Database Concepts (V)

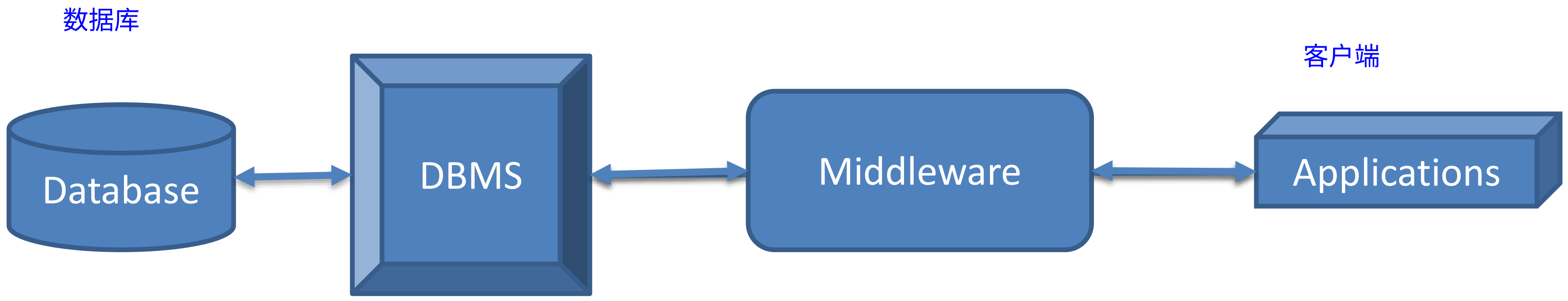
Database Connectivity

Chaokun Wang

School of Software, Tsinghua University
chaokun@tsinghua.edu.cn

April 26, 2021

Motivation



Outline

- Database Connectivity Foundation
- PostgreSQL C Connector
- Project
- Introduction to Python

Q1：怎么在【n由客户决定】的情况下，向数据库中插入n个元组？
A1：SQL里面好像也有循环哦！

```
 WHILE ... LOOP
 END LOOP
```

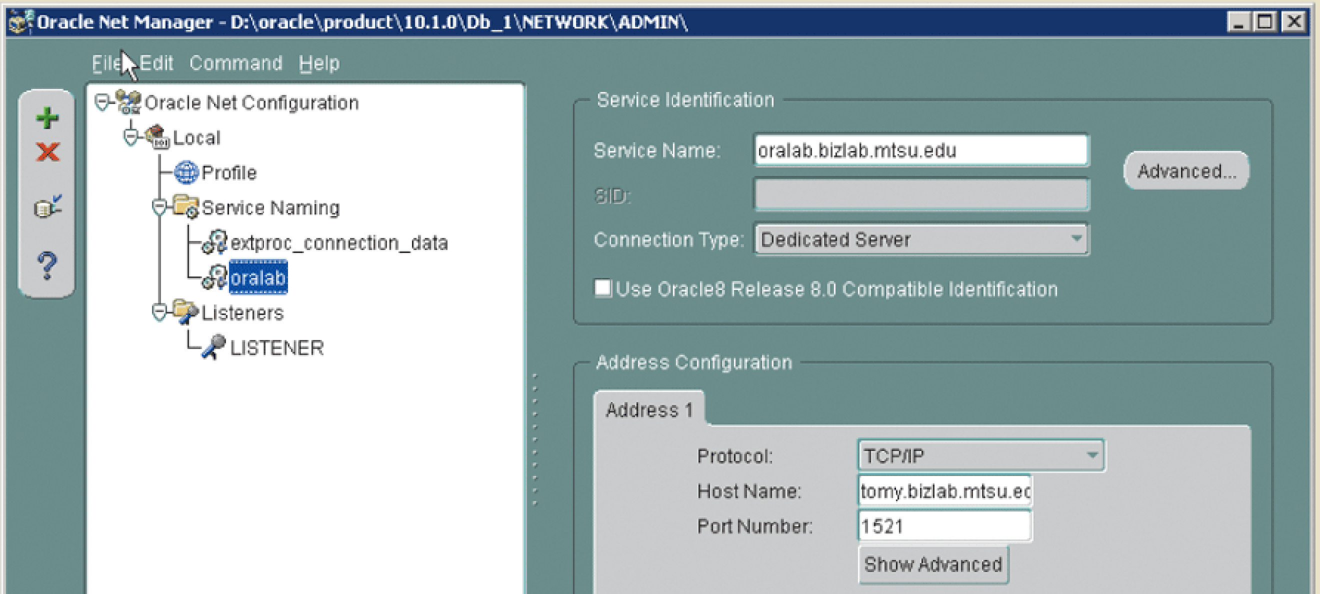
Database Connectivity

- Mechanisms through which application programs connect and communicate with data repositories
  - Database middleware: provides an interface between the application program and the database
  - Data repository: data management application used to store data generated by an application program
  - Universal Data Access (UDA): collection of technologies used to access any type of data source and manage the data through a common interface
    - ODBC, OLE-DB, and ADO.NET form the backbone of MS UDA architecture

Native SQL Connectivity

- Connection interface provided by database vendors, which is unique to each vendor
  - Interfaces are optimized for particular vendor's DBMS
  - Maintenance is a burden for the programmer

FIGURE 15.1 ORACLE NATIVE CONNECTIVITY



Service Identification

Service Name: oralab.bizlab.mtsu.edu

SID:

Connection Type: Dedicated Server

☐ Use Oracle8 Release 8.0 Compatible Identification

Address Configuration

Address 1

Protocol: TCP/IP

Host Name: tomy.bizlab.mtsu.ec

Port Number: 1521

ODBC, DAO, and RDO (1 of 3)

- Open Database Connectivity (ODBC): Microsoft’s implementation of a superset of SQL Access Group Call Level Interface (CLI) standard for database access
  - Widely supported database connectivity interface
  - Allows Windows application to access relational data sources by using SQL via standard application programming interface (API)
- Data Access Objects (DAO): object-oriented API used to access desktop databases such as MS Access and FileMaker Pro
  - Provides an optimized interface that expose functionality of Jet data engine to programmers

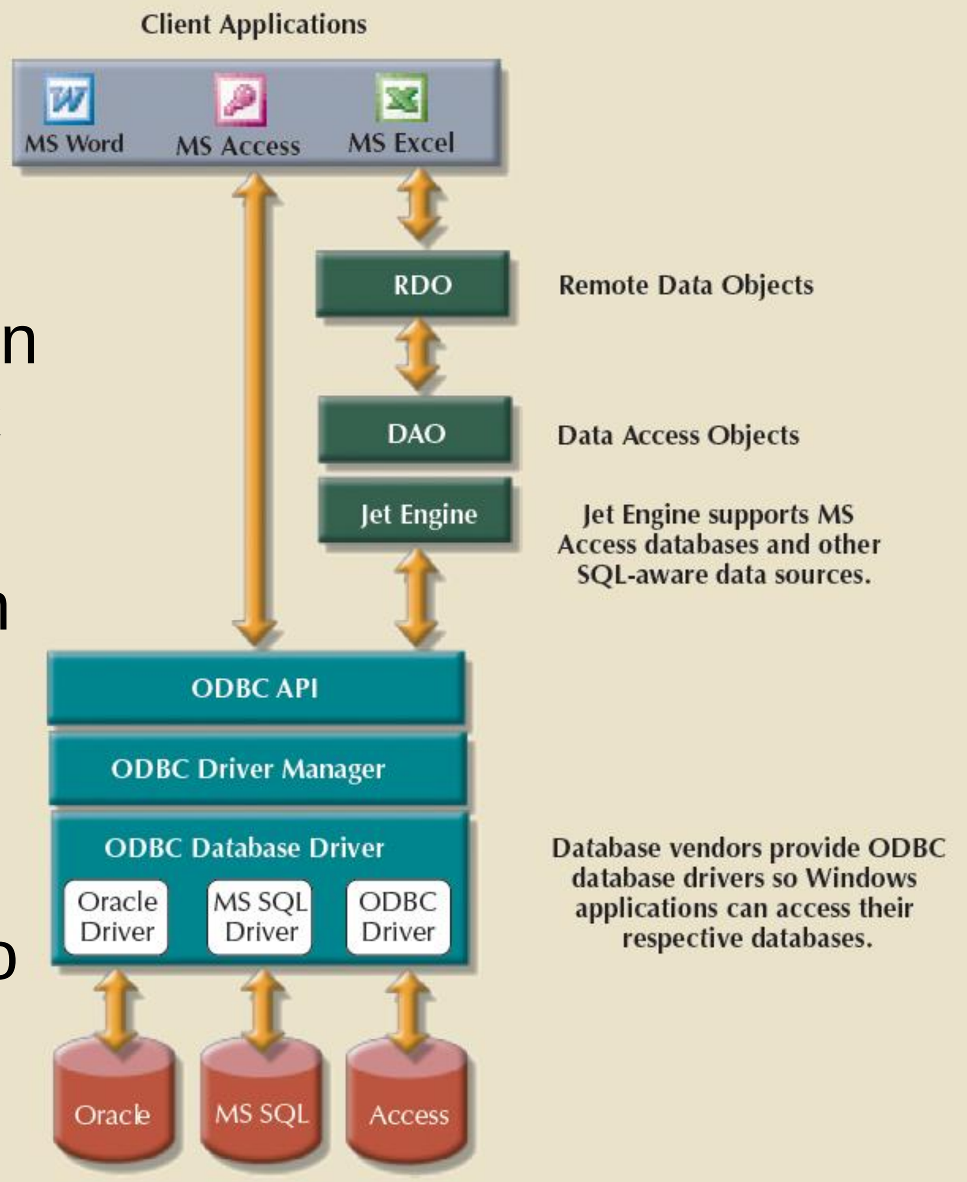
ODBC, DAO, and RDO (2 of 3)

- Remote Data Objects (RDO): higher-level object-oriented application interface used to access remote database servers
  - Optimized to deal with server-based databases
- Dynamic-link libraries (DLLs): implements ODBC, DAO, and RDO as shared code that is dynamically linked to the Windows operating environment

ODBC, DAO, and RDO (3 of 3)

- Components of ODBC architecture
  - High-level ODBC API through which application programs access ODBC functionality
  - Driver manager that is in charge of managing all database connections
  - ODBC driver that communicates directly to DBMS

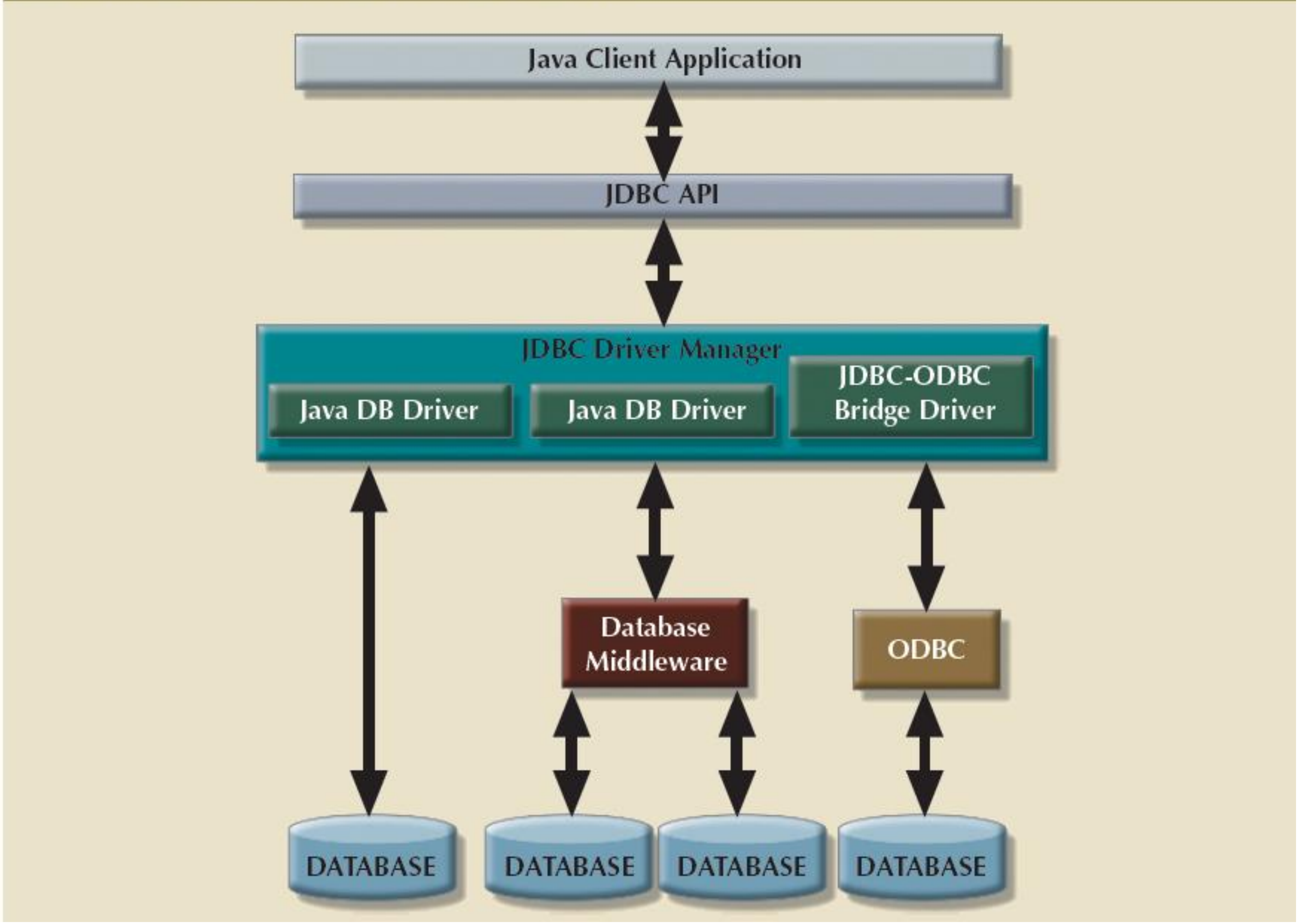
FIGURE 15.2 USING ODBC, DAO, AND RDO TO ACCESS DATABASES



Java Database Connectivity (JDBC)

- Application programming interface that allows a Java program to interact with a wide range of data sources
- Advantages of JDBC
  - Company can leverage existing technology and personnel training
  - Direct access to database server or access via database middleware
  - Programmers can use their SQL skills to manipulate the data in the company's databases
  - Provides a way to connect to databases through an ODBC driver

FIGURE 15.7 JDBC ARCHITECTURE



Database Internet Connectivity (1 of 2)

- Allows new innovative services
  - Permit rapid response by bringing new services and products to market quickly
  - Increase customer satisfaction through creation of innovative data services
  - Allow anywhere, anytime data access using mobile smart devices via the Internet
  - Yield fast and effective information dissemination through universal access

Database Internet Connectivity (2 of 2)

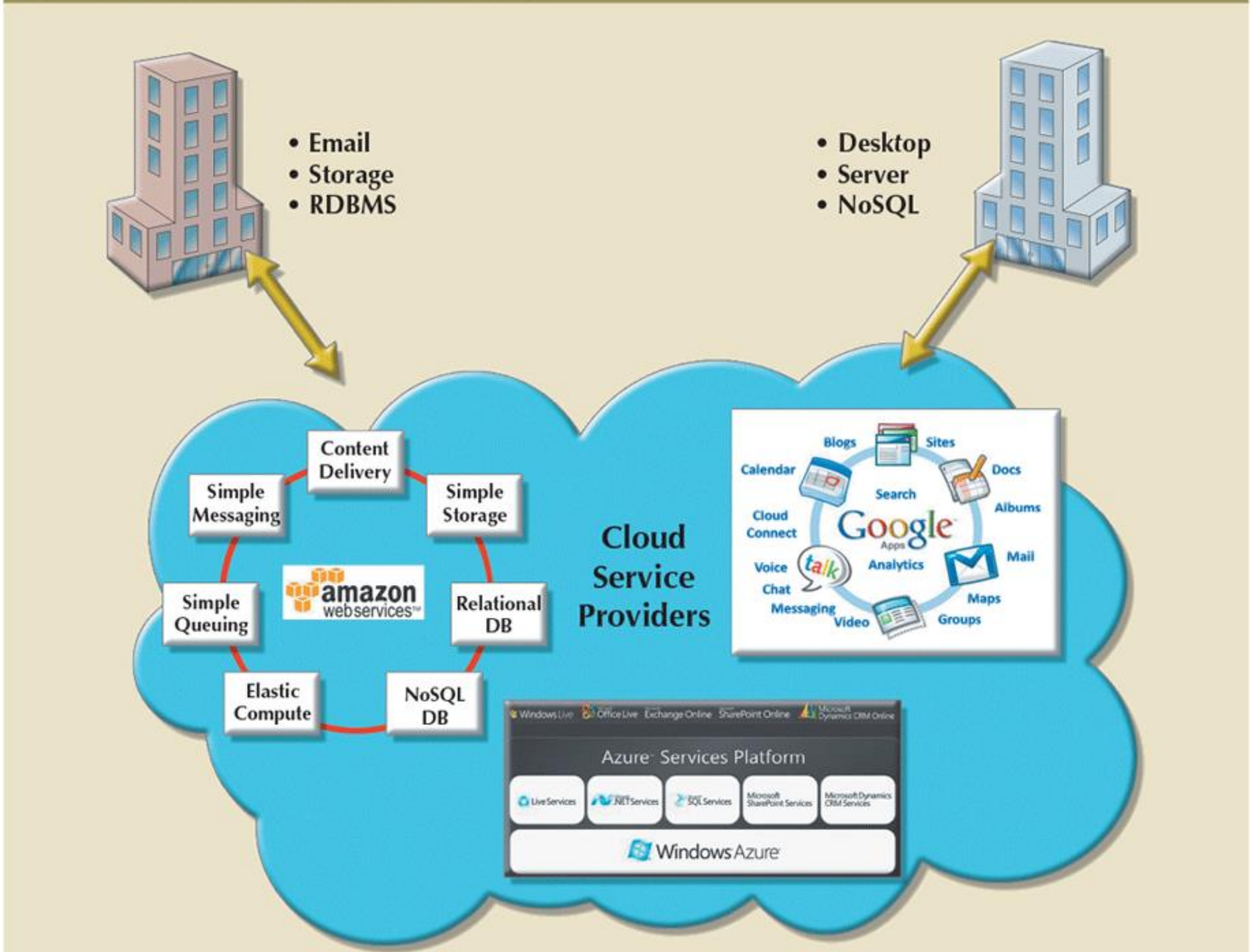
Table 15.3: Characteristics and Benefits of Internet Technologies	
Internet Characteristic	**Benefit**
Hardware and software Independence	Savings in equipment and software acquisition Ability to run on most existing equipment Platform independence and portability No need for multiple platform development
Common and simple user interface	Reduced training time and cost Reduced end-user support cost No need for multiple platform development
Location independence	Global access through Internet infrastructure and mobile smart devices Creation of new location-aware services Reduced requirements (and costs!) for dedicated connections
Rapid development at manageable costs	Availability of multiple development tools Plug-and-play development tools (open standards) More interactive development Reduced development times Relatively inexpensive tools Free client access tools (web browsers) Low entry costs; frequent availability of free web servers Reduced costs of maintaining private networks Distributed processing and scalability using multiple servers

Cloud Computing Services

- Computing model that enables access to a shared pool of configurable computer resources
  – Can be rapidly provisioned and released with minimal management effort or service provider interaction
  – Potential to become a game changer; eliminates financial and technological barriers

