Modules in Python

Created by :- The Easylearn Academy
9662512857

introduction

- Some functions are used in multiple programs in python. We can put these such functions into another program known as module.
- A module is a file which has Python code.
- A module has functions, classes and variables. A module can also include runable code (normal python code).
- We can create multiple modules. Each module should contain logically related functions that we will use in other programs.
- Any Python file can be used as a module.
- Module allows you to logically organize your Python code which makes code easier to understand and use.
- There are actually three different ways to use any module in Python:

  A module can be written in Python itself.

  A built-in module is contained in the interpreter.

  A module can be written in C and loaded dynamically at run-time, like the re (regular expression) module.

How to create module?

- you can create your own Python modules since modules are actually Python .py files.
- Now let us create a very simple module.
- We call it mymath() module.
- Here hello is mymath.py file and world is python method in the same file. Which look like following.

```
def getSquare(number):
    return number * number
```

- Now create another file moduledemo.py which use (call) getSquare method of mymodule module.
- **Both file moduledemo.py and mymath.py should be stored in same folder.**

```
import mymath
# Call function
Square = mymath.getSquare(10));
Print(Square)
```

Now run moduledemo.py function to see what does it print on screen,

- if you want to import more then one module, then separate module name by comma

Where python looks for module?

- When you import a module, the Python interpreter searches for the module in the following sequences −

1. The current directory.
2. **If the module is not found, Python then searches each directory in the shell variable PYTHONPATH.**
3. If all else fails, Python checks the default path. On UNIX, this default path is normally /usr/local/lib/python3/.

   The module search path is stored in the system module sys as the **sys.path** variable. The sys.path variable contains the current directory, PYTHONPATH, and the installation-dependent default.

The PYTHONPATH Variable

- The PYTHONPATH is an environment variable, consisting of a list of directories. The syntax of PYTHONPATH is the same as that of the shell variable PATH.

one extra way to tell system where is the python module

- There is actually one more way to include & use module in python:
- Using this way, you can put the module file in any directory of your choice and then modify sys.path at run-time so that it contains that directory.
- For example, in this case, you could put mymath.py in directory C:\Users\ankit
- and then use the following statements:

- **Example**
- import **sys**
-

  #first give path of drive and directory from where you want to import module
- **sys**.path.**append**('F:\\')
-

  #import module from path given above
- import externalmodule2

Namespaces and Scoping …

- Variables are names (identifiers) that map to objects.
- A *namespace* is a dictionary of variable names (keys) and their corresponding objects (values).
- A Python statement can access variables in a *local namespace* and in the *global namespace*.
- **If a local and a global variable have the same name, the local variable has higher priority over the global variable.**
- Each function has its own local namespace.
- Class methods follow the same scoping rule as ordinary functions.
- Python assumes that any variable assigned a value in a function is local.
- Therefore, in order to assign a value to a global variable within a function, you must first use the global statement.
- The statement *global VarName* tells Python that VarName is a global variable. For such global variable name python do not search local namespace for the variable.

example

- For example, we define a variable *amount* in the global namespace. Within the function Money, we assign amount a value, therefore Python assumes amount as a local variable.
- However, we accessed the value of the local variable *amount* before setting it, so an UnboundLocalError is the result.
- Uncommenting the global statement fixes the problem.

```
amount = 2000 #amount is global variable as it is declared
   outside function
def AddMoney():
    # Uncomment the following line to fix the code:
    global amount
    amount = amount + 1 #it will access and change global
   amount variable
print (amount )
AddMoney()
print (amount )
```

The dir( ) Function

- The dir() is built-in function which returns a sorted list of strings which has the list of the names defined by a module.
- The list contains the names of all the modules, variables and functions that are defined in a module. Following is a simple example −
- Live Demo
- import math
- output = dir(math)
- print(output)

