Level 2 **Slicing (Revisited)**

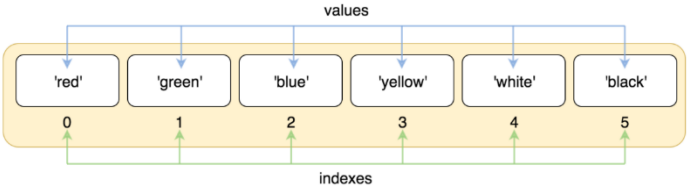
Currently, you should be able to use Python to select items from a **list** or **string** using the following method:

```
name = "Sammy Shark!"
print(name[4])

Output:
y
```

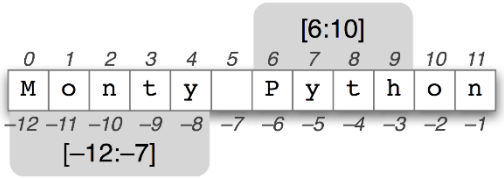
Each element in a **string** or **list** has a location and **address** that we can use to access each element.

```
 colors = ['red', 'green', 'blue', 'yellow', 'white', 'black']
 colors[0]
'red'
 colors[1]
'green'
 colors[5]
'black'
```



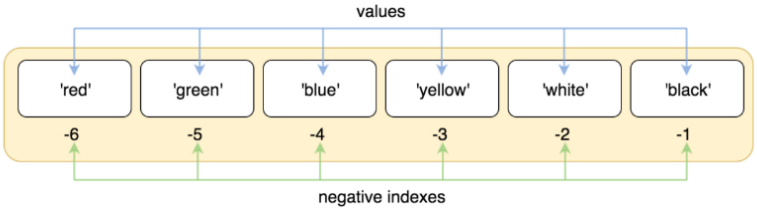
IMPORTANT!!! Don't forget **negative indexing**.
We can also do the following:

Try this:

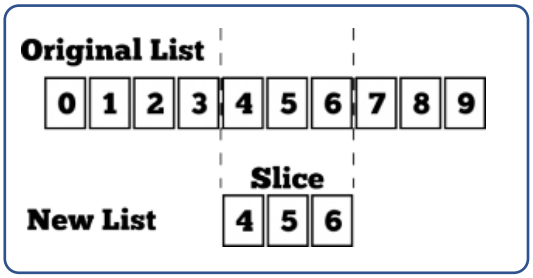



```
colors = ['red', 'green', 'blue', 'yellow', 'white', 'black']

colors[-1]
'black'
colors[-2]
'white'
colors[-6]
'red'
```



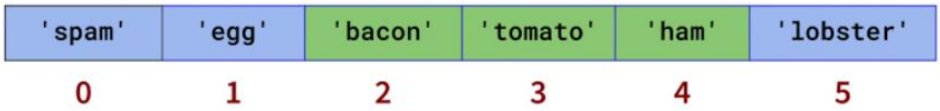
But what if you want to select certain **sections** or a **slice** of a list or string like this?



We can you a technique called: Slicing!

Selecting a **section** of a string or list is called ***Slicing*** and here is how to do it:

To slice a string/list in python you all have to think about the **starting position** and the **ending position** of the section you want, then and use the following method:




```
Food = [‘spam’, ‘egg’, ‘bacon’, ‘tomato’, ‘ham’, ‘lobster’]
```

To select **bacon, tomato, and ham** in this list call food you would Write:

```
food [2:5]
```

The *end* (5) is not inclusive so we won’t get lobster.

Slicing syntax: `list[start:end]`

Exercise#1

Try each of the following (best to **type**…Not cut and paste):

```
alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
print(alphabet[1:5])
print(alphabet[2:7])
print(alphabet[20:25])
print(alphabet[15:0])
print(alphabet[:10])
print(alphabet[:5])
print(alphabet[-5:])
print(alphabet[5:-5])

list2 = [50, 60, 70, 80, 90, 100, 101]
print(list2[:5])
print(list2 [-5:])
print(list2 [5:-5])
```

Slicing syntax: `list[start:end]`

From the above exercise on the previous page you should notice the following:

- This syntax will extract all the characters from position *start* up to **but not including** position *end*.
- If the *start* number is empty, the substring will start from position 0.
  If *end* number is empty, the substring will end at the last character of the string.
- A **negative** *start* value or *end* value is used to identify a position (number of characters) from the end of the string. This can be useful to extract the last few characters of a string.

Exercise#1 …continued:

Try each of the following (best to **type**…Not cut and paste):
Watch *carefully* what each does (some new tricks in here).

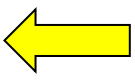
```
letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
print (letters[3:])
print (letters[:2])
print (letters[1:7:2])
print (letters[0:8:2])
print (letters[0:8:3])
print (letters[::2])
print (letters[::-1])
print (letters[::-2])
```

Slicing syntax: `list[start:end:step]`

You should notice that the **third** number in our slicing method is the "**step**". This allows us to select *every* number in the range or every *second* number or every *third* number etc).

```
                  0    1    2    3    4    5    6
>>> letters = ['c', 'h', 'e', 'c', 'k', 'i', 'o']
                       2

>>> letters[1:4:2]
['h', 'c']
```



2 is the "*Step*" – here we will select every **second** number.

