Python Programming

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope LocalMachine

from pyspark.sql import SparkSession

*# Step 01 - Create SparkSession*

spark = SparkSession.builder.master("local").appName("RDD Demo App").getOrCreate()

numbers = list(range(1,101))

sc = spark.sparkContext

*# Creates an RDD*

numbers\_rdd = sc.parallelize(numbers)

*# Operations*

*# Transformations and Actions*

print(numbers\_rdd.collect())

*#*

<https://www.linkedin.com/in/naveen-pn/>

<https://drive.google.com/drive/folders/1OkOom9kNKSgeaChlMYZBzUmllG0_958m?usp=sharing>

* [VM Workstation](https://drive.google.com/file/d/1S7n1eksmKs5KQsgfe8x8GrjV1Ut8IWcz/view?usp=sharing)
* [Cloudera Quickstart VM](https://drive.google.com/file/d/1dkca_z9u7hcfhQr0yOyevq8wifMHPhIB/view?usp=sharing)
* [Python Installation](https://docs.google.com/document/d/1As_yq91cJCN9yFfj4QK40eaEIbXW4xxbssxP1doY5L0/edit?usp=sharing)
* [Java 11](https://docs.google.com/document/d/1QutHNgQWG1hibegY9RzM_SLS5r_f3zItvLva6VVR8hk/edit?usp=sharing)
* while loop
* Operators > < >=

x = float(input("Enter x"))

print(type(x))

y = float(input("Enter y"))

print(type(y))

print(x+y)

print("A", end="%")

print("B", end="%")

print("C")

print("A","B","C", sep="-")

if 10>91:

print("Statemen 01")

print("Statement 02")

else:

print("else bloc")

print("Outside block")

x = int(input("Enter x"))

y = int(input("Enter y"))

z = int(input("Enter z"))

if x>y and x>z:

print("x is greater")

elif y>x and y>z:

print("y is greater")

elif z>x and z>y:

print("z is greater")

else:

print("Else block")

str = "abcd$989h"

is\_dollar\_found = False

count = 0

for e in str:

count+=1

print(f"Iterating: {count}")

if e == '$':

print("Dollar Found")

is\_dollar\_found=True

break

print(is\_dollar\_found)

str = "11356911,2357"

for e in str:

if int(e)%2 !=0:

print("Found an odd number", e)

continue

print("Found an evern number", e)

break

x = 10

x = x+5

x+=5

print(x)

x = True

y = True

print(True and True)

print(9>8 and 10>5)

print(True or False)

print(not False)

w = 20

x = 20

y = 10

z = 11

print(x>y)

print(y>x)

print(w==x)

print(x!=y)

x = 10

y = 20

if x > y:

print(x)

else:

print(y)

result = x if x>y else y

print(result)

# Day 03

## Functions

def calculate\_percentage(obtained\_marks, total\_marks):

return ((obtained\_marks\* 100) / total\_marks)

### Positional Arguments

percentage = calculate\_percentage(60, 100)

print(percentage)

print(calculate\_percentage(70,100))

### Named Arguments

print(calculate\_percentage(total\_marks=100, obtained\_marks=98))

### Default Arguments

def calculate\_percentage(obtained\_marks, total\_marks=400):

return ((obtained\_marks\* 100) / total\_marks)

calculate\_percentage(60)

calculate\_percentage(60,300)

calculate\_percentage(total\_marks=200, obtained\_marks=60)

### Variables Arguments

def add(\*x):

print(x)

add(1)

add(3,5,6)

add(5,6,78,8)

## Lambda Functions

def add(a,b):

return a+b

print(add(1,2))

add = lambda a,b : a+b

print(add(3,4))

<https://docs.google.com/forms/d/e/1FAIpQLSewCs9F5Ueh6KkDQnOFz_o8wqH_61bAIX8S7NixYoXpozNIQA/viewform?usp=sf_link>

MENU = '''

1. Add a book

2. Delete a book

3. Display all the books

q. TO Quit

'''

print(MENU)

choice = input("Enter your choice")

while choice!='q':

print(MENU)

choice = input("Enter you choice")

# Day 04

D1 = {'apple':'red', 'grapes':'green'}

print(type(D1))

print(D1)

D2 = dict(apple='red', grapes='green')

print(D2)

D3 = dict(D2, oranges='orange')

print(D3)

D4 = dict([('apples','red'), ('oranges','orange')])

print(D4)

D5 = dict([('apples','red'), ('oranges','orange')], strawberry='red')

print(D5)

### Accessing Dictionary

D5 = dict([('apples','red'), ('oranges','orange')], strawberry='red')

print(D5)

print(D5['apples'])

print(D5['mango'])

