Python Tutorial

Written by: **Mickey Nguyen**

(linkedin: mickeytrong73@yahoo.com)

2015

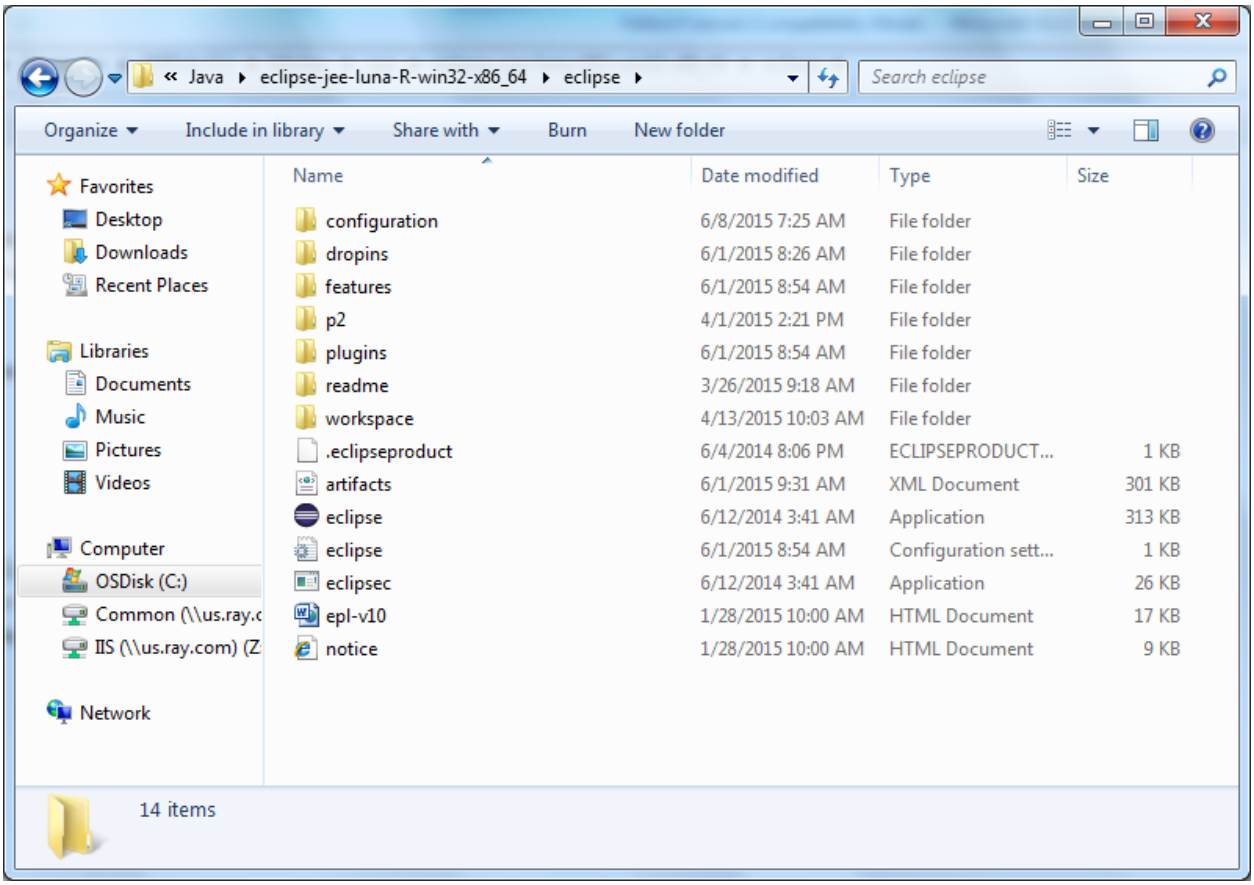
Python Tutorial
Mickey Nguyen

1 Python Installation

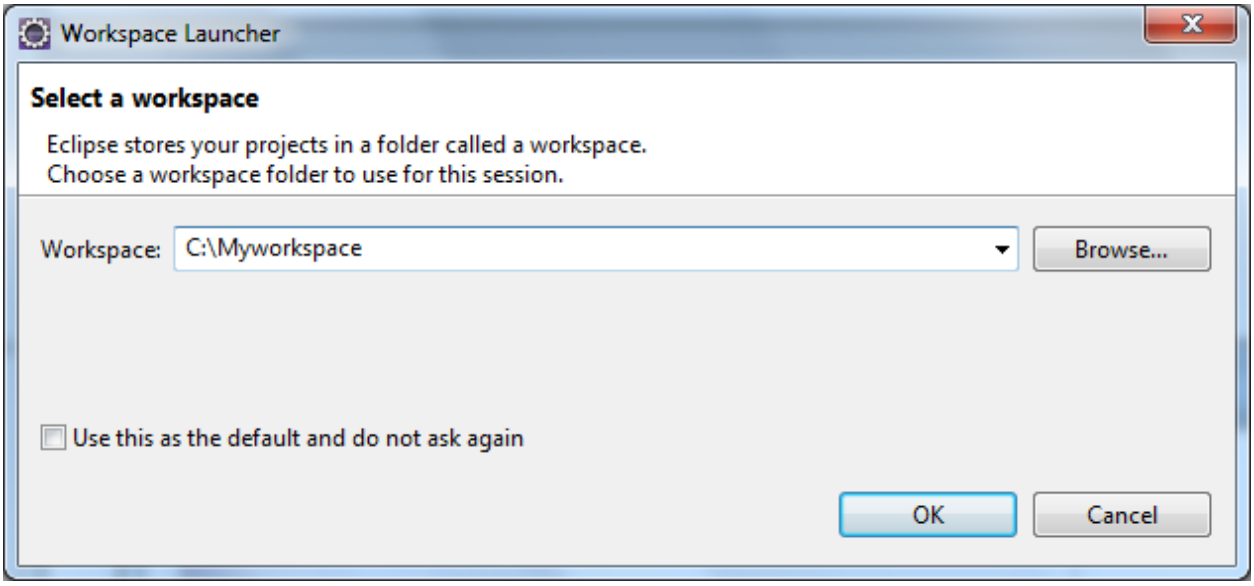
I am going to use Eclipse as IDE (Integrated development environment) compiler. Download Eclipse at

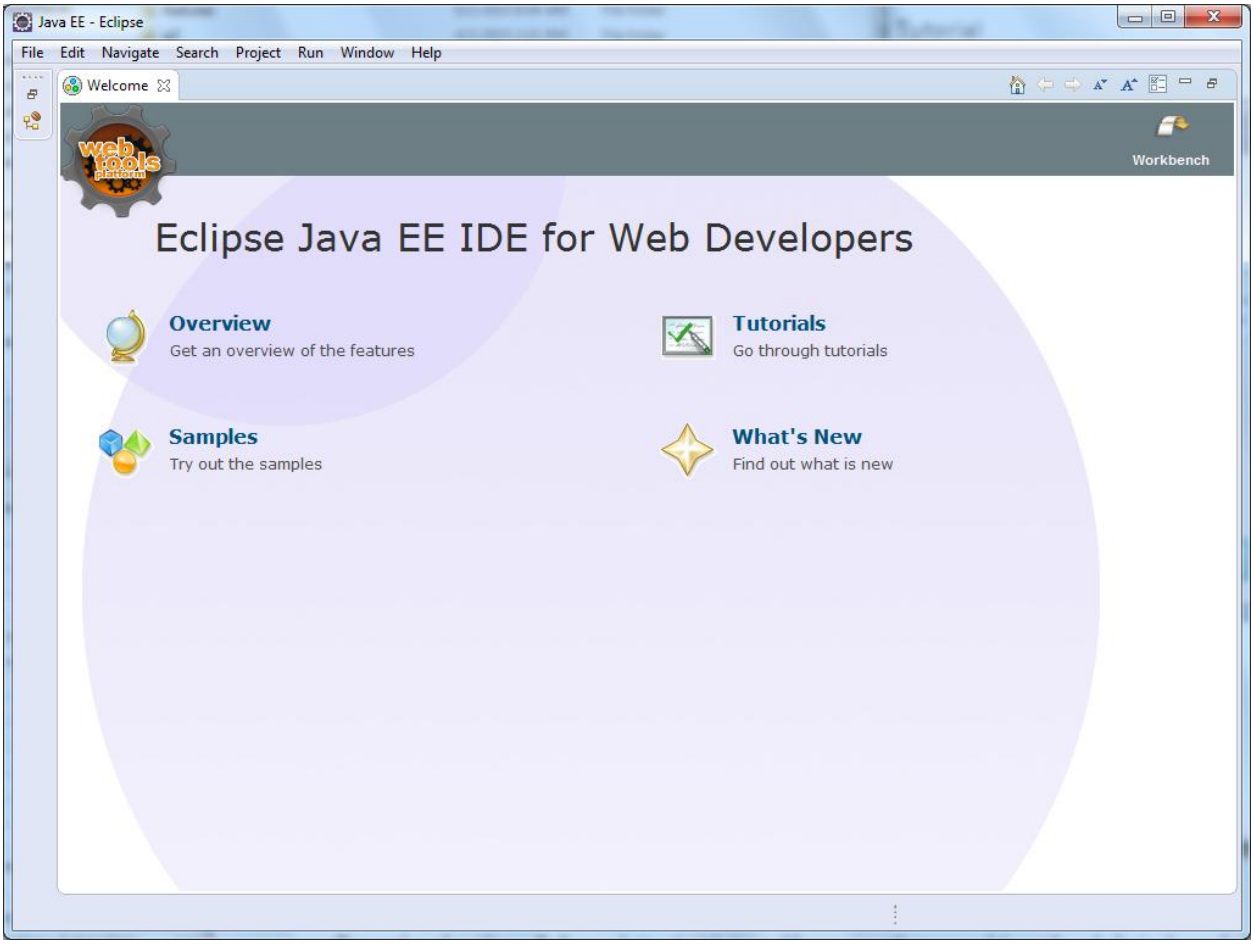
http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/luna/R/eclipse-jee-luna-R-win32-x86_64.zip

Save Eclipse zip file in your local, then extract all. After extract all, it looks like this

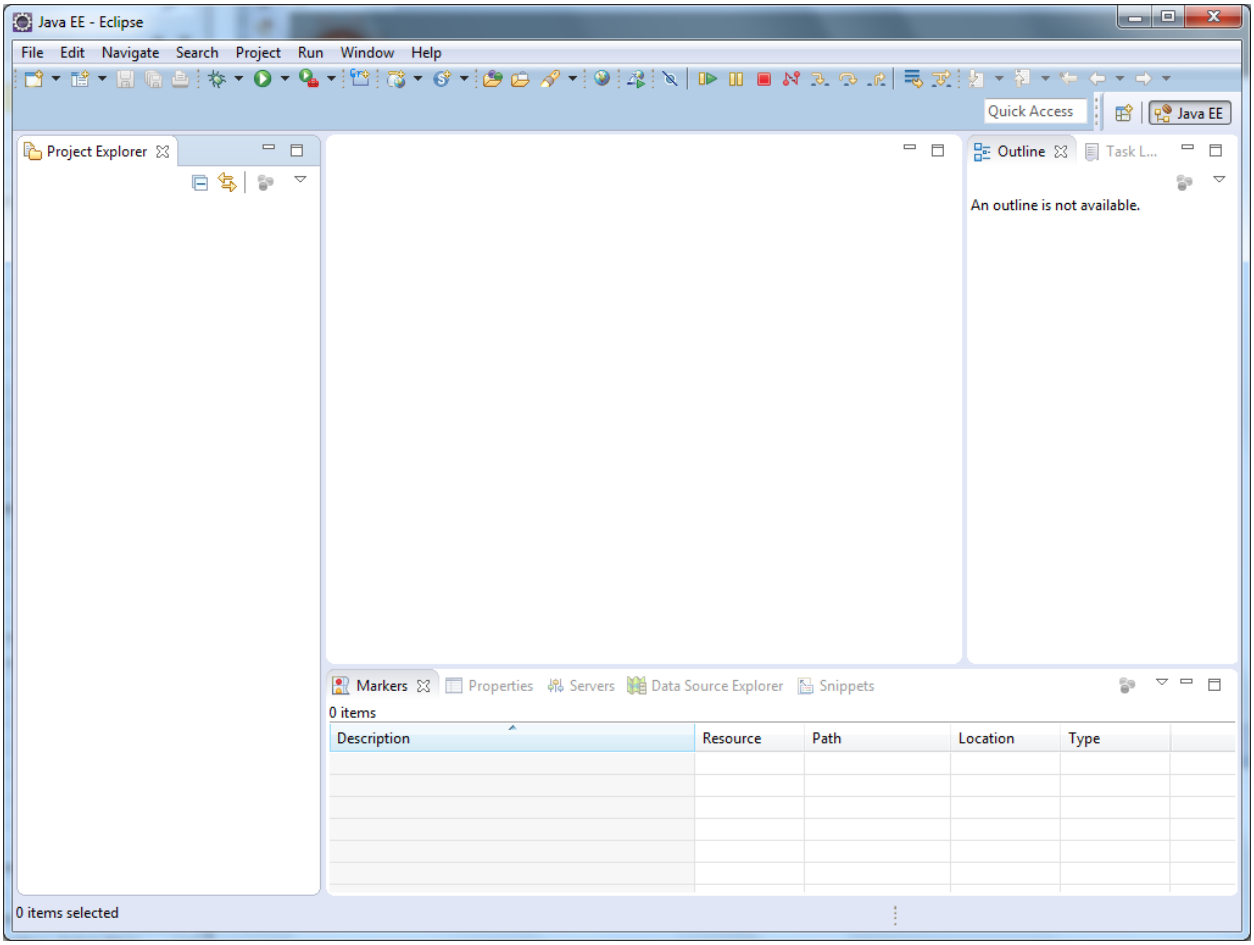


To launch Eclipse simply just double click on eclipse. Before do this, create a folder in your local drive name it as “Myworkspace”, you can name it whatever you want. It will be a director contain all of your work.





You can close the Welcome screen



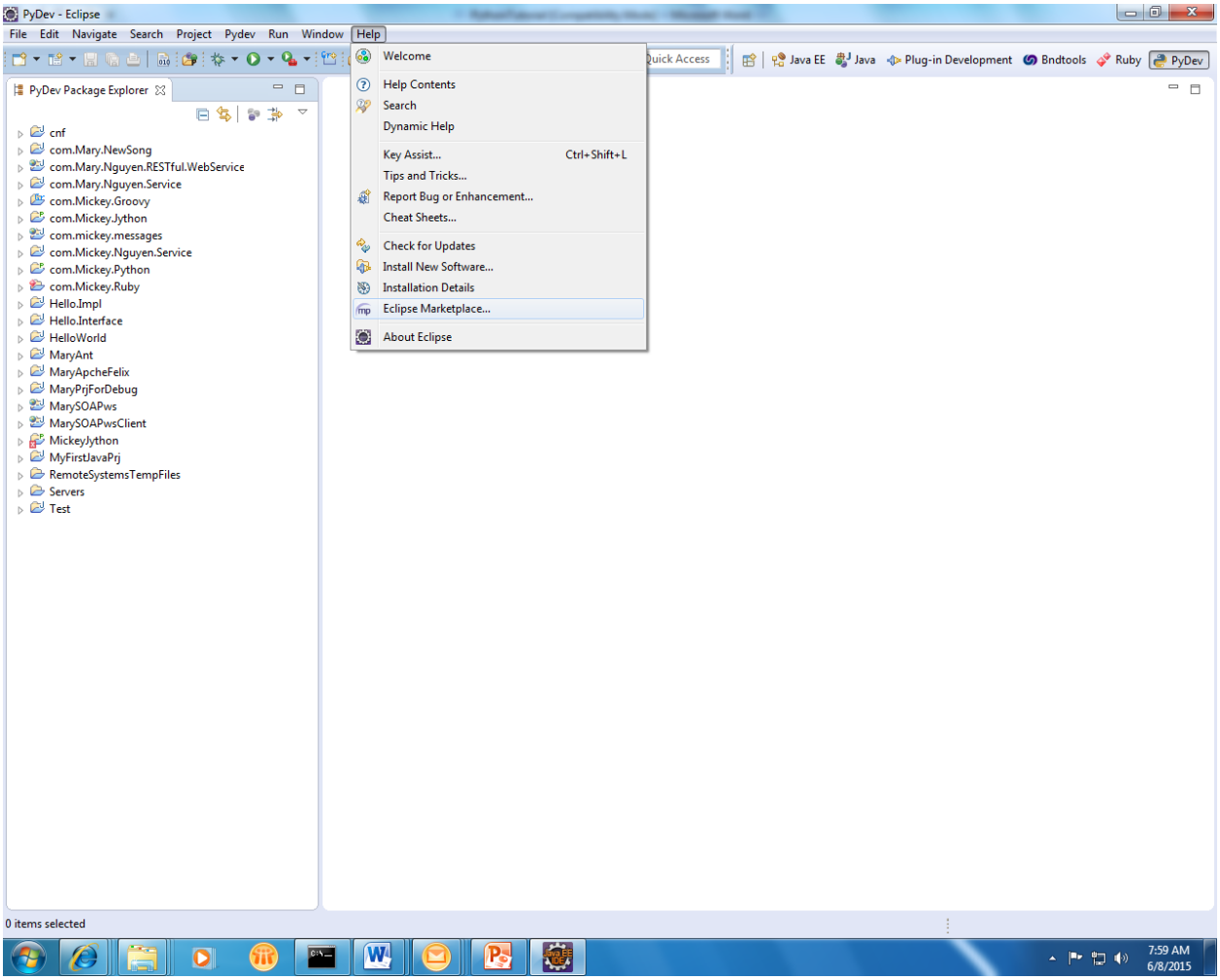
Learning Python 3.x version

There are many differences between Python 2.x and 3x. In this study, I am using Python 3.x; therefore, I used all the new syntaxes.

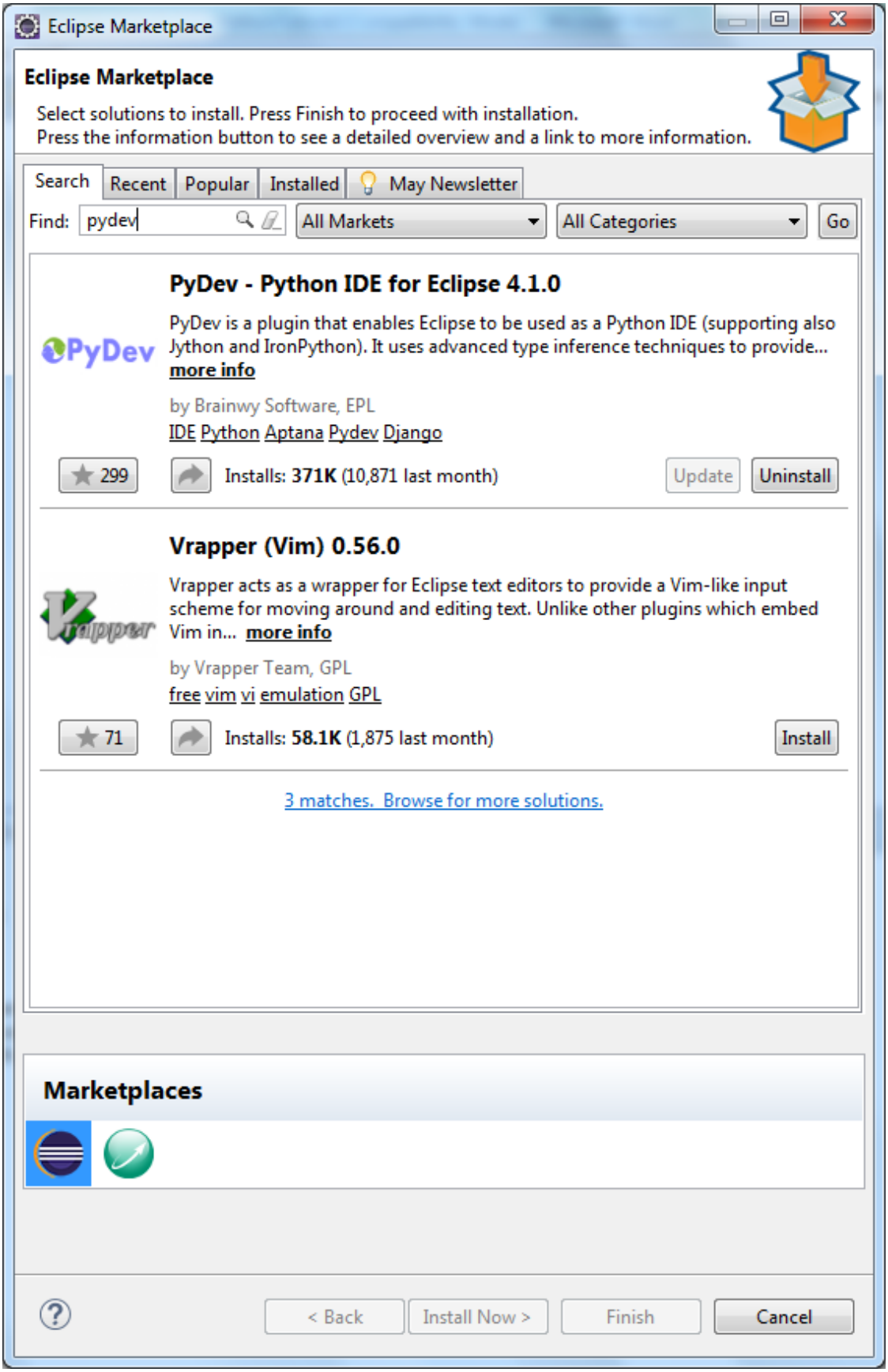
Download python 3.4.x or latest at https://www.python.org/downloads/windows/ , I used 3.4 version. You can use 3.5 if you want to. Here is link for 3.4 version Windows x86-64 MSI installer

There are acouple way to install pydev for Eclipse IDE.

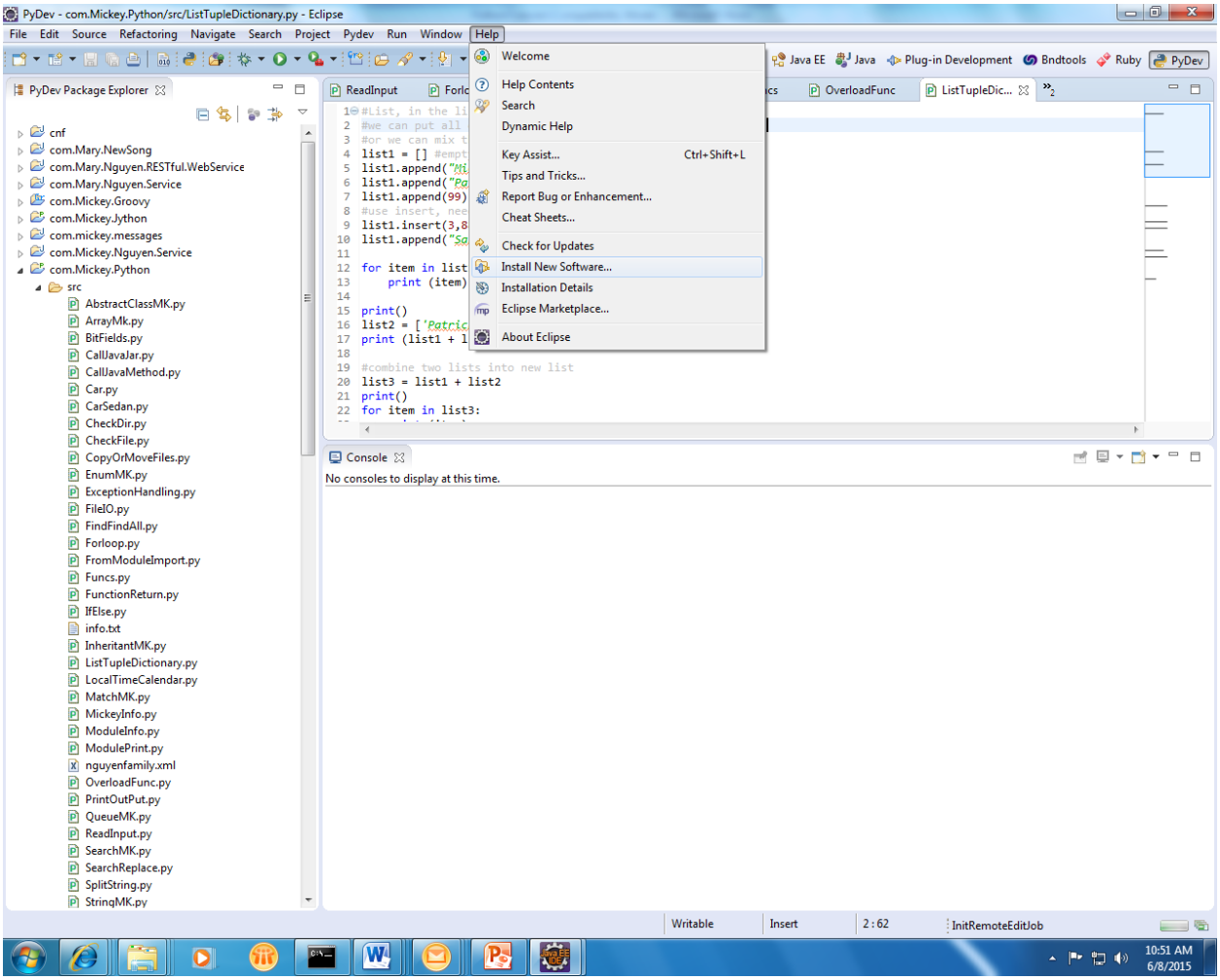
- From Eclipse, go to Help->Market place, search for l PyDev and install it.



