COMP3311 22T1 Database Systems

Week 08
Python/Psycopg2

1. What is the difference between a *connection* and a *cursor* in Psycopg2? How do you create each?

   Answer:

   A *connection* provides authenticated access to a database. You create a *connection* as follows

   ```
   connection = psycopg2.connect(ConnectionParameters)
   ```

   Typical connection parameters are: `dbname` or `database` (required), `user` and `password` for authentication, and `host` and `port` to specify the "location" of the PostgreSQL server.

   Some examples:

   ```
   conn = psycopg2.connect("dbname=mydb")
   conn = psycopg2.connect(database="mydb")
   conn = psycopg2.connect("user=jas password=abc123 dbname=
   conn = psycopg2.connect("host=db.server.org port=5432 dbn
   ```

   A *cursor* provides a pipeline between the Python program and the PostgreSQL database. You create a *cursor* as follows:

   ```
   cursor = connection.cursor()
   ```

   Cursors can be used to send queries to the database and read back rsults as in:

   ```
   cursor.execute("SQL Query")
   results = cursor.fetchall()
   ```

   where `results` is a list of tuples.

2. [Question courtesy of Clifford Sesel] The following Python script (in a executable file called `opendb`) aims to open a connection to a database whose name is specified on the command line:

   ```
   #!/usr/bin/python3
   import sys
   import psycopg2
   if len(sys.argv) < 2:
       print("Usage: opendb DBname")
       exit(1)
   db = sys.argv[1]
   try:
       conn = psycopg2.connect(f"dbname={db}")
       print(conn)
       cur = conn.cursor()
   except psycopg2.Error as err:
       print("database error: ", err)
   finally:
   ```

```
15.     if conn is not None:
16.         conn.close()
17.     print("finished with the database")
18.
```

When invoked with an existing database, it behaves as follows

```
$ ./opendb beers2
<connection object at 0x7fac401799f0; dsn: 'dbname=beers2
finished with the database
```

but when invoked with a non-existent database it produces

```
$ ./opendb nonexistent
database error:  FATAL:  database "nonexistent" does not

Traceback (most recent call last):
  File "./opendb", line 16, in
    if conn :
NameError: name 'conn' is not defined
```

rather than

```
$ ./opendb nonexistent
database error:  FATAL:  database "nonexistent" does not

finished with the database
```

What is the problem? And how can we fix it?

Answer:

The scope of the `conn`object does not extend into the `finally` clause if `conn` is only initialised in the `try` clause. This is essentially what the error message is telling you:

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

Ask the Python developers why variable/object scope works this way.

To fix the problem, you need to initialise `conn` in the outer scope. You can't simply move the `connect()` call there, because it would then be outside the `try` clause and the exception would be handled by the standard Python exception handler rather than our exception handler, e.g.

```
Traceback (most recent call last):
  File "./opendb", line 9, in
    conn = psycopg2.connect(f"dbname={db}")
  File "/Library/Frameworks/Python.framework/Versions/3.7
    conn = _connect(dsn, connection_factory=connection_fa
psycopg2.OperationalError: FATAL:  database "xyzzyyyy" do
```

The solution is to initialise `conn` outside the `try` clause:

```
db = sys.argv[1]
conn = None
try:
   ...
```

3. Using the `mymyunsw` database (`/home/cs3311/web/22T1/tutes/week08/mymyunsw.dump`), write a Python script called `course-roll` that takes two command-line arguments (subject code and term code) and produces a list of students enrolled in that course. Each line should contain: studentID, name. The name should appear as "familyName, givenNames" and the list should be ordered by familyName then by givenName. e.g.