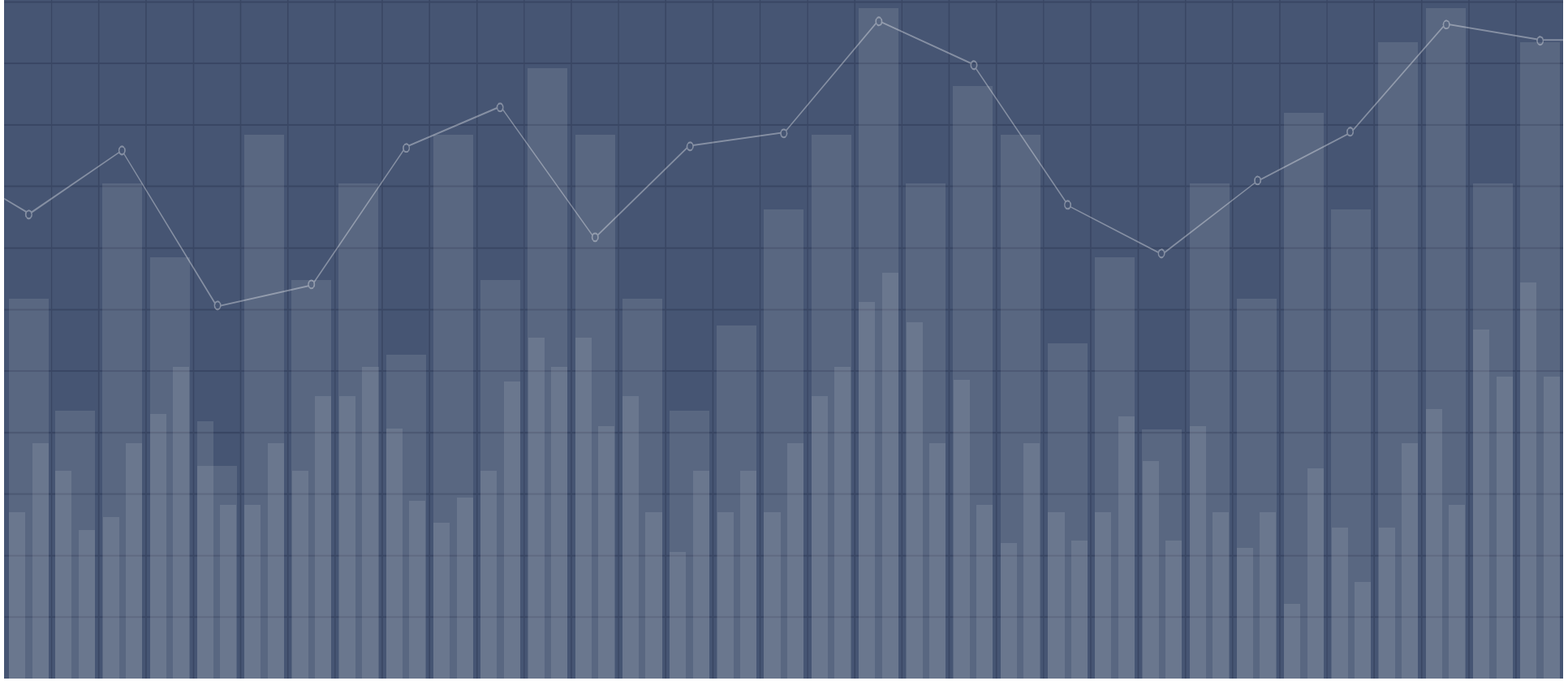
The Python Standard Library …

- Python comes with a library of standard modules, Some modules are built into the interpreter; these provide access to operations that are not part of the core of the language but are nevertheless built in, either for efficiency or to provide access to operating system primitives such as system calls.
- The set of such modules is a configuration option which also depends on the underlying platform.
- It also includes some of the optional components that are commonly included in Python distributions.

The Python Standard Library …

- Python's standard library is very extensive, offering a wide range of facilities that one has to use to perform different task in python.
- The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to
- Python programmers.
- modules written in Python provides standardized solutions for many problems that occur in everyday programming.
- Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.
- The Python installers for the Windows platform usually include the entire standard library and often also include many additional components.
- For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

Math module in Python



What is math module in Python?