Cloud Computing Services

FIGURE 15.21 CLOUD SERVICES



Cloud Implementation Types

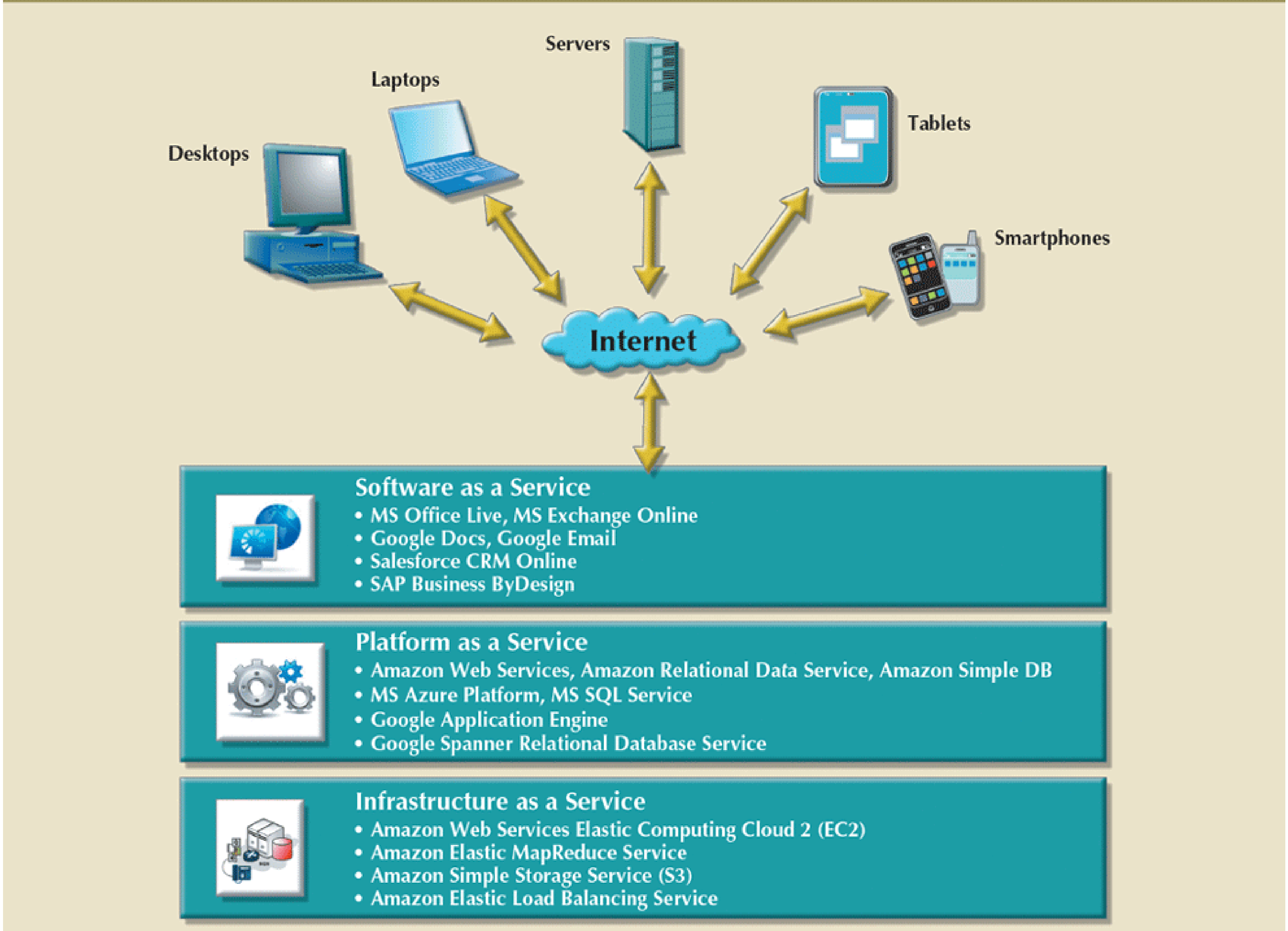
- Public cloud
  - Built by a third-party organization to sell cloud services to the general public
- Private cloud
  - Built by an organization for the sole purpose of servicing its own needs
- Community cloud
  - Built by and for a specific group of organizations that share a common trade

Characteristics of Cloud Services

- Cloud computing services share a set of guiding principles
  - Ubiquitous access via Internet technologies
  - Shared infrastructure
  - Lower costs and variable pricing
  - Flexible and scalable services
  - Dynamic provisioning
  - Service orientation
  - Managed operations

Types of Cloud Services

FIGURE 15.23 TYPES OF CLOUD SERVICES



Software as a Service
- MS Office Live, MS Exchange Online
- Google Docs, Google Email
- Salesforce CRM Online
- SAP Business ByDesign

Platform as a Service
- Amazon Web Services, Amazon Relational Data Service, Amazon Simple DB
- MS Azure Platform, MS SQL Service
- Google Application Engine
- Google Spanner Relational Database Service

Infrastructure as a Service
- Amazon Web Services Elastic Computing Cloud 2 (EC2)
- Amazon Elastic MapReduce Service
- Amazon Simple Storage Service (S3)
- Amazon Elastic Load Balancing Service

Cloud Services: Advantages and Disadvantages

Table 15.4: Advantages and Disadvantages of Cloud Computing	
Advantage	**Disadvantage**
Low initial cost of entry. Cloud computing has lower costs of entry when compared with the alternative of building in house.	Issues of security, privacy, and compliance. Trusting sensitive company data to external entities is difficult for most data-cautious organizations.
Scalability/elasticity. It is easy to add and remove resources on demand.	Hidden costs of implementation and operation. It is hard to estimate bandwidth and data migration costs.
Support for mobile computing. Cloud computing providers support multiple types of mobile computing devices.	Data migration is a difficult and lengthy process. Migrating large amounts of data to and from the cloud infrastructure can be difficult and time-consuming.
Ubiquitous access. Consumers can access the cloud resources from anywhere at any time, as long as they have Internet access.	Complex licensing schemes. Organizations that implement cloud services are faced with complex licensing schemes and complicated service-level agreements.
High reliability and performance. Cloud providers build solid infrastructures that otherwise are difficult for the average organization to leverage.	Loss of ownership and control. Companies that use cloud services are no longer in complete control of their data. What is the responsibility of the cloud provider if data are breached? Can the vendor use your data without your consent?
Fast provisioning. Resources can be provisioned on demand in a matter of minutes with minimal effort.	Organization culture. End users tend to be resistant to change. Do the savings justify being dependent on a single provider? Will the cloud provider be around in 10 years?
Managed infrastructure. Most cloud implementations are managed by dedicated internal or external staff. This allows the organization's IT staff to focus on other areas.	Difficult integration with internal IT system. Configuring the cloud services to integrate transparently with internal authentication and other internal services could be a daunting task.

SQL Data Services

- Cloud computing-based data management service
  - Provides relational data management to companies
  - Hosted data management and standard protocols
  - Standard protocols
  - Common programming interface
- Advantages
  - Reliable and scalable at a lower cost than in-house systems
  - High level of failure tolerance
  - Dynamic and automatic load balancing
  - Automated data backup and disaster recovery are included
  - Dynamic creation and allocation of processes and storage

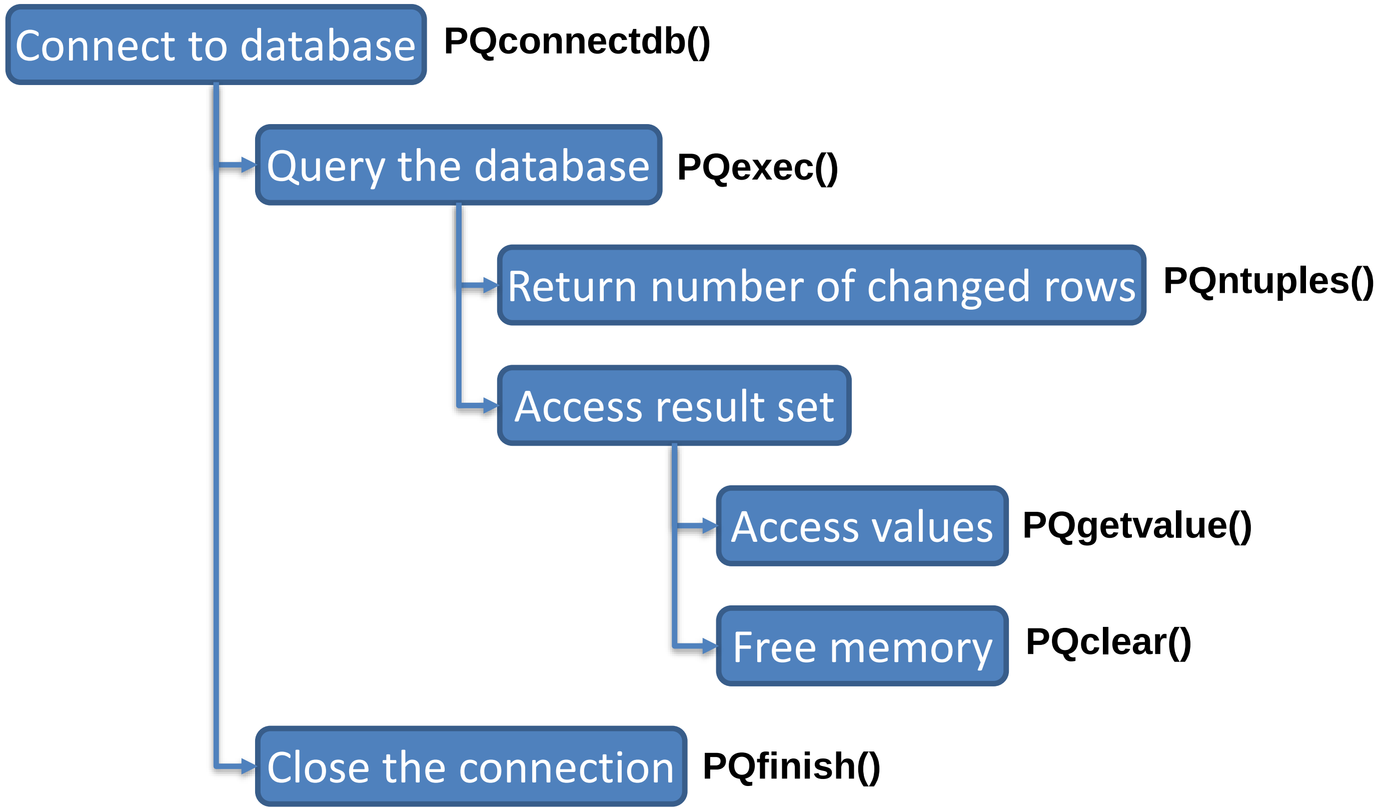
Outline

- Database Connectivity Foundation
- ✈ PostgreSQL C Connector
- Project
- Introduction to Python

Tutorial: PostgreSQL C Connector

- Pre-requirement 有一些需要安装的库文件
  - PostgreSQL Connector/C
    - x86 vs. x64
    - Can be found in PostgreSQL installation directory
  - C IDE & Compiler
    - Eclipse CDT + MinGW or Microsoft Visual Studio
    - Include “PgInstallationDir\include” in the include path
    - Add “PgInstallationDir\lib\libpq.lib” into the linker library
    - Include “PgInstallationDir\bin” to the system path

Tutorial: PostgreSQL C Connector



Please refer to https://www.postgresql.org/docs/10/libpq.html

Tutorial: PostgreSQL C Connector

```
#include <stdio.h>
#include <stdlib.h>
#include <libpq-fe.h>

/* Handle errors */
void do_exit(PGconn* conn, PGresult* res) {
    fprintf(stderr, "%s\n", PQerrorMessage(conn));
    PQclear(res);
    PQfinish(conn);
    exit(1);
}

int main() {
    const char* user = "postgres";
    const char* password =  “mypassword";
    const char* dbname = "postgres";
    char connInfo[256];
    sprintf(connInfo, "user=%s password=%s dbname=%s", user, password, dbname);

    /* PQconnectdb: setup connection */
    PGconn* conn = PQconnectdb(connInfo);
    /* PQstatus: check status*/
    if (PQstatus(conn) == CONNECTION_BAD) {
        fprintf(stderr, "Connection to database failed: %s\n", PQerrorMessage(conn));
        PQfinish(conn);
        exit(1);
    }
```

Tutorial: PostgreSQL C Connector

```
/* PQexec: drop table if needed */
PGresult* res = PQexec(conn, "DROP TABLE IF EXISTS Cars");
if (PQresultStatus(res) != PGRES_COMMAND_OK) {
    do_exit(conn, res);
}
/* PQexec: drop table if needed */
PQclear(res);    用于释放资源

/* PQexec: create table */
res = PQexec(conn, "CREATE TABLE Cars(Id INTEGER PRIMARY KEY, Name VARCHAR(20), Price INT)");
if (PQresultStatus(res) != PGRES_COMMAND_OK) {
    do_exit(conn, res);    意思是：出问题就退出
}
PQclear(res);

/* PQexec: insert data */
res = PQexec(conn,
    "INSERT INTO Cars VALUES (1,'Audi',52642), (2,'Mercedes',57127), (3,'Skoda',9000), " \
    "(4,'Volvo',29000), (5,'Bentley',350000), (6,'Citroen',21000), (7,'Hummer',41400)");
if (PQresultStatus(res) != PGRES_COMMAND_OK) {           红色的字就是在pgadmin能写的SQL语句
    do_exit(conn, res);
}
PQclear(res);
```

Tutorial: PostgreSQL C Connector

```
    /* PQexec: query */
    res = PQexec(conn, "SELECT * FROM Cars LIMIT 5");   进行select操作
    if (PQresultStatus(res) != PGRES_TUPLES_OK) {
        printf("No data retrieved\n"); 如果出问题了就提示
        do_exit(conn, res);
    }

    /* PQntuples: get result rows */
    int rows = PQntuples(res);
    /* PQgetvalue: get result values */
    for (int i = 0; i < rows; i++) {  这个的意思是：把输出挨个print
        printf("%s %s %s\n", PQgetvalue(res, i, 0),
            PQgetvalue(res, i, 1), PQgetvalue(res, i, 2));
    }
    PQclear(res);

    /* PQfinish: close the connection */
    PQfinish(conn);
    return 0;
}
```

Outline

- Database Connectivity Foundation
- PostgreSQL C Connector
- Project
- ✈ Introduction to Python
    - Python in a Nutshell
    - PG Python Connector

Introduction to Python

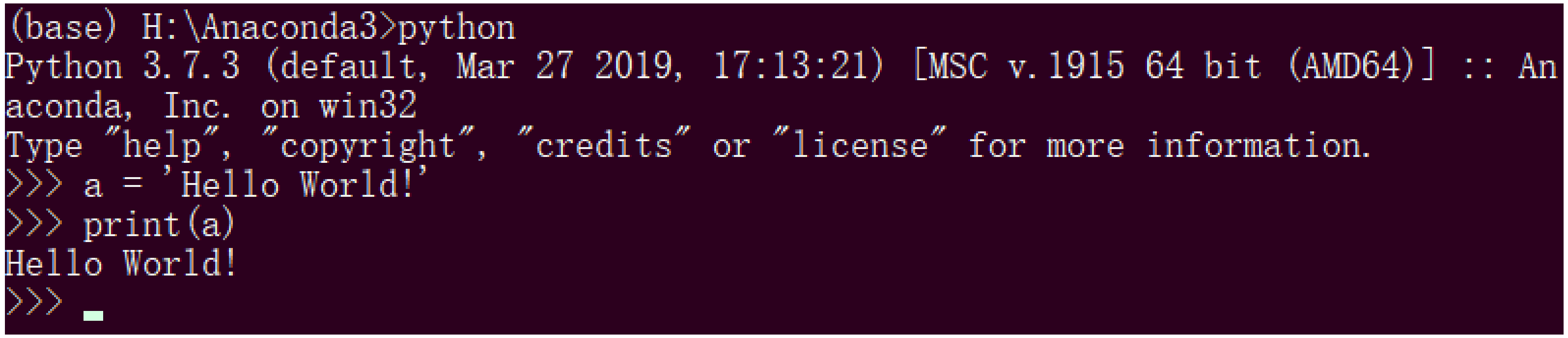
- Open source general-purpose language
- Object Oriented, Procedural, Functional
- Easy to interface with C/C++/ObjC/Java/Fortran
- Great interactive environment

Introduction to Python

- Environment
  - Python 2.x vs Python 3.x
  - Package managers and distributions
    - Anaconda/Pip/…
  - Recommend for Windows user
    - Install Anaconda2/Anaconda3
      - https://www.anaconda.com/download/ or
      - https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/
    - Default install path
      - C:\ProgramData\AnacondaX
        - » C:\ProgramData\AnacondaX\Scripts
          - conda install *packagename*
          - pip install *packagename*

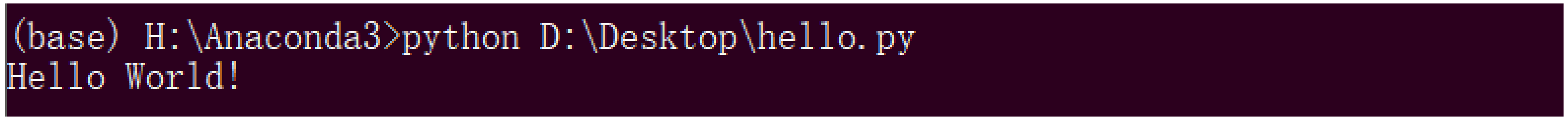
First Python Program

- Interactive mode



```
(base) H:\Anaconda3>python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: An
aconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 'Hello World!'
>>> print(a)
Hello World!
>>> _
```

- Script mode



```
(base) H:\Anaconda3>python D:\Desktop\hello.py
Hello World!
```

Basic Python Syntax

- Statements style

```
balance = 200
withdraws = 150
if withdraws % 100 == 0:
   if balance >= withdraws:
       print("Withdraws %d successfully, current balance is %d" % (withdraws, (balance - withdraws)))
else:
   print("Only notes in 100 yuan is available!")
```

- Multi-Line Statements

```
total = item_one + \
        item_two + \
        item_three
```

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

- Quotation & comment

```
In [29]: # First comment
    ...: print("Hello, 'Python'!") # second comment
    ...: '''This is a multi line comment
    ...: print("You can't see me!")
    ...: In fact it is a paragraph'''
    ...: print('Have fun!')
    ...:
Hello, 'Python'!
Have fun!
```

