1ST Class – Tuesday evening, January 6, 2015

Instructor: Chris Oson – [coson@coleman.edu](mailto:coson@coleman.edu)

Programming Assignment 1 (15 points) COM410-08427-01.py

Programming Assignment 2 (20 points) COM410-08427-02.py

Programming Assignment 3 (15 points) COM410-08427-03.py

Midterm (25 points)

Final (25 points)

100 Points, No quizzes, no reviews.

Submitting assignments: Filename format: *COM410-xxxxxx-yy.py, zip*, 7z, rar, tar (make sure it compiles)

WEBCLASS:

1. Which version of Python? 2.7

2. Tools to write Python scripts On Windows or Unix, use Eclipse or Visual Studio.

(Not going to use the Python IDLE tool.)

Free IDEs:

PyScripter,

Plug-in Iron Python,

PyCharm (JetBrains).

Non-IDE – Sublime Text 2 – Outstanding support for regular expressions.

3. Overview of Assignment #1

4. Introduction to data types.

5. Introduction to strings

Greetings = ‘Hello World’

Statically-typed code: int a = 25;

Python is

1. It is tokenized,

2. Lexed/parse

3. Compile

4. Run

Python parses dynamically-typed code

It’s not checked at compile-time it’s checked at run-time.

It infers the data type.

Example data types:

Python has numbers: Int, float, complex

Integer: 4 bytes (32bits), Long: 8 bytes (64bits)

Compared to C#: int

JavaScript:

*Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32*

*Type "copyright", "credits" or "license()" for more information.*

*>>> first = 35*

*>>> second = 25*

*>>> result = first + second*

*>>> print result*

*SyntaxError: invalid syntax*

*>>> print (result)*

*60*

*>>> a = 5*

*>>> b = 5.5*

*>>> type a*

*SyntaxError: invalid syntax*

*>>> type(a)*

*<class 'int'>*

*>>> type (b)*

*<class 'float'>*

*>>> d = 3j*

*>>> type (d)*

*<class 'complex'>*

*>>> expression = 'How is your day?'*

*>>> expression = "How is your day?"*

*>>> expression = """How*

*is*

*your*

*day?"""*

*>>> expresion4 = 'Howard O'Brien'*

*SyntaxError: invalid syntax*

*>>> expression4 = "Howard O'Brien"*

*>>> myName = 'Moe' + ' ' + 'Howard'*

*>>> print (MyName)*

*Traceback (most recent call last):*

*File "<pyshell#21>", line 1, in <module>*

*print (MyName)*

*NameError: name 'MyName' is not defined*

*>>> print MyName*

*SyntaxError: invalid syntax*

*>>> formattedName = '%s %s' % ('Shemp', 'Howard')*

*>>> print*

*<built-in function print>*

*>>> print (formattedName)*

*Shemp Howard*

*>>>*

*>>> differentFormat = "{0} {1}".format('Joe', 'Bessner')*

*>>> print (differentFormat)*

*Joe Bessner*

*>>> numberExpression = "{0} +{1} = {2}".format(7, 5, 7 + 5)*

*>>> print (numberExpression)*

*7 +5 = 12*

*>>> print str(x) + ' + ' + str(5) + ' = ' + str(7 + 5)*

*SyntaxError: invalid syntax*

*>>> print (str(x) + ' + ' + str(5) + ' = ' + str(7 + 5))*

*Traceback (most recent call last):*

*File "<pyshell#32>", line 1, in <module>*

*print (str(x) + ' + ' + str(5) + ' = ' + str(7 + 5))*

*NameError: name 'x' is not defined*

*>>>*

Imaginary number ‘i’ and is a -1.

Complex number ‘j’ and is a -1.

String is unusual because there is no char!

String can be made with single quote’ ‘, double-quote “ ”, or triple quote “”” “”” (multiple lines).

String is immutable which means the compiler writes over the original value in the same memory space.

We can use a formatted string

The Alt-p will repeat the most recent command.

Format expression vs Format Placeholders

Control Panel – System Settings – Environmental Variables Edit Path variable

– Append the new path to the path environment variable string.

Install Python (Typically to C:\Python27\Python.exe)

Import os

givenPath = ‘c:\\TEMP\

print os.givenPath

Format Specifiers:

%s = string

%d = int

%f = float

%x = hex

Boolean: True, False, and None

None is the same as null in other languages.