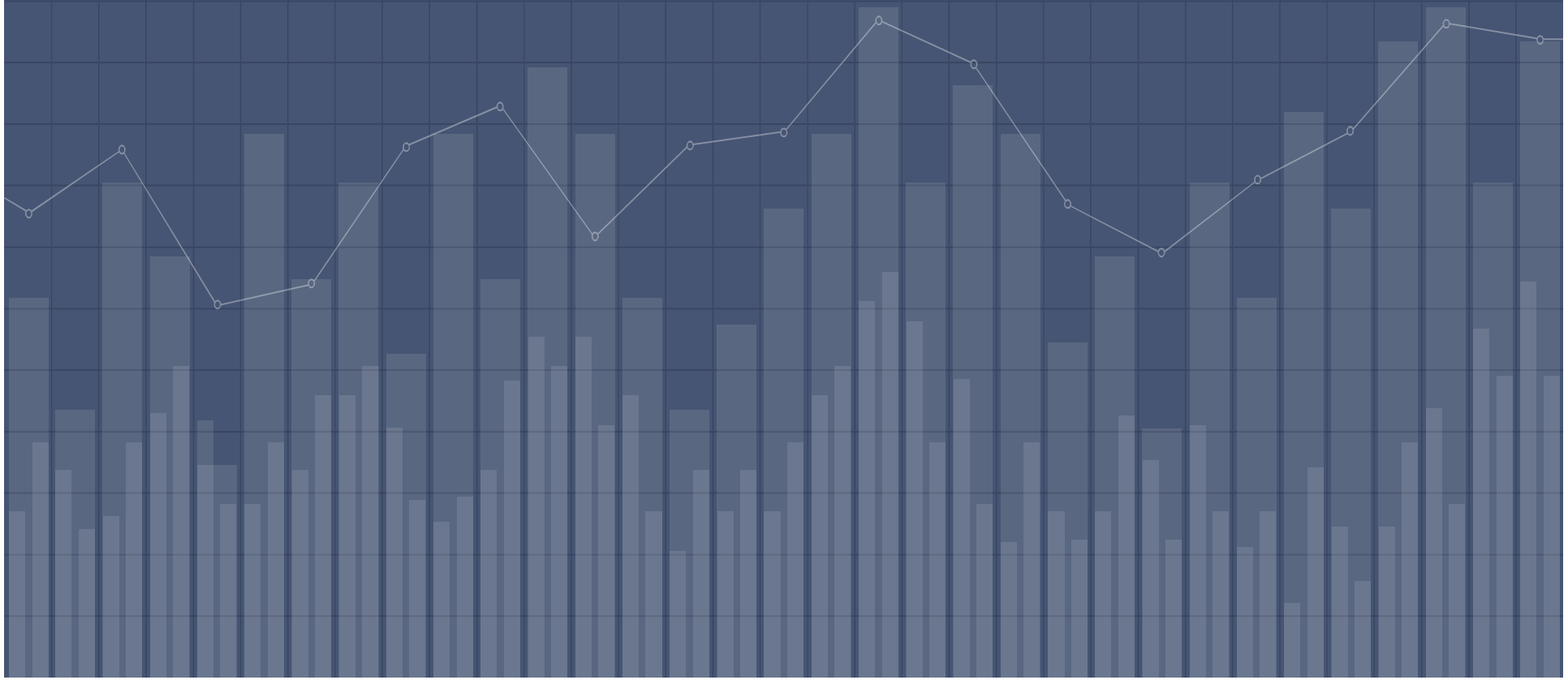
- The math module is a standard module in Python and is always available.
- To use mathematical functions under this module, you have to import the module using
- import math.
- Let us see all function in maths module.

- **ceil(x)**
- Returns the smallest integer greater than or equal to x.
- **copysign(x, y)**
- Returns x with the sign of y
- **fabs(x)**
- Returns the absolute value of x
- **factorial(x)**
- Returns the factorial of x
- **floor(x)**
- Returns the largest integer less than or equal to x
- **fmod(x, y)**
- Returns the remainder when x is divided by y
- **isfinite(x)**
- Returns True if x is neither an infinity nor a NaN (Not a Number)
- **isinf(x)**
- Returns True if x is a positive or negative infinity
- **isnan(x)**
- Returns True if x is a NaN
- **ldexp(x, i)**
- Returns x * (2**i)

