
LEVEL - 2

Objectives

- Python becomes a basic language
- Python language is top in IT industry
- Learning python can easily grab knowledge on Advanced courses like Machine learning and Data science
- Hackerearth is the best platform to solve many problems and various companies using this platform to conduct recruitment examinations to students

Problem solving skills using Python

What is Python?

- Python is an interpreted, high-level, general-purpose programming language
- Created by Guido van Rossum and first released in 1991, that can be used for a wide variety of applications.
- It includes data structures, dynamic typing, dynamic binding, and many more features that make it as useful for complex application development

Why Python becomes popular?

Python is used in a variety of purposes, ranging from web development to data science to DevOps. Data science and machine learning becoming more common in many types of companies, and Python becoming a common choice for that purpose. This is particularly visible in the growth of the *pandas* package. Industries are using Python such as electronics, manufacturing, software, government, and especially universities.

Anaconda

Def: Is a platform/navigator to run python

Why should we use Anaconda for Python?

- Anaconda is popular because it brings many of the tools used in data science and machine learning.
- Anaconda contains,popular python libraries that can be used in data science .
- It also comes with the jupyter notebook and Ipython distribution. So, it saves you from importing numerous libraries separately

Link for installation of Anaconda Software: https://www.anaconda.com/distribution/

Jupyter

- Jupyter is a web - application
- Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R

Jupyter Notebook: The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

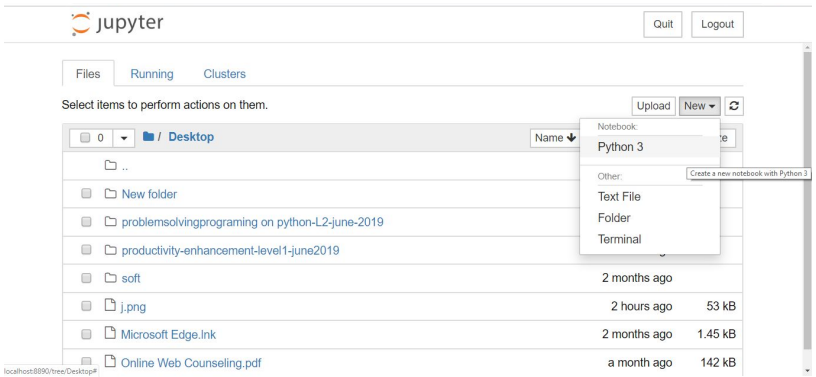
Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Advantages: Best for data exploration, data preparation, data validation, Productionalization

Working with Jupyter Notebook

- Open Anaconda Navigator
- Open Jupyter Notebook by clicking on “Launch” option below the Jupyter Notebook application

Opening of a new Python Notebook:



- After launching the Jupyter Notebook
- Create a folder on the Desktop by entering Desktop folder
- Click on New (at the right side) >> Folder
- Give a name(Problem Solving Programing on Python) to that folder
- Enter that folder New >> Python 3
- Give a name to that Notebook (today’s date)

Markdown Format for Documentation

Markdown cell displays text which can be formatted using markdown language. In order to enter a text which should not be treated as code by Notebook server, it must be first converted as markdown cell either from cell menu or by using keyboard shortcut M while in command mode. The In[ ] prompt before cell disappears.

The cell will appear like

Heading	#H1 ##H2 ###H3
Bold	** Text **
Italic	*Text*
Bold and Italic	***Text***
Normal Text and sublist	* Text * Sublist 1 * Sublist 2
Sublist Within the sublist	> 1.Ordered list1 > 2.Ordered list2

Markdown Basics

- **BOLD**
- *Italic*
- ***Bold&Italic***
- Normal Text
  - Sublist 1
  - Sublist 2

1. Ordered List1
2. Ordered List2

Reference for Markdown Commands: https://www.markdownguide.org/cheat-sheet/

Python Basics

Print Good morning

Print Hello Python Using Python language

```
In [2]:
1  # Phthon Comments
2
3  print("Good morning")
4
5  print("Hello Python")
```

```
Good morning
Hello Python
```

Here end = " " is used to print the two statements in one line. The two statements are separated by space

Python Variables:

A variable in programming is used to store the values/data Different types in variable assignments

```
In [21]:
1  n1 = 123456              # Single Variable Assignment
2  n2 = n3 = n4 = n1        # Multiple Variable Assignment
3
4  a,b,c = 123,456,789      # Multiple variable Assignment with Values
5
6  print(a,b,c)
7  print(n2)
```

```
123 456 789
123456
```