### Accessing using .get()

print(D5.get('apples'))

print(D5.get('mango'))

print(D5.get('apples','NULL IN SOURCE'))

print(D5.get('mango','NULL IN SOURCE'))

print(D5)

print(D5.popitem())

print(D5)

print(D5.pop('apples1',"NO Value"))

print(D5)

for k in D5.keys():

print(k)

for k in D5:

print(k)

for k in D5:

print(D5[k])

for v in D5.values():

print(v)

for k,v in D5.items():

print(k,v)

D = {'name': [{'k1':'v1'}, {'k2':'v2'}]}

**Problem Statement**

1. Given a dictionary

D = {'k1': 1, 'k2': 2 , 'k3' :5}

Write a logic to sum all the values

**Problem Statement**

Given a list

L1 = ['k1', 'k2', 'k3' , 'k4']

L2 = [11,12,13,13]

Write a logic to construct a dictionary

{'k1' : 11, 'k2':12 ,'k3':13, 'k4' : 13}

L = [1,2,3,4,5] # [3,3,5,5,7]

# O = [e for e in L e+1 if e%2 == 0 else e+2]

O = [e+1 if e%2==0 else e+2 for e in L]

print(O)

O = [column for row in nested\_list for column in row ]

print(O)

GIven a list of lists

L = []

L = [[1,-1,2], [0,-5,3,5,-2], [1,2,1,0,-2,-3]]

# Class and Objects

class Student:

def \_\_init\_\_(self):

print("Constructor Invoked")

# Create an Object

s1 = Student()

s2 = Student()

class Student:

def \_\_init\_\_(self, name, roll\_no, marks):

self.name = name

self.roll\_no = roll\_no

self.marks = marks

s1 = Student("Balaji", 1001, [100,100,100,100])

s2 = Student("Sweta",1002, [100,100,100,100])

class Student:

def \_\_init\_\_(self, name, roll\_no):

self.name = name

self.roll\_no = roll\_no

def set\_marks(self, marks):

self.marks = marks

def \_\_str\_\_(self):

return f"Name={self.name}, Roll No:{self.roll\_no}"

s1 = Student("Balaji", 1001)

print(str(s1))

s2 = Student("Sweta",1002)

s2.set\_marks([100,100,100,100])

class Student:

def \_\_init\_\_(self, name, roll\_no, marks):

self.name = name

self.roll\_no = roll\_no

self.marks = marks

def calculate\_total\_marks(self):

return sum(self.marks)

def calculate\_percentage(self):

return (self.calculate\_total\_marks()/(len(self.marks)\*100) ) \*100

# (obtained\_marks / total\_marks) \* 100

# 168 / (4 \* 100) \* 100

s1 = Student("Naveen", 1001, [35,43,54,36])

print(s1.calculate\_total\_marks())

print(s1.calculate\_percentage())

student\_dict = [

{"name":"Naveen", "roll\_no":1001,"marks":[40,50,60,70]},

{"name":"Balaji", "roll\_no":1002,"marks":[94,70,90,100]}

]

'''

student\_dict = [

{"name":"Naveen", "roll\_no":1001,"marks":[40,50,60,70], "percentage":67},

{"name":"Balaji", "roll\_no":1002,"marks":[94,70,90,100], "percentage":80}

]

'''

student\_list = [

{"name":"Naveen", "roll\_no":1001,"marks":[40,50,60,70]},

{"name":"Balaji", "roll\_no":1002,"marks":[94,70,90,100]}

]

for student in student\_list:

obj = Student(student['name'], student['roll\_no'], student['marks'])

student['Percentage'] = obj.calculate\_percentage()

print(student\_list)

class User:

# static variable

company = "Edyoda"

def \_\_init\_\_(self, username, password):

self.username = username

self.password = password

def \_\_str\_\_(self):

return f"Username= {self.username} Password:{self.password} Company: {User.company}"

obj1 = User("admin","admin123")

obj2 = User("scott","tiger")

User.company = "Edyoda PVT Ltd"

print(obj1)

print(obj2)

class Clubcabana:

def \_\_init\_\_(self, name):

self.count=0

self.count+=1

self.name = name

def get\_total\_count(self):

return self.count

p1 = Clubcabana("Ram")

p2 = Clubcabana("Krishna")

print(p1.get\_total\_count())

print(p2.get\_total\_count())

class Club:

total\_count = 0

def \_\_init\_\_(self, name):

Club.total\_count += 1

self.name = name

@staticmethod

def get\_total\_count():

return Club.total\_count

p1 = Club("Ram")

p2 = Club("Krishna")

p3 = Club("khush")

print("Total Count:", Club.get\_total\_count())

class A:

def m1(self):

self.x = 10

def m2(self):

print(self.x)

obj1 = A()

obj1.m1()

obj1.m2()

class A:

