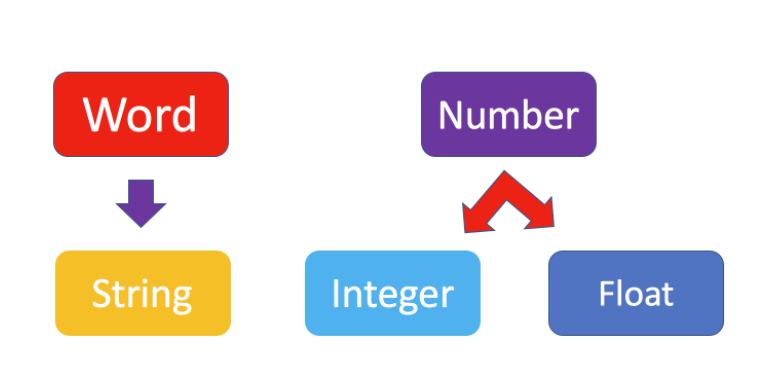
Python for Data Science, AI & Development

# Types of data



|  |  |
| --- | --- |
| Integers – 11 | int |
| Real numbers (Floats) – 21.213 | float |
| Words – “Hello Python 101” | str |
| Boolean |  |

We can see the actual data type in Python by using the type command.

Type(True):bool

Convert Types.

float(2):2.0

int(1.1):1

int(‘1’):1

Boolean can take 2 values – TRUE and FALSE

# Expressions and Variables

Expressions describe a type of operation the computers perform.

Mathematical operations

|  |  |
| --- | --- |
| 43+60+16+41 | „43“ – operand  „+“ – operator |
| 160 |  |

+ structure

\* multiplication

- subtraction

/ division

// integer division, where the result is rounded

We can use variables to store values.

my\_variable=1

# String Operations and Methods

\n new string

\t tab

\\ \

r in front of the string

Methods

.upper() A=Name

B=A.upper()

B:NAME

.replace()

.find()

# List and Tuples

**Tuples**

Tuple1 =(“Disco”, 10,1.2)

Tuple1[0]=”Disco” [-3]

Tuple1[1]=10 [-2]

Tuple1[2]=1.2 [-1]

Tuple1+(“hard rock”, 10)

(“Disco”,10,1.2,”hard rock”, 10)

Tuple2[0:2]=(“Disco”, 10,1.2)

len(Tuple1)=5 number of items in tuple

sorted(Tuple1) sort items in tuple

**Lists**

L=[“Michael Jackson”, 10.1, 1982]

L.extend() add new items in the list

L.append([]) add only one element in the list

del(A[0]) delete one element from the list

.split() convert all elements separated with space in elements of the list

.split(,) the same only using special symbol

Aliasing names referencing to the same list

Clones B=A[:] create a copy of the list

help(A) help command

# Dictionaries

DICT[“Back in Black”]:”1980”

DICT[‘Graduation’]=’2007’ add new key and value in a dictionary

del del(DICT[‘Thriller’]) delete key and value from a dictionary

in ‘The bodyguard’ in DICT check, if this key is in a dictionary (TRUE or FALSE)

DICT.keys() to see all keys in a dictionary

DICT.values() to see all values in a dictionary

# Sets

Set1={“A”, “B”, “C”}

set() convert list to a set

Set1.add(“NSYNC”) add new element to a set

Set1.remove(“NSYNC”) delete element from a set

in “AC/DC” in Set1 verify if item is in a set

& Set1 & Set2 find common items in two sets to create new set

union Set1 union Set2 Add all elements from 2 sets to create a new set

issubset() Set1.issubset(Set2) To check if all elements from 1 set is within 2 set

# Conditions and Branching

**Comparison operations**

A=6

A==7 False

A==6 True

**Conditions**

i>5 i=6 TRUE

i>=5 i=5 TRUE

i=2 FALSE

i!=6 all except 6 will be TRUE

**Branching**

**If** (age>18):

print(“you can enter”)

print (“move on”)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If (age>18):

print(“you can enter”)

**else:**

print(“”go see Meat Loaf”)

print (“move on”)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If (age>18):

print(“you can enter”)

**elif**(age==18):

print(“go see Pink Floid”)

else:

print(“”go see Meat Loaf”)

print (“move on”)