Data Types and Conversion

- **int** - holds signed integers of non-limited length.
- **float** - holds floating precision numbers and it's accurate upto 15 decimal places.
- **string** - holds a sequence of characters In Python we need not to declare datatype. We can simply just assign values in a variable. But if we want to see

```
In [19]:
1 type(a)
2 s1 = "Python"
3 type(s1)

Out[19]: str
```

```
In [22]:
1 f1 = 12.345
2 int(f1)

Out[22]: 12
```

Type of f1 is float, but here by using int(f1), f1 i converted into integer value

```
In [23]:
1 f1 = 12.345
2 str(int(f1))

Out[23]: '12'
```

By using str(f1), int(f1) is converted into string type

By using float(f1), str(int((f1))) is converted into string type

```
In [24]:
1 f1 = 12.345
2 float(str(int(f1)))

Out[24]: 12.0
```

Indentation

Python relies on indentation, using whitespace, to define scope in the code. Other programming languages often use curly-brackets for this purpose. If there is no proper indentation, we will get an error.

```
In [11]:
1 a = 33
2 b = 200
3 if b > a:
4 print("b is greater than a") # you will get an error
5

  File "<ipython-input-11-4276c1871af7>", line 4
    print("b is greater than a") # you will get an error
    ^
IndentationError: expected an indented block
```

Reading the input Dynamically

For this purpose, Python provides the function input(). It always returns a string. Methods to take different types of inputs String as input: intput() This will take both numbers and strings as input

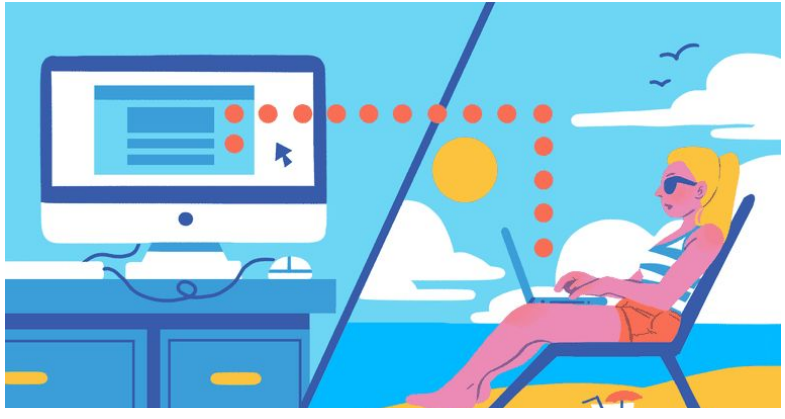
```
In [17]:
1 n1 = input()  # The inut will take both numbers and strings
2 print("Given input is:", n1)

APSSDC
Given input is: APSSDC
```

Conditional Statements

- **If Statement:** If the given condition is True,then the statements inside the body of “ if ” execute. If the given condition is False, it doesn't print the result
- **If...else Statement:** If the given condition is True, then the statements inside the body of “if” are executed. If the given condition is False, then the statements inside the body of “else” are executed.
- **If..elif...else:** ELIF is nothing but else that allows the user to check the number of users. If the “if” block evaluates as False, then condition checks next “elif” block. If all “elif” conditions evaluate as False, then the “else” statement will executed.
- **Nested if Statement:** When an if else statement is present inside the body of another “if” or “else” then this is called nested if else.

Task

- Check the given year is a Leap year or not
- Check if the number exist in a given range (inputs: number(n), range(lower number and upper number)
- Check the number of digits in a given number
- Check if the given number is a factor of 1000
- Calculate the square root of a given number



String Slicing

The Python string data type is a sequence of one or more individual characters that could consist of letters, numbers, whitespace characters, or symbols. We access strings through indexing, slicing them through their character sequences, and go over some counting and character location methods.

How Strings are Indexed? Each of a string's characters correspond to an index number, starting with the index number 0. In this spaces also be indexed. Example: Index of HELLO WORLD!

H	E	L	L	O		W	O	R	L	D	!
0	1	2	3	4	5	6	7	8	9	10	11

Accessing Characters by Positive Index Number
Accessing Characters by Negative Index Number

Methods in String Slicing

- **split() Method** - split() method returns a list of strings after breaking the given string by the specified separator.