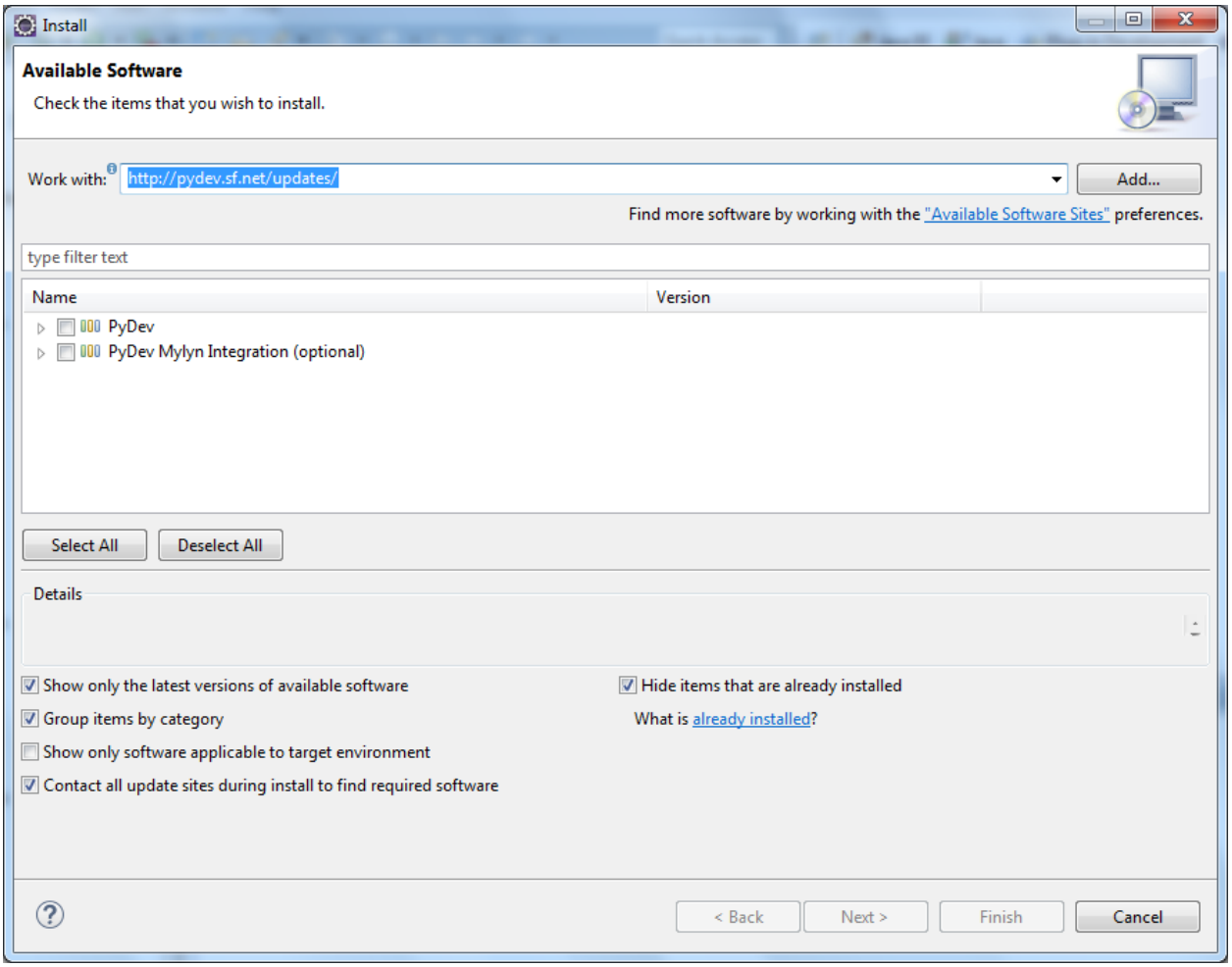
Search for PyDev and install it.


Eclipse Marketplace
Eclipse Marketplace
Select solutions to install. Press Finish to proceed with installation.
Press the information button to see a detailed overview and a link to more information.
Search
Recent
Popular
Installed
May Newsletter
Find: pydev
All Markets
All Categories
Go
PyDev - Python IDE for Eclipse 4.1.0
PyDev is a plugin that enables Eclipse to be used as a Python IDE (supporting also Jython and IronPython). It uses advanced type inference techniques to provide... more info
by Brainwy Software, EPL
IDE Python Aptana Pydev Django
299
Installs: 371K (10,871 last month)
Update
Uninstall
Vrapper (Vim) 0.56.0
Vrapper acts as a wrapper for Eclipse text editors to provide a Vim-like input scheme for moving around and editing text. Unlike other plugins which embed Vim in... more info
by Vrapper Team, GPL
free vim vi emulation GPL
71
Installs: 58.1K (1,875 last month)
Install
3 matches. Browse for more solutions.
Marketplaces
< Back
Install Now >
Finish
Cancel

- If you are done install from Marketplace, skip this install option. Help->Install New Software



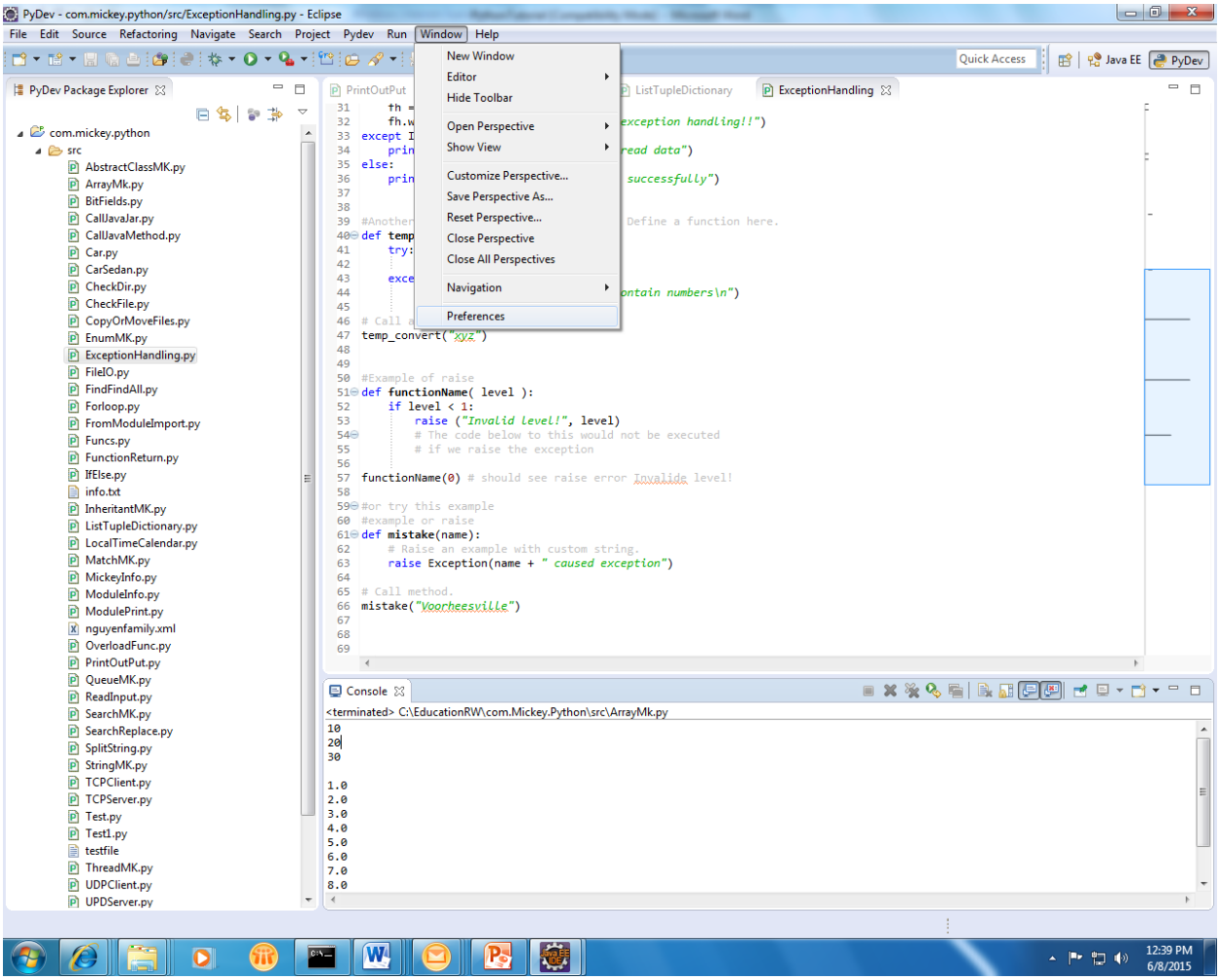
- Then use this link http://pydev.sf.net/updates/ to install

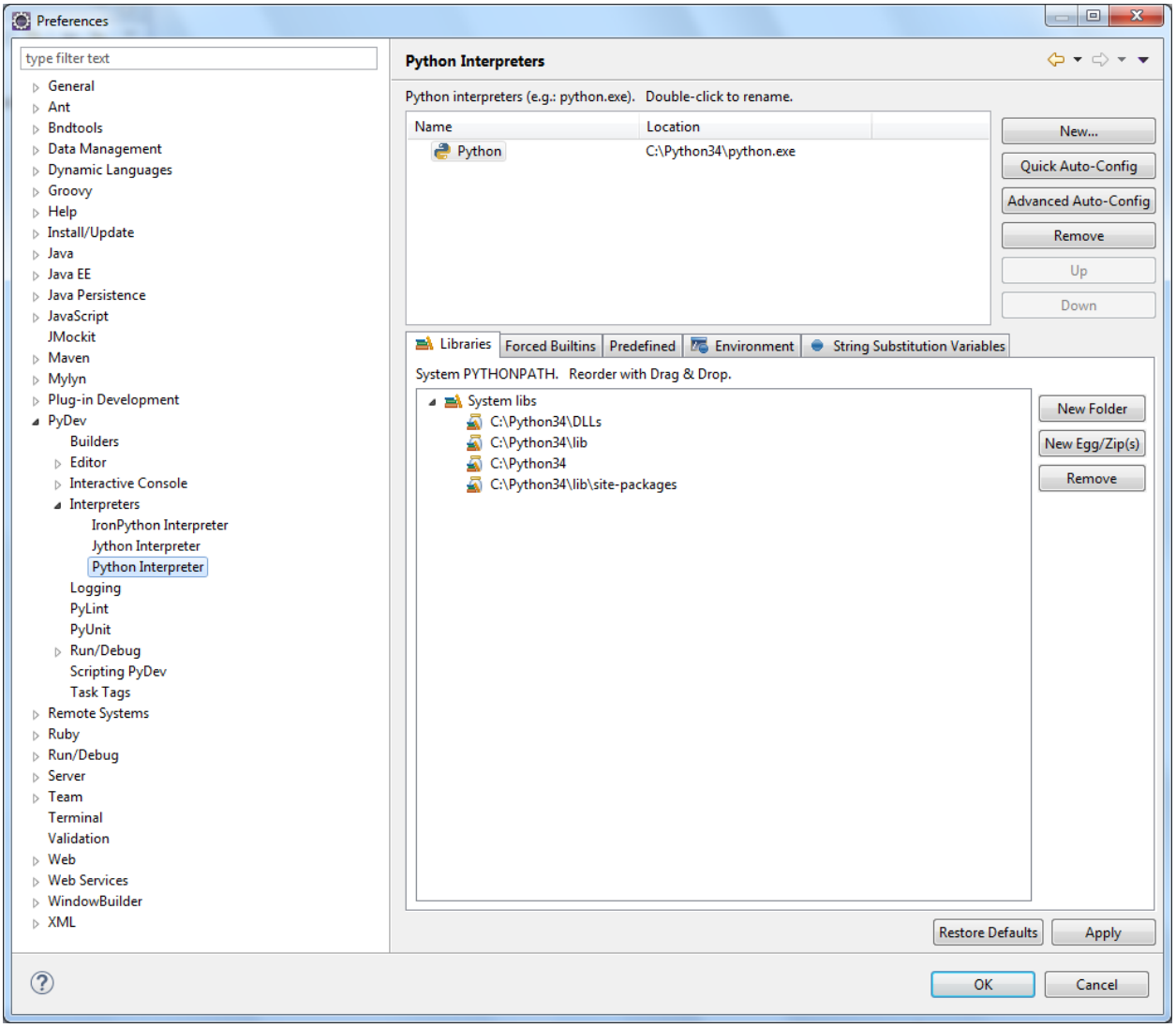


Finish install PyDev

Then,

From Eclipse IDE go to window references->PyDev->Interpreters->Python Interpreter, Click on new then browse to where you install Python3.4 (C:\Python34\python.exe)


PyDev - com.mickey.python/src/ExceptionHandling.py - Eclipse
File Edit Source Refactoring Navigate Search Project Pydev Run Window Help
New Window
Editor
Hide Toolbar
Open Perspective
Show View
Customize Perspective...
Save Perspective As...
Reset Perspective...
Close Perspective
Close All Perspectives
Navigation
Preferences
PyDev Package Explorer
com.mickey.python
src
AbstractClassMK.py
ArrayMk.py
BitFields.py
CallJavaJar.py
CallJavaMethod.py
Car.py
CarSedan.py
CheckDir.py
CheckFile.py
CopyOrMoveFiles.py
EnumMK.py
ExceptionHandling.py
FileIO.py
FindFindAll.py
Forloop.py
FromModuleImport.py
Funcs.py
FunctionReturn.py
IfElse.py
info.txt
InheritantMK.py
ListTupleDictionary.py
LocalTimeCalendar.py
MatchMK.py
MickeyInfo.py
ModuleInfo.py
ModulePrint.py
nguyenfamily.xml
OverloadFunc.py
PrintOutPut.py
QueueMK.py
ReadInput.py
SearchMK.py
SearchReplace.py
SplitString.py
StringMK.py
TCPClient.py
TCPServer.py
Test.py
Test1.py
testfile
ThreadMK.py
UDPClient.py
UPDServer.py
PrintOutPut
ListTupleDictionary
ExceptionHandling
def temp_convert("xyz")
#Example of raise
def functionName(level):
if level < 1:
raise ("Invalid level!", level)
functionName(0) # should see raise error Invalide level!
#or try this example
#example or raise
def mistake(name):
# Raise an example with custom string.
raise Exception(name + " caused exception")
# Call method.
mistake("Voorheesville")
Console
<terminated> C:\EducationRW\com.Mickey.Python\src\ArrayMk.py
10
20
30
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
12:39 PM
6/8/2015



Download Jython at http://www.jython.org/downloads.html

I am using Jython 2.5.3

From Eclipse IDE go to window references->PyDev->Interpreters->Jython Interpreter, then browse to where you install Jython (C:\jython2.5.3\jython.jar)

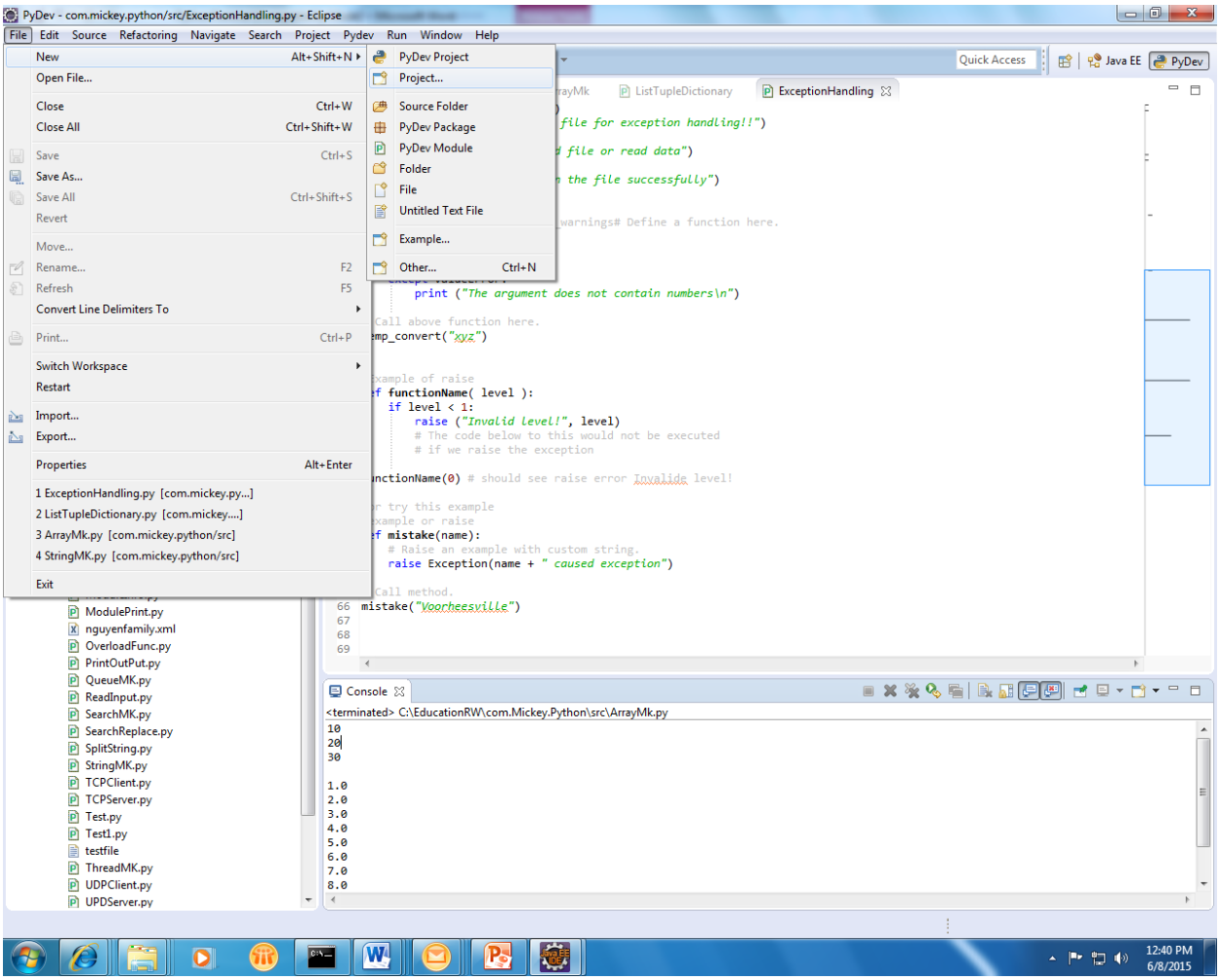
2 Introduction

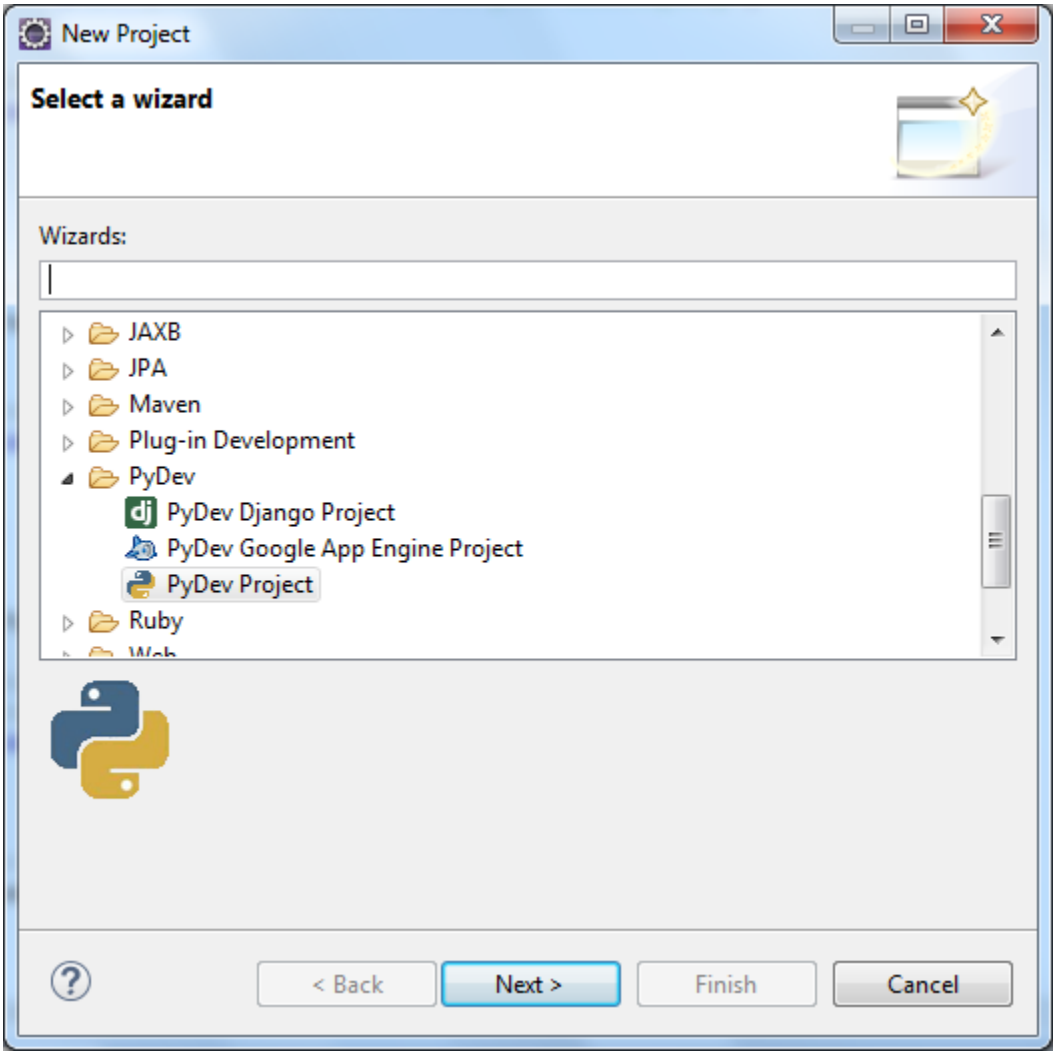
This is high level principle concepts of Python language.

A few important things I want to mention about Python. You don't have to declare data type for variable as in C/C++ . Base on the value you assign the value to your variable, it will understand what data type for that variable. The indentation is very important for Python, that is tell where the scope of that variable belong to. Python 3 you have to use ( ) for print operation. You don't have to use ";" at the end of print operation or function call.