With a Database connection, if there is no connection then it is a ‘None’.

Strings:

In Python – no char, no array.

In Python – Use ‘Slice’

Python gives great importance on ‘exclusivity’ and ‘inclusivity.’

In other languages there is a substring function with two overloads, in Python use a slice:.

A Python strange quirk: >>>exp[0:4] is not: start at offset 0 and give me 4 characters…

It means: start at offset 0 (‘inclusive’) and go until the second operation (which is ‘exclusive’ )

Another e.g.: >>> exp[-6:-2] gives ‘uesd’

Also, >>> exp[:3] defaults start at inclusive 0.

Also, >>> exp[:-3] start at inclusive -3 and defaults to exclusive 0.

Unix Shell Script: 0 = true, !0 = false

VB: --1 = true, -0 = false

C/C++: 1 = true, 0=false

None:

data= ‘This is a test’

results = ‘x’ in data

Now I want to extract a slice mySlice = data[0:3]

If mySlice == “This”:

----you need the colon to continue to the next line and it automatically indents 4 spaces.

Indentations must be 4 characters or you’ll get an “indentation error”

Use the # symbol for commenting in Python.

Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32

Type "copyright", "credits" or "license()" for more information.

>>> exp = 'This is a test, today is Tuesday'

>>> exp[0]

'T'

>>> len(exp)

32

>>> exp[len(exp) - 1]

'y'

>>> exp[-1]

'y'

>>> exp[0:4]

'This'

>>> exp[-6:-2]

'uesd'

>>> exp[:3]

'Thi'

>>> exp[-3]

'd'

>>> exp[-3:]

'day'

>>> exp[-3:4]

''

>>> piece = exp[-3:]

>>> print (piece)

day

>>> print exp[:]

SyntaxError: invalid syntax

>>> print (exp[:])

This is a test, today is Tuesday

>>> test =1

>>> test = number > 10

Traceback (most recent call last):

File "<pyshell#16>", line 1, in <module>

test = number > 10

NameError: name 'number' is not defined

>>> number = 11

>>> test = number > 10

>>> test == 1

True

>>> test == True

True

>>> test == 0

False

>>> test == false

Traceback (most recent call last):

File "<pyshell#22>", line 1, in <module>

test == false

NameError: name 'false' is not defined

>>> test2 = number < 10

>>> test == False

False

>>> print (test2)

False

>>> test2 == 0

True

>>>

data = "This is a test"

results = "x" in data

isValid =

mySlice == 'Thi'

if isValid:

print 'The pattern exists':

isValid = True

elseIf mySlice == ‘Crazy’

doSomethingElse

else:

print 'The pattern does not exist':

endIf;

#!/user/bin/python # On a Unix machine, this is where Python is installed

import OS # Import the operating system module

os.system(“cls”) # Clears the screen

.lower()

.upper() # for upper

.title() # proper text case

# Will not be allowed to use.

.capitalize # Will not be allowed to use.

Make your own version of the title class

Get the string then…

Implement the title string function in Python but you can’t use title.

xyz.center(25, ‘\*’)

\*\*\*\*\*\*\*\*\*\*\*\*xyz\*\*\*\*\*\*\*\*\*\*\*\*

Tokens = exe.split(‘ ‘)

Returns a list of tokens

Django – Python is used in Web Programming.

Use the Define method

Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit (Intel)] on win32

Type "copyright", "credits" or "license()" for more information.

>>> prog = "Python is used in web programming."

>>> print prog.find('used')

10

>>> print prog.find('exe')

-1

>>> print prog.index('used')

10

>>> print prog.index('exe')

Traceback (most recent call last):

File "<pyshell#4>", line 1, in <module>

print prog.index('exe')

ValueError: substring not found

>>>

It has the ‘index’ method but it gives you an error when not found, so we prefer to use the ‘find’ method.