  Syntax: We will write split() method as - string.split()

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

  Note: Python strings are case sensitive

- **islower()** - To check all characters in a string is lowercase alphabets or not
- **isupper()** - To check all characters in a string is uppercase alphabets or not
- **lower()** - To change all the characters of a string to lowercase
- **upper()** - To change all the characters of a string to uppercase
- **count()** - To find how many times a specific character repeated in a string

Task

Take a string “Problem Solving Using Python”

- Find the length of the string
- Change the string to lowercase
- Print the string in reverse order
- Split the string
- Print the string without spaces
- Find how many times “s” is repeated in a given string

Take a string “python programming by python platforms”

- Find how many times “python” is repeated

Functions

Python provides built-in functions like int(), input(), float(), print(), abs(), chr(), etc. but we can also create your own functions. These functions are called user-defined functions.

Parameters: A parameter is a variable used to define a particular value during a function definition.

Arguments: An argument is a value passed to a function at the time of function calling.

Examples on Functions:

Function to find the factorial of a given number

In [3]:
```
# Function to calculate the factorial of a given number

def factorial(n):
    fact = 1
    for i in range(1,n+1):
        fact = fact*i
    return fact
factorial(5)

```

Out[3]: 120

Types of functions in Python

- Without arguments & without return values
- Without arguments & with return Value
- With arguments & without return values
- With arguments & return values

Recursive Functions

When a function call itself is known as recursive functions.

Advantages of recursive function

- Recursive functions make the code simple.
- A complex task can be broken down into sub tasks by using recursion.

Example on Recursive function

Write a function to find the factorial of a given number with recursion

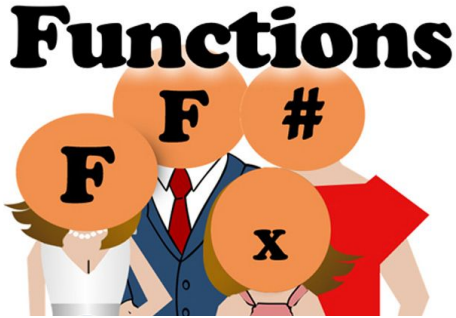
In [26]:

```
# Find factorial of a given number using recursive function
def factorialrecursion(n):
    if n == 1:
        return 1
    else:
        return (n * factorialrecursion(n-1))
    # calling the funcion with in the function

n = 5
print(factorialrecursion(n))
```

```
120
```

Task

- Function to print all numbers divisible by 6 and not a factor of 100 in a given range(lb, ub) inclusive
- Function to find the average of cubes of all even numbers in a given range(lb, ub) inclusive
- Function to generate the list of factors for a given number
- Function to calculate the factorial of a given number
- Function to check if a given number is Prime
- Function to calculate the average first N Prime numbers
- Function to generate all Perfect numbers in a given range



Advanced Problem Set(Optional)

- Function to calculate the average of all factorials in a given range
- Function to generate N odd armstrong numbers
- Function to generate Multiplication table for a number in a given range
  - 10 in the range(100, 102) inclusive
  - 10 x 100 = 1000
  - 10 x 101 = 1010
  - 10 x 102 = 1020

Packages and Modules

Module A single python file containing functions. A module is a software that has a specific functionality. It contains python statements and definitions to do a particular task. Modules are useful to break the large program into pieces. Some modules in Anaconda Navigator are math.sqrt, math.pi etc..

Packages Collection of Modules. Anaconda Navigator, in-build comes with a huge number of packages and libraries like numpy, sympy, math etc…

Example:

```
In [68]: import random

         def generateNRandomNumbers(n, lb, ub):
             for i in range(0, n):
                 print(random.randint(lb, ub), end=" ")

         generateNRandomNumbers(10, 1, 100)
```

```
86 29 81 79 83 67 87 99 24 6
```

Task:

Add 3 contacts to your contacts application, which you created at the time of working on dictionaries

Regular Expressions

A Regular Expression (RegEx) is a sequence of characters that match a pattern. It is a symbolic representation. To use regular expressions, we have to import regular expressions(import re) while writing a program.