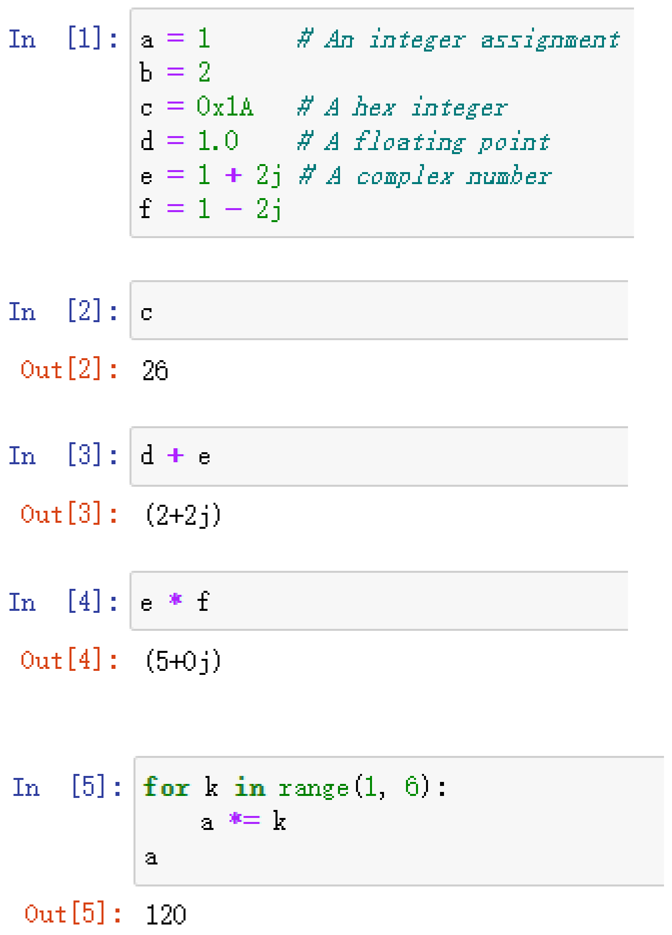
Basic Python Syntax

- Operators
  - +- * / // % **
  - < <= > >= == !=
  - and or not

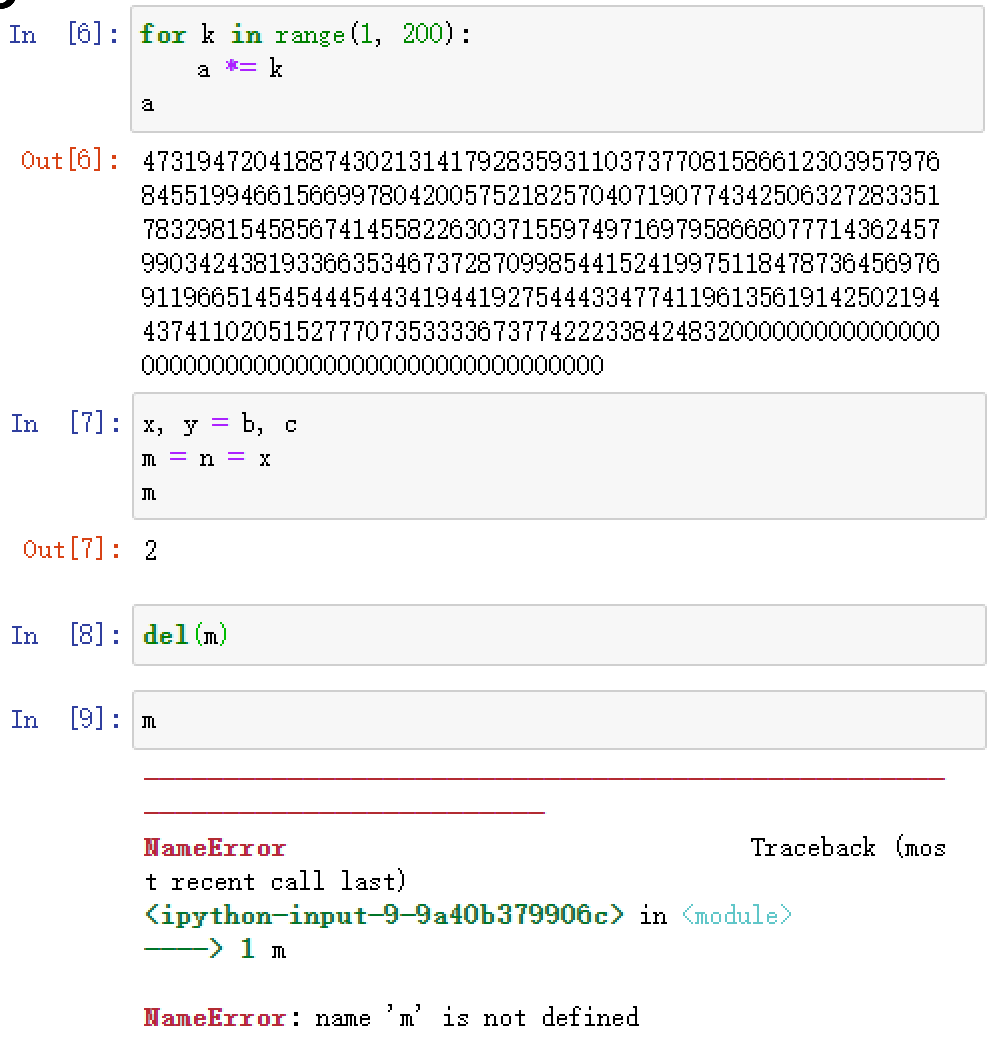
```
>>> a = 5
>>> -2 * 4 + a ** 2
17
>>> a / 2
2.5
>>> a / 2.0
2.5
>>> a // 2.0
2.0
>>> 77 > 66 == 66  # same as (77 > 66) and (66 == 66)
True
>>> a == a / 2 * 2 + a % 2
False
>>> a == a // 2 * 2 + a % 2
True
>>> a = 'Hello' + " " + 'World!'
>>> print(a)
Hello World!
```

Python Data Types

- Numbers and assignment



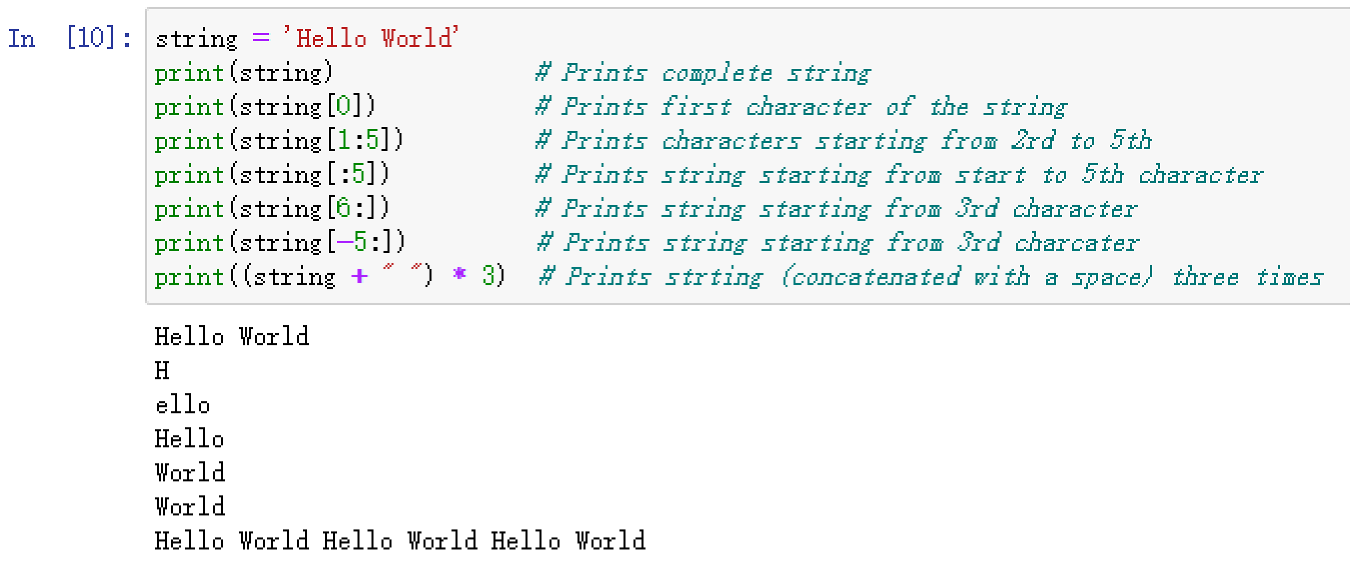
```
In  [1]: a = 1       # An integer assignment
         b = 2
         c = 0x1A    # A hex integer
         d = 1.0     # A floating point
         e = 1 + 2j  # A complex number
         f = 1 - 2j

In  [2]: c

Out[2]:  26

In  [3]: d + e

Out[3]:  (2+2j)

In  [4]: e * f

Out[4]:  (5+0j)

In  [5]: for k in range(1, 6):
             a *= k
         a

Out[5]:  120
```



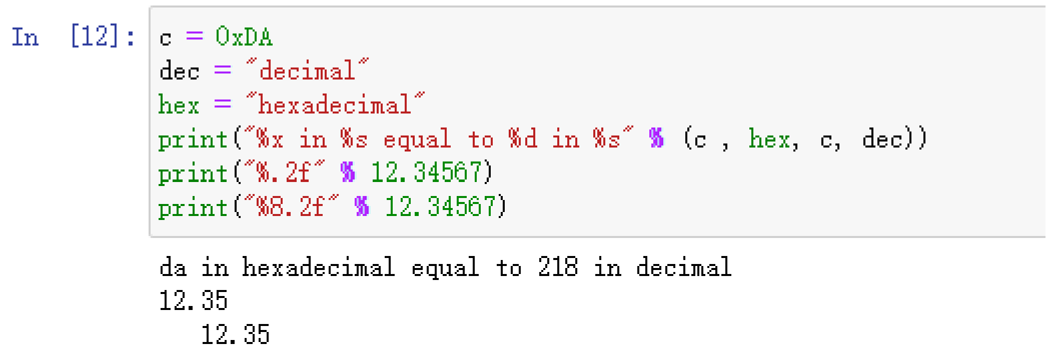
```
In  [6]: for k in range(1, 200):
             a *= k
         a

Out[6]:  473194720418874302131417928359311037377081586612303957976
         845519946615669978042005752182570407190774342506327283351
         783298154585674145582263037155974971697958668077714362457
         990342438193366353467372870998544152419975118478736456976
         911966514545444544341944192754443347741196135619142502194
         437411020515277707353333673774222338424832000000000000000
         000000000000000000000000000000000

In  [7]: x, y = b, c
         m = n = x
         m

Out[7]:  2

In  [8]: del(m)

In  [9]: m

---------------------------------------------------------------------------
NameError                                 Traceback (mos
t recent call last)
<ipython-input-9-9a40b379906c> in <module>
----> 1 m

NameError: name 'm' is not defined
```

Python Data Types

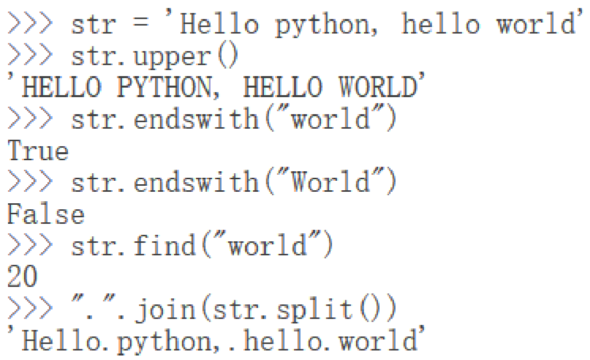
- String



```
In  [10]: string = 'Hello World'
          print(string)            # Prints complete string
          print(string[0])         # Prints first character of the string
          print(string[1:5])       # Prints characters starting from 2rd to 5th
          print(string[:5])        # Prints string starting from start to 5th character
          print(string[6:])        # Prints string starting from 3rd character
          print(string[-5:])       # Prints string starting from 3rd charcater
          print((string + " ") * 3)  # Prints strting (concatenated with a space) three times

          Hello World
          H
          ello
          Hello
          World
          World
          Hello World Hello World Hello World
```



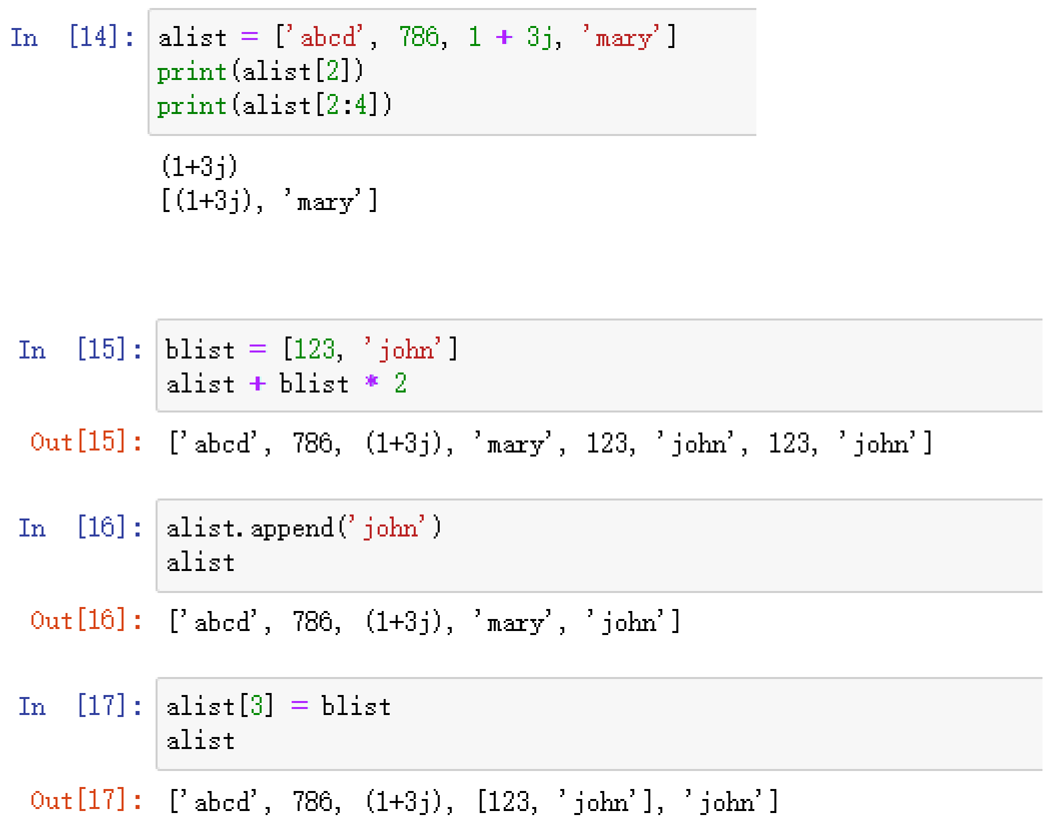
```
In  [12]: c = 0xDA
          dec = "decimal"
          hex = "hexadecimal"
          print("%x in %s equal to %d in %s" % (c , hex, c, dec))
          print("%.2f" % 12.34567)
          print("%8.2f" % 12.34567)

          da in hexadecimal equal to 218 in decimal
          12.35
             12.35
```



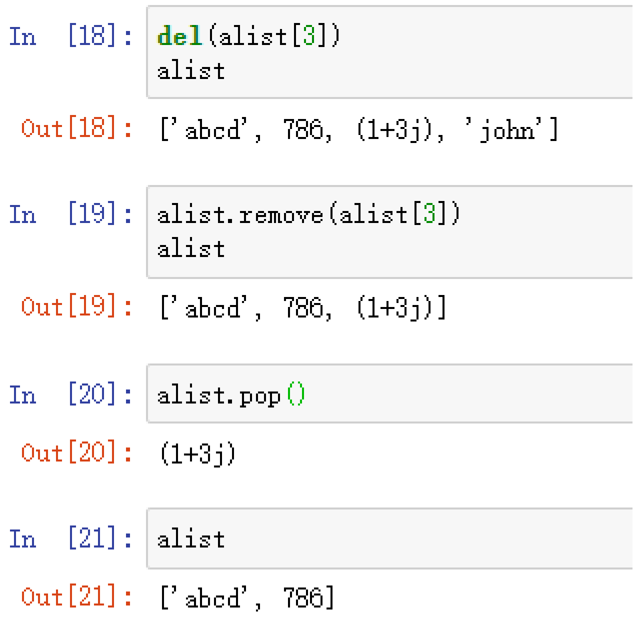
```
>>> str = 'Hello python, hello world'
>>> str.upper()
'HELLO PYTHON, HELLO WORLD'
>>> str.endswith("world")
True
>>> str.endswith("World")
False
>>> str.find("world")
20
>>> ".".join(str.split())
'Hello.python,.hello.world'
```

Python Data Types

- List



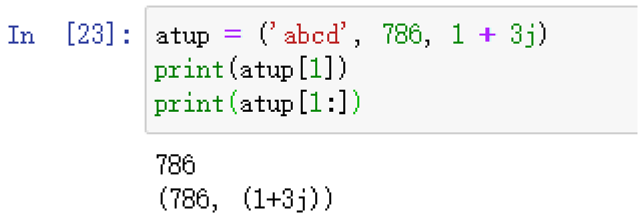
```
In [14]: alist = ['abcd', 786, 1 + 3j, 'mary']
         print(alist[2])
         print(alist[2:4])

         (1+3j)
         [(1+3j), 'mary']

In [15]: blist = [123, 'john']
         alist + blist * 2

Out[15]: ['abcd', 786, (1+3j), 'mary', 123, 'john', 123, 'john']

In [16]: alist.append('john')
         alist

Out[16]: ['abcd', 786, (1+3j), 'mary', 'john']

In [17]: alist[3] = blist
         alist

Out[17]: ['abcd', 786, (1+3j), [123, 'john'], 'john']
```



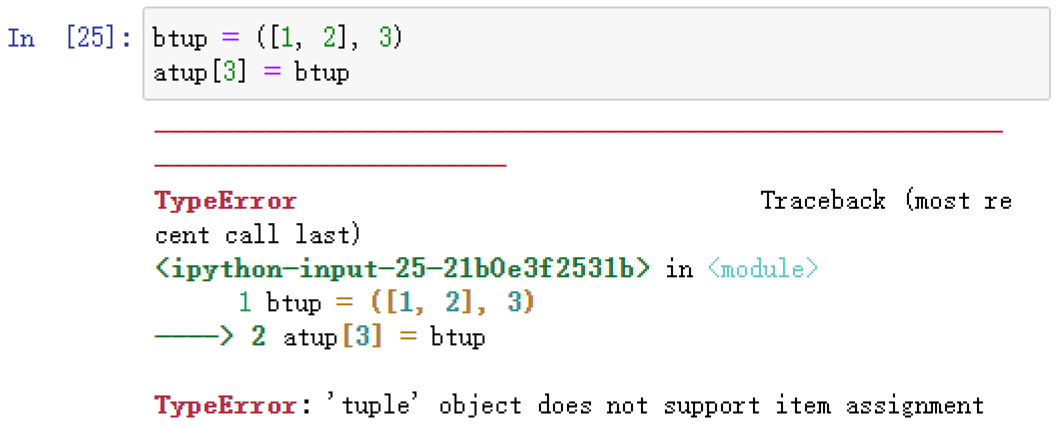
```
In [18]: del(alist[3])
         alist

Out[18]: ['abcd', 786, (1+3j), 'john']

In [19]: alist.remove(alist[3])
         alist

Out[19]: ['abcd', 786, (1+3j)]

In [20]: alist.pop()

Out[20]: (1+3j)

In [21]: alist

Out[21]: ['abcd', 786]
```

Python Data Types

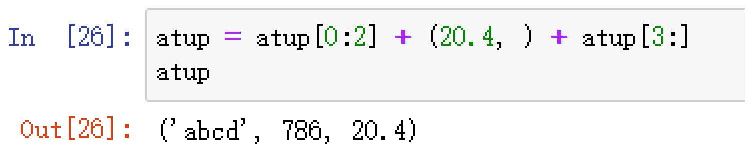
- Tuple
  – Tuples can be thought of as **read-only** lists



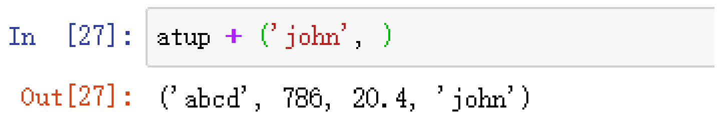
```
In  [23]: atup = ('abcd', 786, 1 + 3j)
          print(atup[1])
          print(atup[1:])

          786
          (786, (1+3j))
```



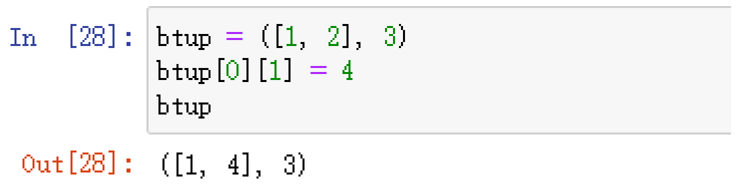
```
In  [25]: btup = ([1, 2], 3)
          atup[3] = btup

          ---------------------------------------------------------------------------
          TypeError                                 Traceback (most re
          cent call last)
          <ipython-input-25-21b0e3f2531b> in <module>
                1 btup = ([1, 2], 3)
          ----> 2 atup[3] = btup

          TypeError: 'tuple' object does not support item assignment
```



```
In  [26]: atup = atup[0:2] + (20.4, ) + atup[3:]
          atup

Out[26]: ('abcd', 786, 20.4)
```



