enter a favorite food, activity, etc.: vanilla sweet cream cold brews
Enter a favorite food, activity, etc.: dark chocolate truffles
Enter a favorite food, activity, etc.: espresso frappucinos
```

```
Enter a favorite food, activity, etc.: cookies of all types

1. chocolate croissants
2. vanilla sweet cream cold brews
3. dark chocolate truffles
4. espresso frappucinos
5. cookies of all types

Enter the number of favorites you want to list: 5
Enter a favorite food, activity, etc.: cruises
Enter a favorite food, activity, etc.: Queen's Bohemian Rhapsody
Enter a favorite food, activity, etc.: working out
Enter a favorite food, activity, etc.: reading
Enter a favorite food, activity, etc.: sleeping

1. cruises
2. Queen's Bohemian Rhapsody
3. working out
4. reading
5. sleeping
```

When your program is working properly, demonstrate it to your TA to get credit.

Task 4: Joining a list of strings

For this task, you'll use the IDLE Editor window to write a program that will prompt for words in a title and then display the title using either the default value for `sep` or one provided by the user. The functions you need to code are described below. Save your program as `lab7_t4.py` under Lab7 in your CS folder.

- **`create_title()`**: A non-void function with one integer parameter `num` that prompts the user `num` times for the words in a title, adds each word to a list, and then returns the list to `main()`. Don't forget to initialize an empty list and use a counting `for`-loop to prompt the user and add the word to the list.
- **`join()`**: A void function with two parameters, a title list and the keyword argument `sep` with a default value of a space. This function should initialize an empty string and then use augmented assignment in a counting `for`-loop to add each word in the list to the string. Don't add the last word in the list to the string or you'll end up with a separator you don't want! See examples below. Hint: Subtract 1 from the length of the list in the argument for the `range()` function, and then add the last word to the string outside the loop (this was demonstrated in class). Remember this in case it shows up on an exam!
- **`main()`**: Void function that prompts the user for the number of words in their title, prompts the user for the separator they want to use (see examples below), and then calls `create_title()` with the number of words as the argument and assigning the result to an lvalue for the list. It should then use an `if-else` construct to call `join()` either with only the list as the argument (i.e., when the default value of `sep` is used) or with the list and the separator as arguments. Hint: The default value of `sep` will be used when a user enters return.

Examples follow.

```
Enter the number of words in your title: 1
Enter your sep [press return for default space]: ***
Enter word of title: Booksmart
Booksmart

Enter the number of words in your title: 3
Enter your sep [press return for default space]:
Enter word of title: Toy
Enter word of title: Story
Enter word of title: 4
Toy Story 4

Enter the number of words in your title: 3
Enter your sep [press return for default space]: -*-
Enter word of title: Toy
Enter word of title: Story
Enter word of title: 4
Toy-*-Story-*-4
```

Once your program is working correctly, run it for your TA to get credit.

You had to combine quite a few concepts in this one: functions, counting `for`-loops, the `range()` function, and conditionals. Well done!