ADVANTO

S O F T W A R E

PDBC with Oracle

- Databases are used to store information like customers info, billing info, calls info etc..
- Based on our requirement we can store data in 2 ways

Temporary storage areas	**Ex: list, tuple, dict etc**
Permanent storage areas	**Ex: files, databases etc**

File Systems:

- Provided by local operating systems
- Suitable for storing less amount of data

Limitations of file:

- We cannot store huge amount of information
- There is no query language support
- Operations will be very complex
- No security for data
- There is no mechanism to prevent duplicate data

PDBC with Oracle

To overcome the drawbacks of file systems, we can use databases

Databases:

- To store large amount of data
- Query language support is available for every database and hence we can perform database operations very easily
- To access data present in database, compulsory username and password is required, hence data is secured
- Databases has no chance for data inconsistency problems

Limitations of databases:

Cannot hold very huge amount of data

Databases support only structured data i.e: Tabular format, but cannot support semi structured data (xml) and unstructured data (videos, audio, images etc.)

PDBC with Oracle

To overcome these problems we should go for more advanced storage areas like big data, data warehouses, Hana etc..

Python database programming:

Python provide inbuilt support for several data bases like Oracle, Mysql, Sql Server, Sqlite, gadfly etc..

Python supports separate module for each database

cx_Oracle	**→ for Oracle**
pymssql	**→ for Microsoft sql server**
Pymysql	**→ for Mysql**

PDBC with Oracle

Standard steps for PDBC

- Import the database specific module

  import cx_Oracle

  import pymysql

- Establish connection between pyton program and database

  we can create this connection object by using **connect()** function

  con= cx_Oracle.connect(database information)

  Ex: con= cx_Oracle.connect('system/system@localhost/orcl')

PDBC with Oracle

To execute sql query and to hold results we have cursor object

cursor= con.cursor()

Execute your sql query

cursor.execute(sql query)	→ for single sql query
cursor.executescript(queries)	→ for group of sql queries
cursor.executemany()	→ for parameterized queries

Fetch the results

cursor.fetchone()	→ to fetch only one row
cursor.fetchall()	→ to fetch all rows
cursor.fetchmany(n)	→ to fetch n rows

PDBC with Oracle

Commit()	→	to save the changes
Rollback()	→	to revert all the changes

- ➢ Close the resources

    cursor.close()

    con.close() → to close the connection

Working with oracle database:

Installing cx_Oracle

From normal command prompt execute the following command

C:\ > pip install cx_Oracle

ADVANTO
SOFTWARE

PDBC with Oracle

To check whether cx_Oracle is installed or not:

From python IDLE type

```
>>> help(“modules”)
```

Check for cx_Oracle module in the list

To connect with oracle database and print version:

```
from cx_Oracle import *
query="create table emp(sno int,name varchar(20), sal int)"
con=connect('system/system@localhost/orcl')
if con!= None:
   print("the current version is: ", con.version)
else:
   print("sorry cannot connect to database")
con.close()
```

Output:

```
the current version is:
11.2.0.1.0
```

ADVANTO
SOFTWARE

PDBC with Oracle

➢ **Program to create employees table in the data base**

```
from cx_Oracle import *
try:
   query="create table emp1(sno int,name varchar(20), sal int)"
   con=connect('system/system@localhost/orcl')
   cursor=con.cursor()
   cursor.execute(query)
   print("Table created successfully")
except DatabaseError as e:
   if con!= None:
      con.rollback()
      print("there is a problem: ",e)

finally:
   if cursor != None:
      cursor.close()
   if con != None:
      con.close()
```

Output:

```
Table created Successfully
```

ADVANTO SOFTWARE

ANAND KOMIRIPALEM

PDBC with Oracle

Program to Drop Employees Table from Oracle Database

```
from cx_Oracle import *
try:
    query="Drop Table emp"
    con=connect('system/system@localhost/orcl')
    cursor=con.cursor()
    cursor.execute(query)
    print("Table Dropped successfully")
except DatabaseError as e:
    if con!= None:
        con.rollback()
        print("there is a problem: ",e)

finally:
    if cursor != None:
        cursor.close()
    if con != None:
        con.close()
```