Exercise#2

```
values = [100, 200, 300, 400, 500, 600, 700, 800]
```

Use **slicing** to print the following from the list above:

- the first 3 items in the list
- the last 2 items in the list
- print the elements *between* (but not including) 200 and 700
- print the last element in the list using negative indexing

```
String = 'I love python cause it is the best'
```

Using **slicing** to print out the following sections in list above:

- the last word in the sentence
- the word python
- the word cause
- the last letter of the sentence

```
list2=[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

Using **slicing** print out the following sections from list above:

- 11, 13, 17
- 19, 23, 29, 31, 37, 41, 43, 47
- 2, 3, 5, 7, 11, 13
- Every second element in the list after the number 11

Exercise#3

Below is a list of "goals per game" of your favorite hockey team. *Sort* the list (using the *sort* method) from highest to lowest and print the **best three games** and the **worst three games** using slicing:

```
goals_per_game = [2,1,0,3,4,7,6,5,8]
```

Exercise #4

Suppose you had a person's name in a variable:

```
myName = "Kenny Carney"
```

However, you want the surname Carney first and then the first name of Kenny. Use slicing to print out Carney Kenny. **Hint**: *You will have to* ***find the poition of the space*** *in the string first*.

Exercise#5

Your boss wants to you *clean* of some data contained in a list. Use a **for loop** and **slicing** to remove the `"id-"` from the following list items:

```
order_items = ["id-999", "id-19098", "id-2", "id-6154"]
```

Output:

```
['999', '19098', '2', '6154']
```

Using slicing to insert or replace items in a list or string

Not only can you **select a** section of string or list with slicing you can also **add** items to a list by **assigning** to a slice. The number of items added doesn't need to match the number of items in the slice. The list grows or shrinks to accommodate the new elements.

For example, Try the following:

```
colors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
colors[0:2] = ['black', 'white']
print(colors)
#Check the output of this code to see how it works!
```

Put the following code into Replit. Here, an **assignment** of a slice is used to replace the first and second elements by the new ones and **add** a new elements to the list. **Check the output of the code to make sure you understand what is going on**.

```
colors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']

colors[0:2] = ['black', 'white', 'gray']
print(colors)

colors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
colors[0:0] = ['black', 'white', 'gray']
print(colors)
```

You can also use slicing to **remove a section of items** from a list or sting using the `del` function.
Try the following example (**check the output to see if it makes sense**):

```
a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
del a[2:4]
print (a)
del a[-1:-4]
print(a)
```

Exercise #6

Your are working in a department store and you have just receive 3 new items to add to your product list. **Use slicing** to **add** 3 new mid-sized products between **toaster** and **oven**.

New product to add: TV, Chair, Table

```
products = ['microwave', 'toaster', 'oven', 'refrigerator', 'dishwasher']
```

Now **replace** microwave and toaster with 'fan' and 'printer'

Exercise #7

Now let's review some other important list methods that we can use in addition to slicing. Review each of the following methods and use each in an example of code that demonstrates **clearly** what each method does.

`extend()`

`append()`

`insert()`

Exercise#8

You are going on a shopping trip with two of your friends and you want to divide up the work as evenly as possible. Create a program that will allow a user to enter a list of *at least* 7 items. Then, using:

- the **`len()`** command,
- the modulo operator **%**,
- floor division **//**
- and **slicing**,

create 3 separate list called: `my_list, friend_list1,` and `friend_list2.` Each list should have the same amount of items in each…. or if list is not divisible by 3, the `firend_list2` should have the extra item or **you** should have one *less* item than both your friends. Print out all three lists.

7 items 2,2,3
8 items 2,3,3
9 items 3,3,3

Exercise#9

You are designing a video game that keeps track of **the different types of enemy's you battle** in a **list**. The enemy's are deployed randomly, but to *make things interesting*, the game's algorithm won't deploy the same type of enemy *if you have battled them **in your last 4 challenges.***
You don't want to keep fighting dragons over and over and over if there are other cool monsters to fight!

Create a program that does the following:

- Randomly generates an opponent to fight (from the list shown below or your own list).
- *Checks* to see if they have been deployed in the **last 4 battles**.
- Then puts the enemy in a new list call `deployed` to keep track of who you are fighting.

The code below is helper code. It is **intentionally** not indented correctly and has errors. You *may* choose to use it to help you out.

```
import random
defeated=[]
answer='y'
list=['zombie','dragon','bear','witch','giant,'elf','wizard','ranger']
while answer=='y':
answer=input('want to battle?')
x=random.choice(list)
if x not in defeated[-2:]:
print x
defeated.append(x)
print defeated
```