Slicing Strings and Lists

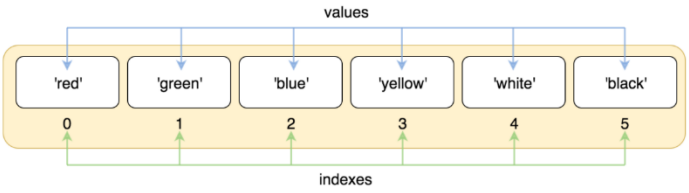
Currently, you should be able to use Python to select items from a list or string using the following method:

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

Output:
y
```

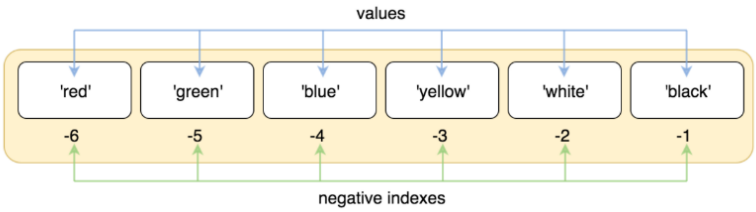
Each element in a string or list has a location and **address** that we can use to access each element.

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

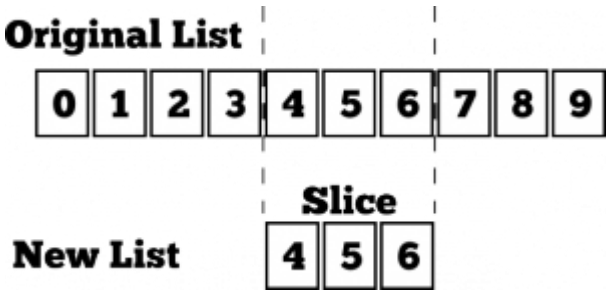


Also, don't forget **negative** indexing. We can also do the following:

```
colors = ['red', 'green', 'blue', 'yellow', 'white', 'black']
colors[-1]
'black'
colors[-2]
'white'
colors[-6]
'red'
```



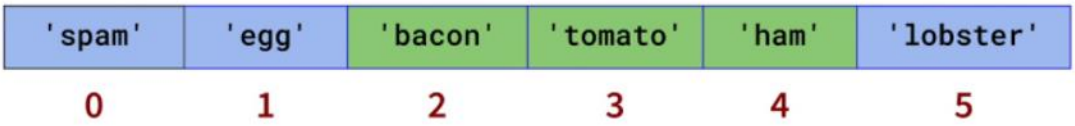
But What if you want to select certain **sections** or a **slice** of a list or string, like this:



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 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:

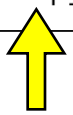
- This syntax will extract all the characters from position *start* up to **but not including** position *end*.
- If the *start* number is empty, the substring will start from position 0.
  If *end* number is empty, the substring will end at the last character of the string.
- A **negative** *start* 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**:

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

Exercise#4

```
name=input('give me your first and last name')
middle_name=input('give me your middle name')
index = name.find(' ')
output = name[:index] + ' ' + middle_name[0] + name[index:]
print(output)
```

Enter the example above and use it to **create a program** that ***uses slicing*** to do the following:

-Ask the user for their full name (First Middle and Last) separated by spaces.
-And then prints out only their Middle name.

Exercise#5

Your company needs you to create email addresses for all your employees. Use the `find()` method above and slicing to create a program that will allow users to enter their first and last name separated by a space. The program will then create an email for them and will **store the email in a list**. The email address should have the following form:
First *letter* of first name followed by their last name followed by @cramazon.ca

Example:

Please enter your name: Jim Basic
Your email address at our company is jbasic@cramazon.ca
This will be stored in our records thank you!

Exercise#6

Create a program that can take the date entered by a user (as a string) in the following form and then store sections of the string in 3 different variables (day, month, year)

Example: (this is NOT working code just an example)

```
Hey! enter the date you arrived in whistler in the following form:
(example: 02 February 2019):  05 January 2019

day = date[:2]    #First 2 characters
year = date[-4:]   #Last 4 characters
month = date[2:-5]  #The remaining characters in the middle
```

Exercise#7

Your company is spending tons of cash on long distance calls. To get an idea where the long distance funds are going, you are asked to create a program that does the following:

Given a **list** of 12 digit numbers:

- Grab area code (first 3 digits of *each* number). This will tell you what **city** the call is being place to**.**
  (use the table below).
- Grab the *last two* digits of *each* number. *This is the* ***minutes*** *the call was made for.*
- Calculate the total money spent calling *each* place (using the table below).

Area Code	City	Cost per minute
604	Vancouver	$0.34
800	International toll free	$0.00
509	Toronto	$0.50
613	Kingston	$0.55
250	Kelowna	$0.40

Example numbers. You can you these numbers or have the user generate their own list of numbers:
(note: the input numbers are strings)

604 989 0001 **23** *(last 2 digits is the number of minutes the call was made for)*
604 789 1001 34
800 876 2324 12
509 484 1888 05
613 686 5555 09
250 592 1112 15
250 876 1425 24

Output example (not accurate to the numbers above):

Company spent $5.89 on phone calls to Vancouver:
Company spent 18.11 on phone calls to Toronto:
Company spent $2.56 on phone calls to Kingston:
Company spent $29.00 spent on phone calls to Kelowna:

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 can 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 defeeted
```