   Some examples of use:

```
$ ./course-roll COMP1521
Usage: course-roll subject term

$ ./course-roll COMP1511 16s1
COMP1511 16s1
No students

$ ./course-roll COMP1151 17s1
COMP1151 17s1
No students

$ ./course-roll PSYC1022 19T0
PSYC1022 19T0
5154325 Georgas, Jose
5141408 Hordern, Priyanshi
5134420 Hulst, Stan
5159982 Madanimelak, Siavash
5163692 Watson, Carine

$ ./course-roll COMP3211 20T1
COMP3211 20T1
5167571 Ahmad Bakir, Nabilah
5160470 Bashir, Zeeshan
5132438 Bown, Yelgun
5147435 Chen, Manjiang
5137415 Hang, Zheren
5134504 Hiwilla, Libin
5182987 Krueger, Isabel
5149728 Lim, Sang
5160734 Maslin, Jocelyn
5160049 McTigue, Courtney
5159809 Murdani, Elsan
5171168 Nadol, Rowan
5169891 Niu, Zihui
5166832 Pong, Chi-Kuang
5161703 Schachat, Kalie
5215073 Shang, Chan
```

```
5173714 Strickland, Vanya
5129717 Styer, Sharon
5208366 Suo, Junjian
5178112 Tandan, Ray
5164849 Thattai, Kajalben
5168959 Virdi, Jawad
```

Answer:

Possible script for `course-roll`:

```python
#!/usr/bin/python3
import sys
import psycopg2
if len(sys.argv) < 3:
    print("Usage:",sys.argv[0],"subject term")
    exit(1)
subject = sys.argv[1]
term = sys.argv[2]
conn = None
query = """
select p.id, p.family, p.given
from   Subjects s
       join Courses c on (c.subject=s.id)
       join Terms t on (c.term=t.id)
       join Course_enrolments e on (e.course=c.id)
       join People p on (e.student=p.id)
where  s.code = %s and t.code = %s
order  by p.family, p.given
"""
try:
   conn = psycopg2.connect("dbname=mymyunsw")
   cur = conn.cursor()
   cur.execute(query, [subject,term])
   students = cur.fetchall()
   print(subject,term)
   if len(students) == 0:
      print("No students")
      exit(0)
   for stu in students:
      print(stu[0],stu[1]+",",stu[2])
except psycopg2.Error as err:
   print("database error: ", err)
finally:
   if conn :
      conn.close()


```

4. The script in the previous question was a little bit sloppy in its error detection, and the messages did not distinguish between a real course with no students and a bad value for either the subject code or the term code, or a valid pair of codes for which there was no course offering. The script should also now print the long name of the subject. Improve the script so it behaves as follows:

```
$ ./course-roll1 COMP1151 18s1
Invalid subject COMP1151

$ ./course-roll1 COMP1511 88xx
```

```
Invalid term 88xx

$ ./course-roll1 COMP3211 20T2
No offering: COMP3211 20T2

$ ./course-roll1 MATH2601 20T2
MATH2601 20T2 Higher Linear Algebra
5194739 Boddam-Whetham, Billi
5160543 Chao, Sixiang
5188339 Ghebrial, Christina
5220431 Lopez Castro, Feyza
5196254 McWilliam, Lianne

$ ./course-roll1 COMP3821 17s1
COMP3821 17s1 Extended Algorithms and Programming Technic
5143563 Aurik, Rumana
5128387 Barsoum, Rona
5128243 Bartlett, Gregg
5133001 Black, Samim
5126171 Demiri, Florida
5128296 Donney, Lisita
5133189 Essam, Swisszanah
5135130 Ghislain, Thien-Ngan
5143373 Howard, Alanna
5141297 MAI, Tiancong
5132763 MAI, Xueqi
5143439 Manolios Reichert, Janette
5141860 Matsuo, Ryann
5125264 Mock, Na-Ima
5128275 Mohd Hassan, Xiao
5147571 Morsalin, Amir
5129681 Neo, Fenzhi
5128164 Park, Toshiki
5142705 Rabelo Castello, Tamara
5142618 Saldanha, Sai
5143532 Sethuramasamy, Tanvi
5141585 Twine, Hughie
5142688 Wenzel, Sohaib
5145043 Wijeratne, Joyce
5127997 Zhen, Kaijian
```