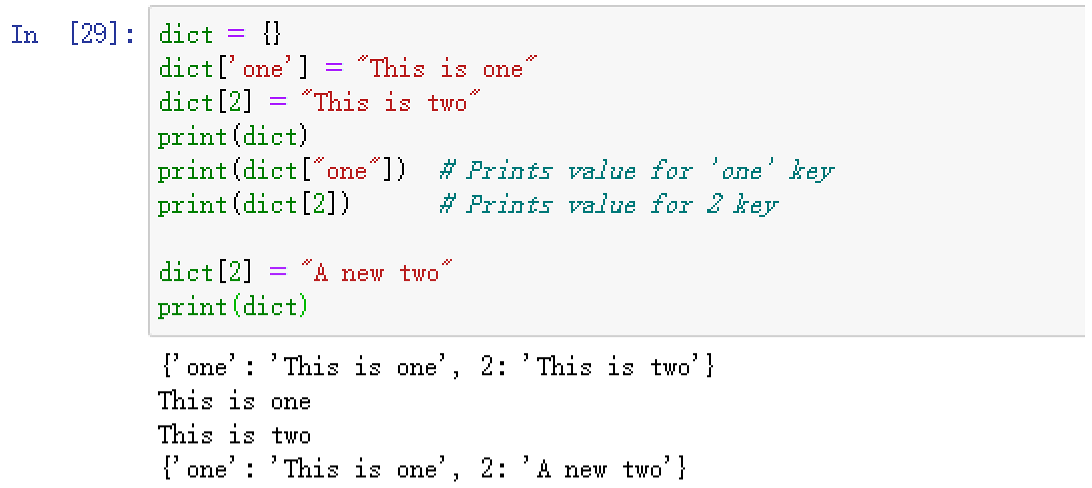
```
In  [27]: atup + ('john', )

Out[27]: ('abcd', 786, 20.4, 'john')
```



```
In  [28]: btup = ([1, 2], 3)
          btup[0][1] = 4
          btup

Out[28]: ([1, 4], 3)
```

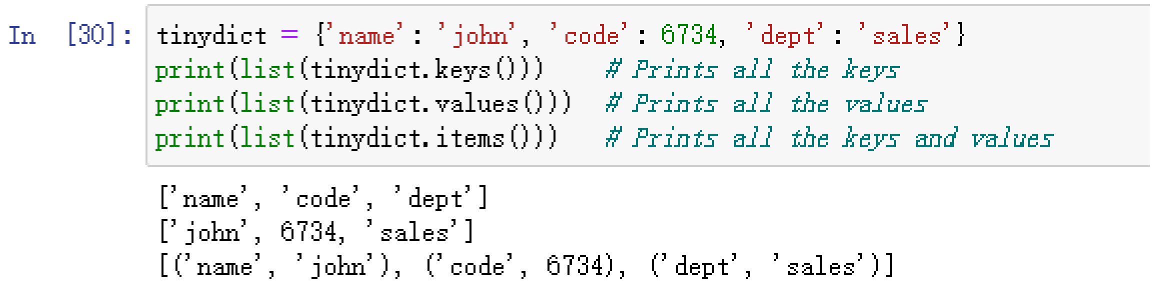
Python Data Types

- Dictionary



```
In [29]: dict = {}
         dict['one'] = "This is one"
         dict[2] = "This is two"
         print(dict)
         print(dict["one"])  # Prints value for 'one' key
         print(dict[2])      # Prints value for 2 key

         dict[2] = "A new two"
         print(dict)
```

```
{'one': 'This is one', 2: 'This is two'}
This is one
This is two
{'one': 'This is one', 2: 'A new two'}
```

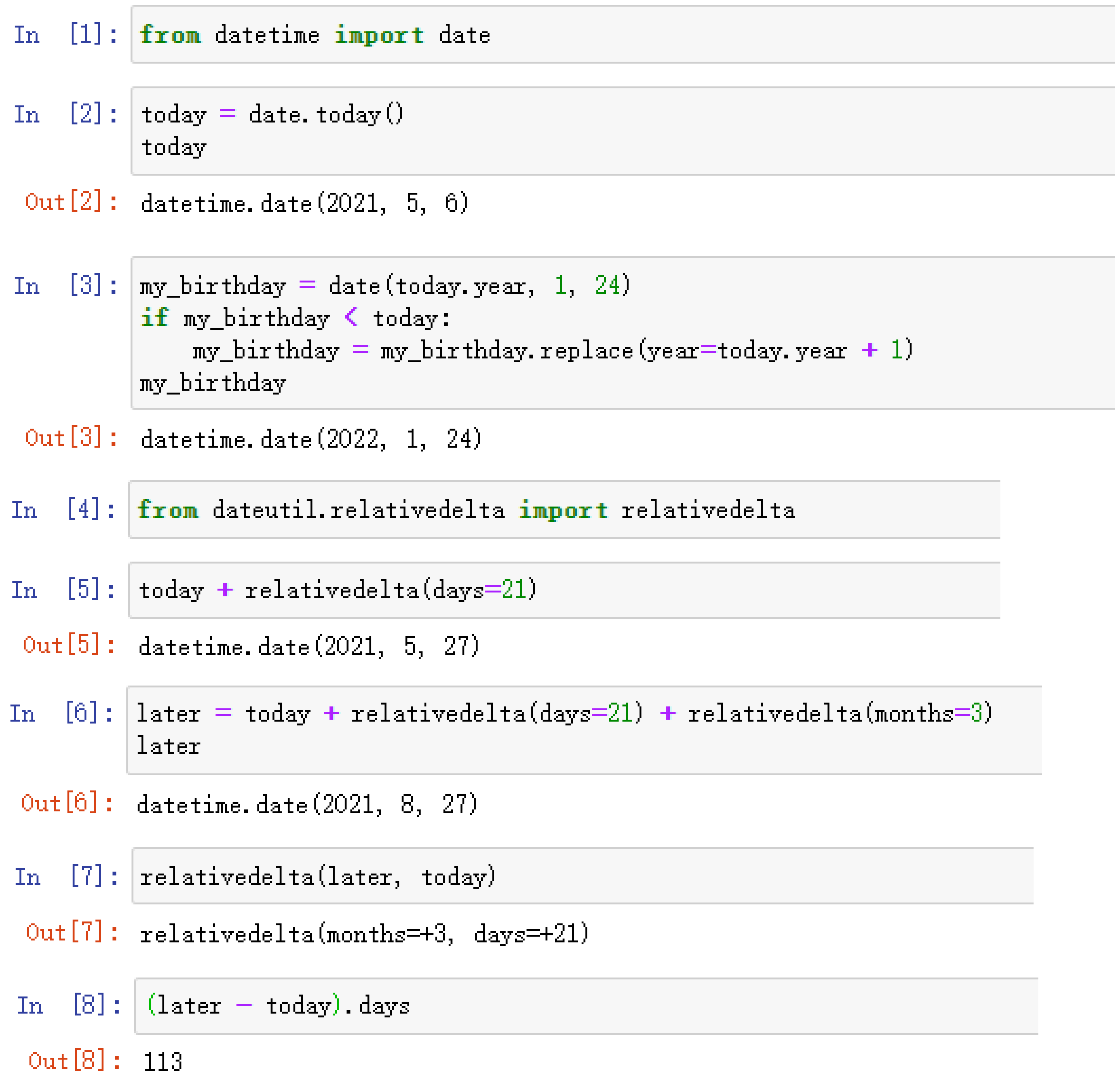


```
In [30]: tinydict = {'name': 'john', 'code': 6734, 'dept': 'sales'}
         print(list(tinydict.keys()))    # Prints all the keys
         print(list(tinydict.values()))  # Prints all the values
         print(list(tinydict.items()))   # Prints all the keys and values
```

```
['name', 'code', 'dept']
['john', 6734, 'sales']
[('name', 'john'), ('code', 6734), ('dept', 'sales')]
```

Python Data Types

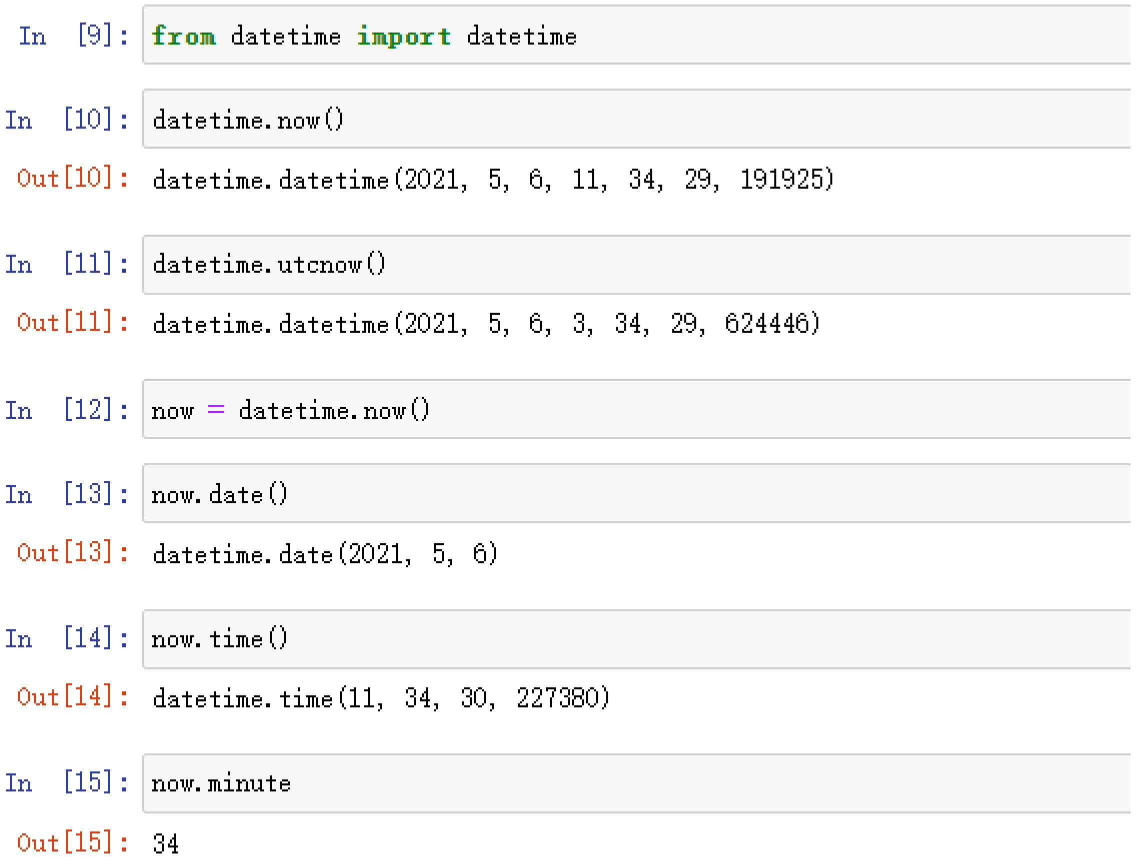
- Date & Time



```
In [1]: from datetime import date

In [2]: today = date.today()
        today

Out[2]: datetime.date(2021, 5, 6)

In [3]: my_birthday = date(today.year, 1, 24)
        if my_birthday < today:
            my_birthday = my_birthday.replace(year=today.year + 1)
        my_birthday

Out[3]: datetime.date(2022, 1, 24)

In [4]: from dateutil.relativedelta import relativedelta

In [5]: today + relativedelta(days=21)

Out[5]: datetime.date(2021, 5, 27)

In [6]: later = today + relativedelta(days=21) + relativedelta(months=3)
        later

Out[6]: datetime.date(2021, 8, 27)

In [7]: relativedelta(later, today)

Out[7]: relativedelta(months=+3, days=+21)

In [8]: (later - today).days

Out[8]: 113
```

Python Data Types

- Date & Time



```
In  [9]: from datetime import datetime

In [10]: datetime.now()

Out[10]: datetime.datetime(2021, 5, 6, 11, 34, 29, 191925)

In [11]: datetime.utcnow()

Out[11]: datetime.datetime(2021, 5, 6, 3, 34, 29, 624446)

In [12]: now = datetime.now()

In [13]: now.date()

Out[13]: datetime.date(2021, 5, 6)

In [14]: now.time()

Out[14]: datetime.time(11, 34, 30, 227380)

In [15]: now.minute

Out[15]: 34
```

Python Statements

- Conditions
  - if… elif… else…

```python
if user.cmd== 'create':
  action = "create item"
elif user.cmd == 'delete':
  action = "delete item"
elif user.cmd == 'update':
  action = "update item"
else:
  action = "invalid command, try again!"
```

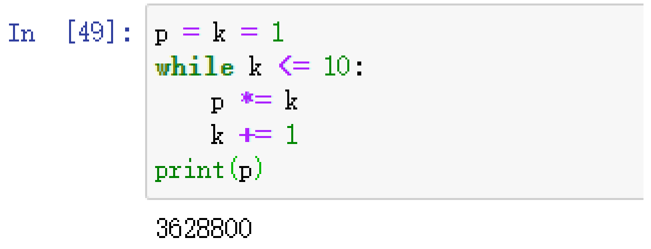
```python
if user.cmd in ('create', 'delete', 'update'):
  action = "%s item" % user.cmd
else:
  action = "invalid command, try again!"
```

  - Ternary conditional operator

```
In  [48]: x, y = 4, 3
          smaller = x if x < y else y
          smaller

Out[48]: 3
```

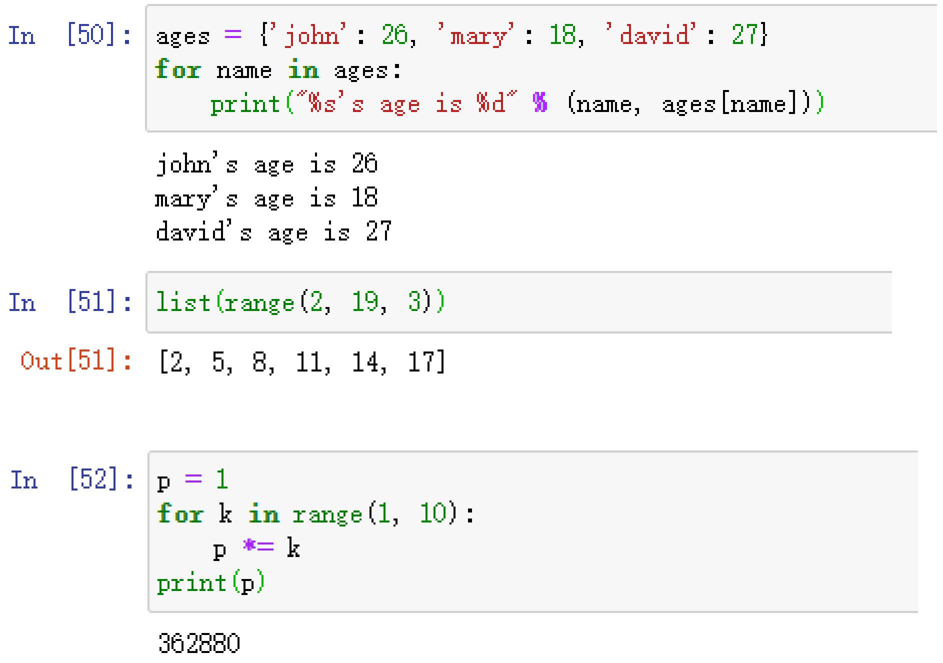
Python Statements

- Loops
  - while



```
In  [49]: p = k = 1
          while k <= 10:
              p *= k
              k += 1
          print(p)

          3628800
```

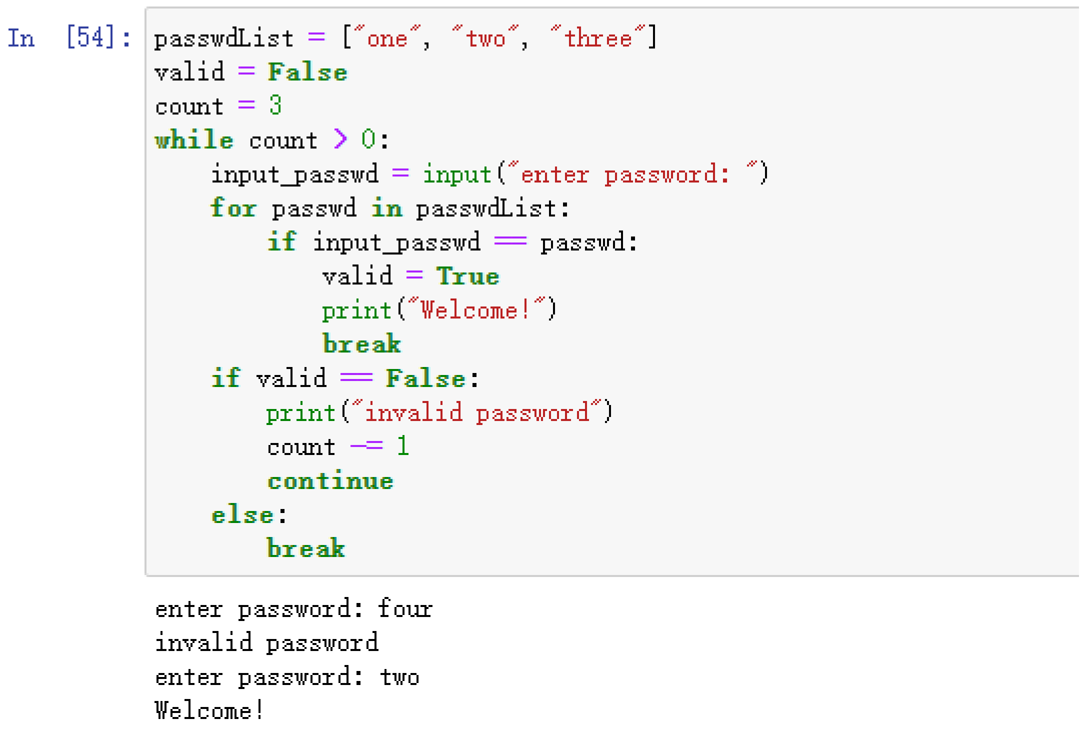
  - for



```
In  [50]: ages = {'john': 26, 'mary': 18, 'david': 27}
          for name in ages:
              print("%s's age is %d" % (name, ages[name]))

          john's age is 26
          mary's age is 18
          david's age is 27

In  [51]: list(range(2, 19, 3))

Out[51]:  [2, 5, 8, 11, 14, 17]

In  [52]: p = 1
          for k in range(1, 10):
              p *= k
          print(p)

          362880
```

Python Statements

- break & continue

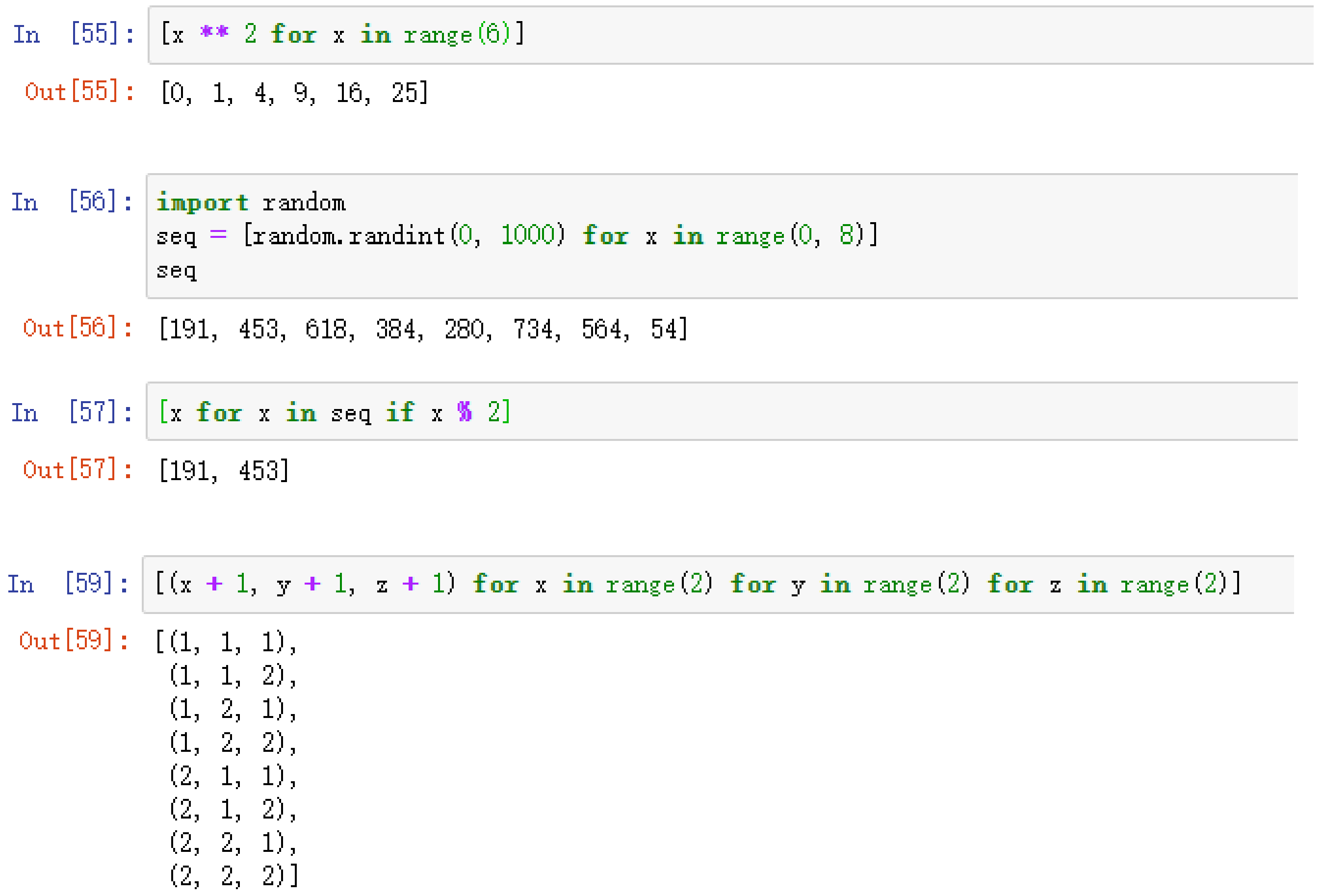


```
In  [54]: passwdList = ["one", "two", "three"]
          valid = False
          count = 3
          while count > 0:
              input_passwd = input("enter password: ")
              for passwd in passwdList:
                  if input_passwd == passwd:
                      valid = True
                      print("Welcome!")
                      break
              if valid == False:
                  print("invalid password")
                  count -= 1
                  continue
              else:
                  break
```

```
enter password: four
invalid password
enter password: two
Welcome!
```