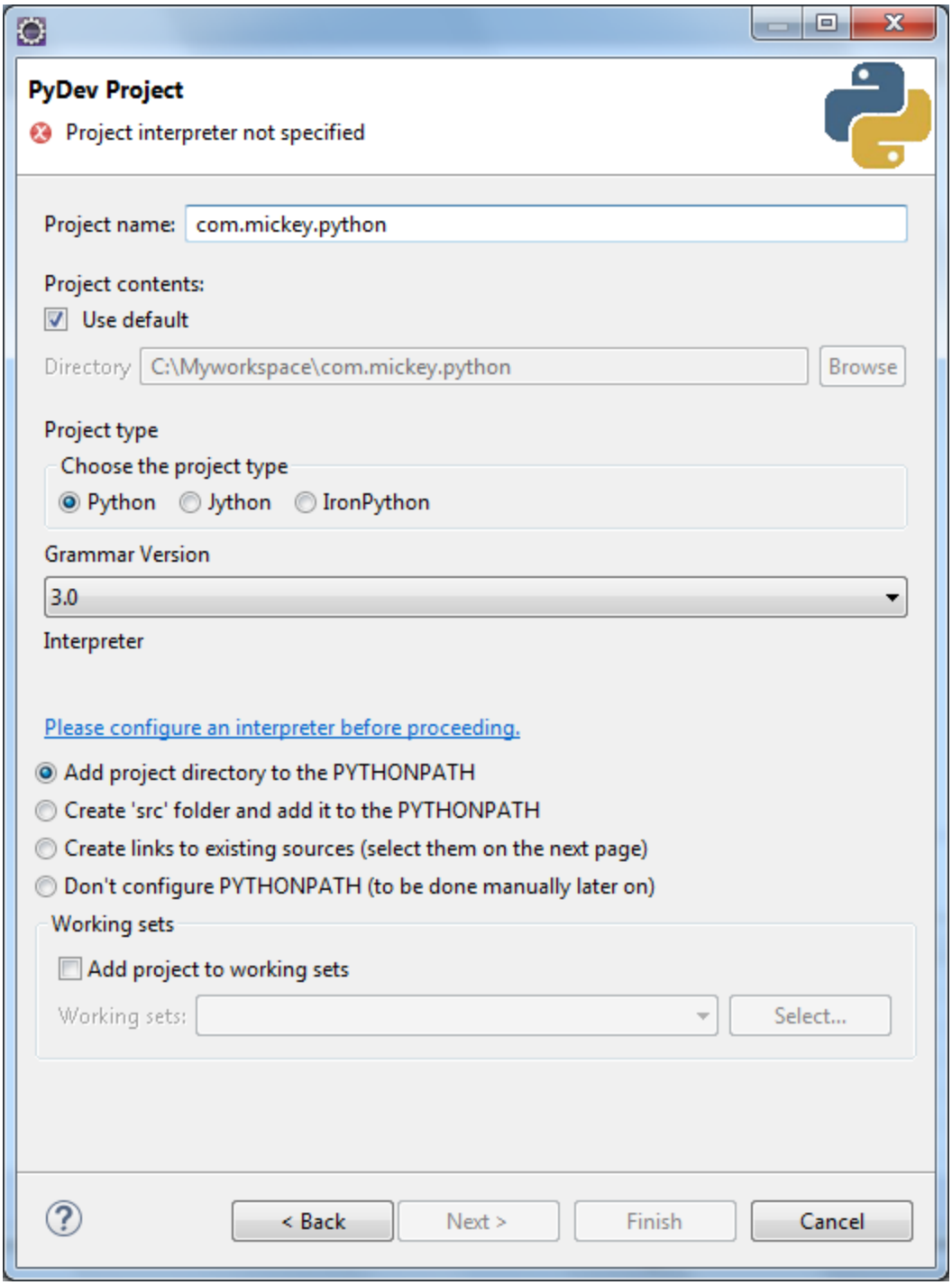
Create Python project using Eclipse

File->New->Project

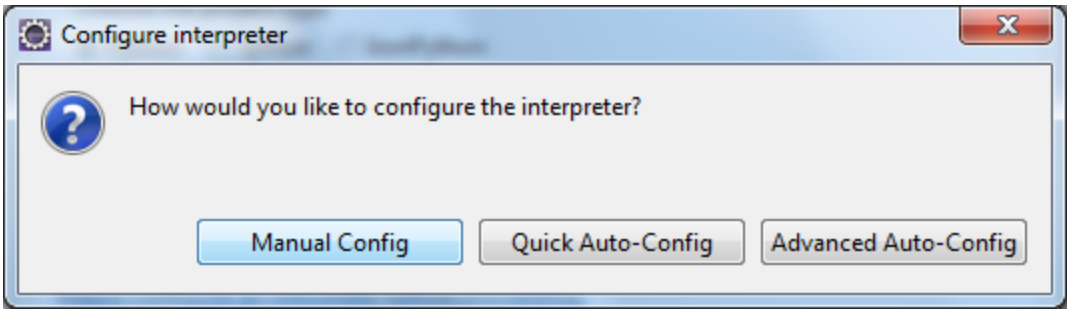




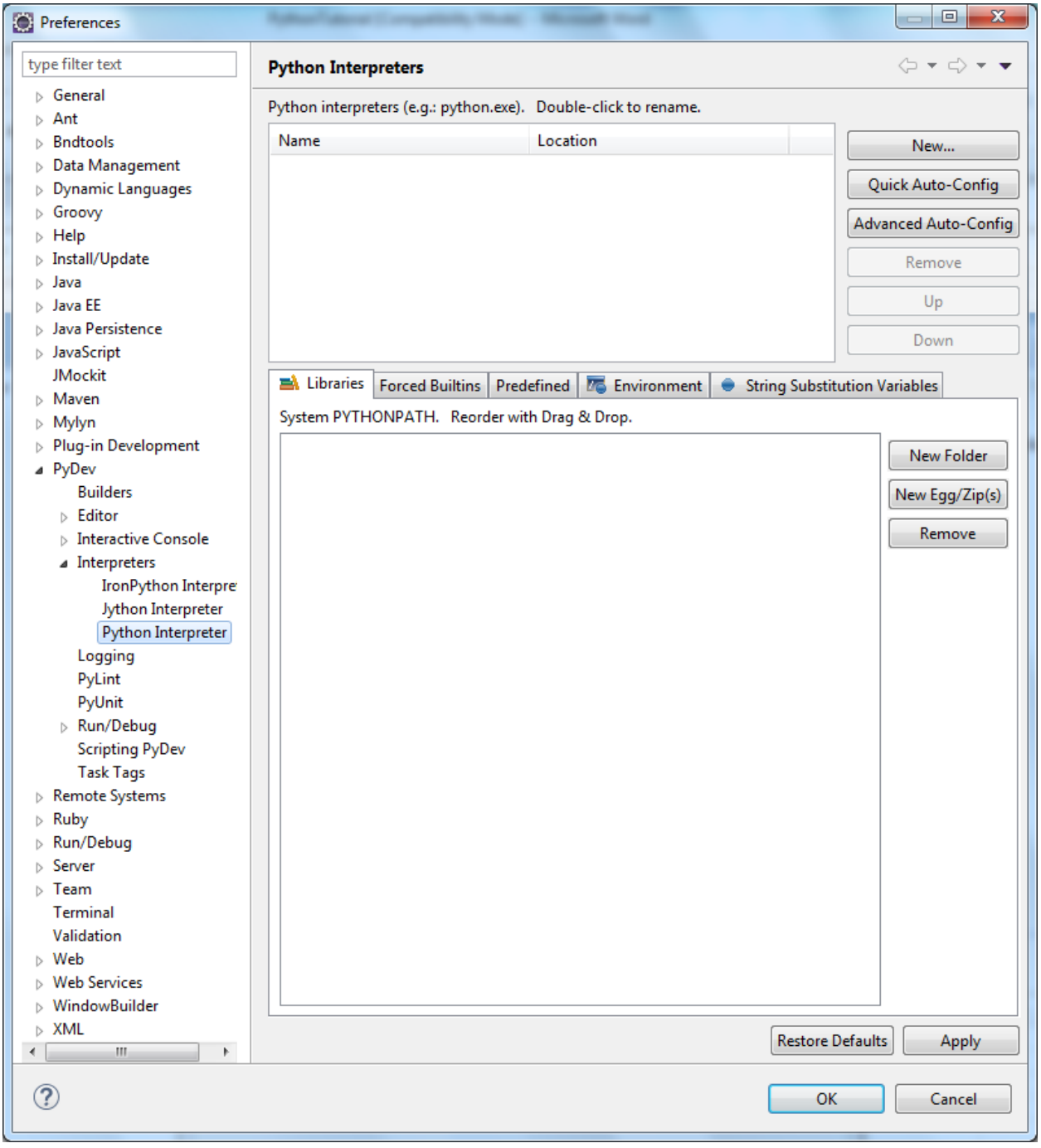
Look for PyDev then select PyDev Project. Name your project whatever you want, select grammar version 3



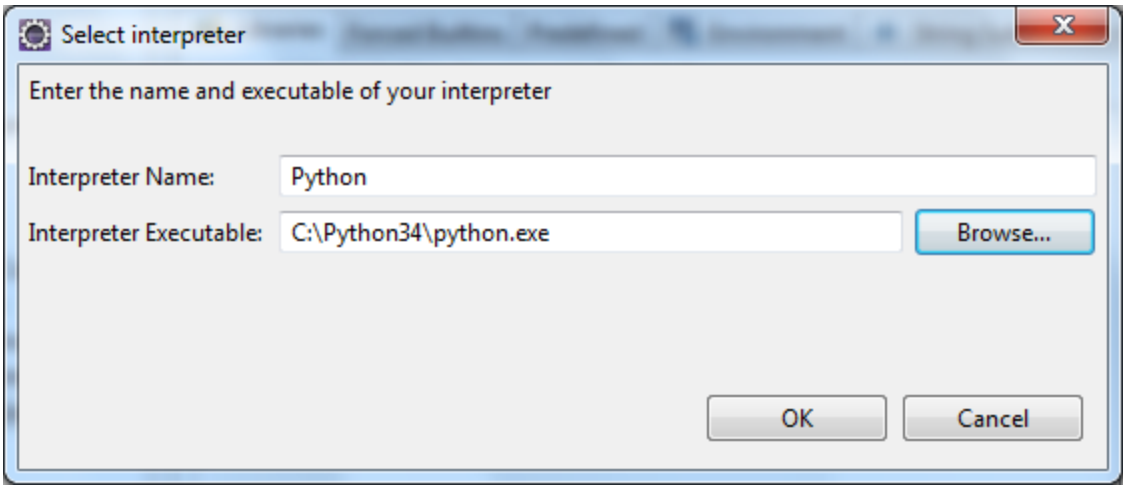
Click on “Please configure an interpreter berfore proceeding”



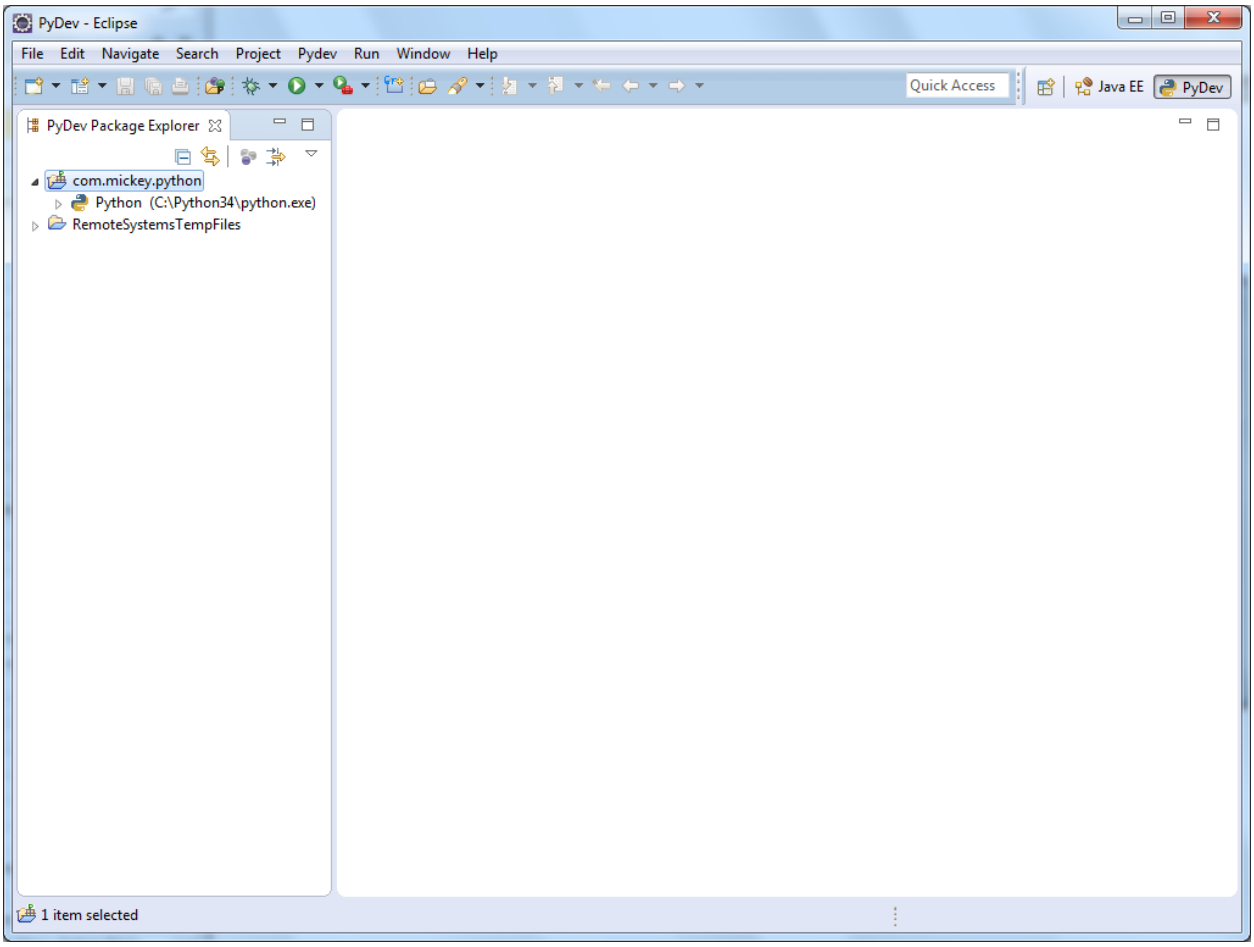
Select Manual Config



Select Python Interpreter, click on new fill out these info

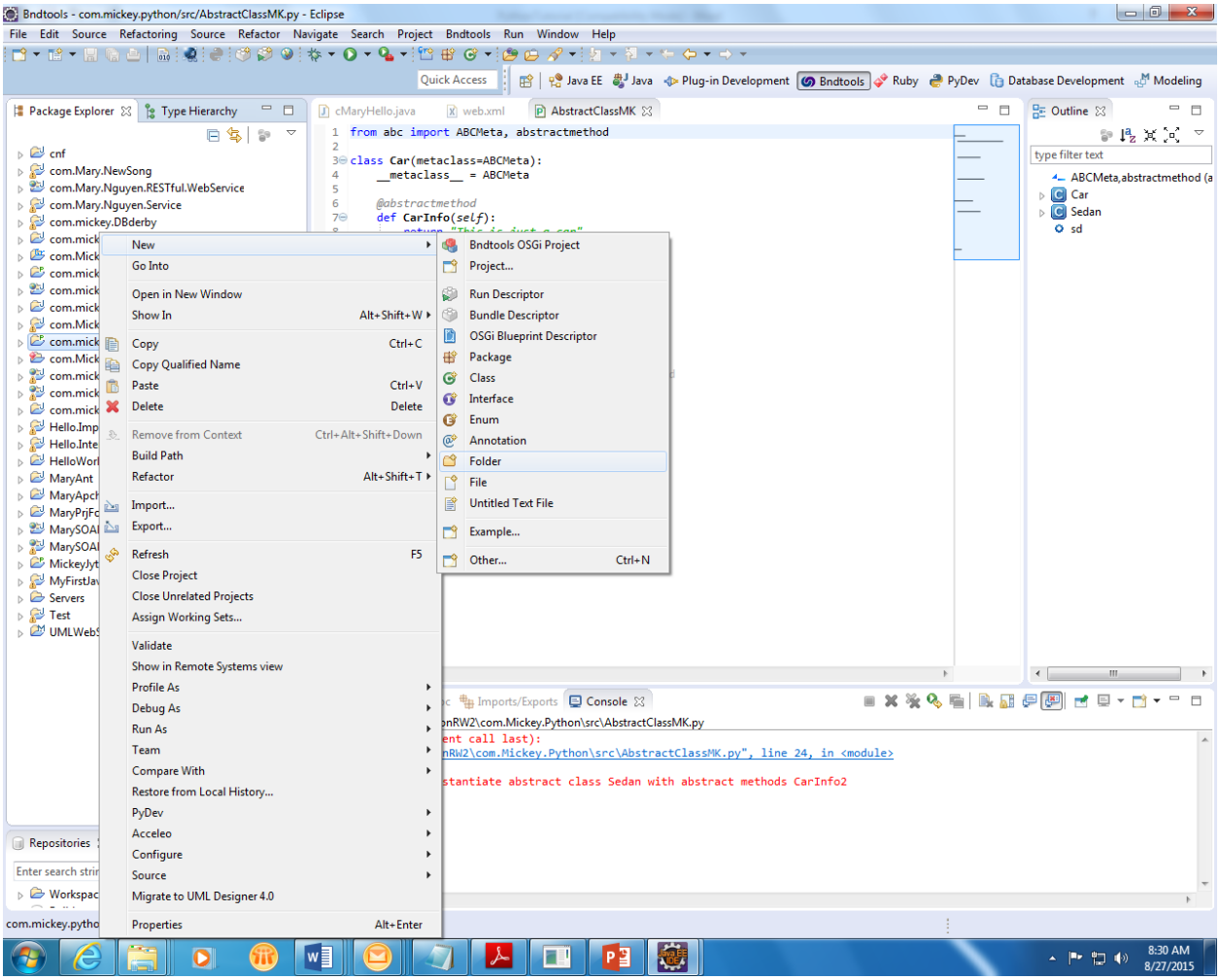


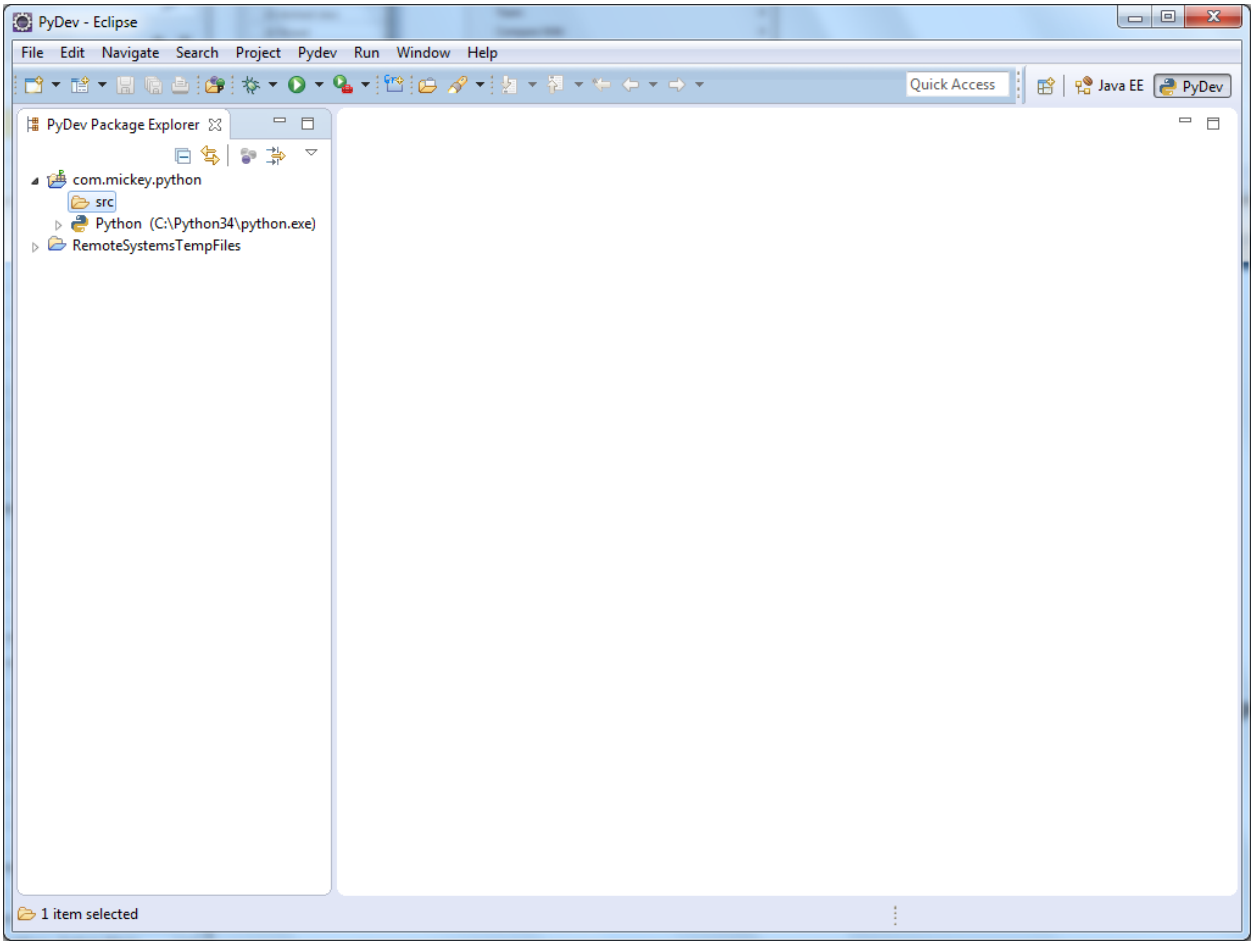
Click on a few oks, then finish, it should go all the way done



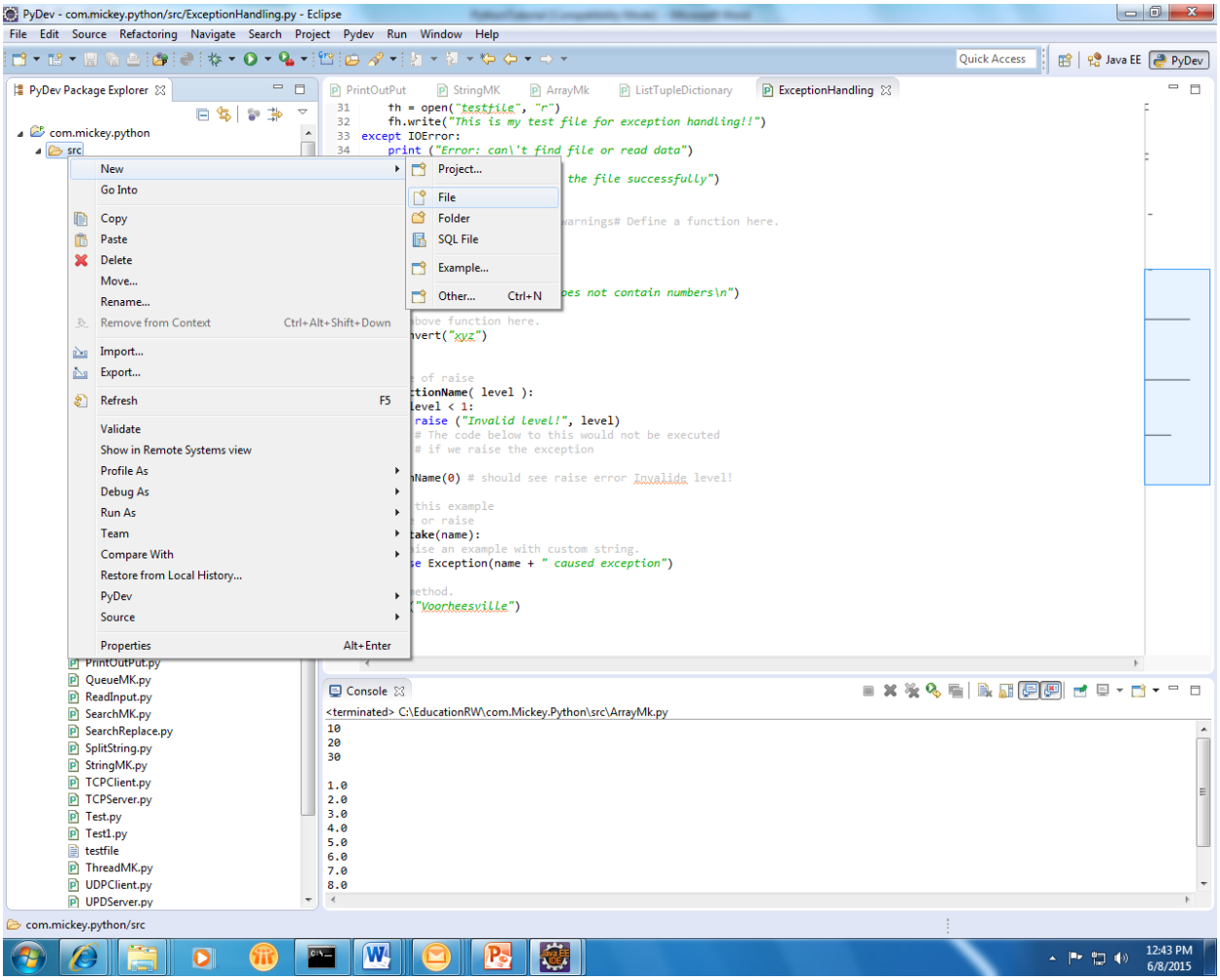
You just create your first Python project on you work space