Output:

```
Table Dropped Successfully
```

PDBC with Oracle

Program to insert record into database

```
from cx_Oracle import *
try:
    query="insert into emp1 values(1,'anand',10000)"
    con=connect('system/system@localhost/orcl')
    cursor=con.cursor()
    cursor.execute(query)
    con.commit()
    print("Record Inserted successfully")
except DatabaseError as e:
    if con!= None:
        con.rollback()
        print("there is a problem: ",e)

finally:
    if cursor != None:
        cursor.close()
    if con != None:
        con.close()
```

DML Commands will be saved into the database only if we commit them

Output:

```
Record Inserted Successfully
```

PDBC with Oracle

Program to insert multiple records

```
from cx_Oracle import *
try:
    query="insert into emp1 values(:sno,:name,:sal)"
    records=[(2,'hai',15000),(3,'hello',30000)]
    con=connect('system/system@localhost/orcl')
    cursor=con.cursor()
    cursor.executemany(query,records)
    con.commit()
    print("Records Inserted successfully")
except DatabaseError as e:
    if con!= None:
        con.rollback()
        print("there is a problem: ",e)

finally:
    if cursor != None:
        cursor.close()
    if con != None:
        con.close()
```

```
Output:
------------

Records Inserted Successfully
```

ANAND KOMIRIPALEM

PDBC with Oracle

Program to Insert Multiple Rows in the Employees Table with Dynamic Input from the Keyboard

PDBC with Oracle

```
from cx_Oracle import *
try:
   con=connect('system/system@localhost/orcl')
   cursor=con.cursor()
   while True:
      sno=int(input("Enter Employee Number:"
      name=input("Enter Employee Name:")
      sal=float(input("Enter Employee Salary:")
      sql="insert into emp1 values(%d,'%s',%d)
      cursor.execute(sql %(sno,name,sal))
      print("Record Inserted Successfully")
      option=input("Do you want to insert one
      if option=="No":
         con.commit()
         break
except DatabaseError as e:
   if con!= None:
      con.rollback()
      print("there is a problem: ",e)
finally:
   if cursor != None:
      cursor.close()
   if con != None:
      con.close()
```

Output:

```
Enter Employee Number:4
Enter Employee Name:ukyk
Enter Employee Salary:76978
Record Inserted Successfully
Do you want to insert one more record[Yes|No]
:y
Enter Employee Number:5
Enter Employee Name:hgkffk
Enter Employee Salary:45566
Record Inserted Successfully
Do you want to insert one more record[Yes|No]
:No
```

PDBC with Oracle

Update Employee Salaries with Dynamic Input

```
from cx_Oracle import *

try:
   con=connect('system/system@localhost/orcl')
   cursor=con.cursor()
   increment=int(input("Enter Salary to Increment: "))
   salrange=int(input("Enter Sal range: "))
   query="update emp1 set sal=sal+%d where sal < %d"
   cursor.execute(query %(increment, salrange))
   print("Record updated Successfully")
   con.commit()

except DatabaseError as e:
   if con!= None:
      con.rollback()
      print("there is a problem: ",e)
finally:
   if cursor != None:
      cursor.close()
   if con != None:
      con.close()
```

Output:

```
Enter Salary to Increment3000
Enter Sal range15000
Record updated Successfully
```

PDBC with Oracle

Program to Delete Employees whose Salary Greater provided Salary as Dynamic Input

```
from cx_Oracle import *

try:
   con=connect('system/system@localhost/orcl')
   cursor=con.cursor()
   salrange=int(input("Enter Sal range: "))
   query="delete from emp1 where sal>%d"
   cursor.execute(query %(salrange))

   print("Record deleted Successfully")
   con.commit()


except DatabaseError as e:
   if con!= None:
      con.rollback()
      print("there is a problem: ",e)

finally:
   if cursor != None:
      cursor.close()
   if con != None:
      con.close()
```