@staticmethod

def m1():

A.x = "ABC"

A.m2()

print("m1")

@staticmethod

def m2():

print(A.x)

A.m1()

class A:

def \_\_init\_\_(self, x):

self.x = x

def \_\_str\_\_(self):

return f"X={self.x}"

obj1 = A(10)

obj2 = obj1

print(obj1)

print(obj2)

class Rectangle:

def \_\_init\_\_(self, width, height):

self.width = width

self.height = height

def set\_width(self, width):

self.width = width

def get\_width(self):

return self.width

def set\_height(self, height):

self.height = height

def get\_height(self):

return self.height

rect1 = Rectangle(10,4)

rect1.set\_width(30)

rect2 = Rectangle(30,4)

rect2.set\_height(310)

print(rect1.get\_width())

print(rect2.get\_width())

print(rect1.get\_height())

print(rect2.get\_height())

class Rectangle:

def \_\_init\_\_(self, width, height):

self.\_width = width

self.\_height = height

@property #get

def width(self):

return self.\_height

@width.setter

def width(self, value):

self.\_width = value

@property #get

def height(self):

return self.\_height

@height.setter

def height(self, value):

self.\_height = value

rect1 = Rectangle(10,4)

rect1.width=30 # Setter

rect2 = Rectangle(30,4)

rect2.height=310

print(rect1.width) # Getter

print(rect2.width)

print(rect1.height)

print(rect2.height)

# Hadoop Commands

Hadoop provides a variety of command line utilities and commands that allow you to interact with the Hadoop ecosystem.

1. HDFS Commands
2. YARN Commands

## HDFS Commands

$> hdfs <Command> <Generic Options>

### File System Commands

#### -ls

| $> hdfs dfs -ls / |
| --- |

#### -mkdir

| $> hdfs dfs -mkdir /workspace  $> hdfs dfs -ls / |
| --- |

#### -rm r

| $> hdfs dfs -rm -r /workspace |
| --- |

There are two ways to copy the data from LFS to HDFS

1. put
2. copyFromLocal

| $> hdfs dfs -put file1.txt /user/$USER/workspace  $> hdfs dfs -ls /user/$USER/workspac  $> hdfs dfs -copyFromLocal file2.txt /user/$USER/worksace |
| --- |

#### There are two ways to copy the data from HDFS to LFS

1. get
2. copyToLocal

| $> rm file1.txt file2.txt  $> ls -lrt  $> hdfs dfs -get /user/$USER/workspace/file1.txt .  $> ls -lrt  $> hdfs dfs -copyToLocal /user/$USER/workspace/file2.txt . |
| --- |

#### setrep

| $> hdfs dfs -setrep 5 /user/$USER/workspace  $> hdfs dfs -ls /user/$USER/workspace |
| --- |

#### -touchz

This option will create a empty file

| $> hdfs dfs -touchz /user/$USER/workspace/file00.txt |
| --- |

#### -appendtoFile

| $> hdfs dfs -appendToFile file1.txt file2.txt file3.txt /user/$USER/workspace/files.txt |
| --- |

#### -getmerge

| $> hdfs dfs -getmerge /user/$USER/workspace/\*.txt hdfsfiles\_merged.txt |
| --- |

#### -test [ezd]

* -e if the path exists, return 0
* -z If the path is zero length, return 0
* -d If the path is a directory, return 0

| $> hdfs dfs -test -e /user/$USER/workspace/file1.txt  $> echo $?  $> hdfs dfs -test -z /user/$USER/workspace/file00.txt  > hdfs dfs -test -d /user/$USER/workspace |
| --- |

### Admin Commands

| $> hdfs dfsadmin -printTopology  $> hdfs dfsadmin -report -live  $> hdfs dfsadmin -safemode enter/get/leave |
| --- |

#### 

### GetConf Command

| $> hdfs getconf -confKey dfs.replication |
| --- |

## YARN Commands

| $> yarn jar hadoop-mapreduce-examples.jar wordcount /user/Cloudera/workspace/ /user/cloudera/output\_01/  $> hdfs dfs -cat /user/cloudera/output\_01/part\* |
| --- |

#### 

| $> |
| --- |

#### 

| $> |
| --- |

#### 

| $> |
| --- |

# Apache Hive

* This property specifies the location where the hive stores the table data.

| hive> SET hive.metastore.warehouse.dir;  /user/hive/warehouse |
| --- |

| hive> SHOW |
| --- |

## Create Database

| hive> CREATE DATABASE IF NOT EXISTS retail\_db COMMENT 'Holds retail information' WITH DBPROPERTIES('edited-by'='Naveen')  hive> CREATE DATABASE retail\_db; |
| --- |