Answer:

Possible script for `course-roll1`:

```python
#!/usr/bin/python3
import sys
import psycopg2
if len(sys.argv) < 3:
    print("Usage:",sys.argv[0],"subject term")
    exit(1)
subject = sys.argv[1]
term = sys.argv[2]
conn = None
subjCheck = "select id,longname from Subjects where
```

```
termCheck = "select id from Terms where code = %s"
offerCheck = "select * from Courses where subject = 
ok = True
query = """
select p.id, p.family, p.given
from   Courses c
       join Terms t on (c.term=t.id)
       join Course_enrolments e on (e.course=c.id)
       join People p on (e.student=p.id)
where  c.subject = %s and t.id = %s
order  by p.family, p.given
"""
try:
   conn = psycopg2.connect("dbname=mymyunsw")
   cur = conn.cursor()
   cur.execute(subjCheck, [subject]);
   res = cur.fetchone()
   if res is None:
      print("Invalid subject",subject)
      ok = False
      subjID = None
   else:
      subjID, subjName = res
   cur.execute(termCheck, [term]);
   res = cur.fetchone()
   if res is None:
      print("Invalid term",term)
      ok = False
      termID = None
   else:
      termID = res[0]
   if subjID is not None and termID is not None:
      cur.execute(offerCheck, [subjID,termID])
      res = cur.fetchone()
      if res is None:
         print("No offering:",subject,term)
         ok = False
   if ok == False:
      exit(1)
   cur.execute(query, [subjID,termID])
   students = cur.fetchall()
   print(subject,term,subjName)
   if len(students) == 0:
      print("No students")
      exit(0)
   for stu in students:
      print(stu[0],stu[1]+",",stu[2])
except psycopg2.Error as err:
   print("database error: ", err)
finally:
   if conn :
      conn.close()

```

5. Write a script that prints the subjects that a specified student is studying in a specified term. No need to check whether the student was enrolled in the given term; simply print that they studied no courses. Examples of use:

```
$ ./courses-studied
Usage: ./courses-studied sudentID term

$ ./courses-studied 1234567 18s1
No such student
```

```
$ ./courses-studied 9300035 20T1
No such student

$ ./courses-studied 5137295 17s2
COMP1521 Computer Systems Fundamentals
COMP1531 Software Eng Fundamentals
MATH1231 Mathematics 1B

$ ./courses-studied 5137295 20T1
COMP3231 Operating Systems
SENG3011 Software Eng Workshop 3
```

Answer:

Possible script for `courses-studied`:

```python
#!/usr/bin/python3
import sys
import psycopg2
if len(sys.argv) < 3:
    print("Usage:",sys.argv[0],"sudentID term")
    exit(1)
student = sys.argv[1]
stuName = None
term = sys.argv[2]
conn = None
stuqry = """
select p.fullname
from   People p join Students s on p.id=s.id
where  p.id = %s
"""
query = """
select s.code, s.name
from   Subjects s
       join Courses c on (c.subject=s.id)
       join Terms t on (c.term=t.id)
       join Course_enrolments e on (e.course=c.id)
       join People p on (e.student=p.id)
where  p.id = %s and t.code = %s
order  by s.code
"""
try:
   conn = psycopg2.connect("dbname=mymyunsw")
   cur = conn.cursor()
   cur.execute(stuqry,[student])
   stuName = cur.fetchone()
   if not stuName:
      print("No such student")
      exit(0)
   cur.execute(query, [student,term])
   courses = cur.fetchall()
   if len(courses) == 0:
      print("No courses")
      exit(0)
   for subj in courses:
      print(subj[0],subj[1])
except psycopg2.Error as err:
   print("database error: ", err)
finally:
   if conn :
      conn.close()
```

```
46.
```