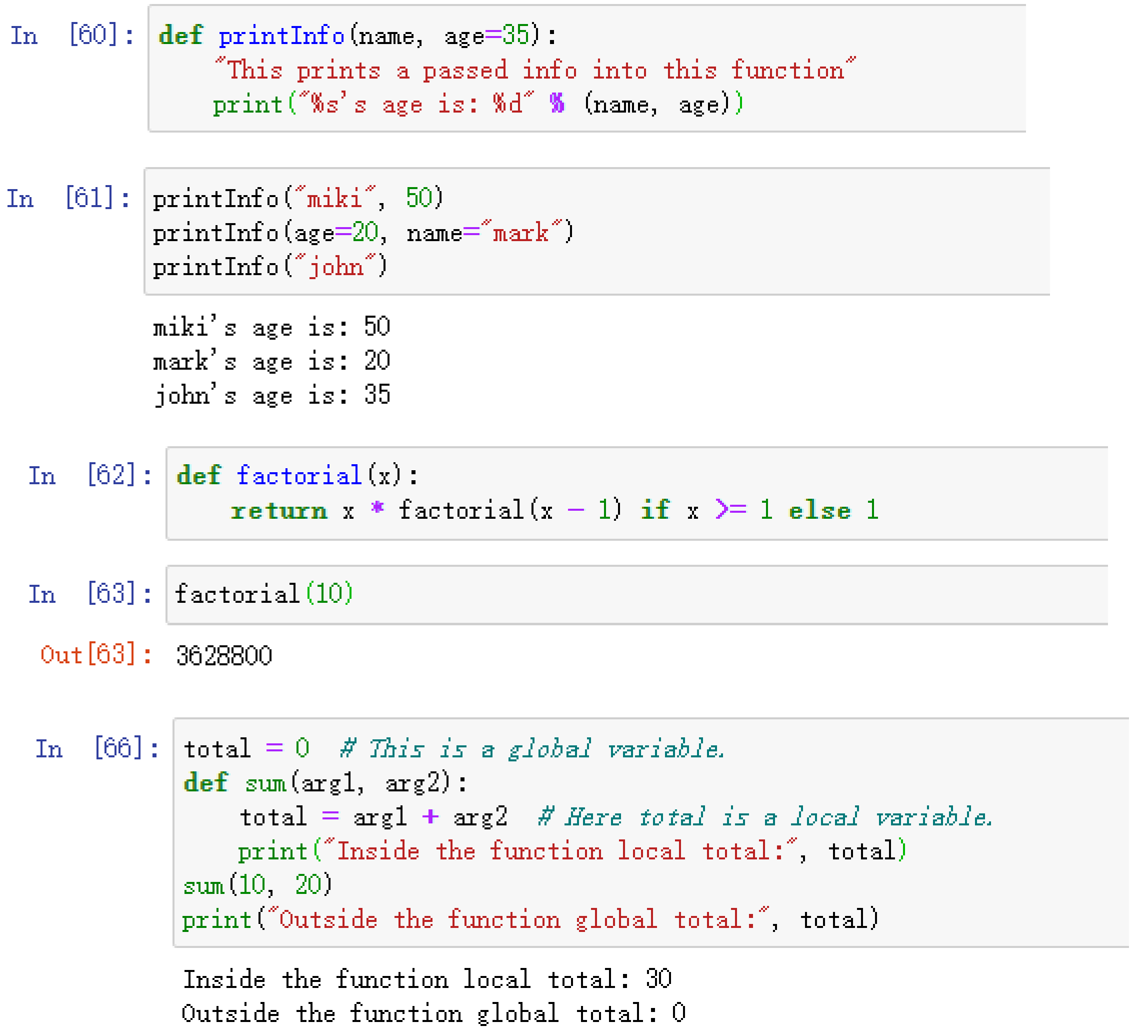
Python Statements

- List comprehensions



```
In  [55]: [x ** 2 for x in range(6)]

Out[55]: [0, 1, 4, 9, 16, 25]


In  [56]: import random
          seq = [random.randint(0, 1000) for x in range(0, 8)]
          seq

Out[56]: [191, 453, 618, 384, 280, 734, 564, 54]

In  [57]: [x for x in seq if x % 2]

Out[57]: [191, 453]


In  [59]: [(x + 1, y + 1, z + 1) for x in range(2) for y in range(2) for z in range(2)]

Out[59]: [(1, 1, 1),
          (1, 1, 2),
          (1, 2, 1),
          (1, 2, 2),
          (2, 1, 1),
          (2, 1, 2),
          (2, 2, 1),
          (2, 2, 2)]
```

Python Functions



```
In  [60]: def printInfo(name, age=35):
              "This prints a passed info into this function"
              print("%s’s age is: %d" % (name, age))
```

```
In  [61]: printInfo("miki", 50)
          printInfo(age=20, name="mark")
          printInfo("john")

          miki’s age is: 50
          mark’s age is: 20
          john’s age is: 35
```

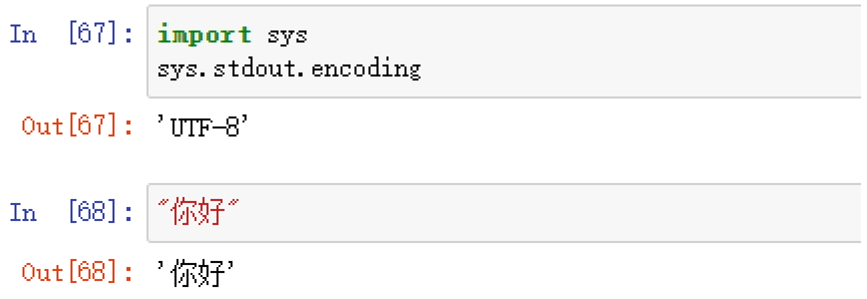
```
In  [62]: def factorial(x):
              return x * factorial(x - 1) if x >= 1 else 1
```

```
In  [63]: factorial(10)

Out[63]: 3628800
```

```
In  [66]: total = 0  # This is a global variable.
          def sum(arg1, arg2):
              total = arg1 + arg2  # Here total is a local variable.
              print("Inside the function local total:", total)
          sum(10, 20)
          print("Outside the function global total:", total)

          Inside the function local total: 30
          Outside the function global total: 0
```

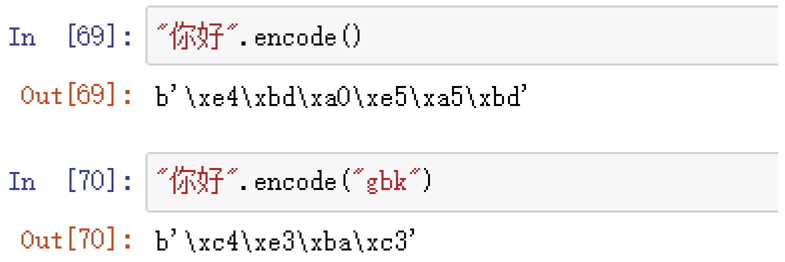
Python Encoding & Decoding

- In python3, the encoding is always “utf8”.



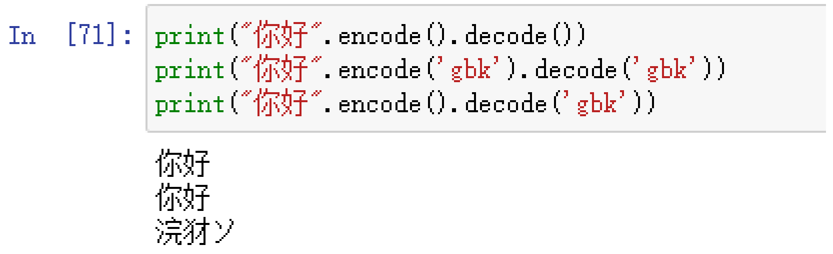
```
In  [67]: import sys
          sys.stdout.encoding

Out[67]: 'UTF-8'

In  [68]: "你好"

Out[68]: '你好'
```

- However, you can encode into other encoding.



```
In  [69]: "你好".encode()

Out[69]: b'\xe4\xbd\xa0\xe5\xa5\xbd'

In  [70]: "你好".encode("gbk")

Out[70]: b'\xc4\xe3\xba\xc3'
```

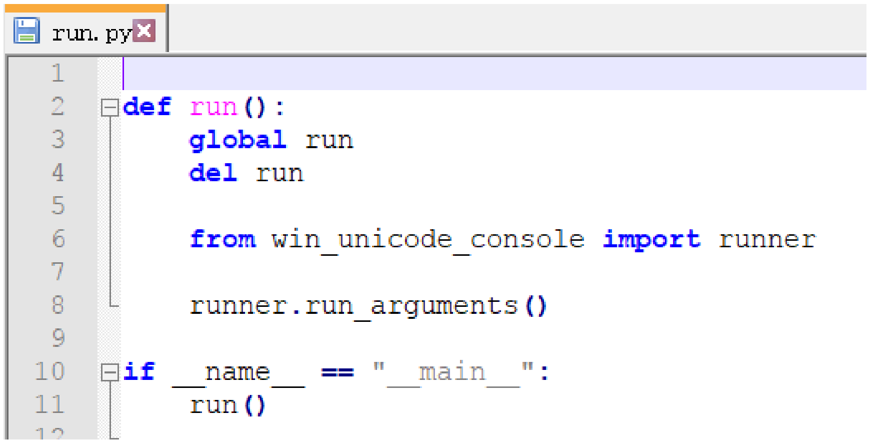
- Or decode after encoding.



```
In  [71]: print("你好".encode().decode())
          print("你好".encode('gbk').decode('gbk'))
          print("你好".encode().decode('gbk'))

          你好
          你好
          浣犲ソ
```

Python Modules

- import
  - import random
  - from datetime import date, time, datetime
- Installed modules
  - C:\ProgramData\AnacondaX\Lib
  - C:\ProgramData\AnacondaX\Lib\site-packages


run.py

```

def run():
    global run
    del run

    from win_unicode_console import runner

    runner.run_arguments()

if __name__ == "__main__":
    run()
```

Python Modules

- PostgreSQL adapter for Python
  - psycopg2
    - pip install psycopg2
    - https://www.psycopg.org/docs/
  - py-postgresql
    - pip install py-postgresql
    - https://pythonhosted.org/py-postgresql/

Outline

- Database Connectivity Foundation
- PostgreSQL C Connector
- Python in a Nutshell
- ✈ PG Python Connector
- Project

Python Modules

- psycopg2
  - Connection establishment

connection类表示数据库连接对象。由psycopg2.connect()方法创建。
conn是connection类下的一个instance

```
In  [74]: import psycopg2
```

引入这个包

本地的话，就写local host或者这个数字

```
In  [75]: conn = psycopg2.connect(database='postgres', user='postgres', password=password, host='127.0.0.1', port=5432)
```

这个用的多

使用具体函数的时候记得写psycopg2.

结构一直是这个

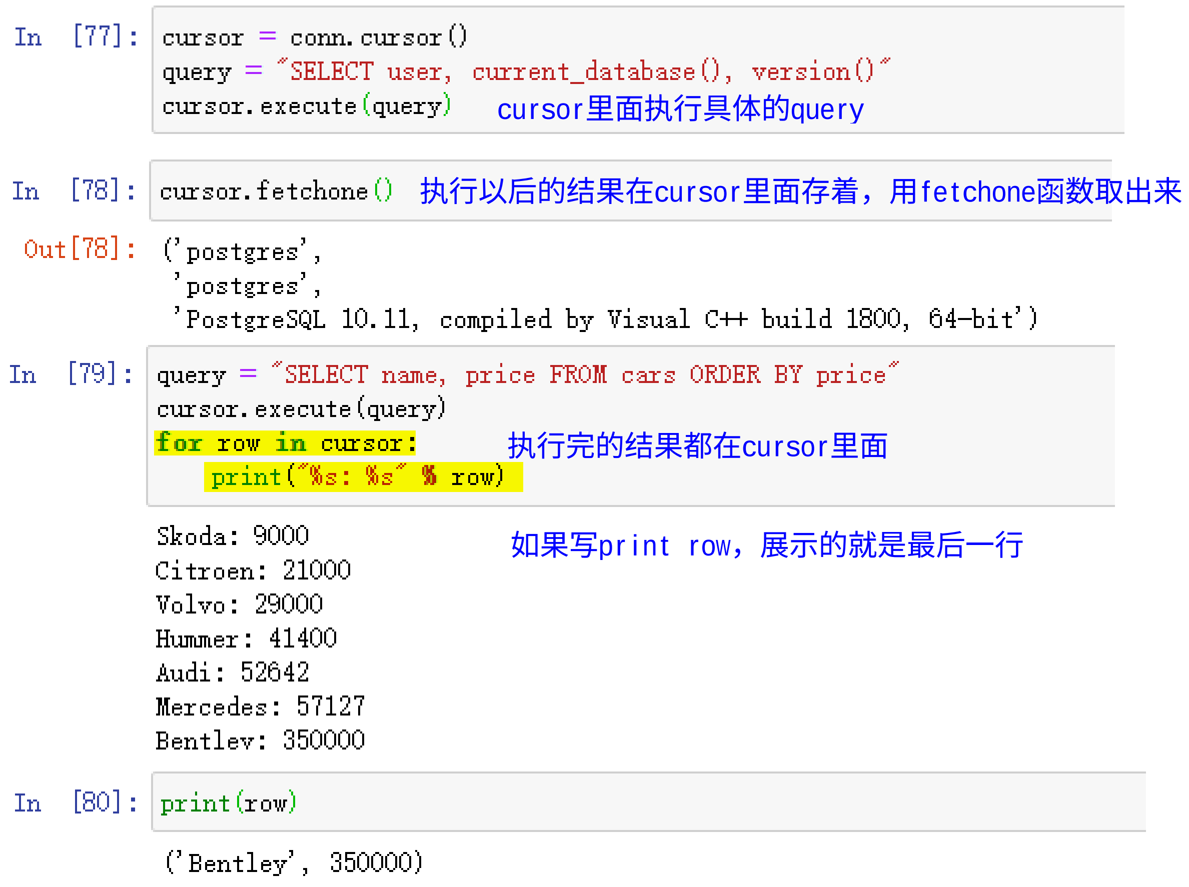
config相当于一个字典

```
In  [76]: config = {
              'user': 'postgres',
              'password': password,
              'host': 'localhost',
              'database': 'postgres',
              'port': 5432
          }

          conn = psycopg2.connect(**config)
```

建立connection是用**config

Python Modules

- psycopg2
  - Querying data



```
In  [77]: cursor = conn.cursor()
          query = "SELECT user, current_database(), version()"
          cursor.execute(query)
```
cursor里面执行具体的query

```
In  [78]: cursor.fetchone()
```
执行以后的结果在cursor里面存着，用fetchone函数取出来

```
Out[78]: ('postgres',
          'postgres',
          'PostgreSQL 10.11, compiled by Visual C++ build 1800, 64-bit')
```

```
In  [79]: query = "SELECT name, price FROM cars ORDER BY price"
          cursor.execute(query)
          for row in cursor:
              print("%s: %s" % row)
```
执行完的结果都在cursor里面

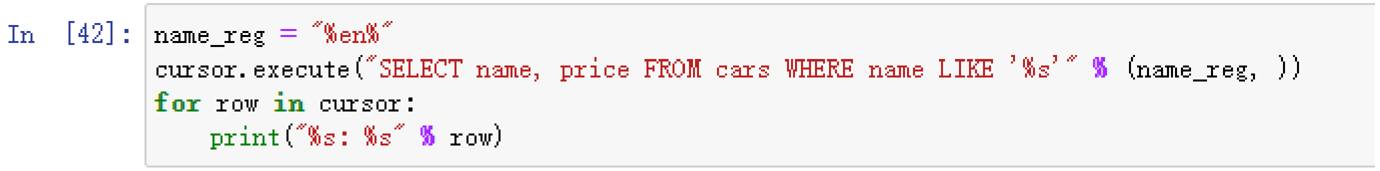
```
Skoda: 9000
Citroen: 21000
Volvo: 29000
Hummer: 41400
Audi: 52642
Mercedes: 57127
Bentley: 350000
```
如果写print row，展示的就是最后一行

```
In  [80]: print(row)
```

```
('Bentley', 350000)
```

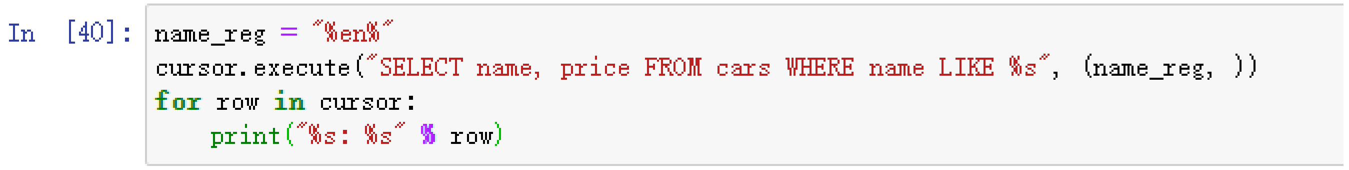
Python Modules

- psycopg2
  - Querying data



```
In  [42]: name_reg = "%en%"
          cursor.execute("SELECT name, price FROM cars WHERE name LIKE '%s'" % (name_reg, ))
          for row in cursor:
              print("%s: %s" % row)
```