To know if a substring exists we use the ‘\_contains\_’ method. But the ‘contains’ method is built-into the Python language; so we use the underscores ‘\_’

It has language specific methods instead of class specific methods.

print exp.\_contains\_(‘is’) # returns boolean True or False

print exp.startswith(‘T’) # returns boolean True or False

print exp.endswith(‘T’) # returns boolean True or False

value = str(input(“Enter something: “))

print len(value)

print value

strip command removes: whitespace of: “\t”, “\n”, “blank”, “|”, ASC(12)

.strip() at the end removes all leading and trailing whitespaces.

2ND Class – Tuesday evening, January 13, 2015

Instructor: Chris Oson – [coson@coleman.edu](mailto:coson@coleman.edu)

Python has no ‘array’ it doesn’t need it.

It can enumerate collections with the [brackets].

numbers = [7,14,21]

charList = [‘a’,’b’,’c’]

floatList = [7.47,1.2,9.8]

But other languages have the ‘array’ and use thee index. And it has a rank of 1, one dimensional.

For(int I=0, max=array.length; i<max; I++){

// do something

}

Python:

user-definedVariable’ can be ‘eachItem’

For user-definedVariable’ in list:

Print eachItem # indent by 4

print # this is on the same line, so flush the buffer

print ‘y’ # now print

- Lists

- List Slices

- Enumerate Lists

Other languages have requirements of arrays:

1) index >= 0 and

2) items are the same data type

Python list can have different data types:

genericList = [[1,2,3],1,’n’,True,None,’Hello’,7.99]

genericList[-1] # the last element

genericList[:3] # slice with starting value is the default of 0 and excludes 3

len(genericList) # 7 elements

**Range Function**

range function # must provide a starting value and an ending value:

firstTenNumbers = range(1, 11)

print firstTenNumbers

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

firstTenNumbers = range(100, 90, -1)

print firstTenNumbers

[100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90]

someValues = [‘abc’, ‘def’, ‘ghi’, ‘jkl’, ‘mno’]

for eachElement in reverse(someValues): # ‘reverse’ statement

print eachElement, # print with the comma

> mno jkl ghi def abc

Separator = ‘-‘ \* 50 # line separator

>---------------------------

#Clear the screen

Os.system(“cls”)

**Iterate an enumeration**

elementToProcess = len(someValues) – 1

loop = 0

while loop < elementToProcess:

print ‘(0) +> (1)’.format(loop, somevalues)) # print index and the value

loop == 1

print

print separator

for index, everyNumber in enumerato(someValue):

print ‘(0) +> (1)’.format(index, somevalues)) # print index and the value

print separator

‘sequence’ as opposed to ‘list’

for-in

# an inline list

for food in (“pate”, “cheese”, “crackers”, “margarine”):

if food == “yogurt”:

print “yogurt encountered…”

break

else: # counter to the for loop, not the if statement

print “There is no yogurt”

print separator

numbers = [7, 15, 22]

# want to skip any even numbers

For everyNumber in numbers:

If everyNumber % 2 == 0:

pass # there is no ‘continue’ keyword. We use ‘pass’ a do-nothing statement

else:

print everyNumber

# Treat a string like a list

lastName = [Douglass’, ‘Jefferson’, ‘Williams’, ‘Frank’, ‘Thomas’]

print ‘lastName: (0)’.format(lastnames[0][0])

print ‘First character of last name: (0)’.format(lastNames[0][0])

# we need to not assume, so we test for at least three elements

For eachElement in lastNames:

for eachCharacter in EachElemet=nt:

print everyCharacter,

print

D o u g l a s s

J e f f e r s o n

… .

listOfLists = [[‘a’. ’b’. ‘c’], [‘a’. ’b’. ‘c’], [‘a’. ’b’. ‘c’]]

2 1

numberList = [5, 15, 25, 35]

print numberList

numberList.append(45)

print numberList

numberList.insert(0, -5) # To add to the list we us the ‘append’ method.

print numberList

numberList.pop() # Removes the last itemprint numberList

print numberList

numberList.pop(0) # Removes the last itemprint numberList

# Python has no push but we can use the insert item and position it at the end of the list

numberList.insert(len(numberList), 55) # Inserts at the item

print numberList

numberList.insert(-1, 65) # It unexpectly, puts it second to the last, not the last

print numberList

del numberList[2]

print numberList

if 105 in numberList:

numberList.remove(105) # it removes only the first instance

# no error thrown

Del numberList[:3] #delete the first three elements ar

The reversed(numberList) method does the same thing as numberList.reverse()

reversed(numberList) # not inline

numberList.reverse() # inline

numberList = range(1, 11) # contains numbers 1 through 10, try xrange if it doesn’t work

squareList = []

newSquareList = []

def Square(x):

return x \* x

print Square(10)