Output:

```
Enter Sal range: 50000
Record deleted Successfully
```

PDBC with Oracle

Program to Select all Employees info

```
from cx_Oracle import *
try:
    con=connect('system/system@localhost/orcl')
    cursor=con.cursor()
    query="select * from emp1"
    cursor.execute(query)
    row=cursor.fetchall()
    print(row)

except DatabaseError as e:
    if con!= None:
        con.rollback()
        print("there is a problem: ",e)
finally:
    if cursor != None:
        cursor.close()
    if con != None:
        con.close()
```

Output:

```
[(1, 'anand', 17000), (2, 'hai', 15000), (3,
'hello', 30000), (5, 'hgkffk', 45566)]
```

ADVANTO
S O F T W A R E

ANAND KOMIRIPALEM

PDBC with Oracle

Program to Select all Employees info using fetchone()

```
from cx_Oracle import *
try:
   con=connect('system/system@localhost/orcl')
   cursor=con.cursor()
   query="select * from emp1"
   cursor.execute(query)
   row=cursor.fetchone()
   while row is not None:
      print(row)
      row=cursor.fetchone()

except DatabaseError as e:
   if con!= None:
      con.rollback()
      print("there is a problem: ",e)

finally:
   if cursor != None:
      cursor.close()
   if con != None:
      con.close()
```

Output:

```
------------
(1, 'anand', 17000)
(2, 'hai', 15000)
(3, 'hello', 30000)
(5, 'hgkffk', 45566)
```

PDBC with Oracle

```
from cx_Oracle import *

try:
    con=connect('system/system@localhost/orcl')
    cursor=con.cursor()
    query="select * from emp1"
    cursor.execute(query)
    n=int(input("Enter n value: "))
    records=cursor.fetchmany(n)
    for row in records:
        print(row)
except DatabaseError as e:
    if con!= None:
        con.rollback()
        print("there is a problem: ",e)

finally:
    if cursor != None:
        cursor.close()
    if con != None:
        con.close()
```

Output:

```
Enter n value: 2
(1, 'anand', 17000)
(2, 'hai', 15000)
```

Regular Expressions

Regular Expressions

- If we want to represent a group of strings according to a particular pattern, then we should go with regular expressions.

Where we use regular Expressions in applications:

- We use regular expressions to perform validations
- To develop pattern matching applications (like ctrl+C, ctrl+v in windows, grep in unix)
- To develop digital circuits
- To develop communication protocol like TCP/IP

Python has a built in module called ‘re’ to work with regular expressions

Regular Expressions

This module contains several inbuilt functions to use Regular Expressions very easily in our applications.

```
>>>import re
>>>dir(re)
```

First we need to convert the pattern into regular object form. We can do it by using **compile()** method

```
import re
pattern = re.complie(“word to search”)
```

Once pattern object Is ready then we should create a matcher object. We use an inbuilt method **finditer()**

Regular Expressions

Matcher object will have the following methods

Start(): returns the starting index of match

End(): returns end+1 index of match

Group(): returns the matched string

```
import re
pattern=re.compile('python')
matcher=pattern.finditer("python, learning python is very easy")
for match in matcher:
   print('match started @:',match.start(),' and ending @:
    ',match.end())
```

Output:

```
match started @: 0  and ending @:  6
match started @: 17  and ending @:  23
```

ADVANTO
SOFTWARE

ANAND KOMIRIPALEM

Regular Expressions

➢ **To print the matches and total number of occurances**

```
import re
count=0
pattern=re.compile('python')
matcher=pattern.finditer("python, learning python is very easy")
for match in matcher:
    count=count+1
    print('match started @:',match.start(),' and ending @: ',match.end())
print("total number of occurances: ", cou
```

```
Output:
------------
match started @: 0  and ending @:  6
match started @: 17  and ending @:  23
total number of occurances:  2
```

Regular Expressions

Note: We can pass pattern directly as argument to finditer() function.

```
import re
count=0
matcher=re.finditer("ab","abaababa")
for match in matcher:
    count+=1
    print(match.start(),"...",match.end(),"...",match.group())
print("The number of occurrences: ",count)
```

```
Output:
------------
0 ... 2 ... ab
3 ... 5 ... ab
5 ... 7 ... ab
The number of occurrences:
3
```

Regular Expressions

- If we want to search for ‘a’ or ‘b’ or ‘c’ or except ‘a’ & ‘b’ & ‘c’ or any such group of characters we can go with character classes.