**Logic operators**

not(True) If the input is true, the result is a false

or the OR operator only produces a false if all the Boolean values are false

and the OR operator only produces a true if all the Boolean values are true

# Loops

squares=[“red”,”yellow”,”green”,”purple”,”blue”]

for i in range(0,5):

squares[i]=”white”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Squares=[‘orange’,’orange’,’purple’,’orange’,’blue’]

Newsquares=[]

i=0

whiles(squares[i]==’orange’):

Newsquares.append(squares[i])

i=i+1

# Functions

Len The function returns the number of elements in the list

Sum Summarizes

Sorted Returns a new sorted list or tuple

Sort The list album ratings will change and no new list will be created

**Creating functions**

def add1(a)

b=a+1

return b

**Multiple parameters**

def Mult (a,b):

c=a\*b

return c

# Objects and Classes

# Reading Files with Open

File1=open(“/resources/data/Example2.txt“,“w“)

File.1=name

‘/resources/data/Example2.txt’

File.1=mode

‘r’

File1.close()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

with open(“Example1.txt”,”r”) as File1:

file\_stuff=File1.**read()** the method **"read"** stores the values of the file in the variable "file\_stuff" as a string

print(file\_stuff) print the file content

print(File1.closed) check if the file content is closed

print(file\_stuff) print the file content outside the indent as well

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

print(file\_stuff)

This is line 1

This is line 2

This is line 3

file\_stuff:

This is line 1 \n This is line 2 \n This is line 3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

with open(“Example1.txt”,”r”) as File1:

file\_stuff=File1.**readlines()** We can output every line as an element in a list using the method **"readlines."**

print(file\_stuff)

file\_stuff: [‘This is line 1 \n’, ‘This is line 2 \n’, ‘This is line 3’]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

with open(“Example1.txt”,”r”) as File1:

**for** line **in** File1: We can use a **loop** to print out each line individually as follows.

print(line)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

with open(“/resources/data/Example2.txt“,“w“) as File1:

File1.write (“This is line A\n”)

File1.write(“This is line B\n”)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lines=[“This is line A \n”, “This is line B \n”, “This is line C\n”]

with open(“/resources/data/Example2.txt“,“w“) as File1:

for line in Lines:

File1.write(line)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

with open(“Example1.txt”,”r”) as readfile: We can copy one file to a new file

with open(“Example3.txt”,”w”) as writefile:

for line in readfile:

writefile.write(line)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Loading Data with Pandas

import pandas

csv\_path=‘file1.csv‘

df=pandas.read\_csv(csv\_path)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Dataframes

import pandas as pd

csv\_path=‘file1.csv‘ This variable stores the path of the CSV.

df=pd.read\_csv(csv\_path)

df.head()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

x=df[[‘Name of column’]] The result is a new data frame comprised of the original column.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

df1=df[df[‘Released’]>=1980] We now have a new data frame, where each album was released after 1979.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

df1.to\_csv(‘new\_songes.csv’) Save as CSV

# One Dimensional Numpy

Numpy is a library for scientific computing.

**Basics and array creation**

Import numpy as np We can cast a list to a numpy array by first importing numpy.

a=np.array([0, 1, 2, 3, 4])

a:array([0, 1, 2, 3, 4]) The value of ‘a’ is stored as follows

type(a): numpy.ndarray If we check the type of the array we get, numpy.ndarray.

a.dtype:dtype(‘int64’) We can use the attribute dtype to obtain the data type of the array's elements. In this case a 64-bit integer.

a.size: 5 The number of elements in the array.

a.ndim: 1 The attribute ndim represents the number of array dimensions or the rank of the array.

a.shape: (5,) The attribute shape is a tuple of integers indicating the size of the array in each dimension.