**Important Points**

* It creates a directory in the hive warehouse (/user/hive/warehouse/)

| $> hdfs dfs -ls /user/hive/warehouse/  $> SHOW DATABASES like '\*reta\*'; |
| --- |

## Create Tables

### Internal Tables

| hive> USE retail\_db;  hive> CREATE TABLE orders(  order\_id INT COMMENT 'Holds sales id',  product\_id INT,  customer\_id INT,  order\_date DATE,  quantity INT,  employee INT  )  ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'  LINES TERMINATED BY '\n'  STORED As TextFile  tblproperties(  "skip.header.line.count"="1"  );  hive> CREATE TABLE products(  product\_id INT COMMENT 'Holds sales id',  product\_name STRING,  category\_id INT,  price DOUBLE  )  ROW FORMAT DELIMITED FIELDS TERMINATED BY ','  LINES TERMINATED BY '\n'  STORED As TextFile  tblproperties(  "skip.header.line.count"="1"  );  hive> CREATE TABLE customers(  customer\_id INT COMMENT 'Holds sales id',  name STRING,  email STRING,  address STRING  )  ROW FORMAT DELIMITED FIELDS TERMINATED BY ','  LINES TERMINATED BY '\n'  STORED As TextFile  tblproperties(  "skip.header.line.count"="1"  );  hive> CREATE TABLE categories(  category\_id INT COMMENT 'Holds sales id',  category\_name STRING)  ROW FORMAT DELIMITED FIELDS TERMINATED BY ','  LINES TERMINATED BY '\n'  STORED As TextFile  tblproperties(  "skip.header.line.count"="1"  );  hive> CREATE EXTERNAL TABLE categories\_ext(  category\_id INT COMMENT 'Holds sales id',  category\_name STRING)  ROW FORMAT DELIMITED FIELDS TERMINATED BY ','  LINES TERMINATED BY '\n'  STORED As TextFile  LOCATION '/user/cloudera/categories'  tblproperties(  "skip.header.line.count"="1"  );  hive> SELECT current\_database()  hive> SET hive.cli.print.current.db=true;  hive> set hive.cli.print.header=true  hive> DESCRIBE orders;  hive> DESCRIBE EXTENDED orders;  hive> DESCRIBE FORMATTED orders; |
| --- |

## Loading Data into Hive Tables

### Loading data from LFS to Hive Tables

| hive> LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/retail\_case\_study/orders.csv' INTO TABLE orders;  hive> LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/retail\_case\_study/products.csv' INTO TABLE products;  hive> LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/retail\_case\_study/customers.csv' INTO TABLE customers;  hive> LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/retail\_case\_study/categories.csv' INTO TABLE categories; |
| --- |

### Loading data from HDFSto Hive Tables

| hive> LOAD DATA INPATH '/user/cloudera/orders.csv' INTO TABLE orders; |
| --- |

## ANalysis

| hive> SELECT p.product\_name, SUM(o.quantity \* p.price) as total\_revenue  FROM orders o  JOIN products p  ON p.product\_id = o.product\_id  GROUP BY p.product\_name  ORDER BY total\_revenue DESC; |
| --- |

## Complex Data Types

### Array Data Type

| hive> CREATE TABLE orders\_cd(  order\_id INT COMMENT 'Holds sales id',  product\_id ARRAY<INT>,  customer\_id INT,  order\_date DATE,  quantity INT,  employee INT  )  ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'  COLLECTION ITEMS TERMINATED BY ','  LINES TERMINATED BY '\n'  STORED As TextFile  tblproperties(  "skip.header.line.count"="1"  );  LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/orders.dat' INTO TABLE orders\_cd; |
| --- |

### Struct Data Type

| hive> CREATE TABLE products\_cd(  product\_id INT,  product\_info STRUCT<product\_name:STRING,price:DOUBLE,category\_name:STRING>  )  ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'  COLLECTION ITEMS TERMINATED BY ','  LINES TERMINATED BY '\n'  STORED As TextFile  tblproperties(  "skip.header.line.count"="1"  );  LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/products.dat' INTO TABLE products\_cd; |
| --- |

### Map Data Types

| hive> CREATE TABLE customers\_cd(  customer\_id INT COMMENT 'Holds sales id',  customer\_info MAP<STRING, STRING>  )  ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'  COLLECTION ITEMS TERMINATED BY ','  MAP KEYS TERMINATED BY ':'  LINES TERMINATED BY '\n'  STORED As TextFile  tblproperties(  "skip.header.line.count"="1"  );  LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/customer.dat' INTO TABLE customers\_cd; |
| --- |

# DataFrame

user\_df.select(col('col\_id'), lower(col('col\_designation')).filter(col('col\_designation')!='technician').show()