Character classes:

[abc]	→	Either a or b or c
[^abc]	→	Except a & b & c
[a - z]	→	Any Lower case alphabet Symbols
[A - Z]	→	Any Upper case alphabet Symbols
[a – zA – Z]	→	Any Alphabet
[a – zA –Z0 - 9]	→	Any Alphabet & digit
[^a - zA - Z 0 - 9]	→	Only Special Characters

Regular Expressions

Ex:

```
import re
matcher = re.finditer('[^abc]', 'a@78bka76')
for m in matcher:
        print(m.start(), '-------', m.group())
```

Output:

```
------------
1 ------- @
2 ------- 7
3 ------- 8
5 ------- k
7 ------- 7
8 ------- 6
```

ADVANTO
S O F T W A R E

Regular Expressions

Predefined Character classes:

\s	→	space Character
\S	→	Except Space Character
\d	→	any digit
\D	→	Except digits
\w	→	Any Word Character (Alpha Numeric)
\W	→	Special Characters
.	→	Any Character

Regular Expressions

Ex:

```
import re
matcher = re.finditer('\W', 'a@78bka76')
for m in matcher:
        print(m.start(), '-------', m.group())
```

Output:

```
------------
1 ------- @
```

Ex-2:

```
import re
matcher = re.finditer(‘\w', 'a@78bka76')
for m in matcher:
        print(m.start(), '-------', m.group())
```

Output:

```
------------
0 ------- a
2 ------- 7
3 ------- 8
4 ------- b
5 ------- k
6 ------- a
7 ------- 7
8 ------- 6
```

Regular Expressions

Quantifiers:

We can use quantifiers to specify the number of occurrences to match.

a → Exactly one 'a’

a+ → Atleast one 'a'

a* → Any number of a's including zero number

a? → Atmost one 'a' ie either zero number or one number

a{m} → Exactly m number of a's

a{m,n} → Minimum m number of a's and Maximum n number of a's.

Regular Expressions

```
import re
matcher=re.finditer("a","abaabaaaba")
for match in matcher:
        print(match.start(),"......",match.group())
```

```
Output:
------------
0 ...... a
2 ...... a
3 ...... a
5 ...... a
6 ...... a
7 ...... a
9 ...... a
```

```
import re
matcher=re.finditer("a+","abaabaaaba")
for match in matcher:
        print(match.start(),"......",match.group())
```

```
Output:
------------
0 ...... a
2 ...... aa
5 ...... aaa
9 ...... a
```

Regular Expressions

```
import re
matcher=re.finditer("a*","abaabaaaba")
for match in matcher:
	print(match.start(),"......",match.group())
```

```
Output:
------------
0 ...... a
1 ......
2 ...... aa
4 ......
5 ...... aaa
8 ......
9 ...... a
10 ......
```

```
import re
matcher=re.finditer("a{3}","abaabaaaba")
for match in matcher:
	print(match.start(),"......",match.group())
```

```
Output:
------------
5 ...... aaa
```

Regular Expressions

Important Functions of Re module:

1) match()
2) fullmatch()
3) search()
4) findall()
5) finditer()
6) sub()
7) subn()
8) split()
9) compile()

```
Output:
------------

Anand
Python
2 Months
```

Regular Expressions

Match():

match function is used to check the given pattern at beginning of target string.

