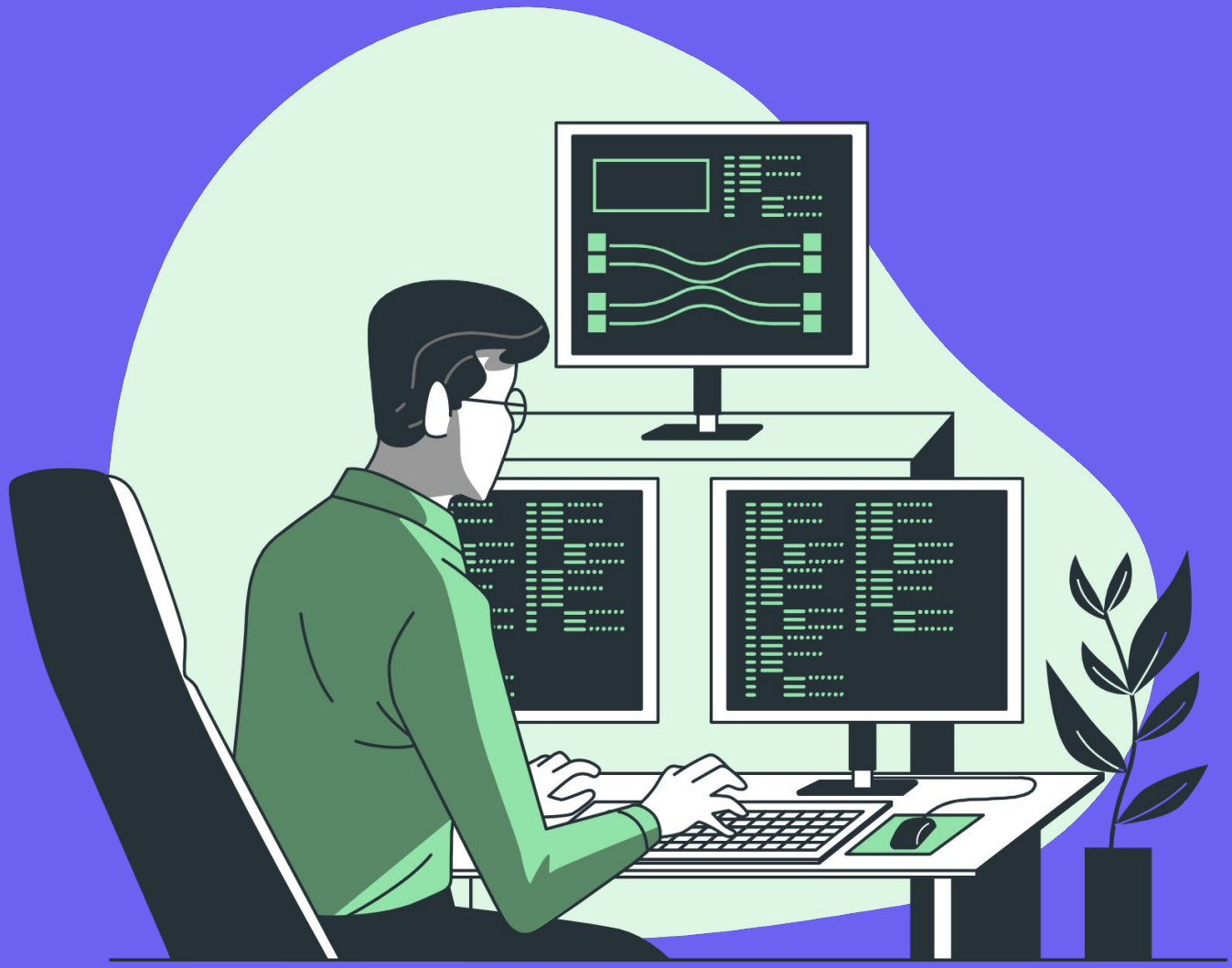
Introduction to NumPy and Pandas



Relevel
by Unacademy

Ensure NumPy and Pandas library is installed and working on everyone's system

Introduction to NumPy and Pandas

Python libraries like NumPy and Pandas are often used together for data manipulations and numerical operations.

Introduction to NumPy and Pandas

NumPy is an abbreviation of Numerical Python.

NumPy. This library has become fundamental, it is hard to imagine a world without it or before its birth. NumPy has been around since 2005, and if you have ever worked with data in Python, you must have used it, one way or the other.

What Makes NumPy So Good?

NumPy has a syntax which is simultaneously compact, powerful, and expressive. It allows users to manage data in vectors, matrices, and higher dimensional arrays. Within those data structures, it allows users to:

1. Access,
2. Manipulate,
3. Compute

Introduction to NumPy and Pandas

NumPy is a Python library that provides a simple yet powerful data structure: the n-dimensional array. This is the foundation on which almost all the power of Python's data science toolkit is built, and learning NumPy is the first step on any Python Data Scientist and Data Analyst's journey.

Importing the NumPy library

Syntax : import NumPy

Once NumPy is imported, we can use the different functions in NumPy libraries to manipulate arrays. Generally, it's imported under the **np** alias.