# Now, we iterate through the first list and square all items

For everyNumber in numberList:

squareList.append(Square(everyNumber))

print squareList

newSquareList = map(Square, numberList) # Passes a delegate and don’t use the parentheses Square()

print squareList

**List Comprehension…**

[operator(x) for x in list]

# mathematical expression, function, lambda

listComprehension = [x \* x for x in numberList]

print squareList

pythonianWay = [x \* x \* x for x in numberList if x % 2 == 1]

print pythonianWay

--

Private int Divide2Numbers(int a, int b) {

If (b == 0) {

Return 0;

}

Return a/b;

}

---

stack = []

ourCode = “””private int Divide2Numbers(int a, int b) {

If (b == 0) {

Return 0;

}

Return a/b;

}”””

for eachCharacter in ourCode:

if eachCharacter == ‘(‘:

stack.append(‘(‘)

if eachCharacter == ‘)‘:

stack.pop()

print stack

---

queue = []

customers = [‘Joe’, ‘Jane’, ‘Joanne’, ‘Jim’, ‘Jen’]

for everyCutomer in customers:

currentCustomer = everyCustomer

queue.append(currentCustomer)

queue.pop(0)

print ‘The number of people in the queue is: (0)’.format(len(queue))

# To find multiple cases of the

from random import randint

print randint(2, 9) # Inclusive

numbers = []

for eachItem in range(1, 21):

numbers. Append(random(1, 11))

print number

value = 5

try:

position = numbers.index(value)

except ValueError:

position = -1

print ‘The position is: (0)’. format(position)

# Move the loop into a helper function to use….

Def findAll(L, value, start = 0):

# generator version

i = start -1

try:

i = L.index(value, I + 1)

yield I # the return result of final is thee current list

except ValueError:

pass

for eachItem in findAll(numbers, 5):

print eachItem

Name, HourlyRate, HoursWorked, FICO, Exemptions

Joe Smith, 14.55, 24.00, 37.51, 3

Gross Pay = HourlyRate \* HoursWorked

Exemptions can be 0 to 4

0 – 25%

1 – 20%

2 – 15%

3 – 10%

4 – 25%

Net Pay = GP – FICO – Exemptions

HourlyRate – Round to two significant digits

HoursWorked – Round to two significant digits

GrossPay – Currency Format

netPay – Currency Format

3RD Class – Tuesday evening, January 20, 2015

Instructor: Chris Oson – [coson@coleman.edu](mailto:coson@coleman.edu)

Dictionary, hash table, .net arrays, name-value pairs. Works on a key.

Key (Cannot have a duplicate key)

Value (doesn’t have to be consistent data types)

Communities = {} # Empty dictionary

Communities = {‘92109’:‘Pacific Beach’,

‘92101’:‘Downtown’}

Communities [‘92101] = ‘Something Else’ # it is valid to change the value

The Big ‘O’: O()

n/2: 100 elements # random integers

A telephone book lists A..Z 8 million entries.

The number of searches in an *ordered list* is 23 searches for “Lebowski”

O(n) -> o(n2)

220 = 10e7 = 1M

221 = 10e8 = 2M

222 = 10e9 = 4M

223 = 10e10 = 8M

42,000 zip codes in the U.S.

communities.setdefault(‘00000’,’’)

**functions**

def functionName(parameters)

square()

The radix of a number is the base of the number

Radix is a named parameter

Named parameters, optional parameters, order

Return more than one variable.