- modf(x)
- Returns the fractional and integer parts of x
- trunc(x)
- Returns the truncated integer value of x
- exp(x)
- Returns e**x
- expm1(x)
- Returns e**x - 1
- log(x[, base])
- Returns the logarithm of x to the base (defaults to e)
- log1p(x)
- Returns the natural logarithm of 1+x
- log2(x)
- Returns the base-2 logarithm of x
- log10(x)
- Returns the base-10 logarithm of x
- pow(x, y)
- Returns x raised to the power y
- sqrt(x)
- Returns the square root of x
- acos(x)
- Returns the arc cosine of x
- asin(x)
- Returns the arc sine of x

- atan(x)
- Returns the arc tangent of x
- atan2(y, x)
- Returns atan(y / x)
- cos(x)
- Returns the cosine of x
- hypot(x, y)
- Returns the Euclidean norm, sqrt(x*x + y*y)
- sin(x)
- Returns the sine of x
- tan(x)
- Returns the tangent of x
- degrees(x)
- Converts angle x from radians to degrees
- radians(x)
- Converts angle x from degrees to radians
- acosh(x)
- Returns the inverse hyperbolic cosine of x
- asinh(x)
- Returns the inverse hyperbolic sine of x
- atanh(x)
- Returns the inverse hyperbolic tangent of x
- cosh(x)
- Returns the hyperbolic cosine of x
- sinh(x)
- Returns the hyperbolic cosine of x
- tanh(x)
- Returns the hyperbolic tangent of x

Random Module in python



- Python offers random module that can generate random numbers.
- It is used to get pseudo-random number.
- The sequence of number generated depends on the seed.
- We use it when
  - We want the computer to pick a random number in a given range I
  - Pick a random element from a list,
  - pick a random card from a deck,
  - flip a coin etc.
  - When making your password database more secure or powering a random page feature of your website.
- The Random module contains some very useful functions

random()

- It is used to generated one random float number between 0.0 and 1.0
- We can call as many times as we need and it will return random number each time which does necessarily unique number. It means it can generate duplicate number also.
-
- Example
- Import random
- print("float random number is ", random.random())

uniform()

- It is used to generate random **float** between given minimum and maximum value.
- We can call as many times as we need and it will return random number each time which does necessarily unique number. It means it can generate duplicate number also.
-
- Example
- print ("float random number through uniform is " ,random.uniform(10,99))

Randint() & randrange(a, b,[step])

- randint function is used to generate **integer** random number between given range.
- randint accepts two parameters: a lowest and a highest number.

- Example
- print random.randint(0, 5)

- randrange function also generate random **integer** number just like randint but in this function random number is always divisible by 3 argument

- Example
- print ("Integer random number through randrange is " ,random.randrange(0,100,10))

Choice() & choices()

- Generate one random value from the sequence.
- colors = ['red', ‘yellow’, 'green‘,’blue’]
- print(random.choice(colors))

- The choice function can often be used for choosing a random element from a list.

- Example
- countries= [“India”,”usa”,”UK”,”Canada”,”Australia”]
- Print random.choice(countries)

- To select more then one random value from list one can use choices method, which has two argument, first argument is list and second argument is no of items that one needs to fetch from list

- Example
- print (random.choices(myList,k=2))

Shuffle()

- The shuffle function, shuffles the elements in list in place, so they are in a random order.
- The random.shuffle shuffles in place and doesn't return anything i.e., It returns None.
- It reshuffles the original list (change the items positions). If you print the original list after calling the shuffle method, you will find the elements positions changed.

- Example

```
number_list = [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
print ("Original list : ", number_list)
random.shuffle(number_list)
print ("List after first shuffle : ", number_list)
random.shuffle(number_list)
print ("List after second shuffle : ", number_list)
```

Sample()

- To get shuffled item from list without actually shuffling original list one can use sample() method.
- Example
- print (random.sample(myList, 4))

String related function in python

String related function in python

- Python has several built-in functions associated with the string data type.
- These functions let us easily modify and manipulate strings. We can think of functions as being actions that we perform on elements of our code.
- Built-in functions are defined in the Python programming language and are readily available for us to use.

