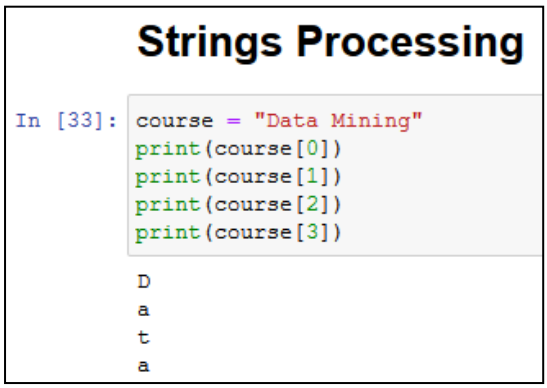
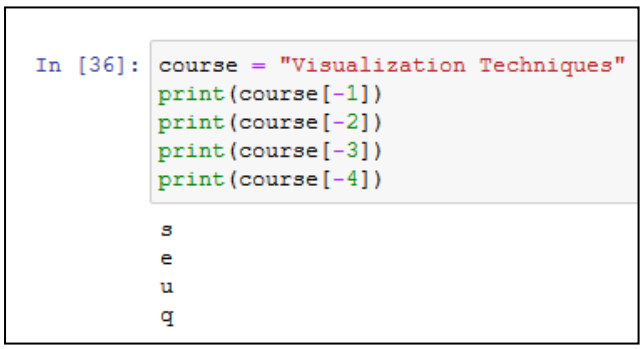
Objective: This worksheet provides few methods to process strings, integers and floating point values. These are useful methods which could be utilized at many places for data processing.

Strings Processing

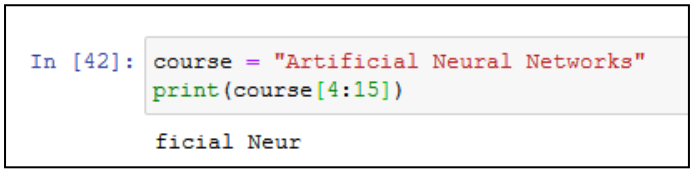
- Let us crate a new section in the notebook under String Processing heading.
- A string comprises of several characters. Each character of a string can be accessed using the location index within the square brackets []. The first character starts from index 0. In the example below the string is "***Data Mining"*** referenced by a variable ***course.*** Its first 4 characters ('D', 'a', 't' and 'a') are printed using the indices 0, 1, 2 and 3. For example ***course[0], course[1]*** etc.



- String characters from the other end can also be processed starting from index -1. Other programming languages like C/C++ do not support it.

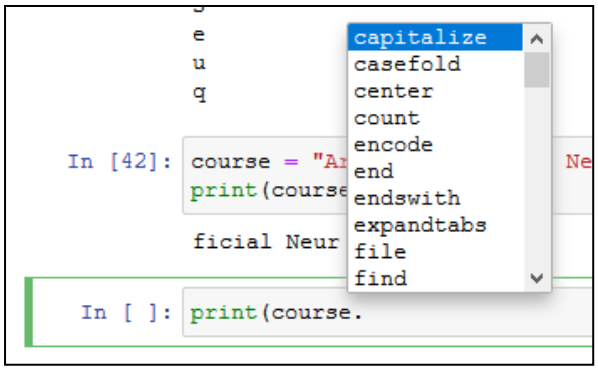


- A sequence of characters can also be accessed from a string using ***[x:y]*** syntax. Where x is the starting index and (y-1) is the last index whose character will be accessed.



Note that characters from the 4th to 14th indices are printed. 15th location is excluded.

- Many in-built functions are available to different data types. Let us review few such functions for strings. In Jupyter Notebook when one types the variable name and a dot and presses ***TAB***, the available functions for that variable data types are shown in a drop-down menu. In the example below, the variable ***course*** is used to check few in-built functions.



- Note the usage of in-built functions in two different ways in the following examples. The function ***lower()*** converts all characters in lowercase and the function ***upper()*** to the uppercase. The third function ***isupper()*** checks if all the characters are in uppercase and returns a boolean value accordingly.

```
In [52]: print(course.lower())
         artificial neural networks

In [54]: course = course.upper()
         print(course)
         ARTIFICIAL NEURAL NETWORKS

In [56]: print(course.isupper())
         True
```

- Another function ***find()*** is very useful to find out the starting location of a substring in the string. For example, if the string is ***"Data Science"***, we can check if the substring ***"Science"*** is present in it or not. If it is present, the function will return the starting index of the substring. If it is not present -1 will be returned.

```
In [57]: myString = "Data Science"
         print(myString.find("Science"))
         5
```

- Another function ***replace()*** is used to replace a part of the string with some other supplied value. For example, in the string "***Data Science",*** the substring ***"Science"*** is replaced by another substring ***"Engineering".*** Notice the usage of two arguments (parameters) in the function ***replace().***

```
In [58]: myString = "Data Science"
         print(myString.replace("Science", "Engineering"))
         Data Engineering
```

Exercise:

1. A given string is "Data Structures and Algorithms Design". What will be printed if its characters at the following locations are accessed: -1, -37 and -38?

Integer and Floating Point Processing

- There are in-built functions for Integer and Float data types also, but many of them are not of immediate use. We will review important operators here.
- In addition to +, -, * and /, Python supports the following arithmetic operators. We will review more library functions as we move along in the course.

// *a floor operator*
**** *an exponent operator*
% *a remainder operator*

```
In [71]: print(12 ** 2)
         print(18 // 5)
         print(12 % 5)

         144
         3
         2
```

*The **floor** of a real number x is the greatest integer lesser than or equal to x. For example, if x = 3.14 then floor of x = 3. Python uses // for floor. The **ceiling** of a real number x is the smallest integer greater than or equal to x. For example, if x = 3.14 then ceiling of x = 4. Python does not have any operator for ceiling, but there are Math library functions to support.*

- The following example initializes a floating point value to a variable *myNumber.* Then an in-built function is called to check if it is an integer or not. This obviously returns a boolean value as False.

```
In [85]: myNumber = 345.234
         print(myNumber.is_integer())

         False
```

- In an arithmetic operation when an integer and floating point number both are involved, the integer is converted to the floating point number and then the operation is performed resulting a floating point answer.

```
In [8]: print(23 + 3.14)
        print (23 - 3.14)

        26.14
        19.86

In [9]: print (23 * 3.14)
        print (23 / 3.14)

        72.22
        7.32484076433121

In [13]: print (23.14 / 3)
         print (23.14 %3)
         print (23 % 3.14)
         print (23.14 % 3.14)

         7.713333333333334
         2.1400000000000006
         1.0199999999999991
         1.1599999999999997
```