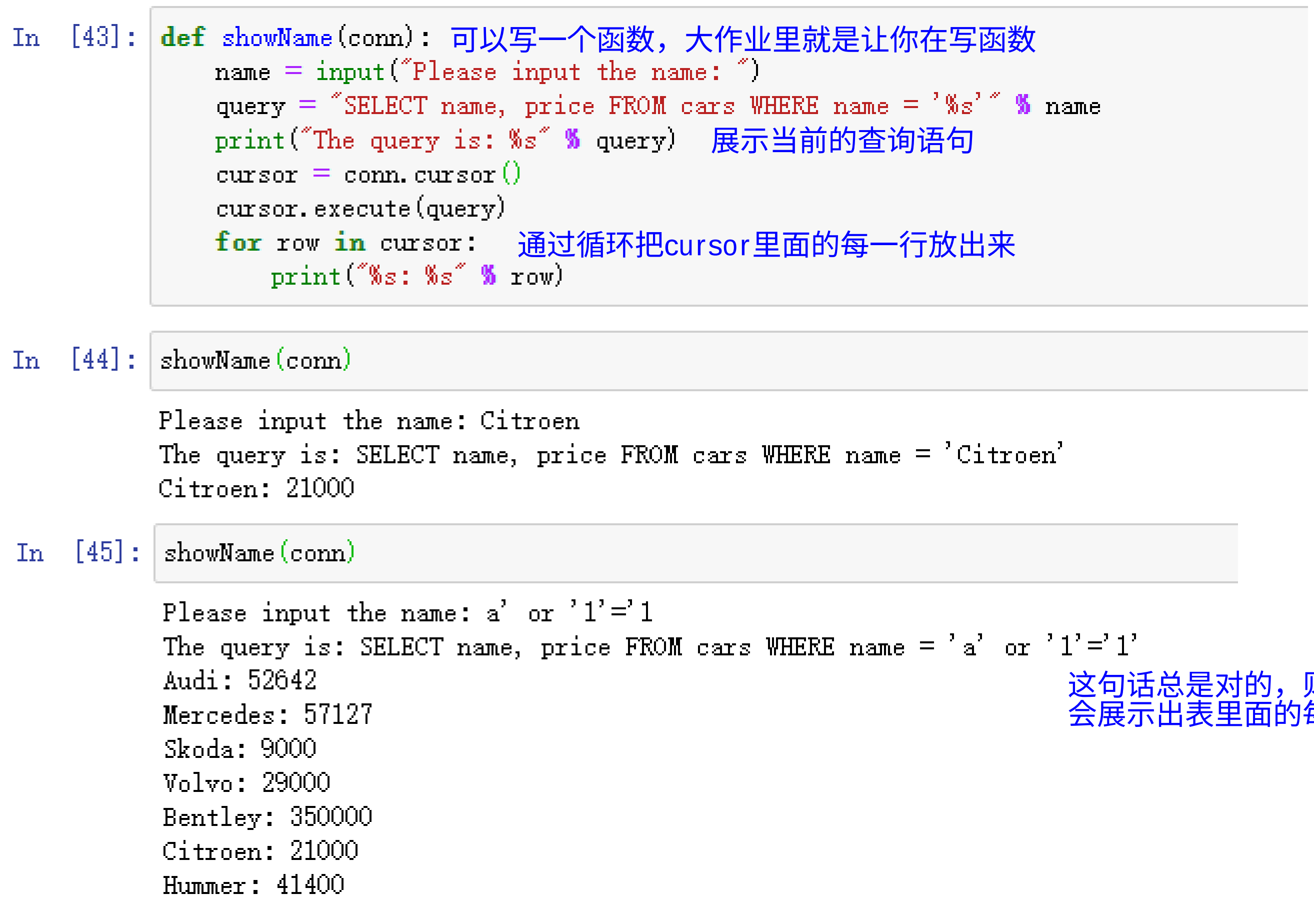
  - Querying data with **variables**
    - Variables are specified either with positional (%s) or named (%(name)s)



```
In  [40]: name_reg = "%en%"
          cursor.execute("SELECT name, price FROM cars WHERE name LIKE %s", (name_reg, ))
          for row in cursor:
              print("%s: %s" % row)
```

Python Modules

- psycopg2
  - Always use **variables**

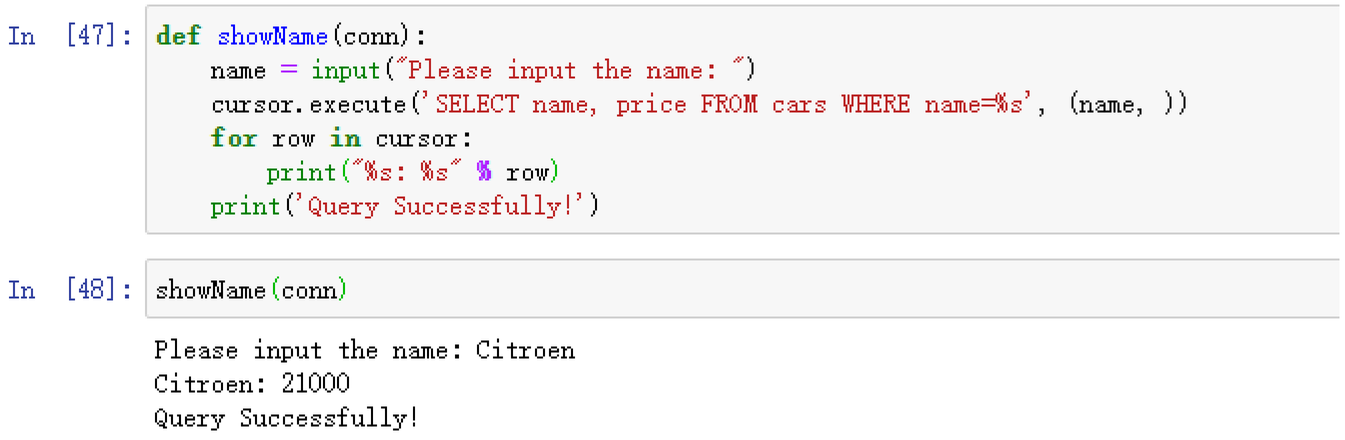


```
In [43]: def showName(conn):  可以写一个函数，大作业里就是让你在写函数
             name = input("Please input the name: ")
             query = "SELECT name, price FROM cars WHERE name = '%s'" % name
             print("The query is: %s" % query)  展示当前的查询语句
             cursor = conn.cursor()
             cursor.execute(query)
             for row in cursor:  通过循环把cursor里面的每一行放出来
                 print("%s: %s" % row)

In [44]: showName(conn)

         Please input the name: Citroen
         The query is: SELECT name, price FROM cars WHERE name = 'Citroen'
         Citroen: 21000

In [45]: showName(conn)

         Please input the name: a' or '1'='1
         The query is: SELECT name, price FROM cars WHERE name = 'a' or '1'='1'
         Audi: 52642
         Mercedes: 57127
         Skoda: 9000
         Volvo: 29000
         Bentley: 350000
         Citroen: 21000
         Hummer: 41400
```

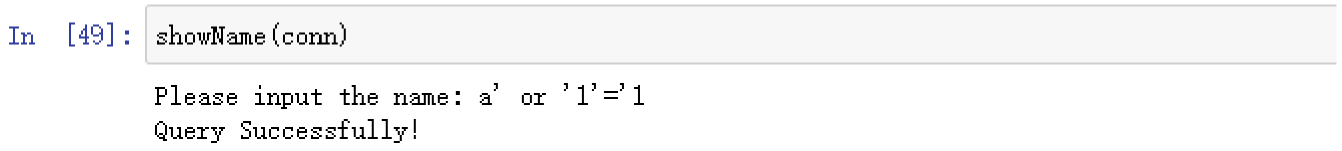
这句话总是对的，则WHERE总是为真
会展示出表里面的每一个车

Python Modules

- psycopg2
  - Always use **variables**



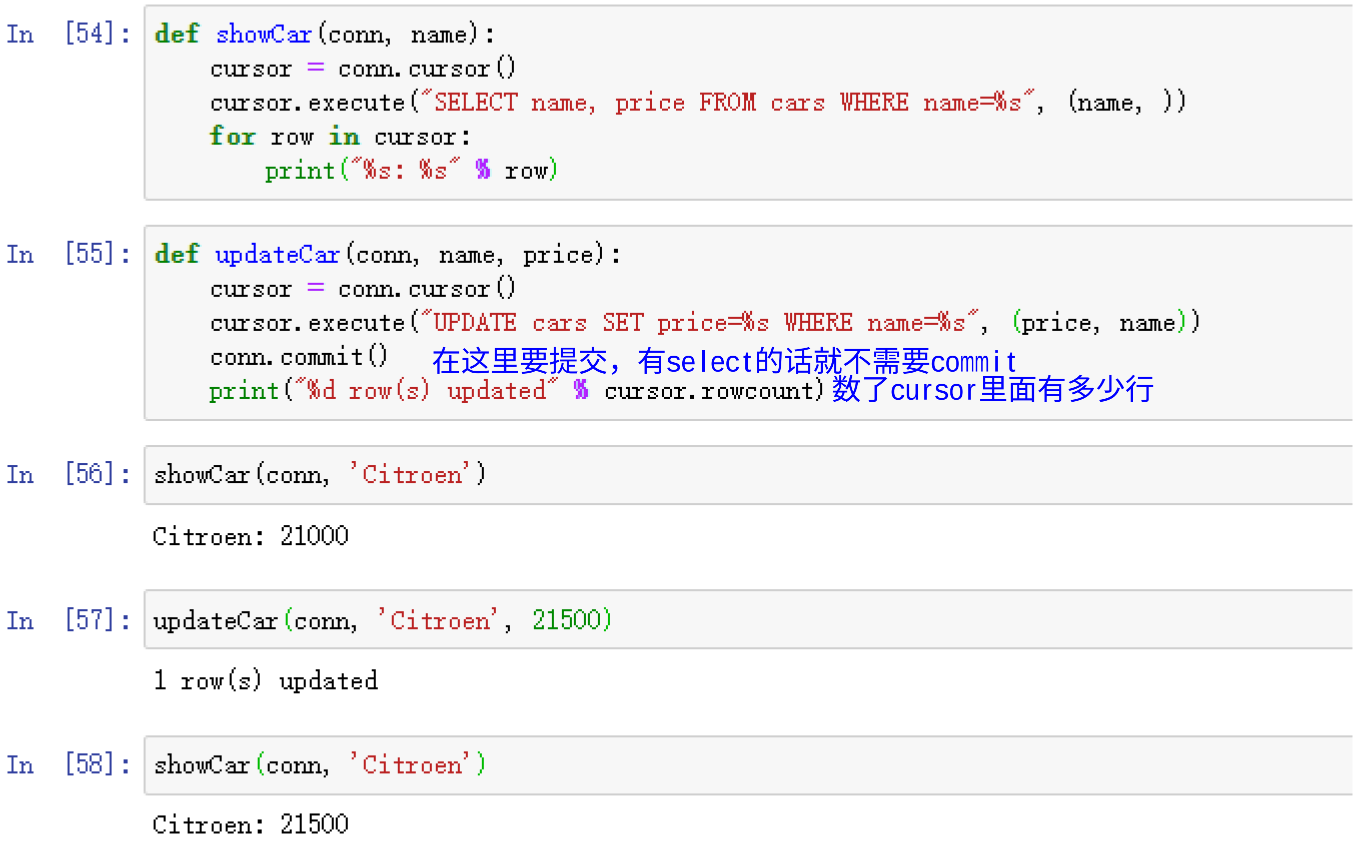
```
In [47]: def showName(conn):
             name = input("Please input the name: ")
             cursor.execute('SELECT name, price FROM cars WHERE name=%s', (name, ))
             for row in cursor:
                 print("%s: %s" % row)
             print('Query Successfully!')

In [48]: showName(conn)

         Please input the name: Citroen
         Citroen: 21000
         Query Successfully!
```



```
In [49]: showName(conn)

         Please input the name: a' or '1'='1
         Query Successfully!
```

Python Modules

- psycopg2
  - Update data



```
In [54]: def showCar(conn, name):
             cursor = conn.cursor()
             cursor.execute("SELECT name, price FROM cars WHERE name=%s", (name, ))
             for row in cursor:
                 print("%s: %s" % row)
```

```
In [55]: def updateCar(conn, name, price):
             cursor = conn.cursor()
             cursor.execute("UPDATE cars SET price=%s WHERE name=%s", (price, name))
             conn.commit()    在这里要提交，有select的话就不需要commit
             print("%d row(s) updated" % cursor.rowcount) 数了cursor里面有多少行
```

```
In [56]: showCar(conn, 'Citroen')

         Citroen: 21000
```

```
In [57]: updateCar(conn, 'Citroen', 21500)

         1 row(s) updated
```

```
In [58]: showCar(conn, 'Citroen')

         Citroen: 21500
```

Python Modules

- psycopg2Tutorial.py

```
import psycopg2
from psycopg2 import pool


def get_database_info():
    try:
        conn = psycopg2.connect(
            user='postgres',
            password='my_password',
            host='localhost',
            database='postgres'
        )
        cursor = conn.cursor()
        query = "SELECT user, current_database(), version()"
        cursor.execute(query)
        print('[user] %s\n[database] %s\n[version] %s' % cursor.fetchone())
    except psycopg2.Error as err:
        print(err)
    else:
        conn.close()
```

Python Modules

- psycopg2Tutorial.py (cont.)

```
def list_cars():
    try:
        config = {
            'user': 'postgres',
            'password': 'my_password',
            'host': '127.0.0.1',
            'database': 'postgres'
        }
        conn = psycopg2.connect(**config)
        cursor = conn.cursor()
        query = 'SELECT name, price FROM cars ORDER BY price'
        cursor.execute(query)
        for row in cursor:
            print('%s: %s' % row)
    except psycopg2.Error as err:
        print(err)
    else:
        conn.close()
```

Python Modules

- psycopg2Tutorial.py (cont.)

初始化一些连接

```
def init_conn_pool():
    config = {
        'user': 'postgres',
        'password': 'my_password',
        'host': '127.0.0.1',
        'database': 'postgres'
    }
    conn_pool = pool.SimpleConnectionPool(minconn=2, maxconn=5, **config)
    return conn_pool


def show_car(conn_pool, name):
    try:
        conn = conn_pool.getconn()
        cursor = conn.cursor()
        query = 'SELECT name, price FROM cars WHERE name=%s'
        cursor.execute(query, (name,))
        for row in cursor:
            print('%s: %s' % row)
    except psycopg2.Error as err:
        print(err)
    else:
        conn.close()
```

Python Modules

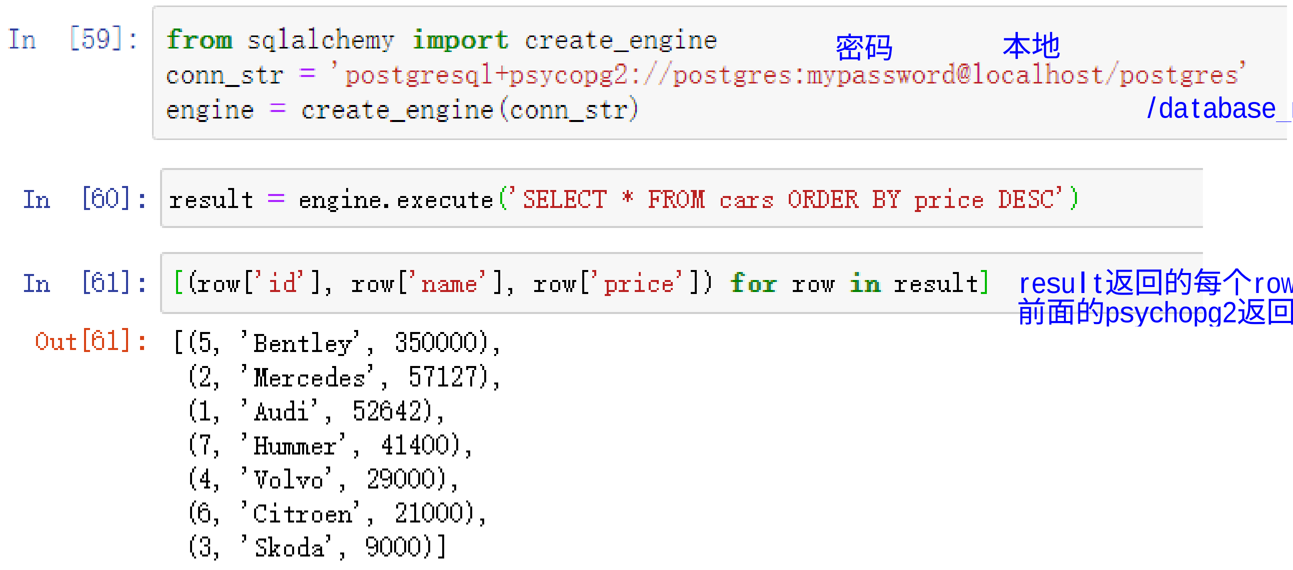
- psycopg2Tutorial.py (cont.)

```
def update_car(conn_pool, name, price):
    try:
        conn = conn_pool.getconn()
        cursor = conn.cursor()
        query = 'UPDATE cars SET price=%s WHERE name=%s'
        cursor.execute(query, (price, name))
        print('%s row(s) updated' % cursor.rowcount)
        conn.commit()
    except psycopg2.Error as err:
        print(err)
    else:
        conn.close()


def main():
    get_database_info()
    list_cars()
    conn_poll = init_conn_pool()
    show_car(conn_poll, 'Citroen')
    update_car(conn_poll, 'Citroen', 21000)
    show_car(conn_poll, 'Citroen')


if __name__ == '__main__':
    main()
```

Python Modules

介绍大作业相关的两个包的应用

- sqlalchemy
  - pip install sqlalchemy 在python里面输这一行
  - Object Relational Mapping (ORM)
  - Require a compatible postgreSQL driver when connect to postgreSQL 把数据库映射成python里的一个类



```
In [59]: from sqlalchemy import create_engine
         conn_str = 'postgresql+psycopg2://postgres:mypassword@localhost/postgres'
         engine = create_engine(conn_str)
```

密码 本地 先创建一个数据库引擎

/database_name

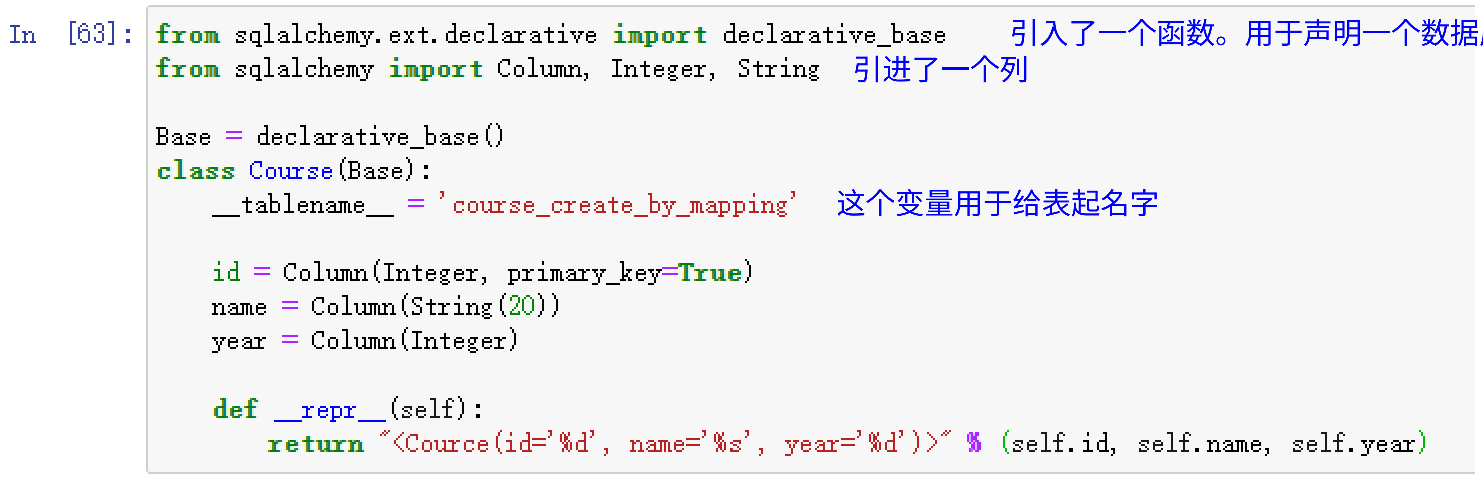
```
In [60]: result = engine.execute('SELECT * FROM cars ORDER BY price DESC')
```

```
In [61]: [(row['id'], row['name'], row['price']) for row in result]
Out[61]: [(5, 'Bentley', 350000),
          (2, 'Mercedes', 57127),
          (1, 'Audi', 52642),
          (7, 'Hummer', 41400),
          (4, 'Volvo', 29000),
          (6, 'Citroen', 21000),
          (3, 'Skoda', 9000)]
```

result返回的每个row都是一个字典！
前面的psychopg2返回的是一个元组

Python Modules

- sqlalchemy
  - Declare a mapping 定义一个映射



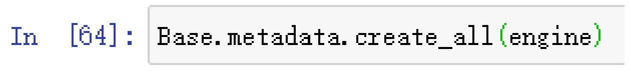
```
In [63]: from sqlalchemy.ext.declarative import declarative_base
         from sqlalchemy import Column, Integer, String

         Base = declarative_base()
         class Course(Base):
             __tablename__ = 'course_create_by_mapping'

             id = Column(Integer, primary_key=True)
             name = Column(String(20))
             year = Column(Integer)

             def __repr__(self):
                 return "<Cource(id='%d', name='%s', year='%d')>" % (self.id, self.name, self.year)
```