Making Strings Upper and Lower Case

- The functions str.upper() and str.lower() will return a string with all the letters of an original string converted to upper- or lower-case letters.
- **Because strings are immutable data types, the returned string will be a new string. Any characters in the original string will not be changed at all.**

- ***mutable*** *object can be changed after it is created, and an* ***immutable*** *object can't.*

- *Example*
- *name = "The Easylearn Academy"*
- *print(name.upper())*
- *print(name.lower())*
- origin

Method that return Boolean values

Method	True if
str.isalnum()	String consists of only alphanumeric characters (no symbols)
str.isalpha()	String consists of only alphabetic characters (no symbols)
str.islower()	String's alphabetic characters are all lower case
str.isnumeric()	String consists of only numeric characters
str.isspace()	String consists of only whitespace characters
str.istitle()	String is in title case
str.isupper()	String's alphabetic characters are all upper case

How to get length of the string

- The string method len() returns the number of characters in a string.
- This method is useful for when you need to enforce minimum or maximum password lengths,
- for example, or to truncate larger strings to be within certain limits for use as abbreviations.
- To demonstrate this method, we'll find the length of a sentence-long string:

- Example
- ***name = "The Easylearn Academy"***
- **print(len(*name*))**

Join()

- The **join()** method returns a string in which the string elements of sequence have been joined by str separator.

- **Syntax**
- Following is the syntax for **join()** method −
- str.join(sequence) **Parameters**

- **sequence** − This is a sequence of the elements to be joined.
- **Return Value**
- This method returns a string, which is the concatenation of the strings in the sequence **seq**. The separator between elements is the string providing this method.

- **Example**
- list = ["India","usa","UK","Canada","Australia"]

- s = "--"
- print (s.join(list))

replace()

- The **replace()** method returns a copy of the string in which the occurrences of old have been replaced with new.

- **Syntax**
- str.replace(old, new[, max]) **Parameters**
- **old** − This is old substring to be replaced.
- **new** − This is new substring, which would replace old substring.
- **max** − If this optional argument max is given, only the first count occurrences are replaced.

- **Return Value**
- This method returns a copy of the string with all occurrences of substring old replaced by new. If the optional argument max is given, only the first count occurrences are replaced.
- **Example**

- line = "this is shop in market. shop is very popular in town"
- print (line.replace("is", “will"))
- print (line.replace("is", “will",1)) #only first occurrence of is will be replaced with was

split()

- The **split()** method returns a list of all the words in the string, using str as the separator (splits on all whitespace if not specified),

- **Syntax**
  str.split(str="", num = string.count(str)).

- **Parameters**
  str − This is any delimeter, by default it is space.
- **num** − this is number of lines to be made

- **Return Value**
  This method returns a list of lines.