Character	Use case
[]	Represent a character
^	Matches the beginning
$	Matches the ending
*	Number of occurrences(zero also)
+	Number of occurrences(from 1)
.	To match a character
?	For zero or one occurrences
|	OR
{}	Range set
()	For enclosing of regular expression

Regular Expressions

- [0-9] -> Any digit
- [a-z] -> Any lower case alphabet
- [2468] -> All single digit multiples of 2
- ^[0-9]{1}$ Only single digit numbers
- ^[0-9]{3}$ Only 3 digit numbers
- [0-9]*0$ -> All multiples of 10
- ^([1-9][0-9]*[05])|([5]) -> All multiples of 5
- [w][o][r][d] or (word) -> Searching for a 'word'

Examples on Regular Expressions:

Write a function to validate a phone number

```
In [6]: import re

        def phoneNumberValidator(number):
            pattern = '^[6-9][0-9]{9}$|^[0][6-9][0-9]{9}|^[+][9][1][6-9][0-9]{9}$'
            if re.match(pattern, str(number)):
                return True
            return False

        phoneNumberValidator(98765432)  # Given Phone number has 8 digits only

Out[6]: False
```

Tasks:

- Search a contact in a dictionary
- Merge a new contact with the existing contacts
- Display all the contacts in a list

Reference link on above tasks:
https://drive.google.com/open?id=1f9HPU0aeISafmrJmMJnXfsG5fhVsNEiC

File Handling

A file is a named location on desk to store relative information. File handling allows users to handle files i.e., to read and write files, along with many other file handling options, to work on files. Compared to other programming languages handling the files is very easy in Python.

Steps in File Handling:

- Open a file
- Doing the operations
- Close a file

Functions in File Handling

- **open() function:** We use open () function in Python to open a file in read or write mode. Open ( ) will return a file object. Syntax: open(filename, mode).
- **close() function:** close() function is used to close the file. Every time you have to close the file at the end of the operation, otherwise the changes you did in the file will not be saved

Modes:

- Read Mode
- Write Mode
- Append Mode

Methods:

split() using file handling: We can also split lines using file handling in Python. This splits the variable when space is encountered. You can also split using any characters as we wish.

readlines() function: The function readlines() reads all lines to the end of the file.

Tasks:

1. Write a Python program to read a file line by line and store it into a list.
2. Append a text “Multi Skill programme” in between APSSDC and Python Programming in your file

- tell(): This method helps us to know the cursor position in the file
- seek(): This method helps us to move the cursor to a particular point in the file

File Data Processing

File data processing means working with the data in a file and doing modifications in that data.

Example: Find the number of lines in a file

```
In [83]: # Count the number of lines in file
         # The file has 5 lines

         def linecount(filepath):
             count = 0
             with open(filepath, 'r') as f:
                 for i in f:
                     count = count+1
             return count
         filepath = "DataFiles/data.txt"
         print(linecount(filepath))
```

```
5
```

Tasks:

- Basic File Data Processing
  - Accessing and Modifying File Data
- Character Count
- Line Count
- File Size
- Word Count
- Unique Word Count

Reference for the above tasks: https://github.com/Akash-Sinha/APSSDC-Python-Programming-June-2019

Tasks on Data Processing

- Contacts Application
  - Add, Search, List, Modify Delete contacts
- Find and Replace Application
  - Count the total number of occurrences of a word
  - Check If a word is existing in a file
  - Replace all occurrences of a word with another word
- Marks Analysis Application
  - Generate marks file for n students
  - Input : Marks text file - each line contains marks of one student
  - Generates a report with the following information
    - Class Average
    - % of students passed
    - % of students failed
    - % of students with distinction
    - Highest Mark Frequency
    - Lowest Mark Frequency

File Encryption and Decryption

Data Encryption: Encryption is a security method in which information is encoded in such a way that only authorized user can read it. It uses encryption algorithm to generate ciphertext that can only be read if decrypted.

0	1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	0	1	2	3

Here 0 is coded as 4, 1 as 5 and so on

Data Decryption: Decryption is the process of taking encoded or encrypted text and converting it back into text that you or the computer can read and understand

4	5	6	7	8	9	0	1	2	3
0	1	2	3	4	5	6	7	8	9

Types:

Symmetric (Private key): It uses a single key that will be shared among the people who need to receive the message

Asymmetric (Public key): Asymmetrical encryption uses a pair of public key and a private key to encrypt and decrypt messages

Tasks:

1. Write a function to encrypt your marks file
2. Write a function to decrypt encrypted marks file

Reference Link on tasks:
https://drive.google.com/open?id=1ooivx-b205V6N3T93KpjUUUnHX-eU3Y8

Iterations

Iterations: If we need to execute a condition for several number of times, then we need to use looping control statements.

For loop: For loop is used to iterate over a sequence(that is either a lists, tuples, dictionaries, sets or a strings).

Syntax: for each in sequence : Body of for

Here ‘ each ‘ is a variable that takes the value inside the sequence. The loop will continue until it reaches the ending value in the sequence.