引入了一个函数。用于声明一个数据库

引进了一个列

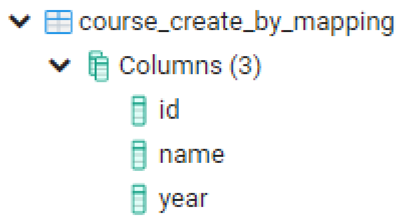
这个变量用于给表起名字

  - Create a schema in database



```
In [64]: Base.metadata.create_all(engine)
```

用这个函数就可以在pg里面建立一个表



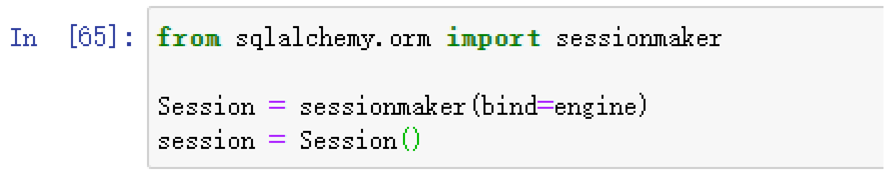
course_create_by_mapping
Columns (3)
id
name
year

重启pgadmin后生效

Python Modules

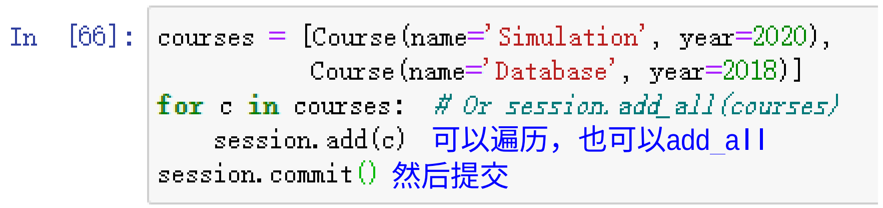
- sqlalchemy
  - Creating a session 怎么去用这个表呢？-> 建立一个session

session建立与数据库的对话conversation



```
In  [65]: from sqlalchemy.orm import sessionmaker

          Session = sessionmaker(bind=engine)
          session = Session()
```

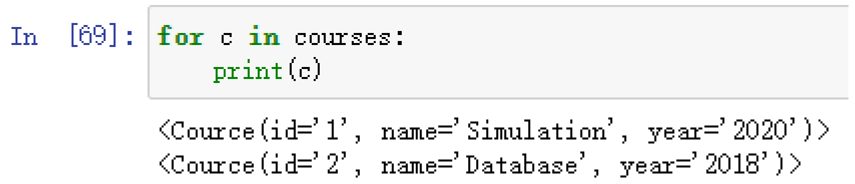
  - Adding & updating objects



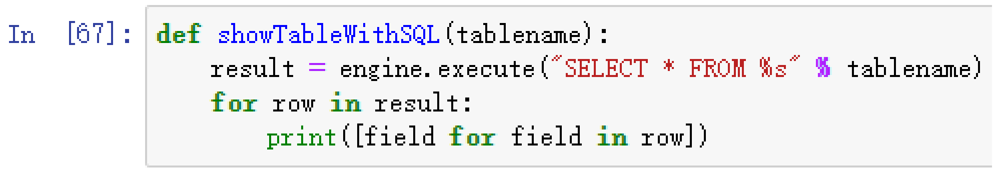
```
In  [66]: courses = [Course(name='Simulation', year=2020),
                     Course(name='Database', year=2018)]
          for c in courses:  # Or session.add_all(courses)
              session.add(c)
          session.commit()
```

可以遍历，也可以add_all

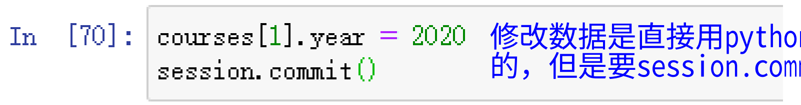
然后提交



```
In  [69]: for c in courses:
              print(c)

          <Cource(id='1', name='Simulation', year='2020')>
          <Cource(id='2', name='Database', year='2018')>
```

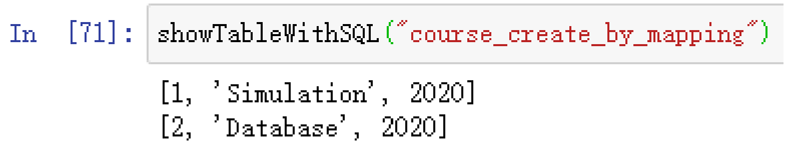


```
In  [67]: def showTableWithSQL(tablename):
              result = engine.execute("SELECT * FROM %s" % tablename)
              for row in result:
                  print([field for field in row])
```

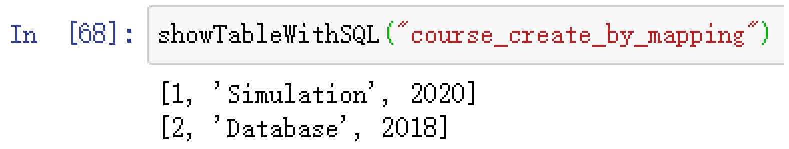


```
In  [70]: courses[1].year = 2020
          session.commit()
```

修改数据是直接用python写的，但是要session.commit



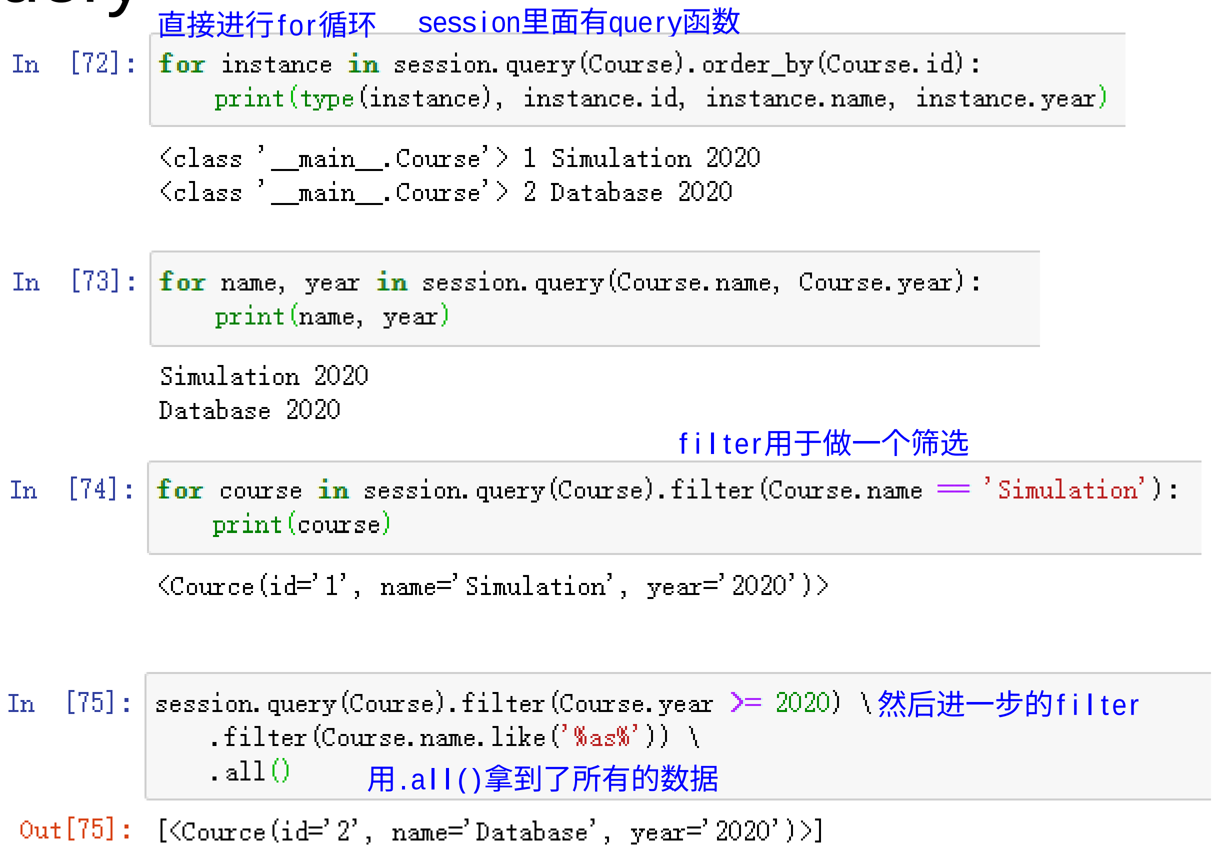
```
In  [71]: showTableWithSQL("course_create_by_mapping")

          [1, 'Simulation', 2020]
          [2, 'Database', 2020]
```



```
In  [68]: showTableWithSQL("course_create_by_mapping")

          [1, 'Simulation', 2020]
          [2, 'Database', 2018]
```

Python Modules

- sqlalchemy
  - Query



直接进行for循环　session里面有query函数

```
In  [72]: for instance in session.query(Course).order_by(Course.id):
              print(type(instance), instance.id, instance.name, instance.year)

          <class '__main__.Course'> 1 Simulation 2020
          <class '__main__.Course'> 2 Database 2020

In  [73]: for name, year in session.query(Course.name, Course.year):
              print(name, year)

          Simulation 2020
          Database 2020
```

filter用于做一个筛选

```
In  [74]: for course in session.query(Course).filter(Course.name == 'Simulation'):
              print(course)

          <Cource(id='1', name='Simulation', year='2020')>

In  [75]: session.query(Course).filter(Course.year >= 2020) \
              .filter(Course.name.like('%as%')) \
              .all()

Out[75]: [<Cource(id='2', name='Database', year='2020')>]
```

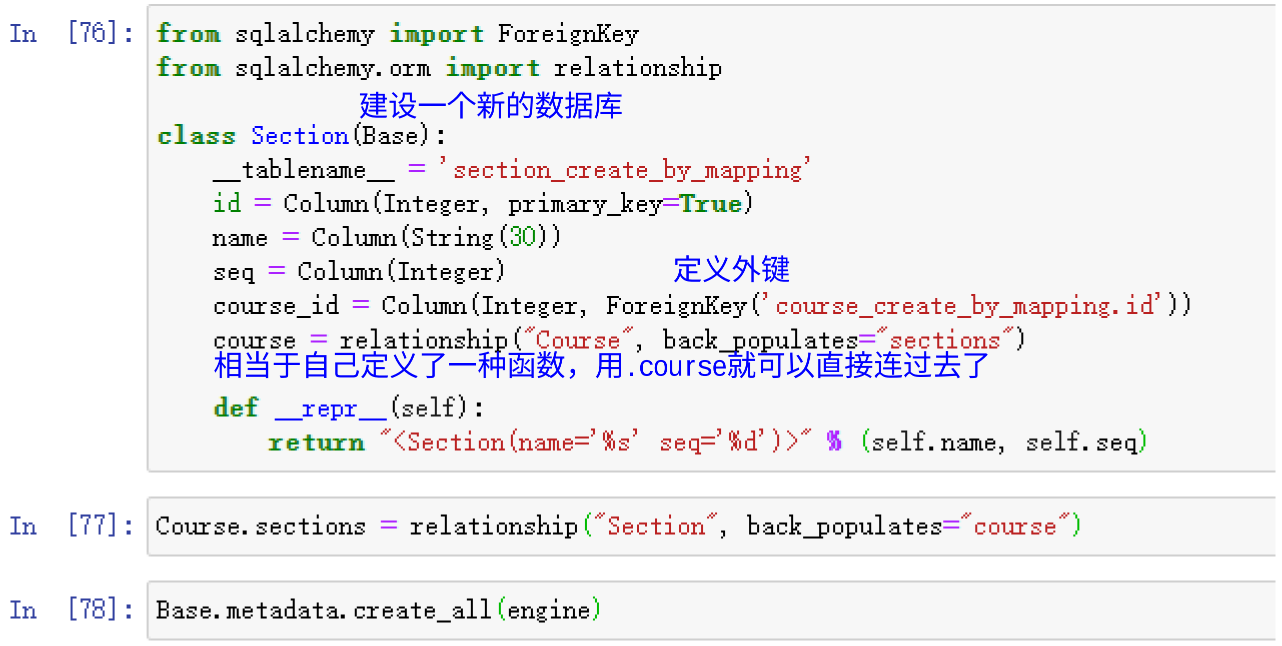
然后进一步的filter

用.all()拿到了所有的数据

Python Modules

- sqlalchemy
  - – Building a relationship

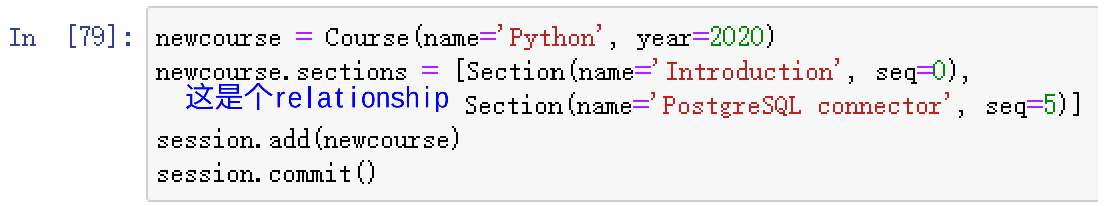
多个表之间的关系



```
In  [76]: from sqlalchemy import ForeignKey
          from sqlalchemy.orm import relationship
                         建设一个新的数据库
          class Section(Base):
              __tablename__ = 'section_create_by_mapping'
              id = Column(Integer, primary_key=True)
              name = Column(String(30))
              seq = Column(Integer)               定义外键
              course_id = Column(Integer, ForeignKey('course_create_by_mapping.id'))
              course = relationship("Course", back_populates="sections")
              相当于自己定义了一种函数，用.course就可以直接连过去了
              def __repr__(self):
                  return "<Section(name='%s' seq='%d')>" % (self.name, self.seq)

In  [77]: Course.sections = relationship("Section", back_populates="course")

In  [78]: Base.metadata.create_all(engine)
```

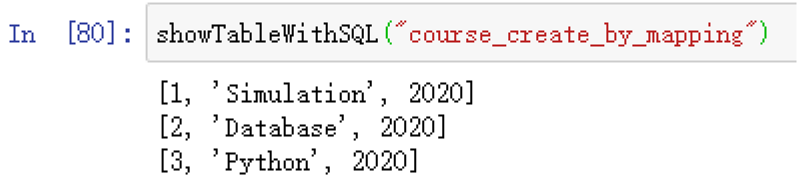
Python Modules

- sqlalchemy
  – Working with related objects

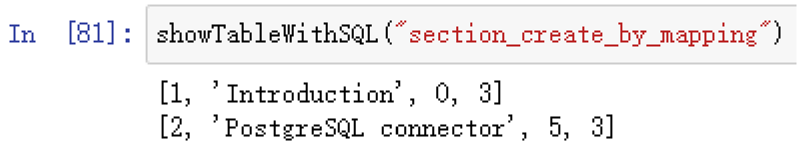


```
In  [79]: newcourse = Course(name='Python', year=2020)
          newcourse.sections = [Section(name='Introduction', seq=0),
                                Section(name='PostgreSQL connector', seq=5)]
          session.add(newcourse)
          session.commit()
```

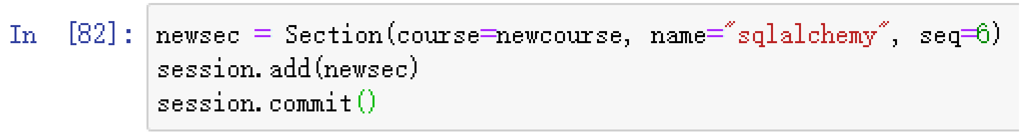
这是个relationship



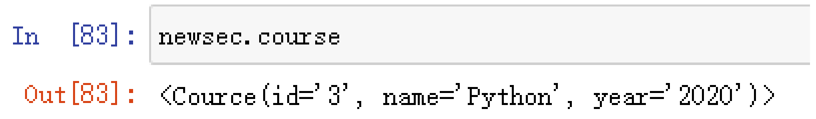
```
In  [80]: showTableWithSQL("course_create_by_mapping")

          [1, 'Simulation', 2020]
          [2, 'Database', 2020]
          [3, 'Python', 2020]
```



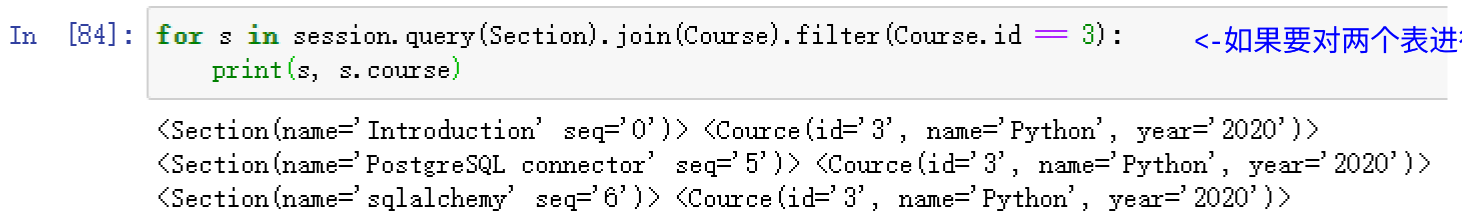
```
In  [81]: showTableWithSQL("section_create_by_mapping")

          [1, 'Introduction', 0, 3]
          [2, 'PostgreSQL connector', 5, 3]
```



```
In  [82]: newsec = Section(course=newcourse, name="sqlalchemy", seq=6)
          session.add(newsec)
          session.commit()
```



```
In  [83]: newsec.course

Out[83]: <Cource(id='3', name='Python', year='2020')>
```



```
In  [84]: for s in session.query(Section).join(Course).filter(Course.id == 3):
              print(s, s.course)

          <Section(name='Introduction' seq='0')> <Cource(id='3', name='Python', year='2020')>
          <Section(name='PostgreSQL connector' seq='5')> <Cource(id='3', name='Python', year='2020')>
          <Section(name='sqlalchemy' seq='6')> <Cource(id='3', name='Python', year='2020')>
```

<-如果要对两个表进行联合查询

Please refer to http://docs.sqlalchemy.org/en/latest/orm/tutorial.html for more possibilities