We can create an alias (alternate name for referring to the same thing) using the keyword "as"

example: import NumPy as np

After this NumPy can be referred to as **np**

Importing the NumPy library

List Vs. Array

Now that we know their definitions and features, we can talk about the differences between lists and arrays in Python:

- Arrays need to be declared. Lists need not to be declared, As List is builtin datatype of Python. In our previous classes, we have seen creating a list simply enclosing a sequence of elements into square brackets. Creating an array, on the other hand, requires a NumPy function i.e NumPy package (i.e., NumPy.array()). Because of this, lists are used more often than arrays.
- Arrays can store data very compactly, and it is efficient to store a large amount of data.
- Want to perform numerical operations? Arrays are great for numerical operations; lists cannot directly handle math operations. For example, you can divide each element of an array by the same number with just one line of code. You'll get an error if you try the same with a list.

Importing the NumPy library

Advantages of using Numpy Arrays Over Python Lists:

Consumes less memory.

Fast as compared to the python List

Convenient to use

Numpy Library functions

Array creation

There are multiple ways to create an array.

- The array can be created using a list or tuple or list of tuples using the array function.
- We can create arrays using initial placeholder content if we know the array size but don't know the elements to be contained.

Array creation

Move to jupyter notebook, for example, under section "creating arrays from multiple methods"

https://colab.research.google.com/drive/158EX1hplEoznQW1h60CneOCtbZEfyNbW?usp=sharing

- **The arange([start,] stop[, step,][, dtype]) :** Returns an array with evenly spaced elements as per the interval. The interval mentioned is half-opened **i.e. [Start, Stop)**
- **NumPy.ones()** function returns a new array of given shapes and types with ones.
- **NumPy.zeros()** function returns a new array of given shapes and types with zeros.
- **NumPy.random.random()** is one function for random sampling in NumPy. It returns an array of specified shapes and fills it with random floats in the half-open interval [0.0, 1.0).
- **linspace** gives evenly spaced values within a given interval. Here num no. of elements is returned.

Array creation

Move to jupyter notebook, for example, under section "NumPy Array dimensions"

Reshaping array: Reshaping NumPy array simply means changing the shape of the given array, shape basically tells the number of elements and dimension of array, by reshaping an array we can add or remove dimensions or change number of elements in each dimension.
In order to reshape a NumPy array we use reshape method with the given array.

Move to jupyter notebook for example, under section "Reshaping array"

Flatten array: Flatten array method is used to get the array copy collapsed into a single dimension. We can use flatten method to get a copy of the array collapsed into one dimension.

Move to jupyter notebook, for example, under section "Flatten array"

Array indexing

Array Indexing: Array indexing is a method to access an array element. we can access to array element by referring to its index number. There are various methods to do array indexing in NumPy

- **Slicing:** NumPy arrays can also be sliced like python lists. In case of multidimesional arrays, we need to specify slice for each dimension.
- **Integer array indexing:** Integer array indexing **allows selection of arbitrary items in the array based on their N-dimensional index**
- **Boolean array indexing:** This method is used to filter and extract elements based on some condition.

Manipulating Arrays

Updating the value in the array is often done in programming.

Basic arithmetic operations

Various arithmetic functions are provided in NumPy

Operations on a single array: We can use overloaded arithmetic operators to do element-wise operations on the array to create a new array. In the case of +=, -=, *= operators, the existing array is modified.

Move to jupyter notebook for operations in a single array example

Binary operators: These operators apply elementwise on the array, and creates a new array. In the case of +=, -=, = operators, the existing array is modified.

Move to jupyter notebook for binary operators example

Sorting in array

np.sort method is used to sort NumPy array.

Move to jupyter notebook for sorting example

Numpy datatype

Below is a list of all data types in NumPy and the characters used to represent them.

i - integer
u - unsigned integer
f - float
c - complex float
m - timedelta
M - datetime
O - object
S - string
U - unicode string
V - fixed chunk of memory for other type (void)

Move to jupyter notebook for example related to NumPy datatype

Numpy array join

In SQL we join table on the basis of keys, whereas in NumPy arrays are joined on the basis of axes. concatentate() function is used to join multiple arrays into a single array.

Note : If no axis is mentioned, default is taken as 0.

Move to jupyter notebook for example related to NumPy array join

Numpy array Search

We can use where() function in NumPy to find the indexes required based on conditional search.

Move to jupyter notebook for example related to NumPy array search

Numpy array Split

Split is reverse of joining. Splitting helps in breaking arrays into multiple. array_split() function is used to split array using NumPy. Array to split and number of array in which it's required to be split is passed while calling the function.

NumPy.hsplit:
Splits the array horizontally into multiple sub-arrays.

Syntax : NumPy.hsplit(arr, indices_or_sections)
Parameters :
arr : Array to be divided into sub-arrays.
indices_or_sections : [int or 1-D array]
Return : A list of sub-arrays

Numpy array Split

NumPy.vsplit

Splits the array vertically into multiple sub-arrays (row-wise)

Syntax : NumPy.vsplit(arr, indices_or_sections)
Parameters :
arr : Array to be divided into sub-arrays.
indices_or_sections : [int or 1-D array]
Return : A list of sub-arrays.