While loop: While loop is used to iterate the statement as long as the condition is satisfied. Syntax: while condition : Body of while

Tasks

For Loop:

1. Print alternate numbers from 1 to 10 using for loop.
2. Print a the given string letter by letter using for loop. (Looping on a String)
3. Program to print the pyramid pattern in python using for loop.
4. Read the given string that contains alphanumeric values. Print all even number, all odd numbers and alphabets separately.

While Loop:

1. Print n Natural Numbers using while loop
2. Program print n number of iterations using while loop with if statement.

Data Structures

Data Structures: Data Structures are used to store the collection of data together. Python has four types of inbuilt data structures.

1. Lists
2. Tuples
3. Dictionaries
4. Sets

1. Lists: Lists in Python Data Structures are used to store the collection of multiple data. Lists are ordered and immutable or changeable. Lists allows duplicate elements or values. Lists are represented as square brackets. Ex: list = [ ]

Accessing Values in Lists

In [10]:
```
1 list1 = [10,20,30,40,50]
2 list2 = ['python','raptor',1,2,3]
3 print(list1,'\n')
4 print(list2)
```

```
[10, 20, 30, 40, 50]

['python', 'raptor', 1, 2, 3]
```

Lists Methods: L1 = [1, 2, 3] L2 = [10, 20, 30] L3 = [19, 12, 25, 12, 28, 12]

append( ) - Adds an item at the end of the list

Ex: L1.append(22) **Result:** L1 = [1, 2, 3, 22]

extend( ) - Adds one entire list into another list.

Ex: L1.extend(L2) **Result:** L1 = [1, 2, 3, 10, 20, 30]

copy( ) - Returns the copy of list

Ex: List1 = L1.copy( ) **Result:** List1 = [1, 2, 3]

count( ) - Returns the count of elements in list

Ex: L3.count(12) **Result:** 3

index( ) - Returns the index position of particular value in the list **Ex:** L2.index(30) **Result:** 2

Lists Methods: L1 = [1, 2, 3] L2 = [10, 20, 30] L3 = [19, 12, 25, 12, 28, 12]

pop( ) - Removes an item at the end of the list

Ex: L1.pop( ) **Result:** L1 = [1, 2]

remove( ) - Removes an item at a particular position of the list

Ex: L2.remove(20) **Result:** L2 = [10, 30]

reverse( ) - Reverse the items in list

Ex: L3.reverse( ) **Result:** L3 = [12, 28, 12, 25, 12, 19]

sort( ) - Sorts the items in ascending order

Ex: L3.sort( ) **Result:** L3 = [12, 12, 12, 19, 25, 28]

clear( ) - Removes all elements in the list **Ex:** L1.clear( ) **Result:** [ ]

Task:
Given list contains all electronic items[mobile,laptop,speakers]. If you find any item which is not related to electronics then remove it using slicing method.

2. Tuples: Tuples are similar to lists which is used to store the collection of multiple data. Tuples are ordered and mutable or unchangeable. Tuple is represented as parentheses " ( ) ". Ex: tuple = ( )

```
1 #### Accessing Values in Tuples
```

```
In [5]:
1 tuple1 = (1, 2, 3, 4, 'Python')
2 print(tuple1)
```

```
(1, 2, 3, 4, 'Python')
```

Tuples have two methods

1. **index( ) :** Returns the index position of particular value in tuple.
2. **count( ) :** Returns the count of items in the tuple.

3.Dictionary: Dictionaries in Python are used to store the collection of data. Dictionaries are unordered and changeable. It is represented as flower brackets. Ex: dict = { }

Dictionaries have a pair of “keys” and “values” separated by “ : “.

```
In [23]:
1  dict1 = {'key1' : 200, 'key2' : 'Python'}
2  print(dict1)

{'key1': 5, 'key2': 'Python'}
```

“key1, key2” are act as keys in dict1 and “200, Python” are act as values in the dictionary “dict1”.

Comparision between Lists and Dictionaries

```
In [30]:
1  list1 = [20,25,'abc']
2  dict1 = {'k1' : 200, 'k2' : 300, 'k3' : 400}
3  print(list1[0])
4  print(dict1['k1'])

20
200
```

In lists, we can get a value using index of that particular value. Similarly, in dictionaries we can get any value using its key. Here Keys are act as an index of the Values.
Key is the unique identifier for a value Value is data that can be accessed with a key.