Right click on project name->new folder, name is “src”


Bndtools - com.mickey.python/src/AbstractClassMK.py - Eclipse
File Edit Source Refactoring Source Refactor Navigate Search Project Bndtools Run Window Help
Quick Access
Java EE Java Plug-in Development Bndtools Ruby PyDev Database Development Modeling
Package Explorer
Type Hierarchy
cMaryHello.java
web.xml
AbstractClassMK
1 from abc import ABCMeta, abstractmethod
3 class Car(metaclass=ABCMeta):
4 __metaclass__ = ABCMeta
6 @abstractmethod
7 def CarInfo(self):
Outline
type filter text
ABCMeta.abstractmethod (a
Car
Sedan
sd
New
Go Into
Open in New Window
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Validate
Show in Remote Systems view
Profile As
Debug As
Run As
Team
Compare With
Restore from Local History...
PyDev
Acceleo
Configure
Source
Migrate to UML Designer 4.0
Properties Alt+Enter
Bndtools OSGi Project
Project...
Run Descriptor
Bundle Descriptor
OSGi Blueprint Descriptor
Package
Class
Interface
Enum
Annotation
Folder
File
Untitled Text File
Example...
Other... Ctrl+N
Imports/Exports
Console
nRW2\com.Mickey.Python\src\AbstractClassMK.py
ent call last):
nRW2\com.Mickey.Python\src\AbstractClassMK.py", line 24, in <module>
stantiate abstract class Sedan with abstract methods CarInfo2
Repositories
8:30 AM
8/27/2015

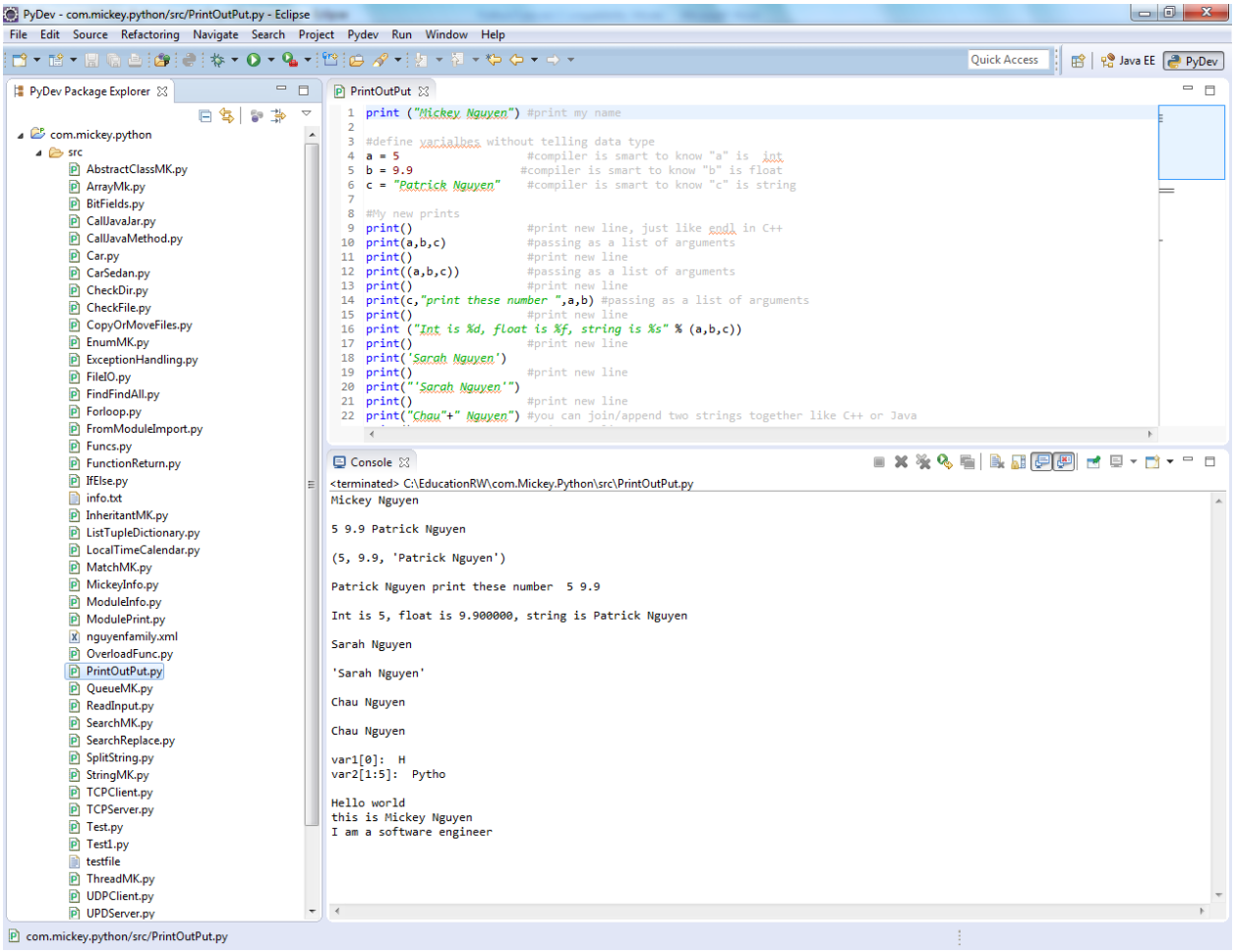


Right click on src->new->file, then enter your scriptname as “PrintOutPut.py”



Copy my code and I wrote and paste it in, you can run it

Right click on PrintOutput->Run As->Python Run



3 Print

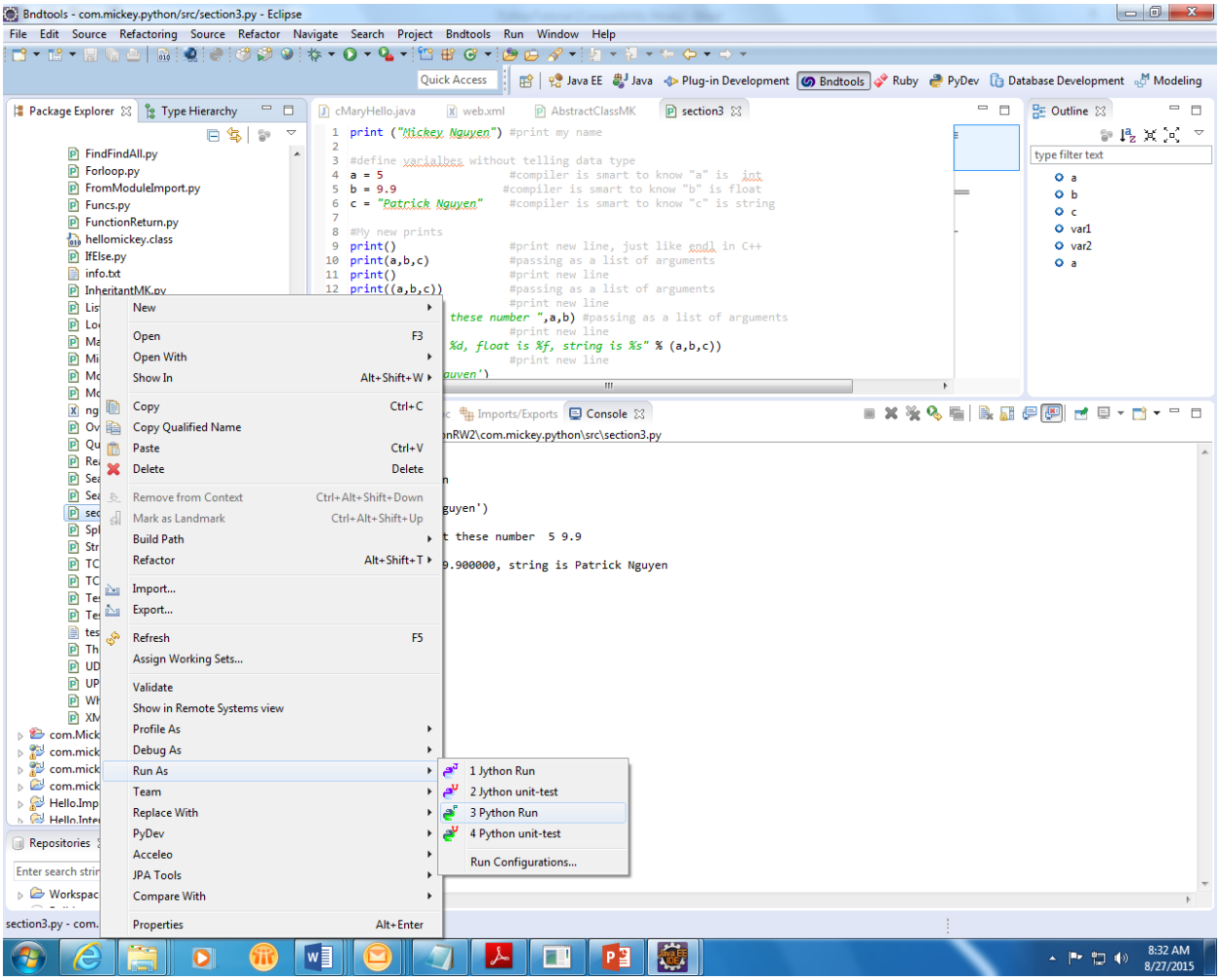
Syntax:
 # is a comment
print("whatever") # whatever is a string
print('whatever') # whatever is a string
print("The string %d,%f"% (arguments)) # you can pass in arguments like you can in C language printf
print() # print a new line

section3.py

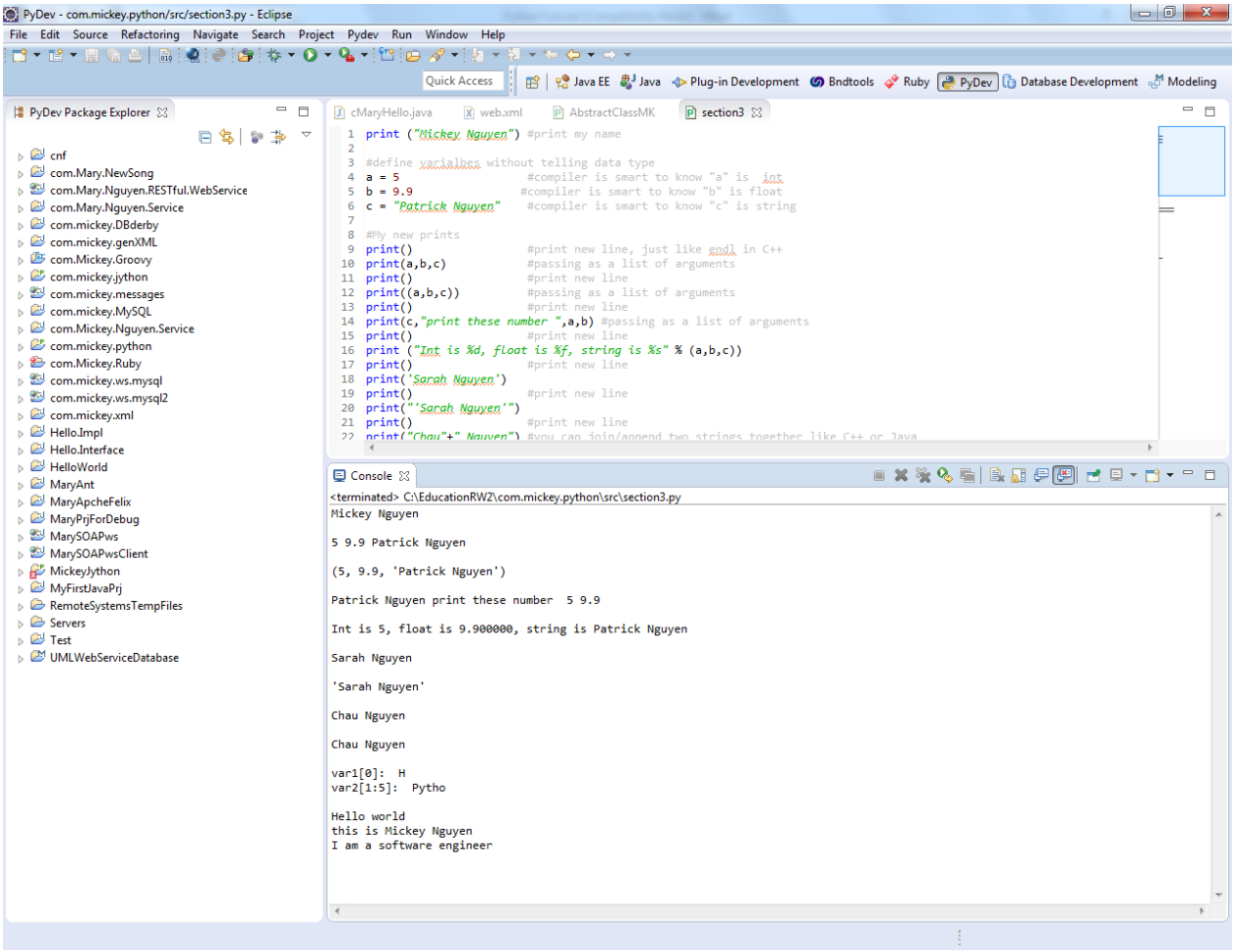
```
print ("Mickey Nguyen") #print my name

#define varialbes without telling data type
a = 5                    #compiler is smart to know "a" is  int
b = 9.9                 #compiler is smart to know "b" is float
```

```
c = "Patrick Nguyen"    #compiler is smart to know "c" is string

#My new prints
print()                 #print new line, just like endl in C++
print(a,b,c)            #passing as a list of arguments
print()                 #print new line
print((a,b,c))          #passing as a list of arguments
print()                 #print new line
print(c,"print these number ",a,b) #passing as a list of arguments
print()                 #print new line
print ("Int is %d, float is %f, string is %s" % (a,b,c))
print()                 #print new line
print('Sarah Nguyen')
print()                 #print new line
print("'Sarah Nguyen'")
print()                 #print new line
print("Chau"+" Nguyen") #you can join/append two strings together like C++ or Java
print()                 #print new line
print('Chau'+' Nguyen') #single ' ', is also work for string also, not like you use
for character in C/C++
#playing with string
print()
var1 = 'Hello World!'
var2 = "Python Programming"

#print first index 0
print ("var1[0]: ", var1[0])
#print index 0 to 5
print ("var2[1:5]: ", var2[0:5])

# see how to write a block of comments
""" this is a very long comments
on and on
on and on
end of long comments """

print() # new line
# or define a long string in multiple lines
a = """Hello world
this is Mickey Nguyen
I am a software engineer """
print(a)
```

To run it, right click on your script->Run As->Python Run


Bndtools - com.mickey.python/src/section3.py - Eclipse
File Edit Source Refactoring Source Refactor Navigate Search Project Bndtools Run Window Help
Quick Access
Java EE Java Plug-in Development Bndtools Ruby PyDev Database Development Modeling
Package Explorer
Type Hierarchy
cMaryHello.java
web.xml
AbstractClassMK
section3
Outline
type filter text
a
b
c
var1
var2
a
print ("Mickey Nguyen") #print my name
#define variables without telling data type
a = 5 #compiler is smart to know "a" is int
b = 9.9 #compiler is smart to know "b" is float
c = "Patrick Nguyen" #compiler is smart to know "c" is string
#My new prints
print() #print new line, just like endl in C++
print(a,b,c) #passing as a list of arguments
print()
print((a,b,c)) #passing as a list of arguments
#print new line
these number ",a,b) #passing as a list of arguments
#print new line
%d, float is %f, string is %s" % (a,b,c))
#print new line
guyen')
Imports/Exports
Console
nRW2\com.mickey.python\src\section3.py
guyen')
t these number 5 9.9
9.900000, string is Patrick Nguyen
FindFindAll.py
Forloop.py
FromModuleImport.py
Funcs.py
FunctionReturn.py
hellomickey.class
IfElse.py
info.txt
New
Open F3
Open With
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Mark as Landmark Ctrl+Alt+Shift+Up
Build Path
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Assign Working Sets...
Validate
Show in Remote Systems view
Profile As
Debug As
Run As
Team
Replace With
PyDev
Acceleo
JPA Tools
Compare With
Properties Alt+Enter
1 Jython Run
2 Jython unit-test
3 Python Run
4 Python unit-test
Run Configurations...
Repositories
Enter search strin
Workspac
section3.py - com.
8:32 AM
8/27/2015

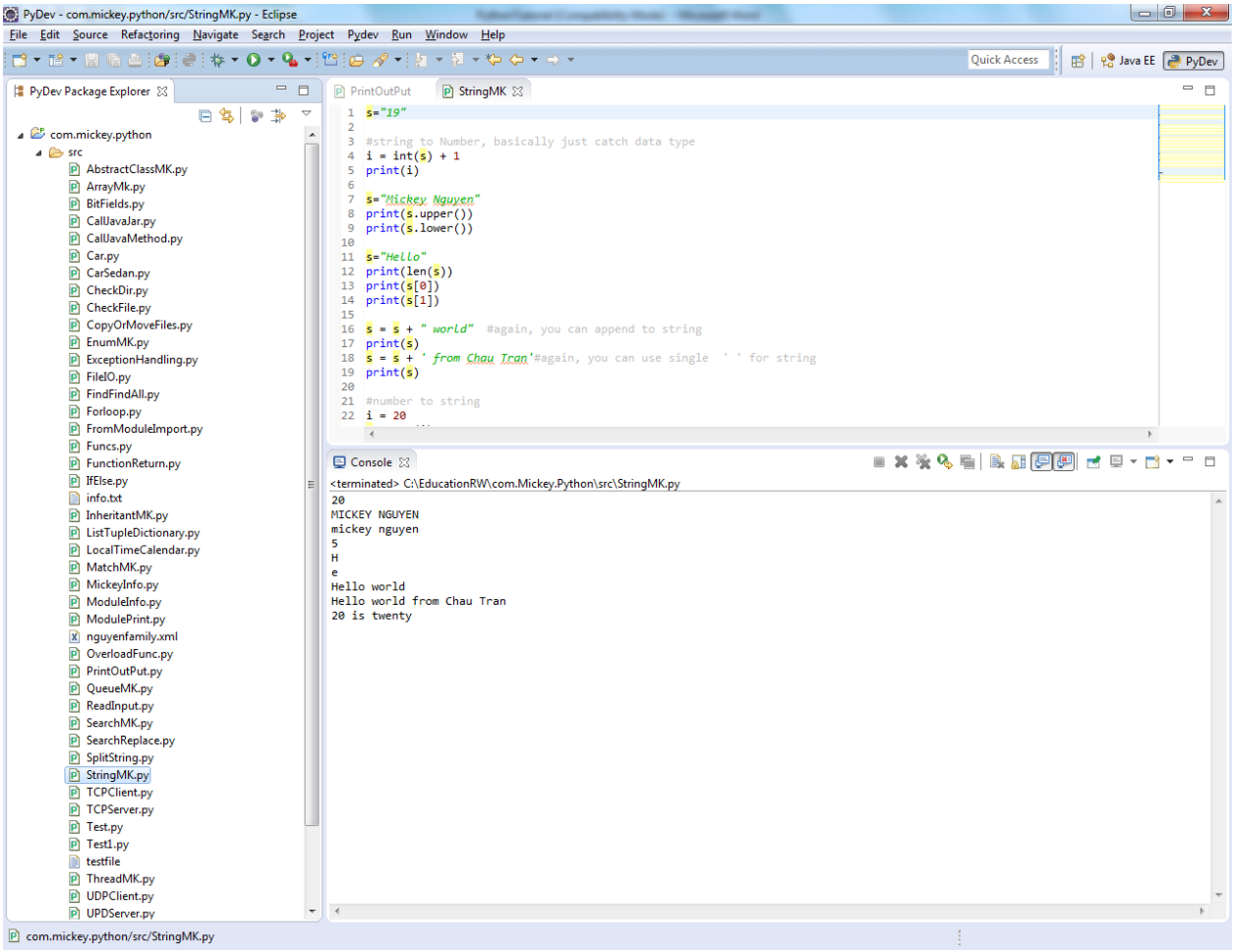
Output:



4 String

Create another script name, so you don't have to override your first one. You can see under my src I have many scripts

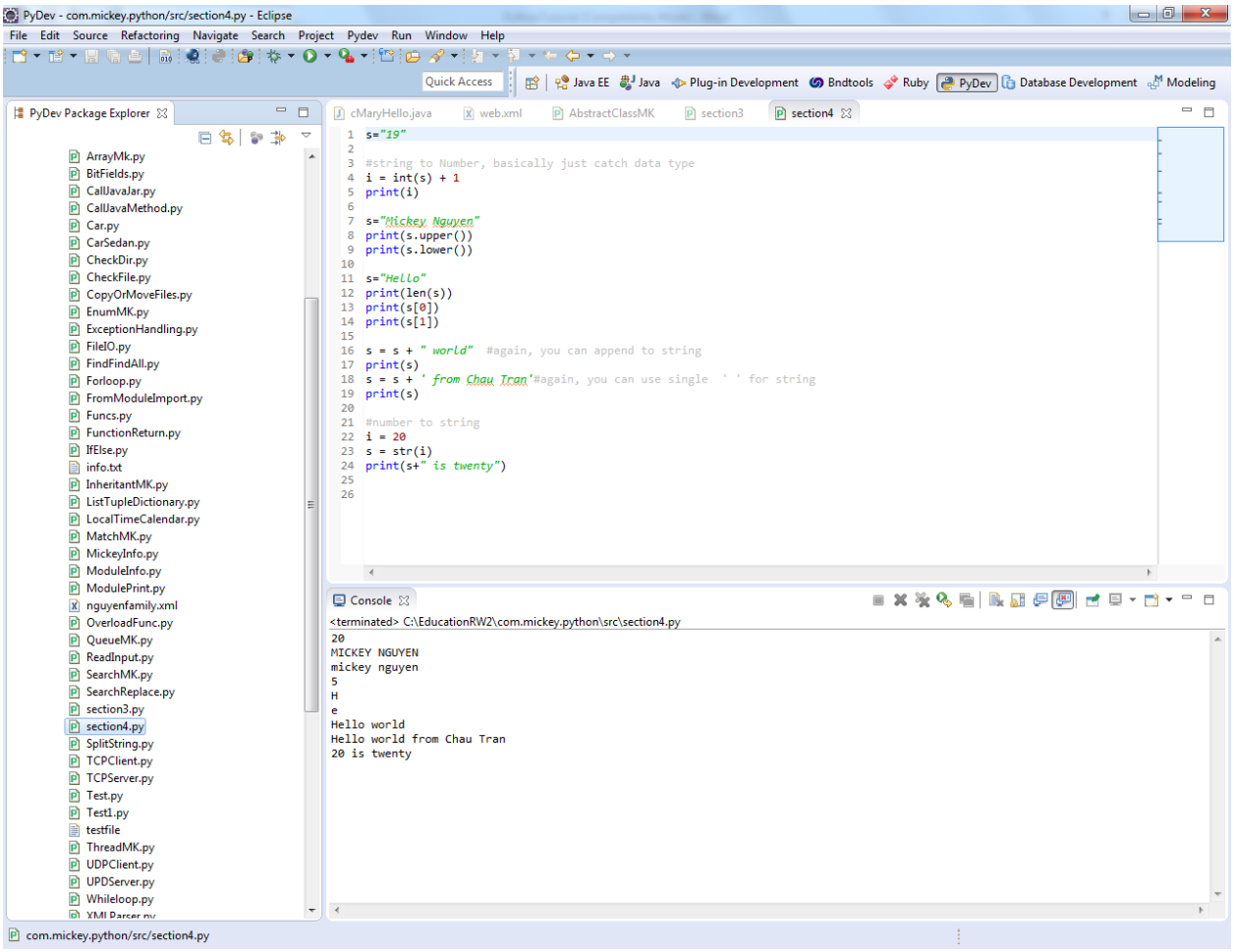
section4.py




```
s="19"

#string to Number, basically just catch data type
i = int(s) + 1
print(i)

s="Mickey Nguyen"
print(s.upper())
print(s.lower())

s="Hello"
print(len(s))
print(s[0])
print(s[1])

s = s + " world"  #again, you can append to string
print(s)
s = s + ' from Chau Tran'#again, you can use single  ' ' for string
print(s)

#number to string
i = 20
s = str(i)
```

```
print(s+" is twenty")
```

output:



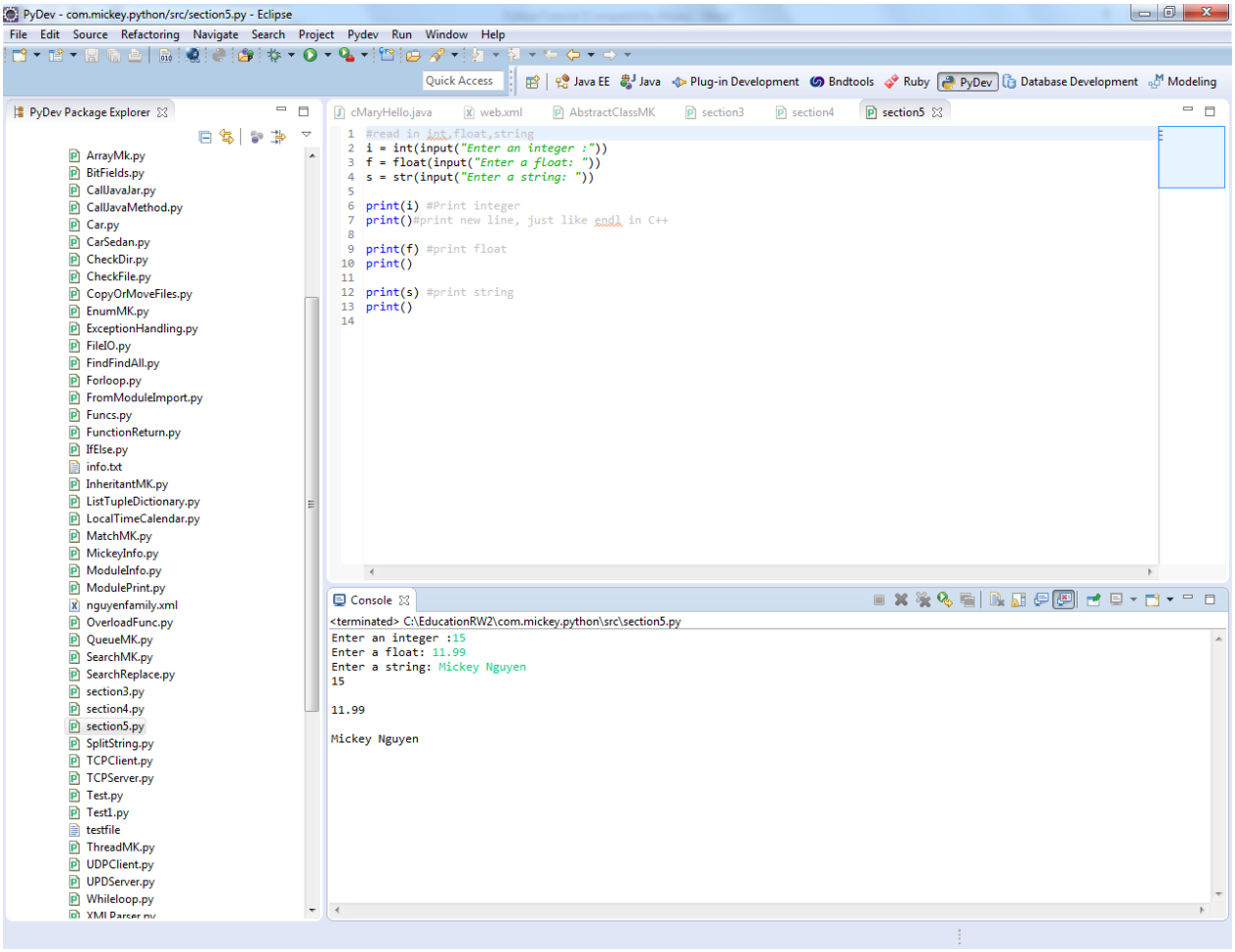
5 Read input then print it out

section5.py

```
#read in int,float,string
i = int(input("Enter an integer :"))
f = float(input("Enter a float: "))
s = str(input("Enter a string: "))

print(i) #Print integer
print()#print new line, just like endl in C++

print(f) #print float
```

```
print()

print(s) #print string
print()
```

Output:



6 While loop

Note: Indentation is very important in this language; In other languages like C/C++ or Java you can have wrong indentation but you cannot have wrong indentation in this language. So, please be careful about it.

Syntax:

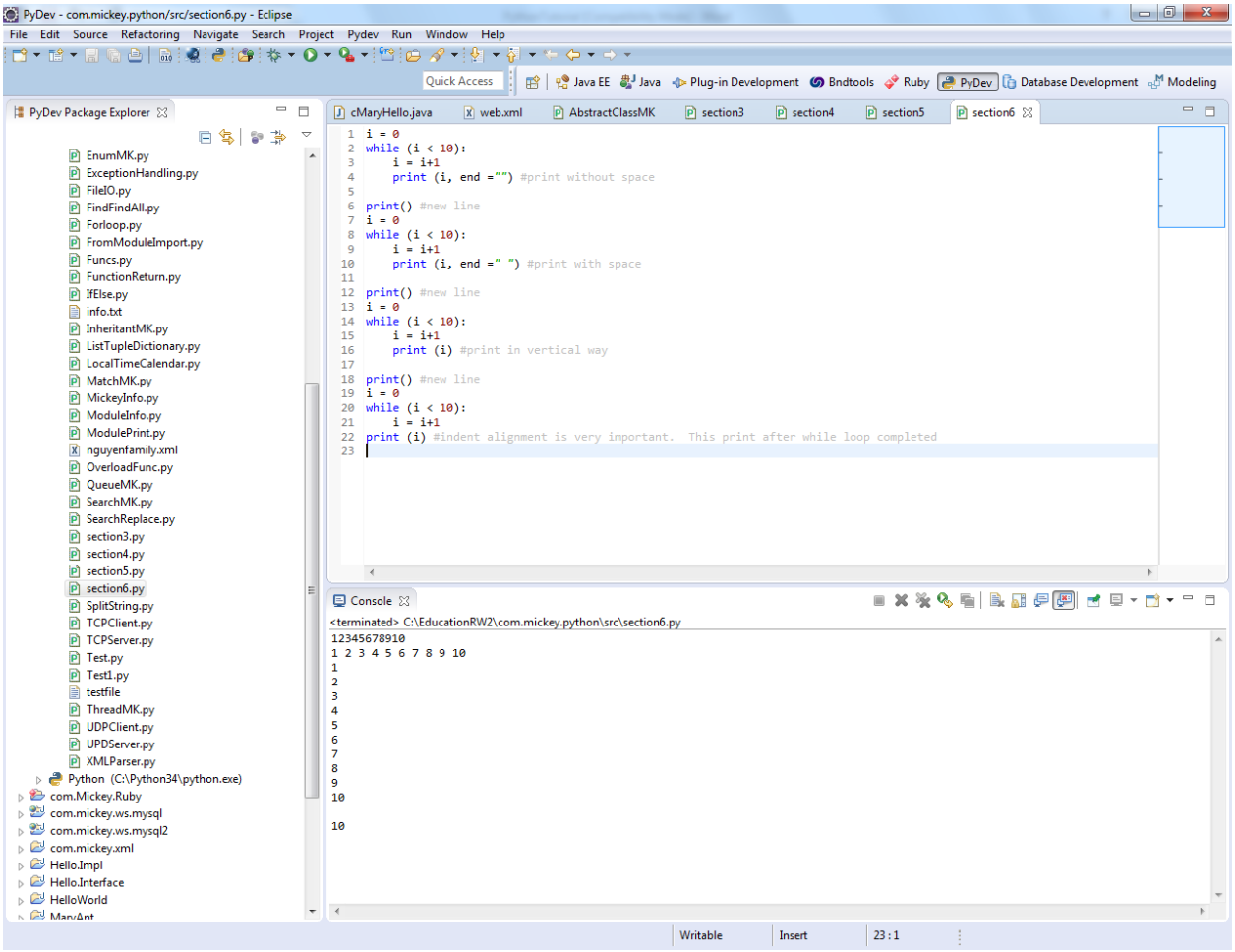
while(condition):

```
    #inside while loop
    #inside while loop
    #still inside while loop
#indent back to the same level of while loop to get out of while loop
```

section6.py

```
i = 0
while (i < 10):
    i = i+1
    print (i, end ="") #print without space

print() #new line
i = 0
while (i < 10):
    i = i+1
    print (i, end =" ") #print with space

print() #new line
i = 0
while (i < 10):
    i = i+1
    print (i) #print in vertical way

print() #new line
i = 0
while (i < 10):
    i = i+1
print (i) #indent alignment is very important.  This print after while loop completed
```

Output:



7 For loop

Syntax:

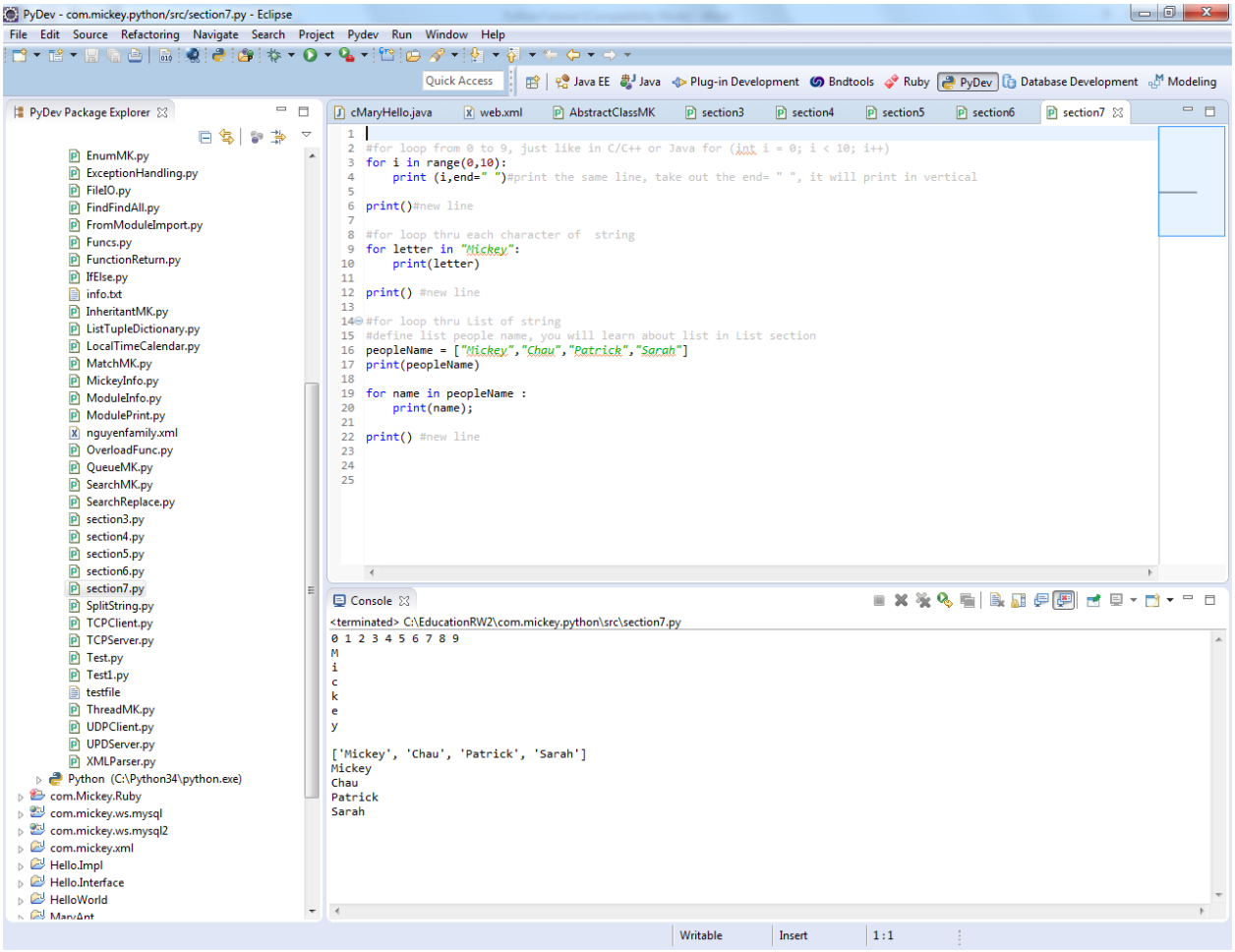
for item in range (start index, stop index)
for item in string/list/enum/…

section7.py

```
#for loop from 0 to 9, just like in C/C++ or Java for (int i = 0; i < 10; i++)
for i in range(0,10):
```

```
    print (i,end=" ")#print the same line, take out the end= " ", it will print in
vertical

print()#new line

#for loop thru each character of  string
for letter in "Mickey":
    print(letter)

print() #new line

#for loop thru List of string
#define list people name, you will learn about list in List section
peopleName = ["Mickey","Chau","Patrick","Sarah"]
print(peopleName)

for name in peopleName :
    print(name);

print() #new line
```

Output:



8 Function

Syntax:

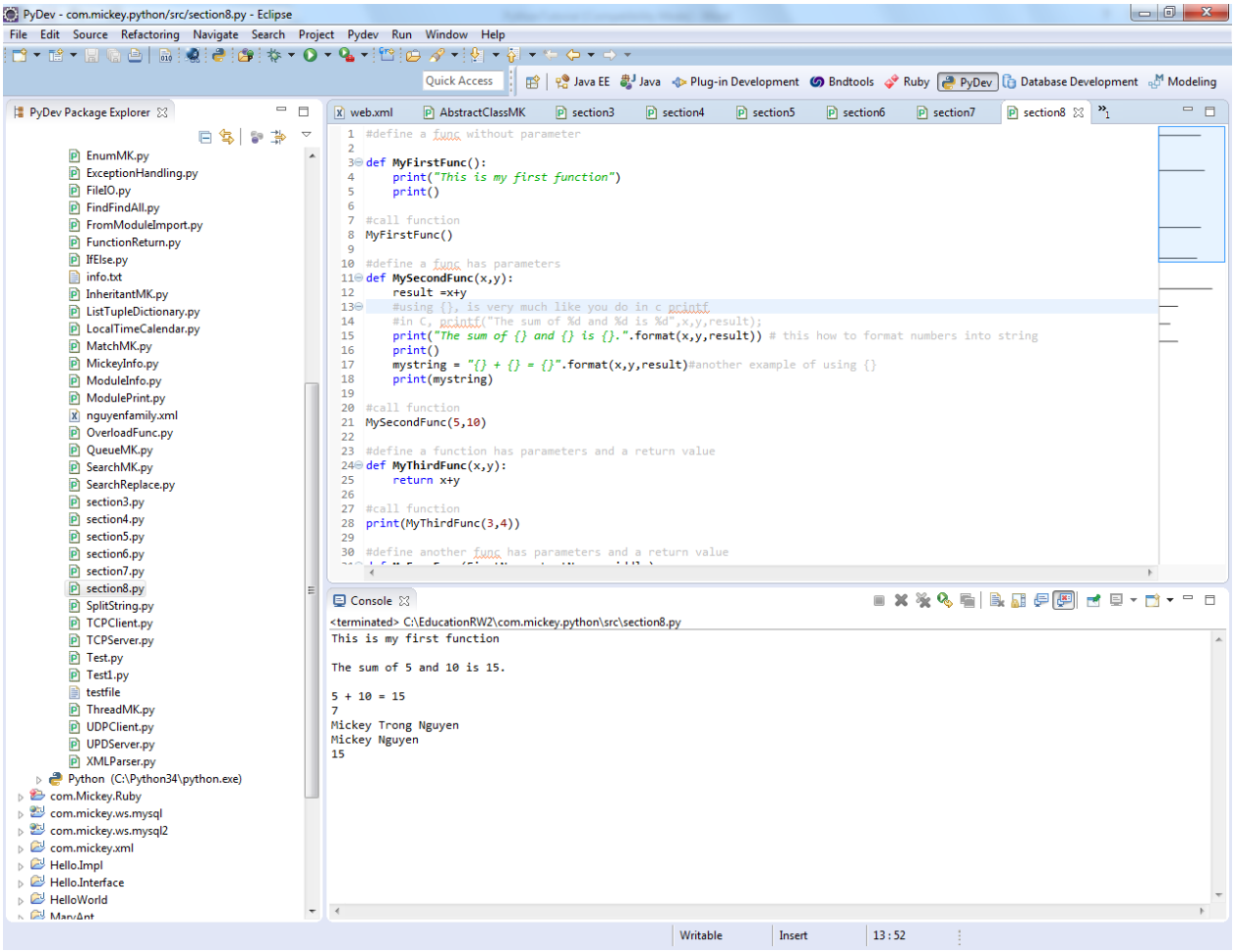
def functionName(): #function without parameter
def functionNanme(par1,par2,..,parn) #function with parameters

section8.py

```
#define a func without parameter

def MyFirstFunc():
    print("This is my first function")
    print()

#call function
MyFirstFunc()
```

```
#define a func has parameters
def MySecondFunc(x,y):
    result =x+y
    #using {}, is very much like you do in c printf
    #in C, printf("The sum of %d and %d is %d",x,y,result);
    print("The sum of {} and {} is {}.".format(x,y,result)) # this how to format
numbers into string
    print()
    mystring = "{} + {} = {}".format(x,y,result)#another example of using {}
    print(mystring)

#call function
MySecondFunc(5,10)

#define a function has parameters and a return value
def MyThirdFunc(x,y):
    return x+y

#call function
print(MyThirdFunc(3,4))

#define another func has parameters and a return value
def MyFourFunc(FirstName, LastName, middle):
    myname = FirstName + " "+ middle + " "+ LastName
    return myname

print(MyFourFunc("Mickey","Nguyen","Trong"))

#Function return value
def FunctionReturn(someStr):
    strValue = someStr
    return strValue
#call function
value = FunctionReturn("Mickey Nguyen")
print(value)

#Function return value
def Sum (a,b):
    return a+b

#call function
total = Sum(5,10)
print(total)
```

Output:



9 Overloading function

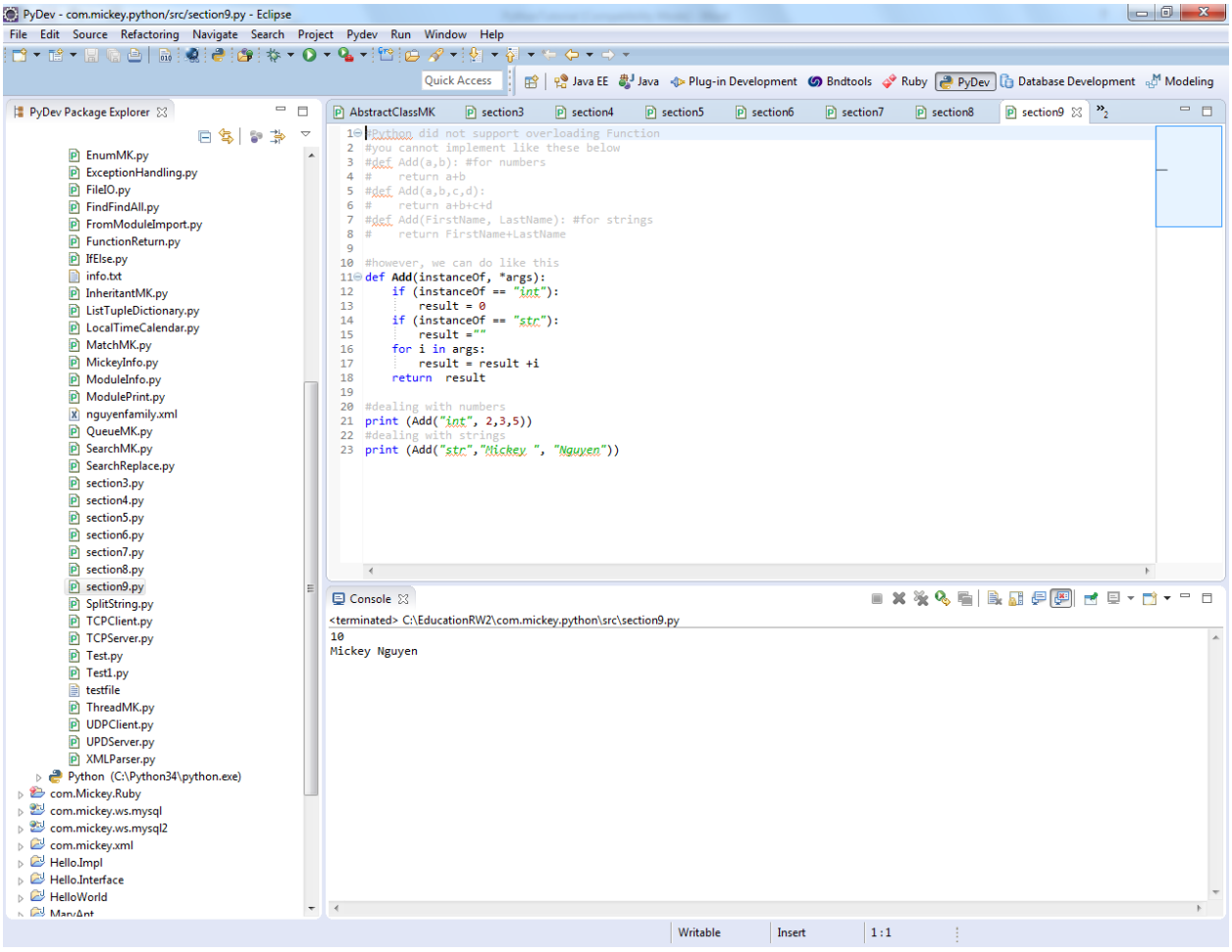
Python did not support overloading function but we in some case, if the parameters are same data type, then wecan work around like this

```
#Python did not support overloading Function
#you cannot implement like these below
#def Add(a,b): #for numbers
#    return a+b
#def Add(a,b,c,d):
#    return a+b+c+d
#def Add(FirstName, LastName): #for strings
#    return FirstName+LastName

#however, we can do like this
```

section9.py

```
def Add(instanceOf, *args):
    if (instanceOf == "int"):
        result = 0
    if (instanceOf == "str"):
        result =""
    for i in args:
        result = result +i
    return  result

#dealing with numbers
print (Add("int", 2,3,5))
#dealing with strings
print (Add("str","Mickey ", "Nguyen"))
```

Output:



10 if-elif-else

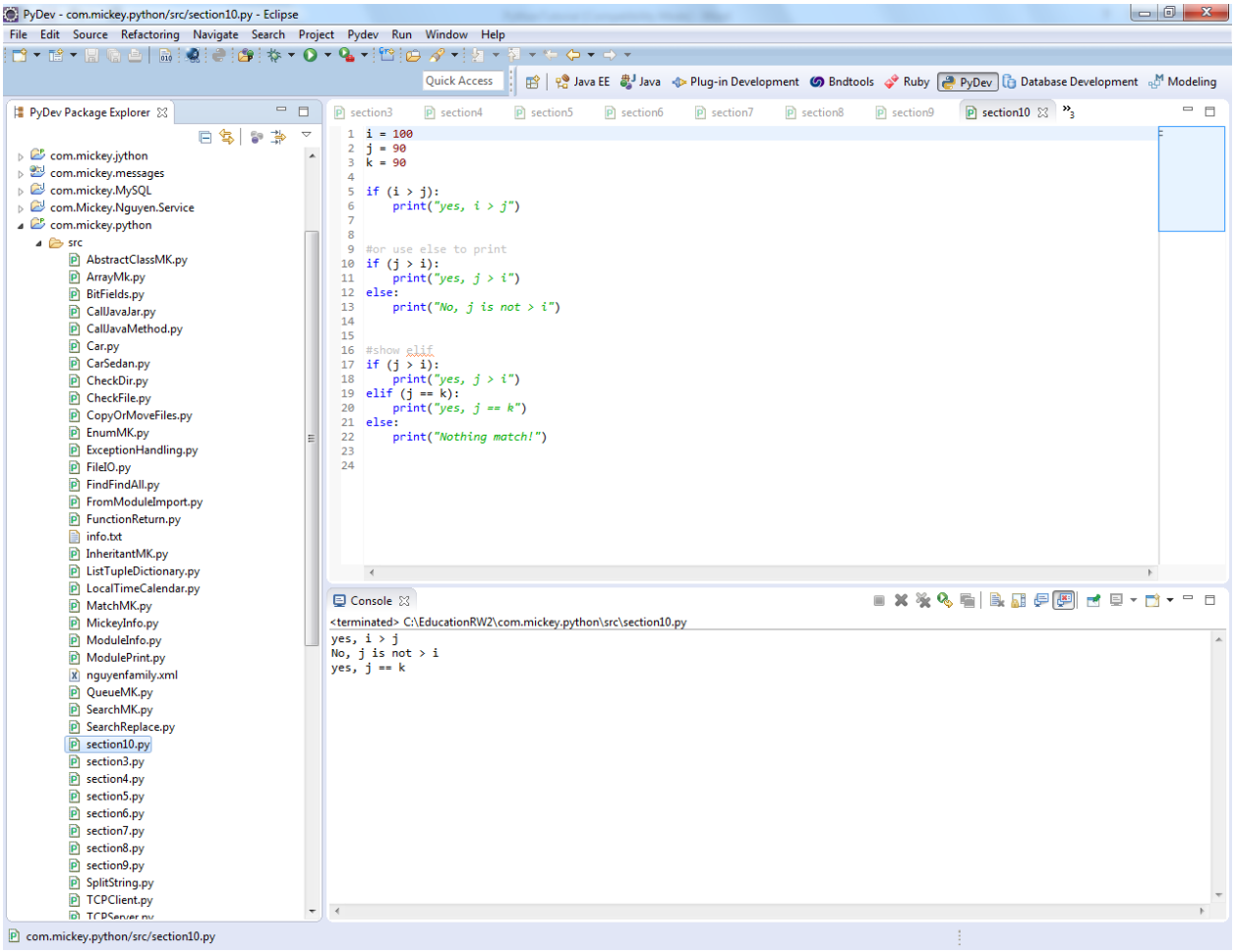
Syntax:

if ():
elif ():
else():

section10.py

```
i = 100
j = 90
k = 90

if (i > j):
    print("yes, i > j")


#or use else to print
if (j > i):
    print("yes, j > i")
else:
     print("No, j is not > i")


#show elif
if (j > i):
    print("yes, j > i")
elif (j == k):
     print("yes, j == k")
else:
    print("Nothing match!")
```

Output:



11 Switch-case

Python did not have switch-case. You nested if-elif-else instead.