- **Example**

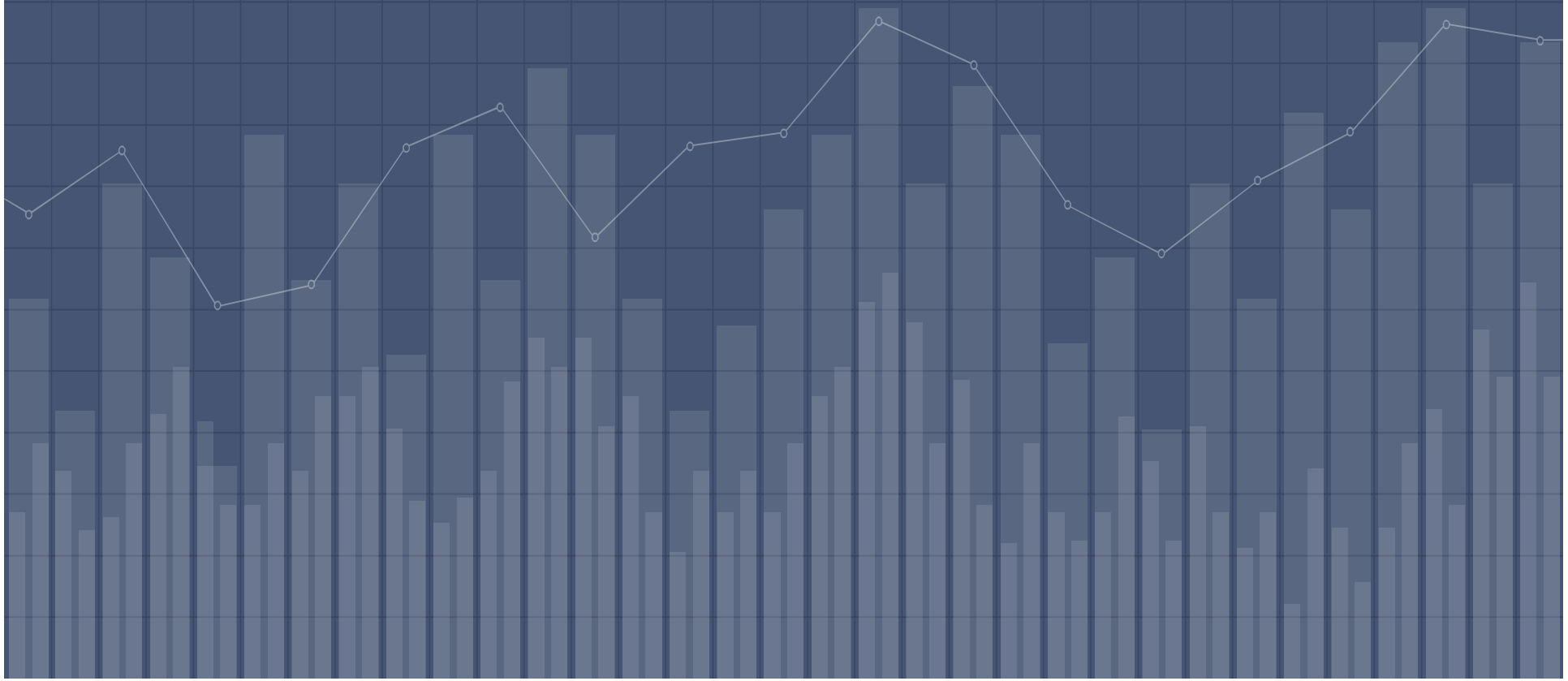
```
alphabets = "a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9"
print(alphabets)
list2 = alphabets.split()
print(list2)
```

Generate Random Strings and Passwords in Python

- String module which contains various string constant which contains the ASCII characters of all cases. String module contains separate constants for lowercase, uppercase letters, digits, and special characters.
- random module to perform the random generations.

```
import random
import string
def GeneratePassword(length=10):
    letters = string.ascii_lowercase + string.ascii_uppercase + string.digits
    size = len(letters) -1
    count=0
    password =  []
    while count<length:
        password.append(letters[random.randint(0,size)])
        count=count+1
    return ''.join(password)
print(GeneratePassword(72))
```

Packages in Python



What is packages?

- Package are group of python modules which are logically related to each other.
- For example you have two different exporter type of module, lets say sqltoexcel which export sql database to excel file & SQLtoSqlite which export sql database to sqlite database file.
- in such as both module can be saved in directory called "DBExporter" which contains above two & other modules which expert the database into various format.
- Package are created by creating directory. once package is created one can add sub packages and so on into main package directory by creating child directory.
- Packages are Python's way of organizing and structuring your code base. Concept of package is very similar to concept of package in java.

How to create a package

- To create a package, We simply need a directory in which we are going to keep our python modules.
- Name of this directory will be the name of package.

How to Import modules from package?

- We can import a module from package using "*import*" keyword only.
- While importing a module from package we'll have to specify the package name along with the module name.
- So that interpreter will first look into specified package and then will import the specified module from it.
- To traverse into the package we can use **dot (.)** operator. So the construct for importing a module from package becomes.
- ***import <package name>.<module name>***

- Suppose we want to import the "karaoke" module which is present inside filters sub-package. So the construct is;

- **import music.filters.karaoke**

- Let us see an example

Create package fruit and apple & banana as child package of it also add one module and one method into both package

```
# fruit/apple.py
def printApple():
    print ("this is an apple")

# fruit/banana.py
def printBanana():
    print ("this is a Banana")

import fruit.apple as m1
import fruit.banana as b1

a1.printApple()
b1.printBanana()
```