Dictionary Methods: dict1 = { ' k1 ' : 200, ' k2 ' : 300, ' k3 ' : 400 }

copy( ) - Returns the copy of dictionary Example: dict2 = dict1.copy( ) Result: dict2 = { ' k1 ' : 200, ' k2 ' : 300, ' k3 ' : 400 }

fromkeys( ) - Returns specified keys and values Example: x = (‘key1’ , ‘key2’) y = 0 newDict = dict.fromkeys(x, y) Result: newDict = [‘key1’ : 0, ‘key2’ = 0]

get( ) - Gets the value using particular key. Example: dict1.get(‘ k1 ’) Result: 200

items( ) - Returns the list containing tuple for each pair of key and value Example: dict1.items( ) Result: dict_items ([('k1', 200), ('k2', 300), ('k3', 400)])

keys( ) - Similar to get( ) method. Example: dict1.keys( ) Result: dict_keys(['k1', 'k2', 'k3'])

values( ) - Returns the list containing tuple for each pair of key and value Example: dict1.items( ) Result: dict_items ([('k1', 200), ('k2', 300), ('k3', 400)])

Dictionary Methods: dict1 = { ' k1 ' : 200, ' k2 ' : 300, ' k3 ' : 400 }

pop( ) - Removes the value using specified key

Example: dict1.pop(‘ k1 ‘)

popitem( ) - Removes the key and value at the end of the dictionary

Example: dict1.popitem( ) Result: dict1 = { ' k1 ' : 200, ' k2 ' : 300 }

setdefault( ) - If key does not exist, this method insert key. Returns the value of particular key

update( ) - Updates the values of specified key

Example: dict1.update({'color' : 'green'}) Result: dict1 = {'k1': 200, 'k2': 300, 'k3': 400, 'color': 'green'}

clear( ) - Removes all items in the dictionary Example: dict1.clear( ) Result: { }

Tasks

Build a Contact Application in an android phone. Add a number in your contact application and search for that number.

Update your contacts in contact application.

A dictionary contains students data(keys → S.No, values → list of name , age). Now add a new student data and delete one old student data. Print result in the format of “S.No : [name , age] “

4.Sets: Sets are used to store the collection of data in python. It is unordered and unindexed. Sets contain unique values. It is also represented as curly brackets. Ex: Set = { } or Set( )

Sets contains Unique values

```
In [47]:  1  sets = {1,2,3,2,6,3,8,12}
          2  print(sets)

{1, 2, 3, 6, 8, 12}
```

Methods in Sets:

- add( ) : Adds an element at the end of the Set
- copy( ) : Returns copy of the
- Set clear( ): removes all the elements in the set
- pop( ) : Removes the element from the end of the Set
- remove( ) : Removes specified element
- update( ) : Adds an entire set into another set.
- difference( ) : Remove the common elements in two sets and Returns the unique elements in the first set.

- difference_update( ): Returns the new updated set after make difference between the two sets.
- intersection( ): Returns the set that is the intersection of two other sets.
- intersection_update( ): Removes the elements in one set which is not there in another and Returns updated set.
- discard( ): Removes specified element.
- isdisjoint( ): Returns True, whether two sets are intersection.
- issubset( ) : Returns True, whether two sets .
- issuperset( ) : Similar to issubset. It will check all the elements in “values2” is present in “values1”, then it returns True.
- symmetric_difference( ): Returns a set of elements that are present in either one set or another set
- symmetric_difference_update ( ) : Returns newly updated set with symmetric difference between the two sets.
- union( ) : Returns a set with elements present in both sets.

Task

Take two sets and print the result using methods like Union, Difference and so on.

```
1  students1 = {'A','A','B','C','C','E','N'}
2  students2 = {'A','N','F','N','G','A'}
3
4  print('Union : ',students1.union(students2))
5  print('Difference : ',students1.difference(students2))
6  print('Intersection : ',students1.intersection(students2))
```

```
Union :  {'N', 'F', 'A', 'B', 'G', 'C', 'E'}
Difference :  {'C', 'B', 'E'}
Intersection :  {'N', 'A'}
```

Functional Programming

1. **List Comprehension:** In python, list comprehension is an easy way to create lists. List comprehensions are more efficient compared to for loop.

```
In [40]:
1 result = []
2 for number in list1:
3     result.append(number)
4 print(result)
```

```
[10, 20, 30, 40]
```

```
In [38]:
1 result = [number for number in range(6,11)]
2 print(result)
```

```
[6, 7, 8, 9, 10]
```

TASK :

1. Given a List contains a paragraph and each sentence is separated with commas. Write a program to print list that contains individual words.
2. Program to print the cubes of the numbers in a given list using list comprehension[1,2,3,4,5,6].