12 Enum

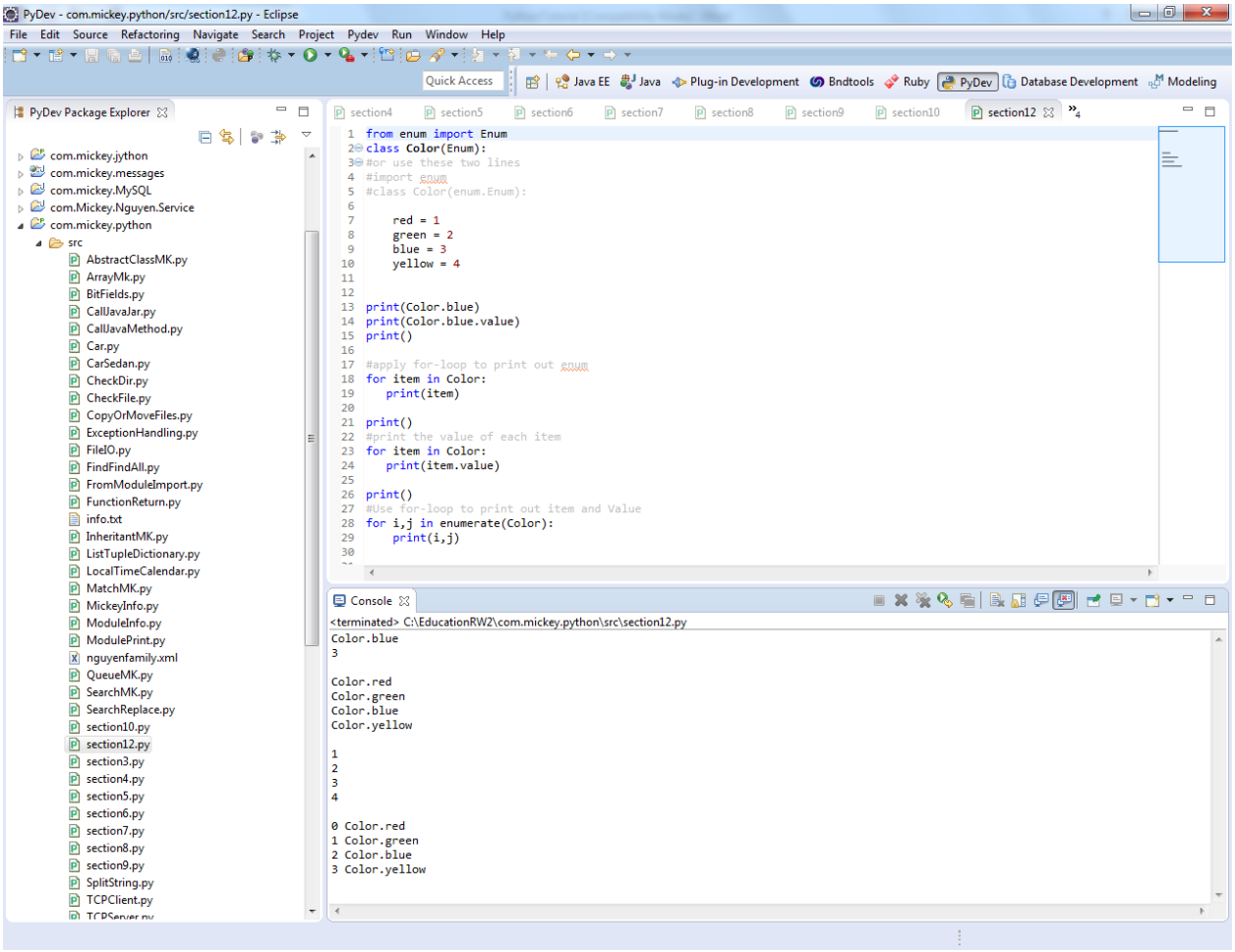
Syntax:

class enumName (Enum):

section12.py

```
from enum import Enum
```

```
class Color(Enum):
#or use these two lines
#import enum
#class Color(enum.Enum):

    red = 1
    green = 2
    blue = 3
    yellow = 4


print(Color.blue)
print(Color.blue.value)
print()

#apply for-loop to print out enum
for item in Color:
   print(item)

print()
#print the value of each item
for item in Color:
   print(item.value)

print()
#Use for-loop to print out item and Value
for i,j in enumerate(Color):
    print(i,j)


Output:
```



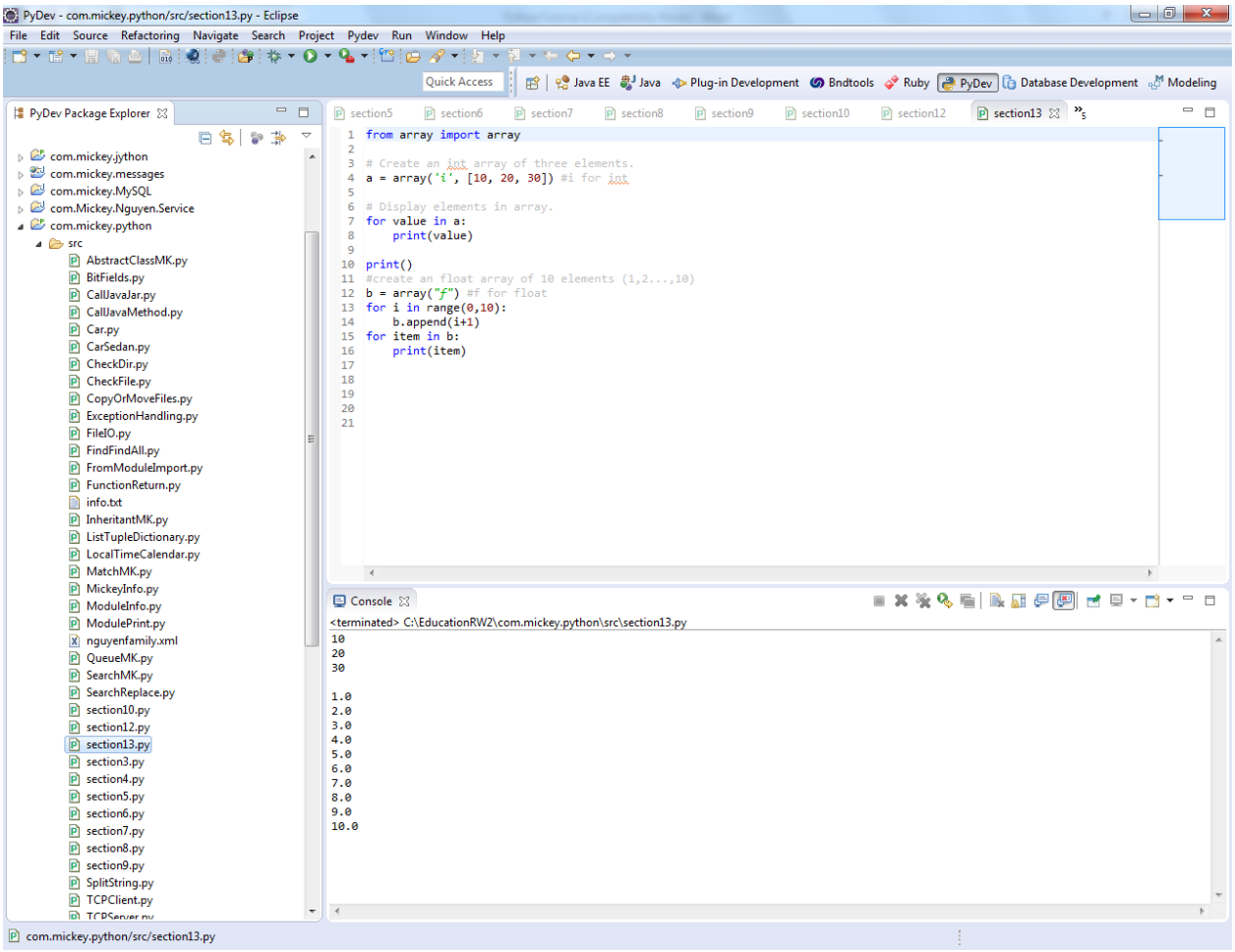
13 Array

section13.py

```
from array import array

# Create an int array of three elements.
a = array('i', [10, 20, 30]) #i for int

# Display elements in array.
for value in a:
    print(value)

print()
#create an float array of 10 elements (1,2...,10)
b = array("f") #f for float
for i in range(0,10):
    b.append(i+1)
```

```
for item in b:
    print(item)
```

Output:



14 List-Tuple-Dictionary

If you know array in C/C++, List and Tuple are container just like array, you can store data in each index of List and Tuple.

In array, you have to specify the array size. Also, you have only and only one type of data type in your array (int, or float, or string,or ….)

In List and Tuple, the size can grow dynamically, you don't have to specify the size. Also, you can mix data type of each element in List and Tuple. For example, first index, you store int, second index, you store float, third index, you store string.

Dictionary is just like a hash table with a pair of “key” and “value”. Just like in C++ we have “map”, in Java we have Dictionary as well.

Note: You can mix all kind of data types of element s on the List and Tuple. You also can mix any key/value data type on Dictionary

myList =[“mickey”,50,189.99,”Male”]
myTuple = (“mickey”,50,189.99,”Male”)
myDic = {“name”:”mickey,”age”:50,”eeight”:189.99,”gender”:”Male”}

Syntax:

listName =[]
tupleName = ()
dictionaryName = {}

List vs Tuple.

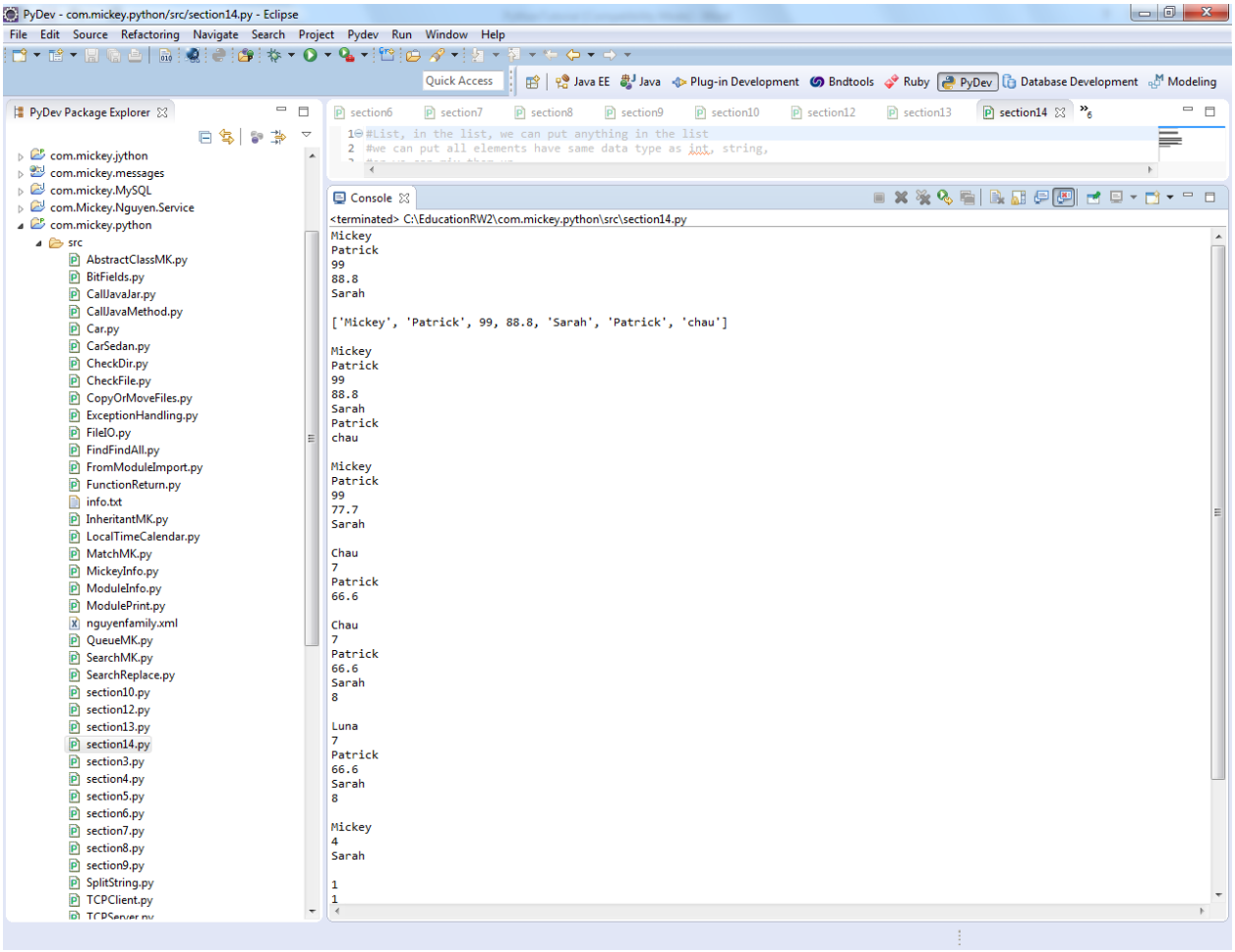
- Element in the list can be updated
- Element in tuple cannot be updated. It is read only. It can work around by convert to list, update it then convert it back to tuple.

section14.py

```
#List, in the list, we can put anything in the list
#we can put all elements have same data type as int, string,
#or we can mix them up
list1 = [] #empty list
list1.append("Mickey") # or use list1.insert(0,"Mickey")
list1.append("Patrick")
list1.append(99)
#use insert, need to tell which index to be inserted
list1.insert(3,88.8)
list1.append("Sarah")

for item in list1:
    print (item)

print()
list2 = ['Patrick', "chau"]
print (list1 + list2)

#combine two lists into new list
list3 = list1 + list2
print()
for item in list3:
    print (item)
```

```
#update index 3
list1[3] =77.7
#reprint
print()
for item in list1:
    print (item)

#Tuple
tuple1 = ("Chau", 7, 'Patrick', 66.6)
print()
for item in tuple1:
    print(item)

tuple2 = ("Sarah",8)
#combine two tuples into new tuple
tuple3 = tuple1 + tuple2
print()
for item in tuple3:
    print(item)

#tuple3[0]= "Luna" #cannot update value of index 0.
#solution is convert tuple to list, update it, then convert back to tuple
list4 = list(tuple3)
list4[0] = "Luna"
tuple3 = tuple(list4)
print()
for item in tuple3:
    print(item)


#Dictionary
dic = {}
dic[1] = "Mickey"
dic["Patrick"] = 4
dic["1"] = "Sarah"
print()
print(dic.get(1))
print(dic.get("Patrick"))
print(dic.get("1"))

#automaticall loop thru, the order is unexpected
print()
for item in dic.keys():
    print(item)


print()
for item in dic.keys():
    print(item)
    print(dic.get(item))
```

output:

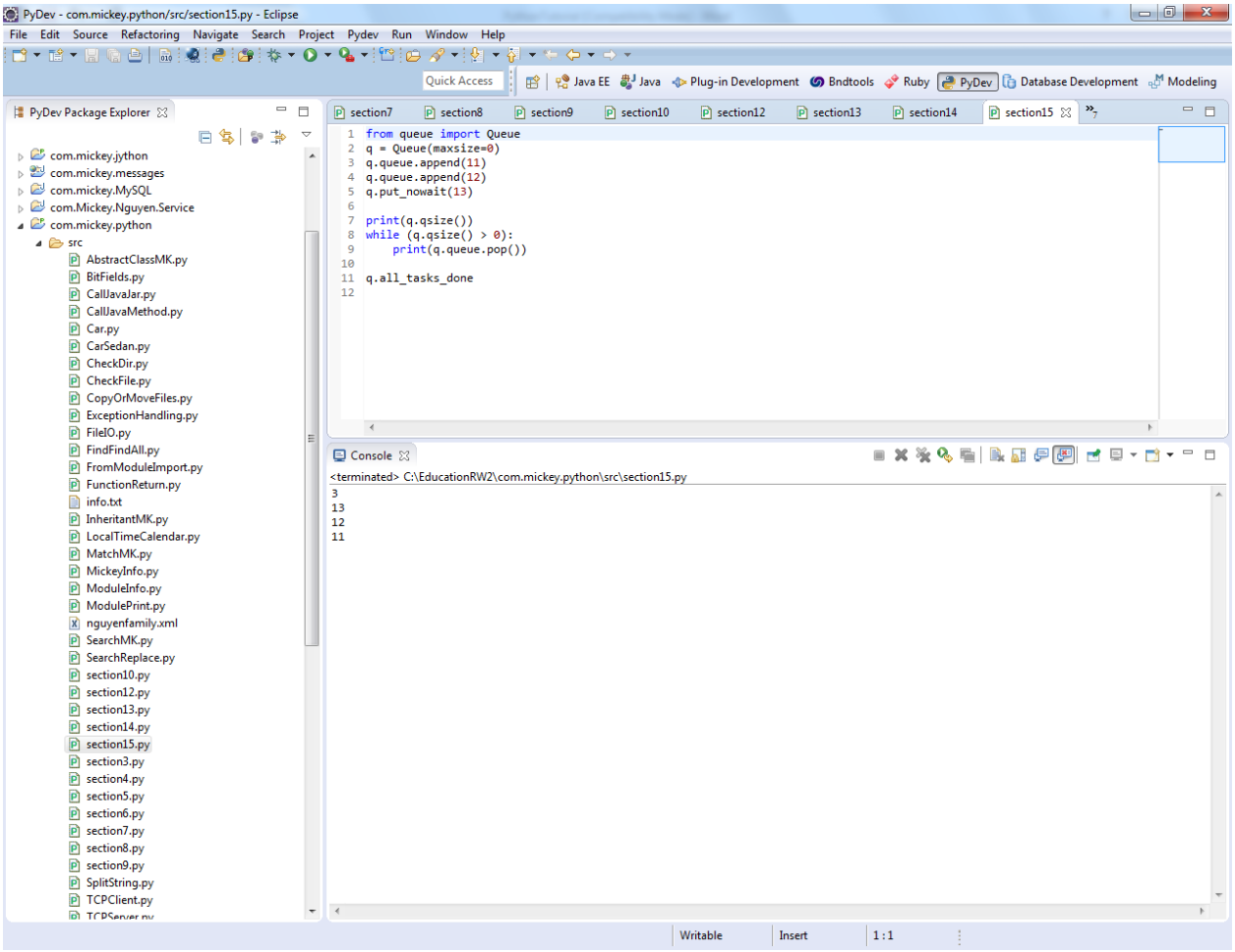


15 Queue

section15.py

```
from queue import Queue
q = Queue(maxsize=0)
q.queue.append(11)
q.queue.append(12)
q.put_nowait(13)

print(q.qsize())
while (q.qsize() > 0):
    print(q.queue.pop())

q.all_tasks_done
```

Output:



16 Modules

Each module.py file can be treated as one module. Other .py file want to use APIs of other .py file, all it needs to do is import that module name.

section16A.py

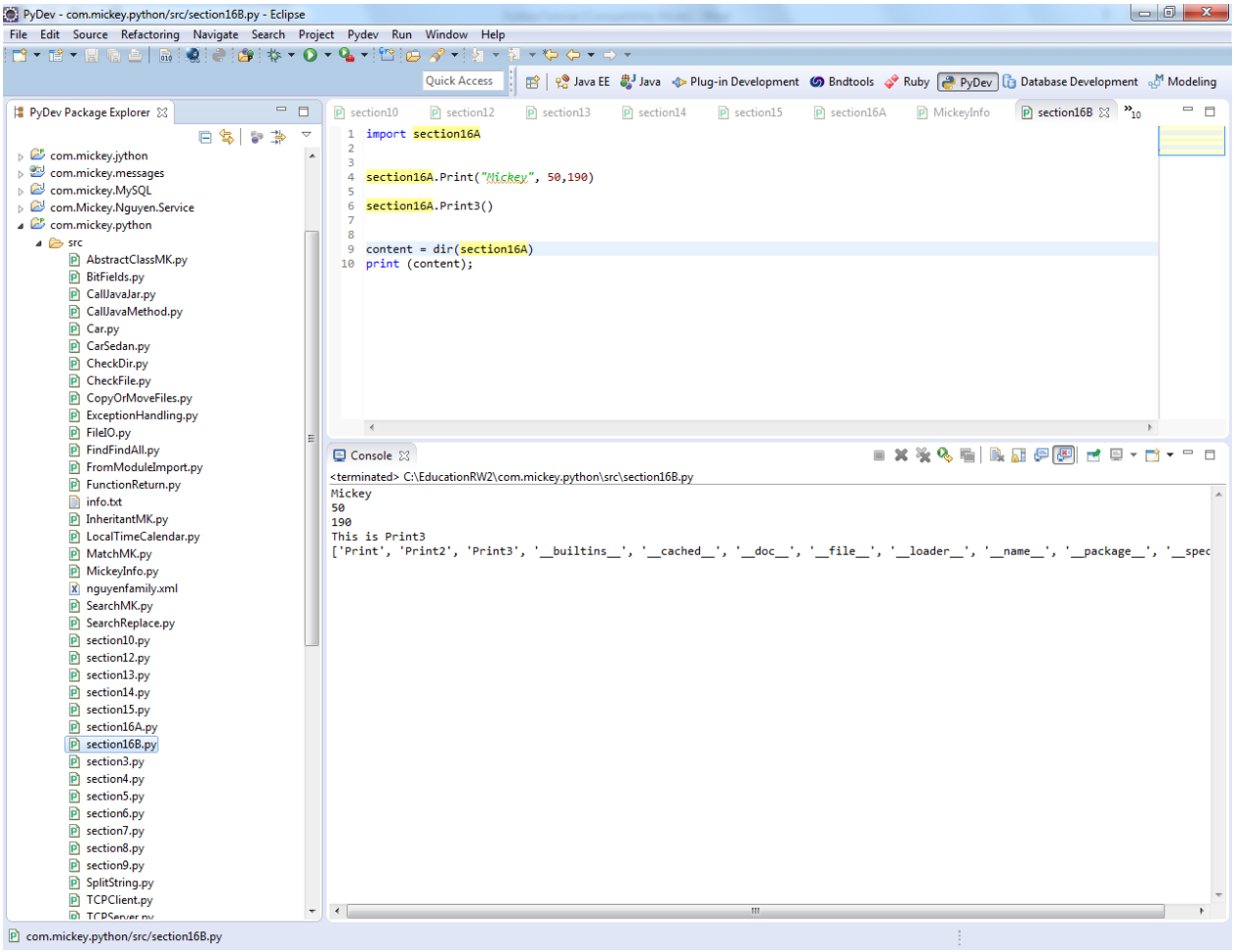
```
def Print(name, age, weigh):
    print(name)
    print(age)
    print(weigh)
    return #without this ok, too

def Print2():
    print("This is Print2")
```

```
def Print3():
    print("This is Print3")
```