Helper methods helps us to perform actions on certain data types, e.g. ‘dictionary helper’

The ‘dictionary helper’ contains a reference to the function helper

Plan:

Have dictionary helper

Function DisplayNumber

Create another file called Main (#a driver)

Main/Driver

1. I want to call

2. DisplayNumber

4TH Class – Tuesday evening, January 27, 2015

Instructor: Chris Oson – [coson@coleman.edu](mailto:coson@coleman.edu)

**Tuples**

# Comment out with an if block...

if printStuff:

tuple1 = ('one', 'two', 'three', 4, 1997, 2000);

tuple2 = (1, 2, 3, 4, 5);

tuple3 = "a", "b", "A", "Z";

emptyTuple = ();

singleValueTuple = (2, )

tuple1[0] = 'five'

print tuple1[0]

# you can use the slice notation

print tuple1[0:3]

# The results will be a tuple of a tuple.

print type(tuple1[0:3])

first = (33, 44, 55)

second = (66,)

third = first + second

print third # results (33, 44, 55, 66)

# You can't change a tuple but you can create a copy of a tuple.

brandNewTuple = (10, 20, 30)

print len(brandNewTuple)

print (5, 9, 4) + (15, 19, 14)

print (5, ) \* 4 # duplicate instances of a tuple.

if 20 in brandNewTuple:

print 'The number is in brandNewTuple'

for eachTupleItem in brandNewTuple:

print eachTupleItem \* 2

# The two functions of data type Tuple is the 'min' and 'max'

print min(brandNewTuple)

print max(brandNewTuple)

print tuple(range(5, 15)) # using tuple to create a new tuple

# Tuples are faster than using a List. It has a smaller big O.

# Tuples are also safer because a user cannot change the data (they could change data in a list)

# A tuple can convert a list into a tuple

#

print id(b)

print a.\_\_sizeof\_\_()

print b.\_\_sizeof\_\_()

a = tuple(range(1000))

b = list(range(1000))

print id(a)

b = [1, 2] # I'm changing the list now because it is immutable

b[0] = 3

a = (1, 2)

a[0] = 3 # results an error because we're trying to change the value of a tuple.

# Python has the id function

b.append(5)

print b

# result 8024, 9088

# Python uses parentheses for tuples

# they are immutable - cannot modify the contents of aa tuple

**Regular Expressions**

Regex Meta Characters:

^ - The hat symbol indicates the beginning of a line

$ - The dollar symbol indicates the end of a line

Are case sensitive by default.

Examples of ‘Anchors’: (anchor positions on a line of input):

^M

E$

Example of ‘Alternation operator’ (like an ‘or):

| - The alternation symbol is the vertical bar

^Gra|ey (for spelling variations, it applies to only one character before and after the | symbol)

Grey (British spelling)

Gray (American spelling)

^(Je|Geo)rey (Group the grouping by parentheses grouping operator)

. – matches any character except a new line character ‘\n’

+ – the plus meta-character is one or more characters preceding a +

Bed

Beeed

\* – the star meta-character is zero or more characters preceding a \* (an asterisk)

Bed

bd

() – grouping or captive variables

^(Je|Geo)ffrey$

{} – Used to indicate a min and/or max range

sweet

sweeeeet

^swe{2,4}t # two to four e’s will match but five e's will not match

swewet # matches

? – optional

^swea?t

swet

sweat

[] – denoted with square brackets

(the range of characters – are contiguous, i.e. [A-M],

OR the list of characters – are non-contiguous, i.e. [AEIOU])

^[0-9A-F]+$ # The $ means the last character must also match

\s – Used to indicate white space: (the blank “”, the tab “\t”, the new line “\n” character)

The Character Class

\w – means any alpha-numeric value: (A-Z, a-z, 0-9) (lowercase w)

\d – means any digit [0-9]

\b – means word boundary

Regex is typically used for data validation

For negative numbers use: ^-? (It’s optional)

"^-?\d+\.\d+$" (for -129.0, 700.00)

"^-?\d+(\.\d+)?$" (for -129, 700, 700.00) # (Regex to match neg float, pos float, and whole number)

"^-?\d+(\.\d+)?$" (for numbers with commas, e.g. 1,575, strip out the comma characters and replace with nothing)

number = number.replace(‘,’, ‘’)

var data = “ 129.99 “; (JavaScript)

data = s/^\s+ | \s+$//g;

Chain functions together:

number = number.replace(‘,’, ‘’)\

.strip()

.title()

def IsValidNumber(number, pattern):

# match only works with the beginning of the string

# search finds matches throughout the string

if len(pattern) == 0:

return True

isMatch = re.search("^\d+\.\d+$") # isMatch is now an object

# There are four types of function parameters

if isMatch is None

return True

return true

print IsValidNumber(59.999, "^\d+\.\d+$")

San Diego Python User Group – February 21, 2015

WiFi – Name: Ansir, PW: startnow

Data Science Panda Tutorial Group –

Topic: Variance and Correlation, regression

Ipynb#

Need:

-Excel Notebook

-Panda (under the hood: numby)

-Seaborn

68% sigma 1

8TH Class – Tuesday evening, February 24, 2015

pip

Python import

Abstract classes

Static methods

How do we connect to the database:

1. Install mysql
2. We

Make 20 records for the database.