2. Iterators: In python, iterator is an object that contains a number of elements. Iterator implements two methods called "_iter_" and "_next_". Lists, tuples, dictionaries and sets are all iterable objects or containers which can get an iterator form. Example:

```
1  it = iter('Python')
2
3  print(next(it))
4  print(next(it))
```

```
P
y
```

TASK: Given list contains the courses like c, c++, python. Print all the courses line by line using iter( ) and next( ) functions in iterator.

```
1  course = ['C','C++','Python']
2  result = iter(course)
3  print(next(result))
4  print(next(result))
5  print(next(result))
6
```

```
C
C++
Python
```

3.Generators: We can create iterators using generators in simple approach. Generator is a function that returns iterators or objects. Generators uses “yield” instead of “return”.

Example:

```
1  def generatorFunction():
2      yield 1
3      yield 2
4      yield 3
5
6  for value in generatorFunction():
7      print(value)
```

```
1
2
3
```

TASK: Function to print square of each number until it reaches the upper bound range using Generators function. [i.e., range(5) → 2^2 = 4, 4^2 = 16, 16^2….]

Maps

Maps: It produces a list of results that it apply given function to each item of a given iterable (list,tuple, sets etc..) Syntax: map(function, iterable)

In [2]:
```
def addition(n):
    return n + n
# 1+1=2, 2+2=4, 3+3=6, 4+4=8
result = list(map(addition, range(1,5)))
print(result)
```

```
[2, 4, 6, 8]
```

TASK: Function to print squares of numbers in given range between 1 to 5

In [39]:
```
def squareOfNumb(n):
    return n * n
result = set(map(squareOfNumb,range(1,5)))
print(result)
```

```
{16, 1, 4, 9}
```

Filter

Filter: Filter function is used to check whether it is boolean or not Syntax: f : x ⟶ { T, F } Example: Given list have alphanumeric values. Print all digits in list. Li = [1, 2, ‘a’, ‘b’, ‘c’, 3]. “Filter” checks elements in “Li” and it will print the result if condition is “True” (1⟶ True, 2⟶ True, ‘a’ ⟶ False…..)

TASK : Print all prime numbers for a given range using filter( ) function.

In [45]:
```
li = [1,2,'a','b','c',3]

def isDigit(c):
    c = str(c)
    if c.isdigit():
        return True
    return False

isDigit('a')
list(filter(isDigit, li))
```

Out[45]: [1, 2, 3]

In [47]:
```
def checkPrime(n):
    if n < 2:
        return False
    for i in range(2,n//2 + 1):
        if n % i == 0:
            return False
    return True
lb, ub = 500, 600
primeList = list(filter(checkPrime, range(lb,ub)))

print(primeList)
```

[503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599]

Lambda

Lambda: Python Lambda is just a function without a name or anonymous function. Syntax: lambda argument : expression Example: lambda x : x ^ 3 (x→ argument, x ^ 3 → expression) Note: Lambda takes 0 or more arguments but only one expression. Ex: lambda x,y : x + y

```
In [57]:
1 result = lambda x:x**3
2 result(4)

Out[57]: 64
```

```
In [66]:
1 def cube(n):
2     return n ** 3
3 result = lambda n : n ** 3
4 result(4)

Out[66]: 64
```

TASK: 1. Print all even number in the range between 20 to 40 using Lambda function

2. Given Two lists contains the name of scientists. List 1 have First Name of scientist and List 2 have Last Name. Write a program to print single list that should contains Full Name of all scientist names and First letter must be Capital.

Python Libraries For Data Science

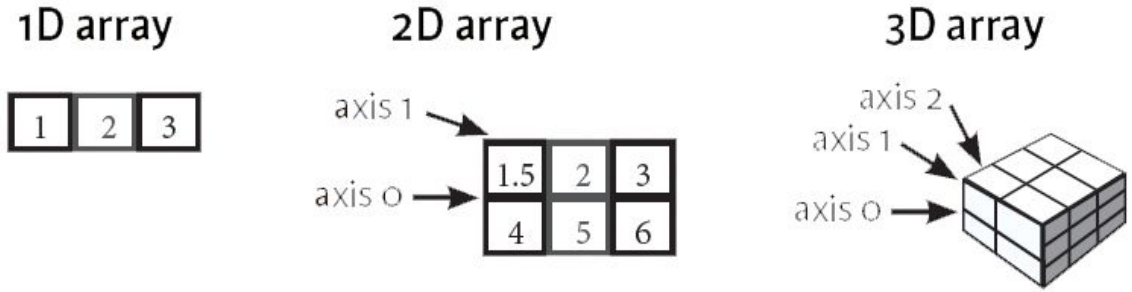
Introduction to Data Science:

Data science is the process of deriving knowledge and insights from a huge and diverse set of data through organizing, processing and analysing the data. It involves many different disciplines like mathematical and statistical modelling, extracting data from it source and applying data visualization techniques.