This section16B.py try to use section16A.py

```
import section16A


section16A.Print("Mickey", 50,190)

section16A.Print3()


content = dir(section16A)
print (content);
```

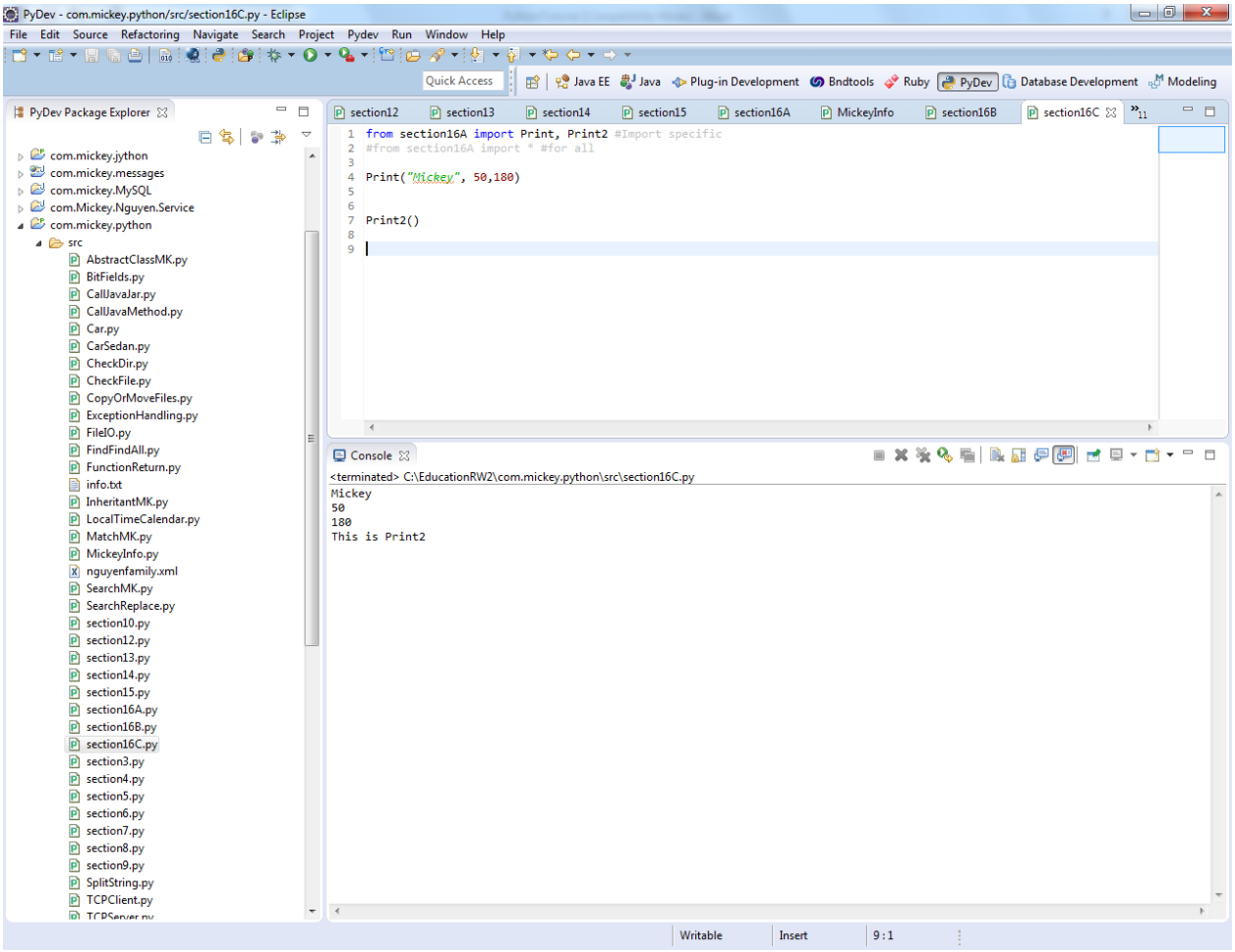
Output from the run of section16B.py:



16.1 Use from module import

This From section16C.py try to import specific thing from section16A

```
from section16A import Print, Print2 #Import specific
#from section16A import * #for all

Print("Mickey", 50,180)


Print2()
```



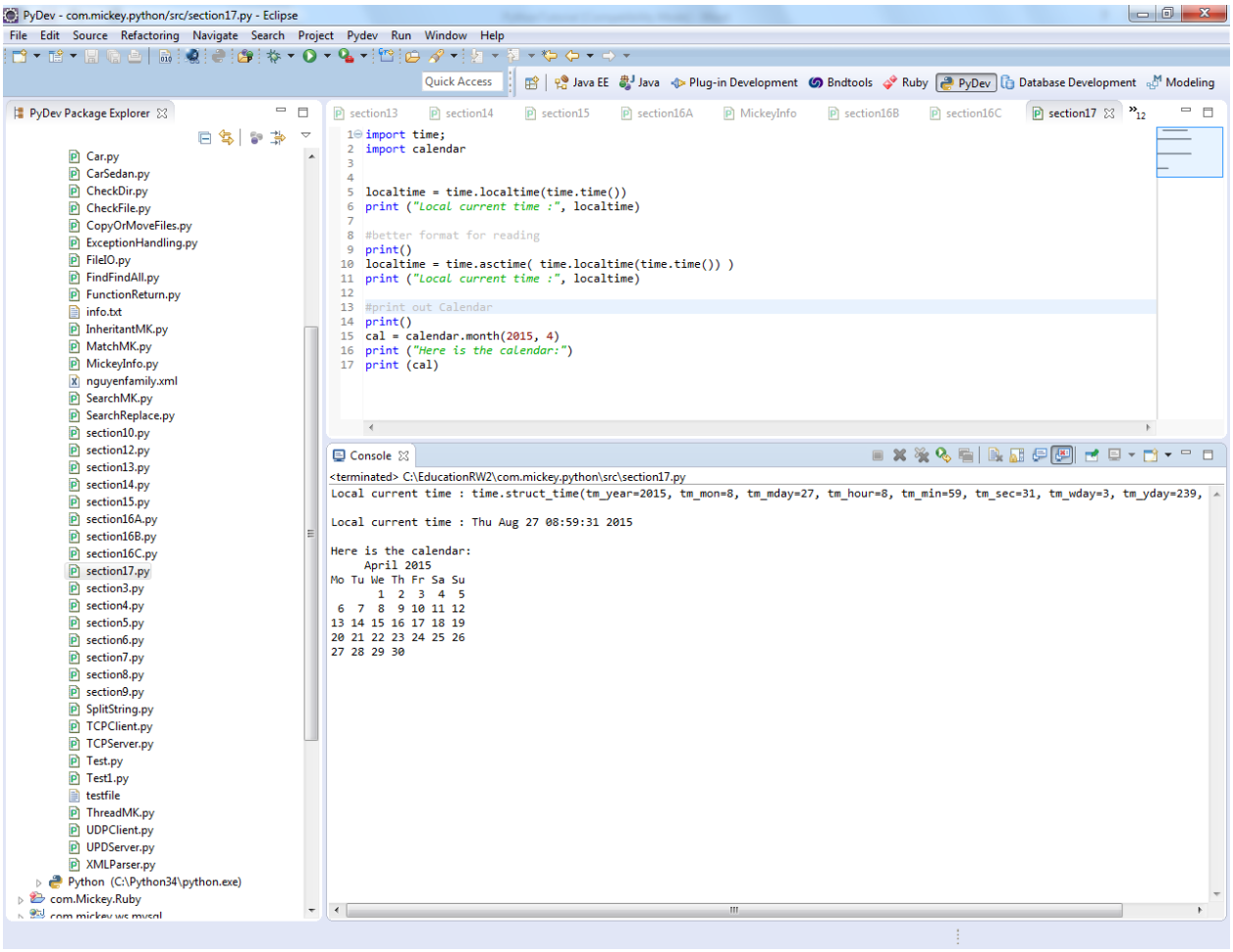
17 Use local time and calendar

section17.py

```
import time;
import calendar


localtime = time.localtime(time.time())
print ("Local current time :", localtime)

#better format for reading
print()
localtime = time.asctime( time.localtime(time.time()) )
print ("Local current time :", localtime)

#print out Calendar
print()
```

```
cal = calendar.month(2015, 4)
print ("Here is the calendar:")
print (cal)
```

Output:



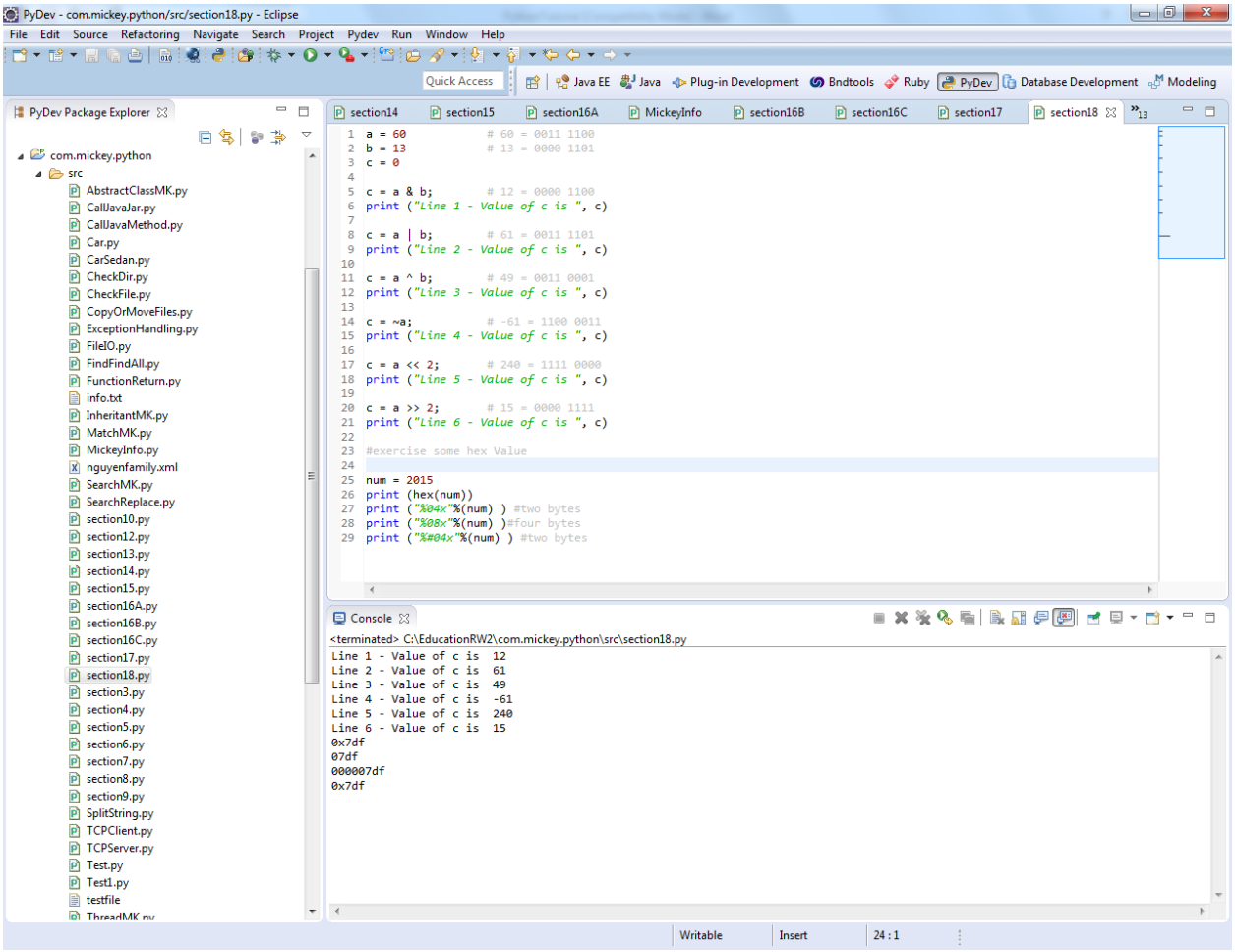
18 Bitwise

section18.py

```
a = 60            # 60 = 0011 1100
b = 13            # 13 = 0000 1101
c = 0

c = a & b;        # 12 = 0000 1100
print ("Line 1 - Value of c is ", c)

c = a | b;        # 61 = 0011 1101
```

```
print ("Line 2 - Value of c is ", c)

c = a ^ b;         # 49 = 0011 0001
print ("Line 3 - Value of c is ", c)

c = ~a;            # -61 = 1100 0011
print ("Line 4 - Value of c is ", c)

c = a << 2;        # 240 = 1111 0000
print ("Line 5 - Value of c is ", c)

c = a >> 2;        # 15 = 0000 1111
print ("Line 6 - Value of c is ", c)

#exercise some hex Value

num = 2015
print (hex(num))
print ("%04x"%(num) ) #two bytes
print ("%08x"%(num) )#four bytes
print ("%#04x"%(num) ) #two bytes
```

Output:

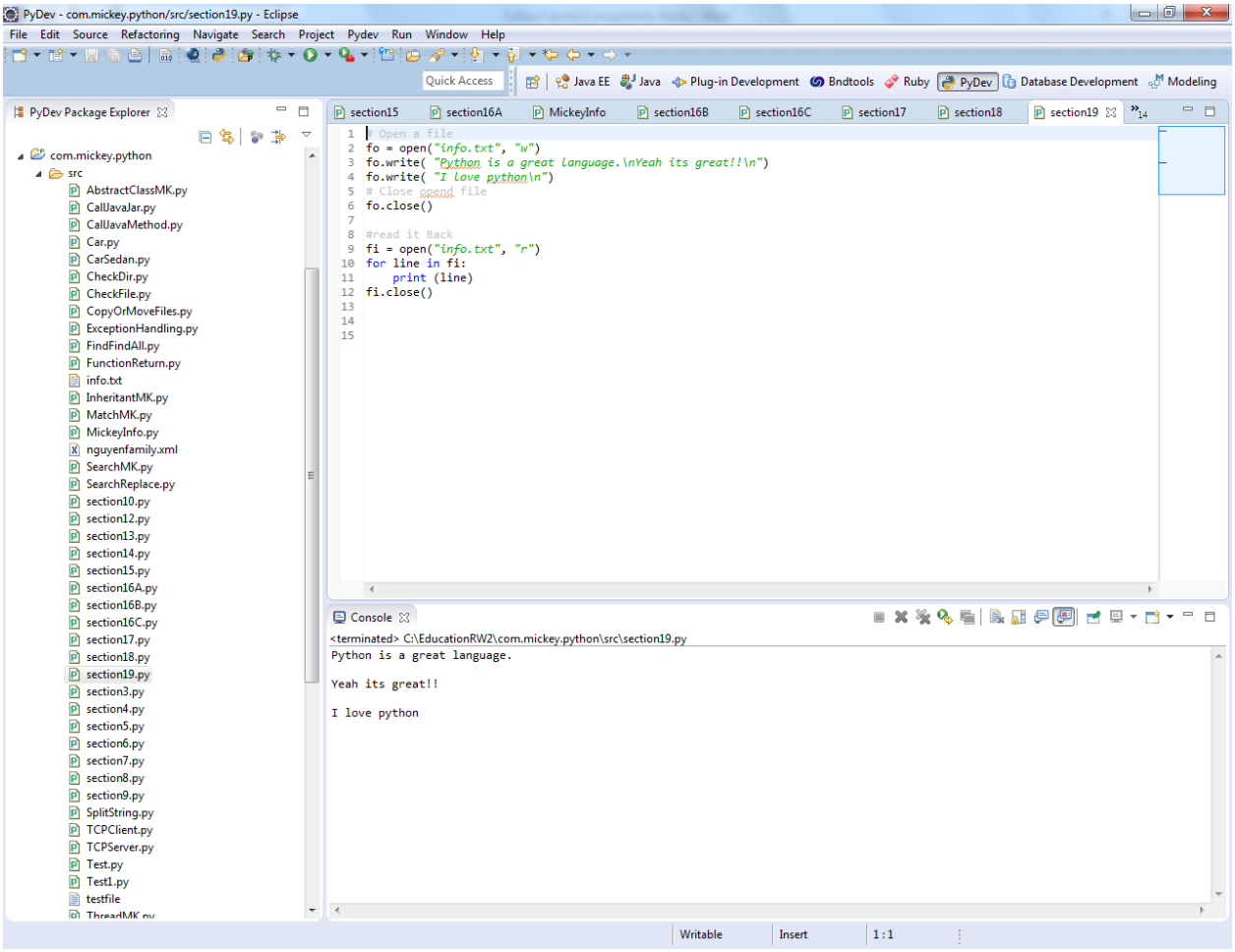


19 FileIO

section19.py

```
# Open a file
fo = open("info.txt", "w")
fo.write( "Python is a great language.\nYeah its great!!\n")
fo.write( "I love python\n")
# Close opend file
fo.close()

#read it Back
fi = open("info.txt", "r")
for line in fi:
    print (line)
fi.close()
```

Output:



20 Class

__init__, is a constructor

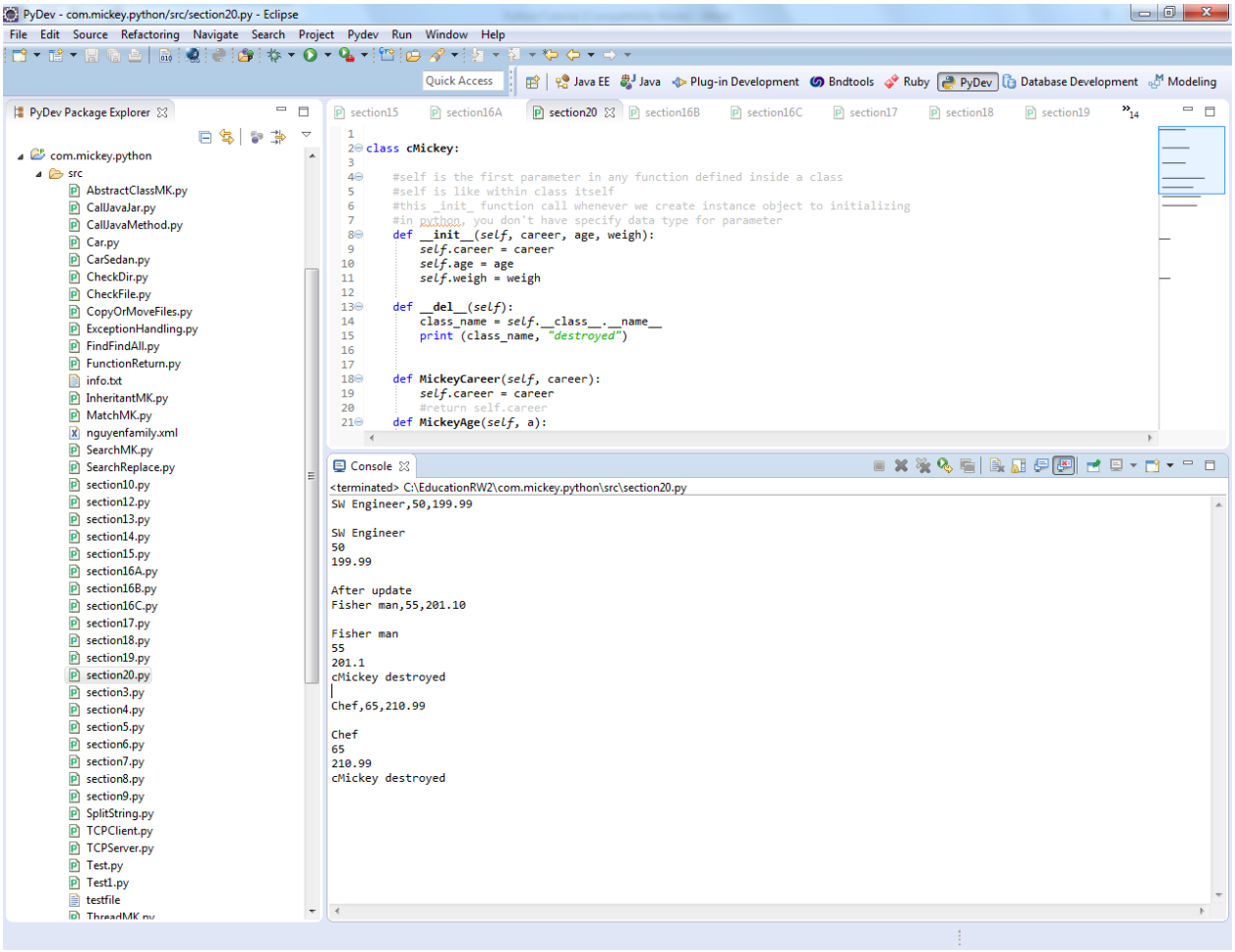
__del__, is a destructor. You don't have to implement destructor.