**Indexing and slicing**

a=np.array([0, 1, 2, 3, 4])

a[0]=100

a: ([100, 0, 1, 2, 3, 4])

d=a[1:3] We can select the elements from one to three and assign it to a new numpy array.

d: array([0, 1, 2])

**Basic operations**

* Vector addition

u=[1,0]

v=[0,1]

z=[]

for n, m in zip(u,v):

z.append(n+m)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

u=np.array([1,0])

v=np.array([0,1])

z=u+v

z: array([1,1])

* Array multiplication with Scalar

y=np.array([1,2])

z=2\*y

z: array([2,4])

* Hadamard product

u=np.array([1,2])

v=np.array([3,2])

z=u\*v

z: array([3,4])

* Dot product (Скалярное произведение)

u=np.array([1,2])

v=np.array([3,1])

result=np.dot(u,v)

result: 5

* Adding constant to an numpy Array

u=np.array([1,2,3,-1])

z=u+1

z: array([2,3,4,0])

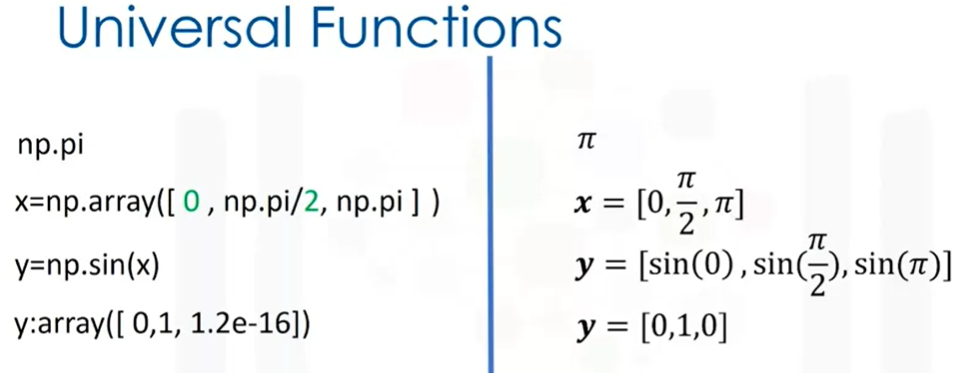
**Universal functions**

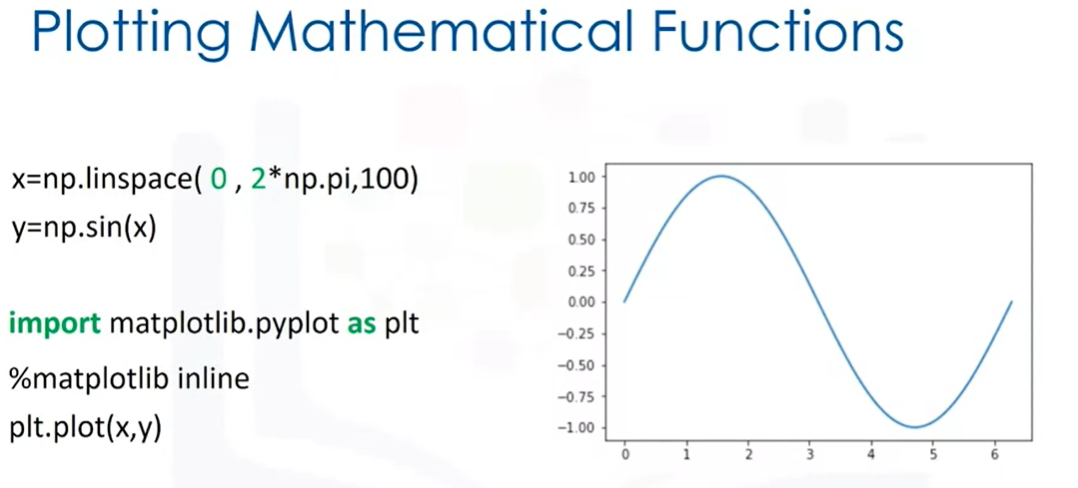
a=np.array([1,-1,1,-1])

mean\_a=a.mean() We can calculate the mean or average value of all the elements.

mean.a: 0.0

max\_a=a.max() We can find the maximum value.





# Two Dimensional Numpy

**Basics and array creation in 2D**

a=([11,12,13],[21,22,23],[31,32,33])

A= np.array(a)

A.ndim:2 We can use the attribute ndim to obtain the number of axes or dimensions referred to as the rank

A.shape: (3,3) As with a 1D array, the attribute shape returns a tuple. The first element in the tuple corresponds to the number of nested lists contained in the original list (rows). The second element corresponds to the size of each of the nested list (array zero).

X[0:2,2]

[1,0,1]

[2,2,2]