Move to jupyter notebook for example related to NumPy array Split

Numpy Stacking

Numpy stack functions are used to join a sequence of array along a new axis.

NumPy.vstack: used to join arrays along the vertical axis

Syntax : NumPy.vstack(tup)

Parameters :
tup : Tuple containing arrays.

Note : The arrays must have the same shape along all but the first axis.

Return : [stacked ndarray] vertically stacked arrays

Move to jupyter notebook for example

Numpy Stacking

NumPy.hstack: used to join arrays along the horizontal axis.

Syntax : NumPy.hstack(tup)

Parameters :
tup : Tuple containing arrays.

Note: The arrays must have the same shape along all but the second axis.

Return : [stacked ndarray] horizontally stacked arrays

Move to jupyter notebook for example

Numpy Stacking

NumPy.column_stack: To stack 1-D arrays as columns into 2-D arrays.

syntax : NumPy.column_stack(tup)

Params :
tup: Tuple containing arrays to be stacked.

Note: The arrays must have the same first dimension.
Return: The stacked 2-D array.

Move to jupyter notebook for example

Broadcasting

NumPy operations are generally done element-by-element, requiring two arrays to have the same shape. Numpy's broadcasting rule helps perform functions with different shape arrays if certain conditions are met.

The Broadcasting Rule: To broadcast the array, the size of the trailing axes for both arrays should be of the same size, or one of them must be one.

for example :

X(2-D array) : 5 x 7
Y(1-D array) : 7
Resulting array : 5 x 7

Broadcasting

But below example could be a mismatch

X : 5 x 7

Y : 5

Move to jupyter notebook under Broadcasting section for examples

More functions in NumPy and pandas will be taught in next class

Quiz

Q.1 What is a correct syntax to create a NumPy array?

a) np.stackarray([1,2,3])
b) np.array([1,2,3])
c) np.NumPy([1, 2, 3])
d) np.pandas([1, 2, 3])

Answer:
b) np.array([1,2,3])

Quiz

Q.2 What is a correct syntax to print the numbers [1, 2, 3] from the array given below:

array_ = np.array([1,2,3,4,5,6,7])

a) array_[0:2]

b) array_[1:3]

c) array_[2:4]

d) array_[3:]

Answer:

a) array_[0:2]

Quiz

Q.3 What is a correct syntax to check the data type of an array?

a) array.dtype

b) array.type

c) array.typeof data

d) array.ntype

Answer:

a) array.dtype

Quiz

Q.4 What is the correct method to join two or more arrays?

a) join_array()

b) array_join()

c) concatenate()

d) array_concatenate()

Answer:

c) concatenate()

Quiz

Q.5 What is the correct method to search for a certain value in an array?

a) search()

b) where()

c) find()

d) identify()

Answer:

b) where()

Quiz

Q.6 What is the correct syntax to print the number 3 from the array below:

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

a) print(arr[0,2])

b) print(arr[1,3])

c) print(arr[3,0])

d) none of the above

Answer:

a) print(arr[0,2])

Quiz

Q.7 Using a NumPy function, how would you create a one-dimensional NumPy array of the numbers from 20 to 200, counting by 10?

a) np.arange(10, 200, 20)

b) np.arange(20, 200, 10)

c) np.arange(20, 200)

d) np.range(20, 200, 10)

Answer:

np.arange(20, 200, 10)

Quiz

Q.8 How could you create a ten-element array of random integers between 1 and 5 (inclusive)?

a) np.random.randint(1, 6)

b) np.random.randint(1, 5, 10)

c) np.random.randint(1, 6, 10)

d) np.random.randint(1, 5)

Answer:

np.random.randint(1, 6, 10)

Quiz

Q.9 # four_by_five:

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20]])
```

Write a statement that prints the first row. (It will be a five-element array).

a) print(four_by_five[0])

b) print(four_by_five[1])

c) print(four_by_five[0:1])

d) print(four_by_five[0:2])

Answer:

print(four_by_five[0])

print(four_by_five[0:1])

Quiz

Q.10 How to replace items that satisfy a condition without affecting the original array?

Replace all odd numbers in arr with -1 without changing arr

Input:
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
Desired Output:
Input arr
#> array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
out
#> array([0, -1, 2, -1, 4, -1, 6, -1, 8, -1])

Quiz

Solution:

```
arr = np.arange(10)
out = np.where(arr % 2 == 1, -1, arr)
print(arr)
out
#> [0 1 2 3 4 5 6 7 8 9]
array([ 0, -1,  2, -1,  4, -1,  6, -1,  8, -1])
```

Quiz

Q.11 Reshape an array - Convert a 1D array to a 2D array with 2 rows

Input:

```
np.arange(10)
#> array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
Desired Output:
#> array([[0, 1, 2, 3, 4],
#>        [5, 6, 7, 8, 9]])
```

Solution:

```
arr = np.arange(10)
arr.reshape(2, -1)  # Setting to -1 automatically decides the number of cols
#> array([[0, 1, 2, 3, 4],
#>        [5, 6, 7, 8, 9]])
```