If the match is available then we will get Match object, otherwise we will get None.

```
import re
s=input("Enter pattern to check: ")
m=re.match(s,"abcabdefg")
if m!= None:
    print("Match is available at the begir
    print("Start Index:",m.start(), "and E
else:
    print("Match is not available at the beginning")
```

Output:

```
------------

Enter pattern to check: ab
Match is available at the
beginning of the String
Start Index: 0 and End Index: 2
```

ADVANTO SOFTWARE

ANAND KOMIRIPALEM

Regular Expressions

Fullmatch():

fullmatch() function is used to match a pattern to all of target string. i.e complete string should be matched according to given pattern. If complete string matched then this function returns Match object otherwise it returns None.

```
import re
s=input("Enter pattern to check: ")
m=re.fullmatch(s,"abcabdefg")
if m!= None:
        print("Match is available at the beginning
        print("Start Index:",m.start(), "and End In
else:
        print("Match is not available at the begin
```

```
Output:
------------

Enter pattern to check: abcabdefg
Match is available at the
beginning of the String
Start Index: 0 and End Index: 9
```

Regular Expressions

search():

search() function is used to search a pattern in the given pattern. If match is available then returns the first occurrence of the match object otherwise it returns None.

```
import re
s=input("Enter pattern to check: ")
m=re.search(s,"abcabdefg")
if m!= None:
	print("Match is available at the beginning
	print("Start Index:",m.start(), "and End In
else:
	print("Match is not available at the beginning")
```

Output:

```
------------

Enter pattern to check: ca
Match is available at the
beginning of the String
Start Index: 2 and End Index: 4
```

Regular Expressions

If we want to ignore case sensitivity , we need to use 3rd argument IGNORECASE

Ex:

```
import re
s=input("Enter pattern to match: ")
m=re.search(s,"abaabbaaabbaaaaabbbb", re.IGNORECASE)
if m != None:
    print("Matcher Available")
    print(m.start(), "-----------", m.end())
else:
    print("Match not available")
```

Output:

```
Enter pattern to match: AA
Matcher Available
2 ----------- 4
```

Regular Expressions

Note:

^x → It will check whether target string starts with x OR not. If target string starts with x it will return string else None

```
import re
s="Learning python is Easy"
m=re.search('^Learn',s)
if m != None:
   print("String starts with Learn")
else:
   print("Target String not starting with Learn")
```

x$ → It will check whether target string ends with x OR not. If target ends with x it will return x else None

Regular Expressions

x$ → It will check whether target string ends with x OR not. If target ends with x it will return x else None

```
import re
s="Learning python is Easy"
m=re.search('Easy$',s)
if m != None:
    print("String Ends with Easy")
else:
    print("Target String not Ending with Easy")
```

ANAND KOMIRIPALEM

Regular Expressions

findall():

To find all occurrences of the match.

This function returns a list object which contains all occurrences.

```
import re
m=re.findall('[0-9]',"a7b9k6z")
print(l)
```

Output:

```
------------

['7', '9', '6']
```

finditer():

Returns the iterator yielding a match object for each match.

```
import re
m=re.finditer('\d','a7b9k6z')
for i in m:
        print(i.start(), i.end(), i.group())
```

Output:

```
------------

1 2 7
3 4 9
5 6 6
```

Regular Expressions

sub():

sub means substitution or replacement.

re.sub(regex,replacement,targetstring)

In the target string every matched pattern will be replaced with provided replacement.

```
import re
s=re.sub("[a-z]","#","a7b9c5k8z")
print(s)
```

Output:

```
------------
#7#9#5#8#
```

Every alphabet symbol is replaced with # symbol

Regular Expressions

subn():

It is exactly same as sub except it can also returns the number of replacements.

This function returns a tuple where first element is result string and second element is number of replacements.

(resultstring, number of replacements)

```
import re
s=re.subn("[a-z]","#","a7b9c5k8z")
print(s)
print("The Result String:",s[0])
print("The number of replacements:",s[1])
```

Output:

```
------------

('#7#9#5#8#', 5)
The Result String: #7#9#5#8#
The number of replacements: 5
```

ANAND KOMIRIPALEM

ADVANTO SOFTWARE

Regular Expressions

split():

If we want to split the given target string according to a particular pattern then we should go for split() function.