这是一个文档连接，可以看看

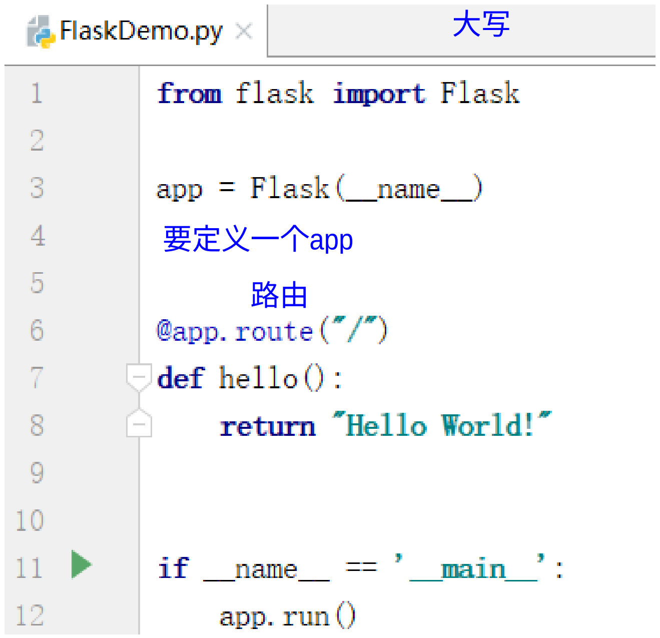
Python Modules

- Flask

大作业代码的第一个要求

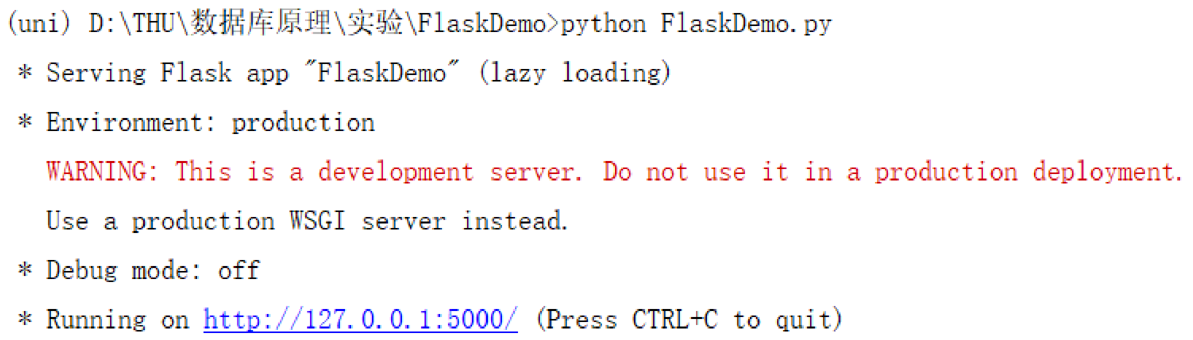
  – A microframework for Python based on Werkzeug, Jinja 2

    - pip install flask 可以用来写轻量级的网络前后端

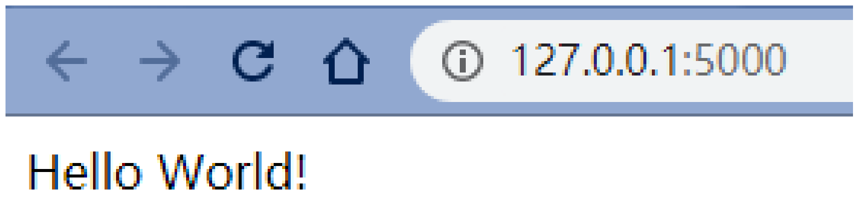


FlaskDemo.py 大写

```python
from flask import Flask

app = Flask(__name__)
要定义一个app

        路由
@app.route("/")
def hello():
    return "Hello World!"


if __name__ == '__main__':
    app.run()
```



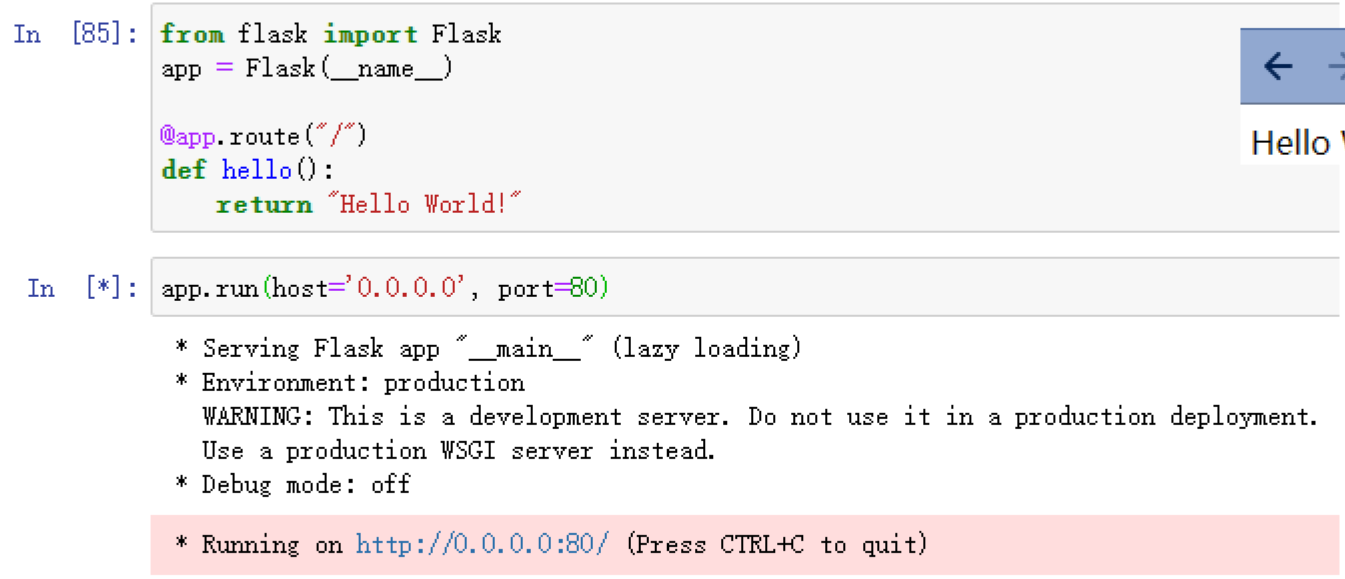
```
(uni) D:\THU\数据库原理\实验\FlaskDemo>python FlaskDemo.py
 * Serving Flask app "FlaskDemo" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



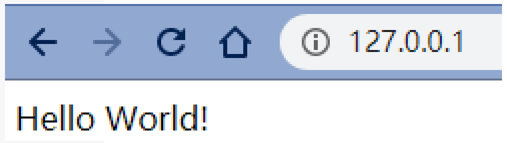
Hello World!

Python Modules

- Flask
  - A microframework for Python based on Werkzeug, Jinja 2
    - pip install flask



```
In  [85]: from flask import Flask
          app = Flask(__name__)

          @app.route("/")
          def hello():
              return "Hello World!"

In  [*]:  app.run(host='0.0.0.0', port=80)

 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
```



127.0.0.1

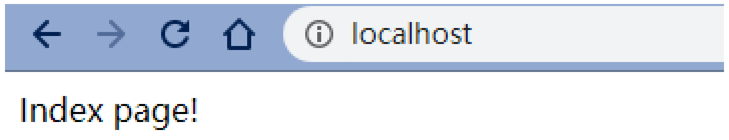
Hello World!

Python Modules

- Flask
  - Routing

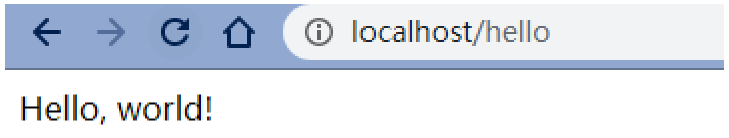
可以用这些方法来访问不同的商品

```
@app.route('/')
def index():
    return 'Index page!'
```


localhost

Index page!

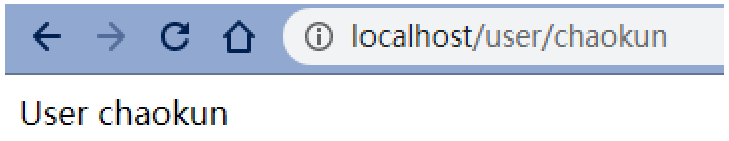
```
@app.route('/hello')
def hello():
    return 'Hello, world!'
```


localhost/hello

Hello, world!

用<>是可以捕获里面的username

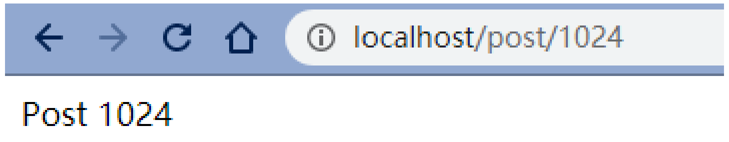
```
@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return 'User %s' % username
```


localhost/user/chaokun

User chaokun

还可以指定要捕获的类型　<类型:>

```
@app.route('/post/<int:post_id>')
def show_post(post_id):
    """show the post with the given id,
    the id is an integer"""
    return 'Post %d' % post_id
```


localhost/post/1024

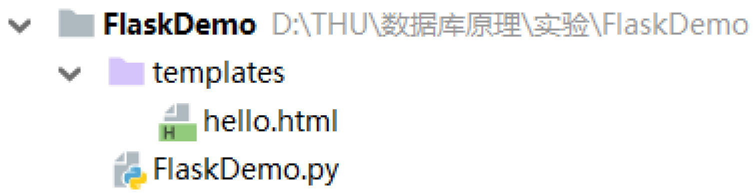
Post 1024

Python Modules

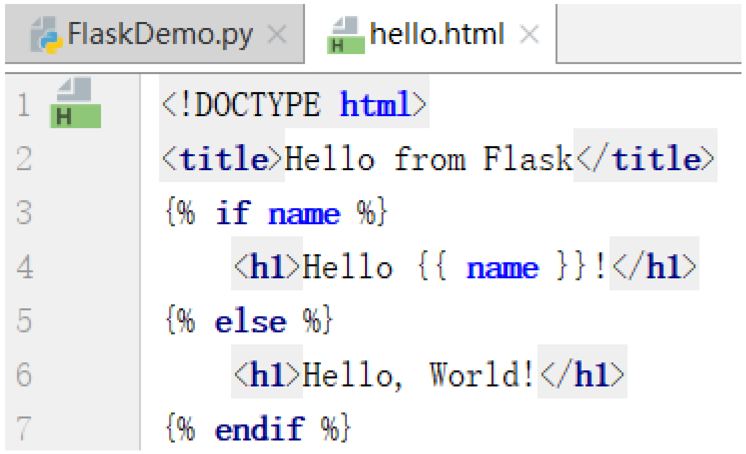
- Flask
  - Rendering templates

```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', name=name)
```

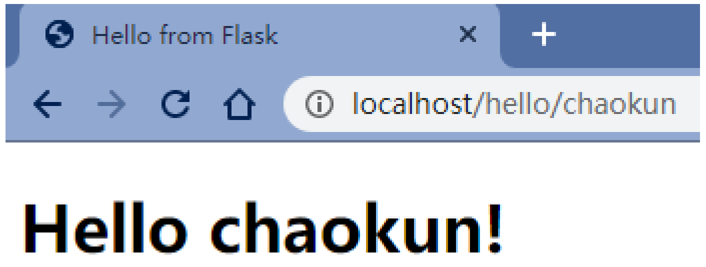
还可以写一些模板



```
FlaskDemo D:\THU\数据库原理\实验\FlaskDemo
  templates
    hello.html
  FlaskDemo.py
```



```
<!DOCTYPE html>
<title>Hello from Flask</title>
{% if name %}
    <h1>Hello {{ name }}!</h1>
{% else %}
    <h1>Hello, World!</h1>
{% endif %}
```



Hello chaokun!

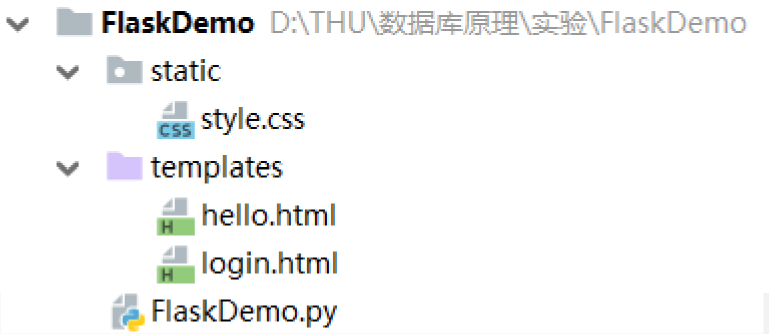
Python Modules

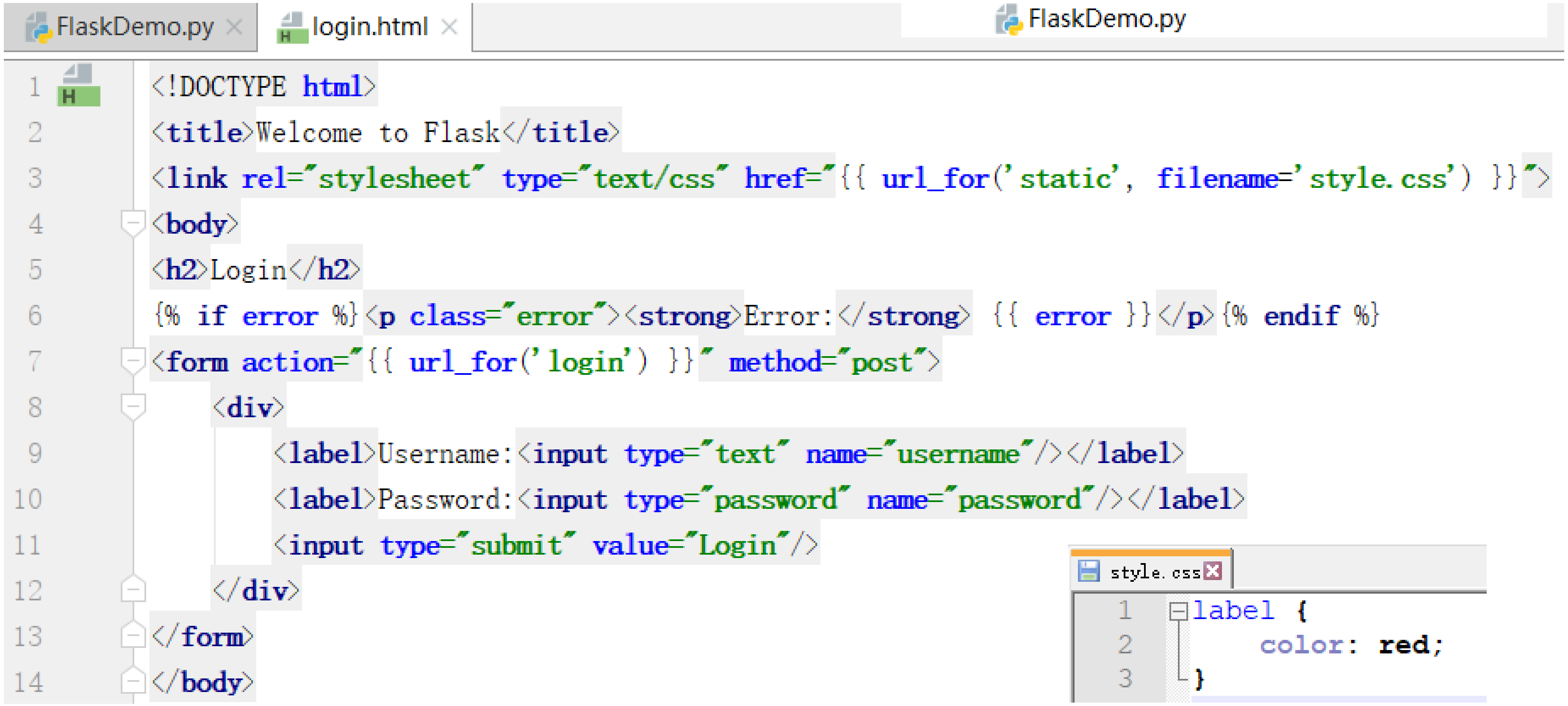
- Flask
  - Handling requests

```
from flask import request, session, url_for, redirect
app.config['SECRET_KEY'] = 'Valar Morghulis'

@app.route('/login', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        if request.form['username'] != "myuser":
            error = 'Invalid username'
        elif request.form['password'] != "mypassword":
            error = 'Invalid password'
        else:
            session['logged_in'] = True
            session['username'] = request.form['username']
            return redirect(url_for('hello') + session.get('username'))
    return render_template('login.html', error=error)
```

Python Modules

- Flask
  - Handling requests (cont.)


```
FlaskDemo D:\THU\数据库原理\实验\FlaskDemo
  static
    style.css
  templates
    hello.html
    login.html
  FlaskDemo.py
```

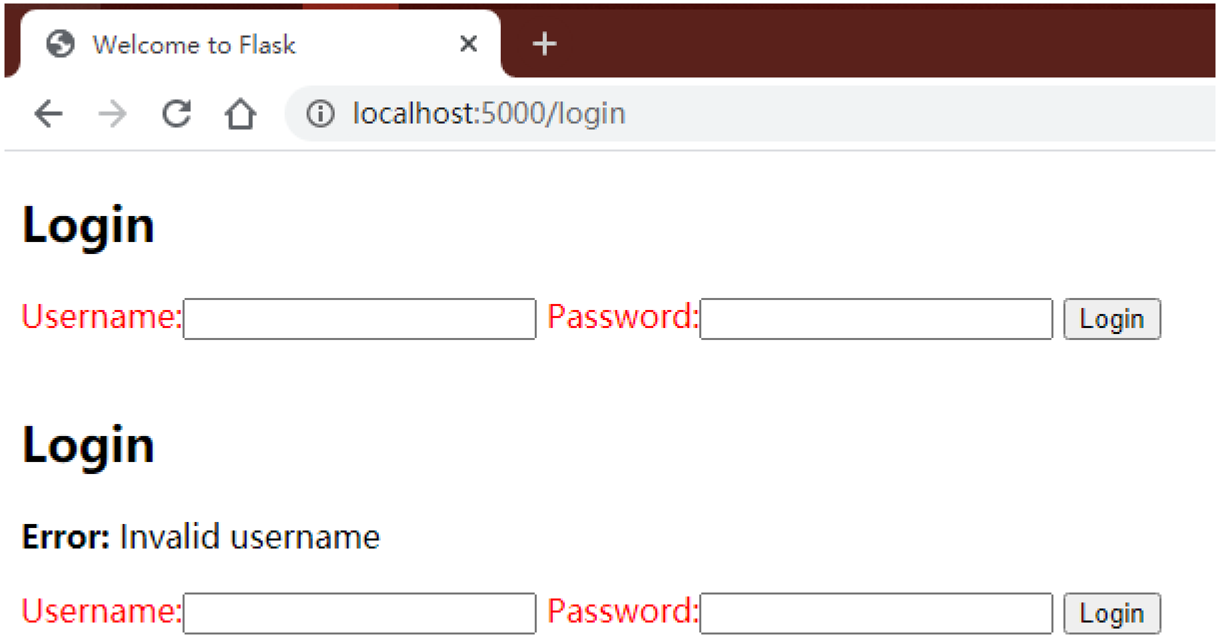


```
<!DOCTYPE html>
<title>Welcome to Flask</title>
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
<body>
<h2>Login</h2>
{% if error %}<p class="error"><strong>Error:</strong> {{ error }}</p>{% endif %}
<form action="{{ url_for('login') }}" method="post">
    <div>
        <label>Username:<input type="text" name="username"/></label>
        <label>Password:<input type="password" name="password"/></label>
        <input type="submit" value="Login"/>
    </div>
</form>
</body>
```

```
label {
    color: red;
}
```

Python Modules

- Flask
  - Handling requests (cont.)



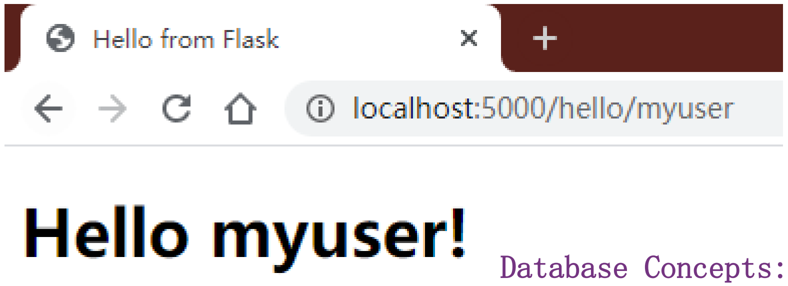
Login

Username: Password: Login

Login

Error: Invalid username

Username: Password: Login



Hello myuser!

Python Modules

- Flask
  - Resources
    - Quick start: http://flask.pocoo.org/docs/1.1/quickstart/
    - Template: http://jinja.pocoo.org/docs/2.11/templates/
    - HTML: https://www.w3schools.com/html
    - Web development tool: Microsoft Expression Web (https://www.microsoft.com/en-us/download/details.aspx?id=36179)