StudentID – Int

FirstName varChar(32)

LastName varChar(32)

FullName (Computed column)

StartDate not null DateTime

GradDate not null DateTime

GPA float Decimal(9,4) precision, scale

Auto increment # MySQL  
Auto identity # SQL Server  
Auto number # MS Access

DateUtils do it in the database,

To determine if we had a module preinstalled in Python – We write a Python script with the pip:

Import pip

Install\_packages = pip.get\_installed\_distributions()

Installed\_packages\_data = sorted (

[“%s == %s”%(i.key, i.version) for I in installed\_packages]

)

print installed\_packages\_data

be in the scripts folder python\scripts\pip install MySQL-python

Download the appropriate connector module.

Mysql-connector-python-2.0.3

MySQL on a Windows machine needs two components:

1) MySQL database engine

2) MySQL Workbench

pip 6.0.8

python-datautil 2.4.0

setuptools 12.2

six 1.9.0

# Use the MySQLdb

use MySQLdb

db = MySQL.connect(“localhost, “username, “password, “DATABASE”)

cursor = db.corsor

-----

Create a union query:

SELECT 1 AS ID, ‘Perl’ AS LANGUAGE

UNION

SELECT 2, ‘PHP’

UNION

SELECT 3, ‘Ada’

UNION

SELECT 4, ‘TCL’

Soft delete:

LanuageID Language IsDeleted

28 Algol 0

30 Algol 1

28 Algol 0

SELECT # FROM languages WHERE ISDELETED = 0;

def SafeQuote(input):

trimmed = input.strip()

if len(trimmed) == 0:

return input

singleQuote = “’”

if singleQuote not in trimmed:

return input

return trimmed.replace(“’”, “\\\’”)

9TH Class – Tuesday evening, March 3, 2015

Abstract class exposes 2 methods: insertStudent and selectStudent

Call the ‘database’ Driver that implements the abstract class and that is where the code is.

Third file is the Student class.

Computed column is in later versions. So we will do this now:

Create class definitions with the properties.

Assignment3.py

Db = DatabaseDriver()

Db.InsertStudent(..data..)

Db.SelectStudent( page, offset )

Here is where we process our inline parameters.

e.g. assignment3.py 7, 7, ‘2001-05-01’ # command line parameters

from here to the current date

For pagination try the limit keyword.

pageNo = sys.argv[1]

At the command line:

4 files

5 Command line parameters. When you access the parameters you’ll use the @args array.

Void main (string args[]){

Printf(“Hello world.”);

Return;

}

First.exe args = 0

e.g. in Perl

On windows: > perl first.pl first and second args

e.g. in Python,

It has its own args

Import the sys.argv module

Print sys.argv[0] # script name

Print sys.argv[1] # page number

Print sys.argv[2] # number of records

Print sys.argv[3] # any date

Exception Handling (of errors)

All exceptions are derived from the Exception Class:

Exception

- errno

- strerror

SomeException IOError (subtype | subclass of Exception)

Class ErrorWithArgs(Exception):

def \_\_init\_\_(self, \*args):

# \*args is used to get a list of the parameters passed in

self.args = [a for a in args]

Raise the exception by using a try block.

# Revisit the map function again

Import the datetime module

Template of all exceptions – in Python

3 things expensive for Time:

- Opening a file

- Printing

- Opening a database connection

‘basename’ of the assignmet filename

Week 6 - Classes, abstract classes, abstract methods, how to create

Week 7 – Pip, dateutils, parse

Week 8 – SQL, sql connection

Week 9 – error handling, map, lambda/anonymous, command-line arguments

Google MySQL Python – Chapter 5 Connector/Python Coding Examples

Exceptions with mysql