This function returns list of all toker

```
import re
s=re.split(",","sunny,bunny,
print(s)
for t in s:
      print(t)
```

Output:

```
------------
['sunny', 'bunny', 'chinny', 'vinny', 'pinny']
sunny
bunny
chinny
vinny
pinny
```

```
import re
s=re.split("\.","www.Advanto.com")
print(s)
for t in s:
      print(t)
```

Output:

```
------------
['www', 'Advanto', 'com']
www
Advanto
com
```

ANAND KOMIRIPALEM

ADVANTO
SOFTWARE

Regular Expressions

Write a Python Program to check whether the given Number is valid Mobile Number OR not?

```
import re
n=input("Enter your Mobile Number: ")
            #m=re.fullmatch('[7-9]\d{9}',n)
m=re.fullmatch('[7-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]',n)
            #m=re.fullmatch('[7-9][0-9]{9}',n)
if m != None:
   print("valid Mobile Number")
else:
   print("invalid Mobile Number")
```

```
Output:
------------
Enter your Mobile Number:
944054013
invalid Mobile Number

Enter your Mobile Number:
9004550139
valid Mobile Number
```

Regular Expressions

Write a Python Program to extract all Mobile Numbers present in file where Numbers are mixed with Normal Text Data

```
import re
f1= open('abc.txt','r')
f2= open('abc1.txt', 'w')
for every_line in f1:
    l=re.findall("[7-9]\d{9}", every_line)
    for i in l:
        f2.write(i+"\n")
print("Extraction completed")
f1.close()
f2.close()
```

Web Scrapping

- The process of collecting information from web pages is called web scraping.
- In web scraping to match our required patterns like mail ids, mobile numbers we can use regular expressions.

Ex: program to get title of the websites

```
import re,urllib
import urllib.request
sites = ["http://google.com", "http://rediff.com"]
for s in sites:
    print("Searching...", s)
    u=urllib.request.urlopen(s)
    text=u.read()
    title=re.findall("<title>.*</title>", str(text),re.IGNORECASE)
    print(title)
```

It gives a list output

ADVANTO
S O F T W A R E

ANAND KOMIRIPALEM

Web Scrapping

```
import re,urllib
import urllib.request
sites = ["google", "rediff"]
for s in sites:
    print("Searching...", s)
    u=urllib.request.urlopen("http://"+s+".con
    text=u.read()
    title=re.findall("<title>.*</title>", str(text),
    print(title[0])
```

```
Searching... http://google.com
<title>Google</title>

Searching... http://rediff.com
<title>Rediff.com: News |
Rediffmail | Stock Quotes |
Shopping</title>
```

The type of text is bytes so we need to
Convert it into string format

ADVANTO
SOFTWARE

ANAND KOMIRIPALEM

Web Scrapping

Program to pint mobile numbers from redbus.in

```
import re,urllib
import urllib.request
sites = ["google", "rediff"]
u=urllib.request.urlopen("https://www.re
text=u.read()
numbers=re.findall("[0-9]{9}[0-9]+", str(tex
for n in numbers:
    print(n)
```

```
Output:
------------
919945600000
919945600000
919945600000
919945600000
919945600000
919945600000
919945600000
919945600000
919945600000
919945600000
```

ADVANTO
S O F T W A R E

ANAND KOMIRIPALEM

Web Scrapping

Program to check whether mail is gmail or not

```
import re
s=input("Enter Mail Id:")
m=re.fullmatch("\w[a-zA-Z0-9_]{3}[a-zA-Z0-9_.]*@gmail[.]com", s)
if m!= None:
    print("valid Email")
else:
    print("Invalid Email ID")
```

Web Scrapping

To check a valid Registration number.

```
import re
s=input("Enter Mail Id:")
m=re.fullmatch("MH[012][0-9][A-z]{2}\d{4}", s)
if m!= None:
    print("valid Email")
else:
    print("Invalid Email ID")
```

ADVANTO
S O F T W A R E

Thank Q