NumPy-Base N-Dimensional Array Package:

Numpy is a python library and a fundamental package for scientific computing in Python. It is used to create multidimensional arrays for fast operations. Use the following import conversion. Ex: import numpy a

NumPy Arrays: NumPy have a fixed size of arrays used to store a collection of elements, unlike python lists. There are three types of arrays called 1D, 2D and 3D arrays.



Task:

1. Creating 1D and 2D arrays using numPy.
2. Program to print the given List into an Array using numpy

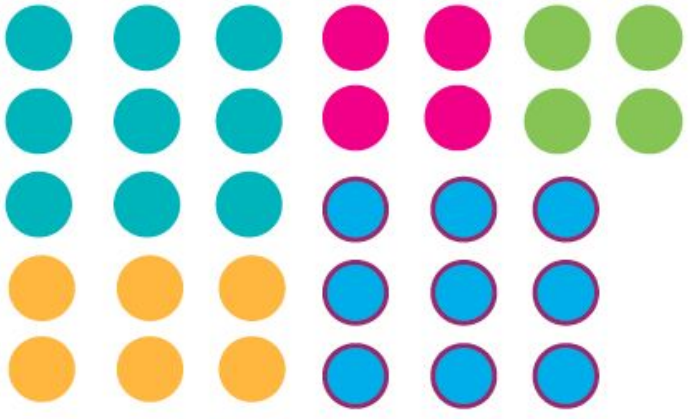
Inspect the structure and Content of array using ndarray

The most basic object in NumPy is the “*ndarray*” object or simply an “array”. The ndarray has important attributes. They are

- ndarray.ndim: No of dimensions of array referred as “rank”
- ndarray.shape: Returns the length of the array in tuple.
- ndarray.size: Returns the size of an array.
- ndarray.dtype: We can create or specify “dtype” using standard Python types. NumPy provides types like numpy.int32, numpy.int16, and numpy.float64
- ndarray.itemsize: Returns the size in bytes of each element in an array

Task

1. Create a 2D Array and inspect size, shape, dimensions and dtype of the array.
2. Program to create an array by taking range of array and reshape it.
3. Reverse the array using slicing
4. Create a 3D array of 1's. Take a list & convert it into a 3D array and add 1 to each element in an array.



Arithmetic Operations: We can perform arithmetic operators in NumPy like addition, subtraction etc., and of course log, sin, cos etc

```
1  a = [1,2,3]
2
3  a1 = np.array(a)
4  print(np.sqrt(a1))
5  print(np.sin(a1))
6  print(np.cos(a1))
7  print(np.log(a1))
```

```
[1.         1.41421356 1.73205081]
[0.84147098 0.90929743 0.14112001]
[ 0.54030231 -0.41614684 -0.9899925 ]
[0.         0.69314718 1.09861229]
```

Advantages of NumPy:

- We can write Vectorised code on numpy arrays, not on lists, which is convenient to read and write.
- Vectorized code typically does not contain explicit looping and indexing etc.
- Numpy is much faster than the standard python ways to do computations

Pandas

Derived from “PANel Data – an Econometrics from Multidimensional data”. It is an open source python library that produces high-performance data manipulation and analysis tool.

- Use Cases
  - Data Cleaning
  - Data Transformation
  - Data Analysis
- Notations
  - Series
  - Data Frames

Data Structures in Python Pandas:

There are two main data structures in Pandas -

1. Series (similar to a numpy array)
    - Creating a pandas series
    - Indexing series
2. Dataframes
    - Creating data frames from dictionaries
    - Importing CSV data files as pandas dataframe
    - Reading and summarising dataframes
    - Sorting data frames

Tasks

- Program to print any numbers with indices of alphabets.
- Program to print numbers with the index of date(from 2019-11-22)
- Print squares of numbers from 1 to n with indices.
- Program to create a dictionary of name, age and occupation. Frames of create dictionary.


Thank You