`section20.py`

```
class cMickey:

    #self is the first parameter in any function defined inside a class
    #self is like within class itself
    #this _init_ function call whenever we create instance object to initializing
    #in python, you don't have specify data type for parameter
```

```
    def __init__(self, career, age, weigh):
        self.career = career
        self.age = age
        self.weigh = weigh

    def __del__(self):
        class_name = self.__class__.__name__
        print (class_name, "destroyed")


    def MickeyCareer(self, career):
        self.career = career
        #return self.career
    def MickeyAge(self, a):
        self.age = a
        #return self.age
    def MickeyWeigh (self, w):
        self.weigh = w
        #return self.weigh
    def MickeyInfo(self):
        print("%s,%d,%.02f" %(self.career,self.age,self.weigh))


        #another way to print
        print() #new line
        print(self.career)
        print(self.age)
        print(self.weigh)

#create instance mk1
mk1= cMickey("SW Engineer",50, 199.99)
mk1.MickeyInfo()
#now update new INFO
mk1.MickeyCareer("Fisher man")
mk1.MickeyAge(55)
mk1.MickeyWeigh(201.1)
#print new info after update
print() #new line
print("After update")
mk1.MickeyInfo()

del mk1 #call destructor
#create instance mk2
mk2 = cMickey("Chef",65, 210.99)
print() #new line
mk2.MickeyInfo()

del mk2 #call destructor
```

Output:



21 Inheritant

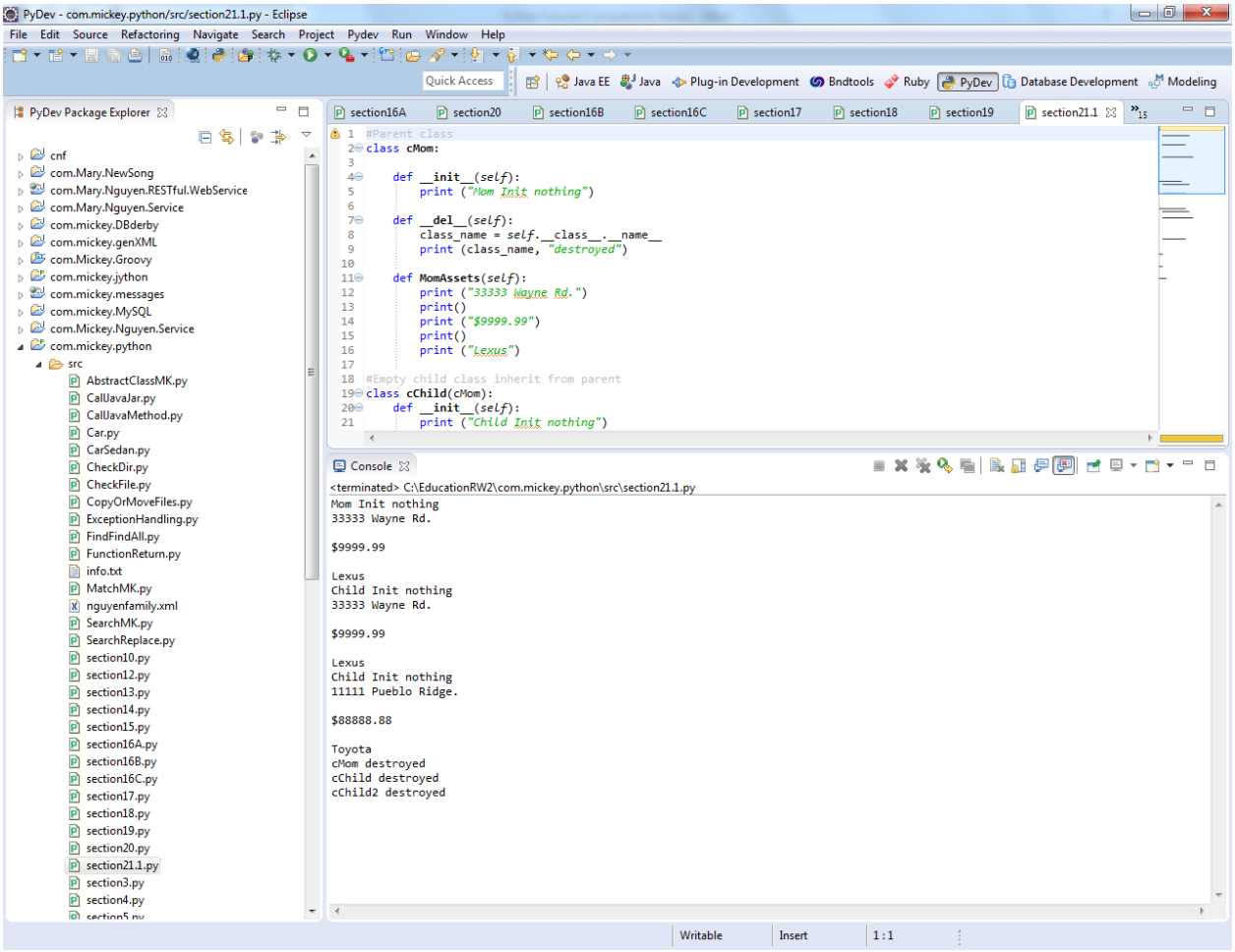
21.1 Child and Parent on same file

section21.1.py

```
#Parent class
class cMom:

    def __init__(self):
        print ("Mom Init nothing")

    def __del__(self):
        class_name = self.__class__.__name__
        print (class_name, "destroyed")

    def MomAssets(self):
        print ("33333 Wayne Rd.")
```

```
        print()
        print ("$9999.99")
        print()
        print ("Lexus")

#Empty child class inherit from parent
class cChild(cMom):
    def __init__(self):
        print ("Child Init nothing")

    def __del__(self):
        class_name = self.__class__.__name__
        print (class_name, "destroyed")

#child2 class inherit from parent, will override Parent MomAssets
class cChild2(cMom):
    def __init__(self):
        print ("Child Init nothing")
    def MomAssets(self):
        print ("11111 Pueblo Ridge.")
        print()
        print ("$88888.88")
        print()
        print ("Toyota")

    def __del__(self):
        class_name = self.__class__.__name__
        print (class_name, "destroyed")

#parent
p = cMom()
p.MomAssets()

#Empty child
c = cChild() #inherit from parent
c.MomAssets()#did not have this method in child but can call from Mom

#Empty child
c2 = cChild2() #inherit from parent
c2.MomAssets()#Will override from mom
```

Output:



21.2 Parent and Child on separate files

Parent cCar class is in separate file with child cSedan class

section21.2.py

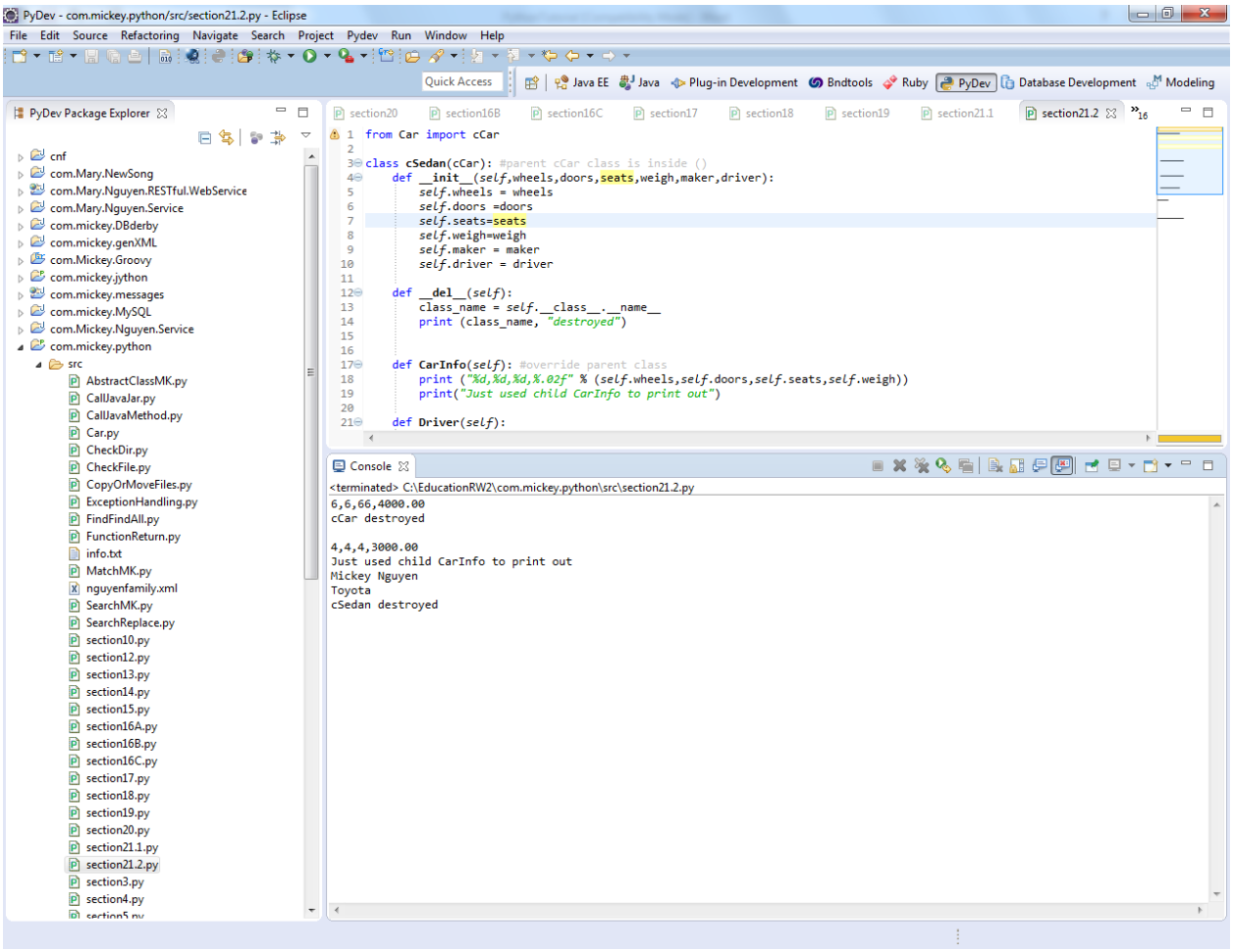
```
class cCar:

    def __init__(self, wheels,doors,seats,weigh,maker):
        self.wheels = wheels
        self.doors = doors
        self.seats = seats
        self.weigh = weigh
        self.maker = maker

    def __del__(self):
        class_name = self.__class__.__name__
        print (class_name, "destroyed")

    def CarInfo(self):
        print ("%d,%d,%d,%.02f" % (self.wheels,self.doors,self.seats,self.weigh))
```

```
    def CarMaker(self):
        print ("%s" %(self.maker))
```

Child cSedan class is in separate file with cCar parent class

```
from Car import cCar

class cSedan(cCar): #parent cCar class is inside ()
    def __init__(self,wheels,doors,seats,weigh,maker,driver):
        self.wheels = wheels
        self.doors =doors
        self.seats=seats
        self.weigh=weigh
        self.maker = maker
        self.driver = driver

    def __del__(self):
        class_name = self.__class__.__name__
        print (class_name, "destroyed")


    def CarInfo(self): #override parent class
        print ("%d,%d,%d,%.02f" % (self.wheels,self.doors,self.seats,self.weigh))
        print("Just used child CarInfo to print out")

    def Driver(self):
        print(self.driver)

#create an instance parent object
car = cCar(6,6,66,4000.0,"Toyota")# parent
car.CarInfo() #call parent method
del car #call destructor
print()#new line

#create my own sedan car
mySedan = cSedan(4,4,4,3000.0,"Toyota","Mickey Nguyen")
mySedan.CarInfo() #use sedan child method (override cCar parent) to display
#call my own Driver info
mySedan.Driver()
#call this from Parent.  child did not have this method
mySedan.CarMaker()

del mySedan #call destructor
```

```
Output run from child class cSedan
```



22 Abstract class

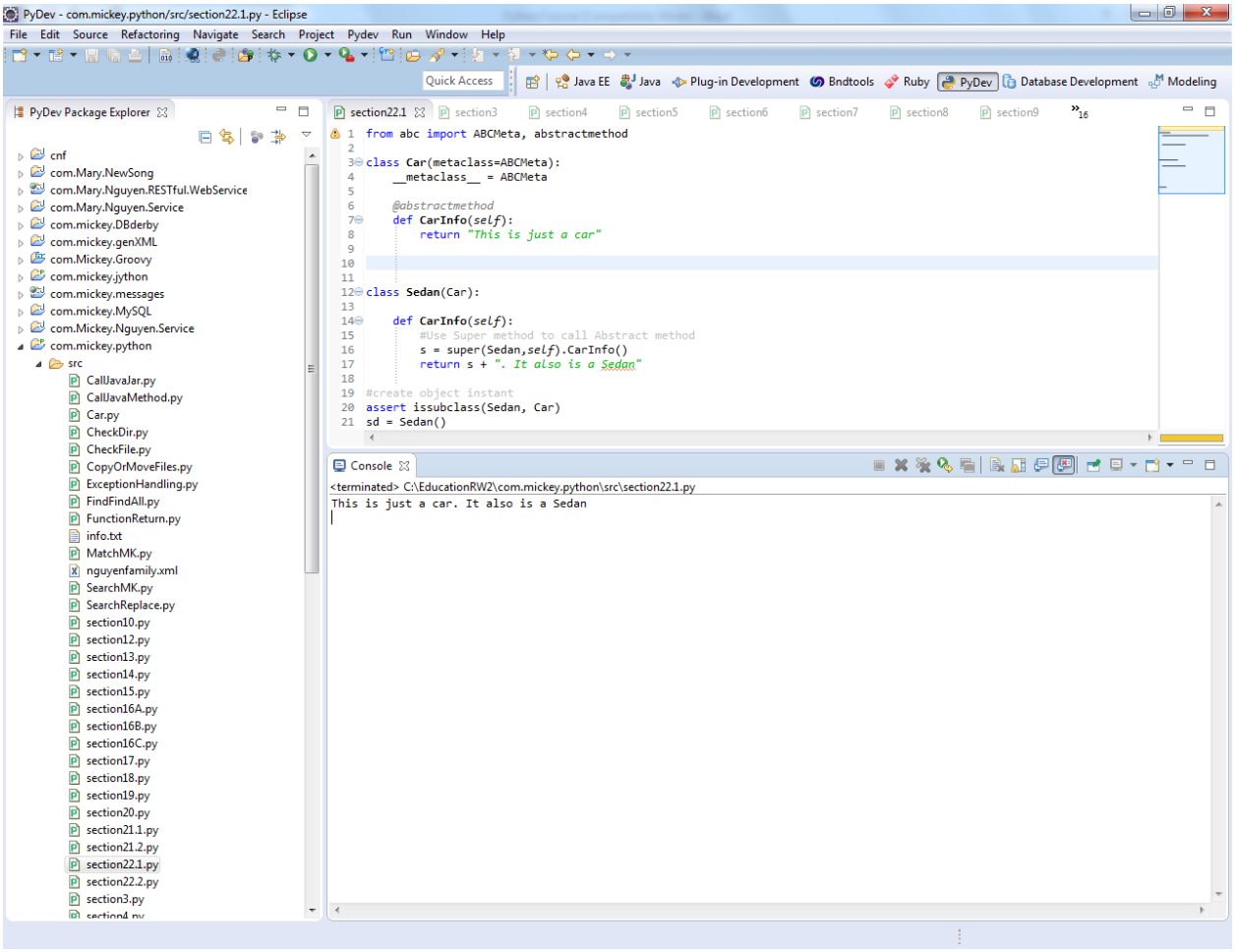
22.1 Child implement all abstract methods of its parent

section22.1.py

```
from abc import ABCMeta, abstractmethod

class Car(metaclass=ABCMeta):
    __metaclass__ = ABCMeta

    @abstractmethod
    def CarInfo(self):
        return "This is just a car"


class Sedan(Car):
```

```
    def CarInfo(self):
        #Use Super method to call Abstract method
        s = super(Sedan,self).CarInfo()
        return s + ". It also is a Sedan"

#create object instant
assert issubclass(Sedan, Car)
sd = Sedan()
print(sd.CarInfo())
```

Output:



22.2 Child did not implement all abstract methods of its parent

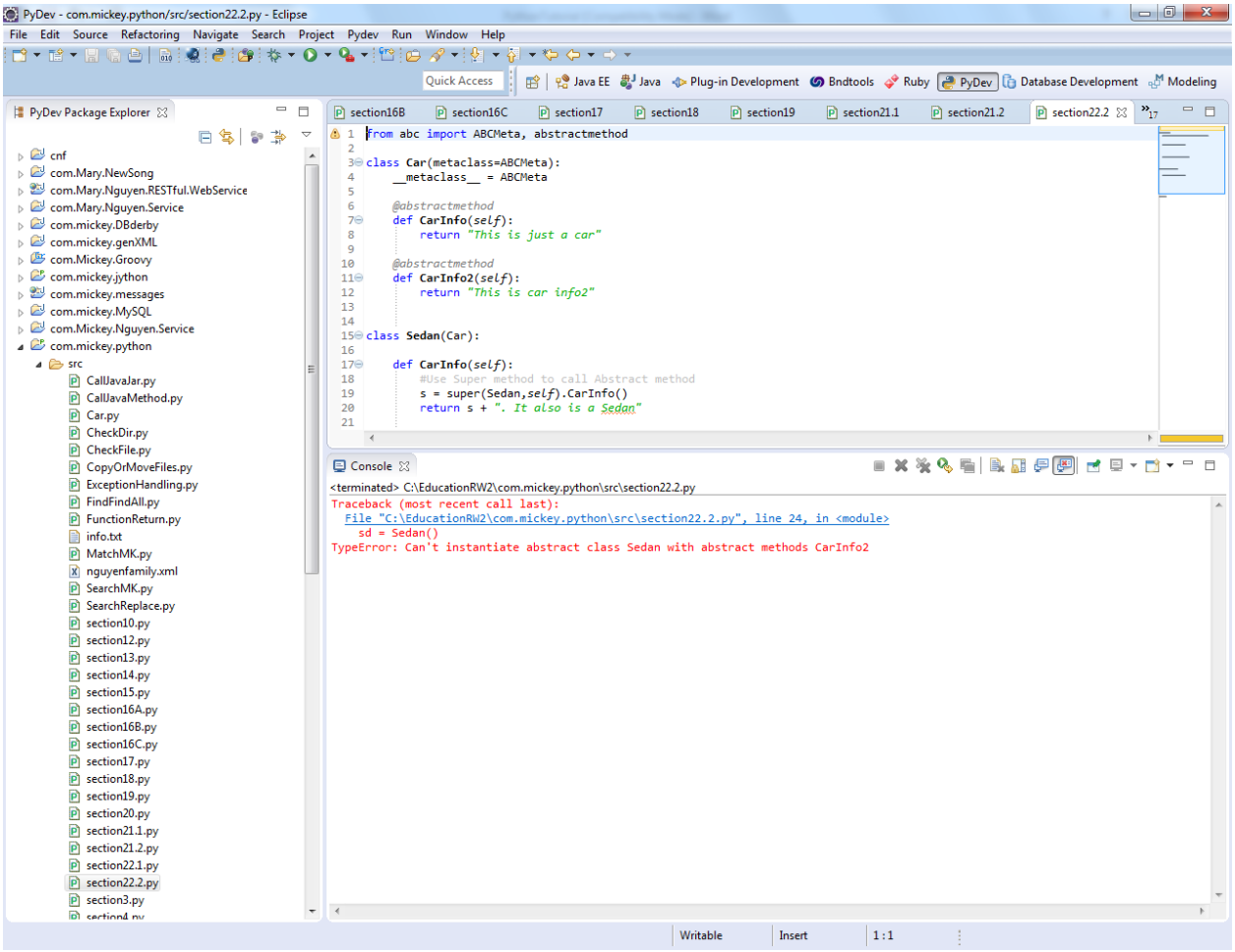
Now, I intentional to add another abstract method to base class but don't have implement on child class. Note: Child class suppose to implement all the abstract method of abstract parent class

section22.2.py

```
from abc import ABCMeta, abstractmethod

class Car(metaclass=ABCMeta):
    __metaclass__ = ABCMeta
```

```
    @abstractmethod
    def CarInfo(self):
        return "This is just a car"

    @abstractmethod
    def CarInfo2(self):
        return "This is car info2"


class Sedan(Car):

    def CarInfo(self):
        #Use Super method to call Abstract method
        s = super(Sedan,self).CarInfo()
        return s + ". It also is a Sedan"

#create object instant
assert issubclass(Sedan, Car)
sd = Sedan()
print(sd.CarInfo())
```

Output:



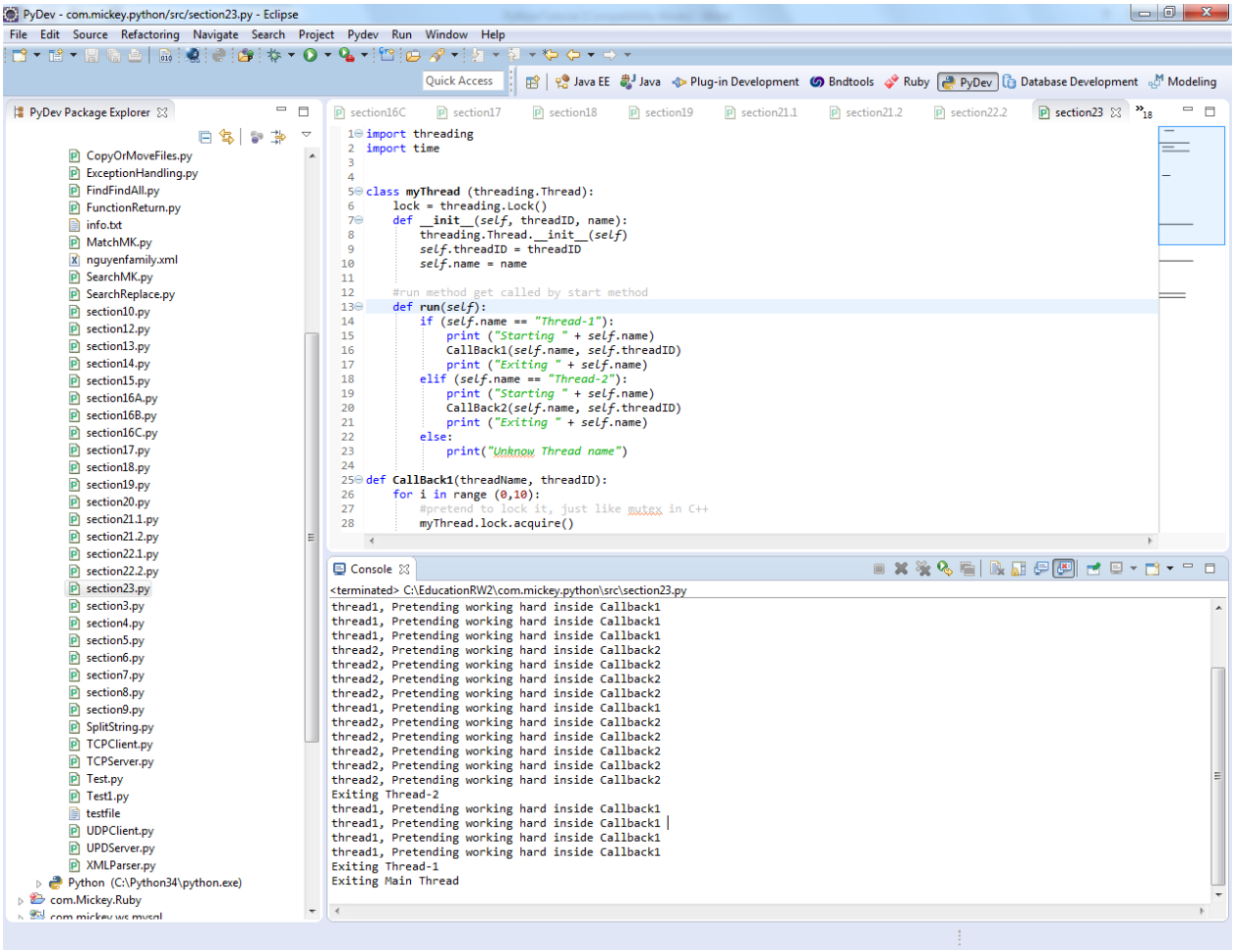
23 Thread

section23.py

```
import threading
import time


class myThread (threading.Thread):
    lock = threading.Lock()
    def __init__(self, threadID, name):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name

    #run method get called by start method
    def run(self):
        if (self.name == "Thread-1"):
```

```
            print ("Starting " + self.name)
            CallBack1(self.name, self.threadID)
            print ("Exiting " + self.name)
        elif (self.name == "Thread-2"):
            print ("Starting " + self.name)
            CallBack2(self.name, self.threadID)
            print ("Exiting " + self.name)
        else:
            print("Unknow Thread name")

def CallBack1(threadName, threadID):
    for i in range (0,10):
        #pretend to lock it, just like mutex in C++
        myThread.lock.acquire()
        time.sleep(2)
        print("thread" + str(threadID) + ", Pretending working hard inside Callback1
")
        #release the lock
        myThread.lock.release()

def CallBack2(threadName, threadID):
    for i in range (0,10):
        myThread.lock.acquire()
        time.sleep(2)
        print("thread" + str(threadID) +", Pretending working hard inside Callback2
")
        myThread.lock.release()

# Create new threads
thread1 = myThread(1, "Thread-1")
thread2 = myThread(2, "Thread-2")

# Start new Threads
thread1.start()
thread2.start()
thread1.join()
thread2.join()

print ("Exiting Main Thread")
```

Output:



24 Networking Socket

24.1 UDP server

section24.1.py

```
import socket

UDP_IP = "127.0.0.1" #local host
UDP_PORT = 5005

sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP
sock.bind((UDP_IP, UDP_PORT))

while (1):
    data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
    print ("received message:", data)
    #send it back to client
    sent = sock.sendto(data, addr)
```

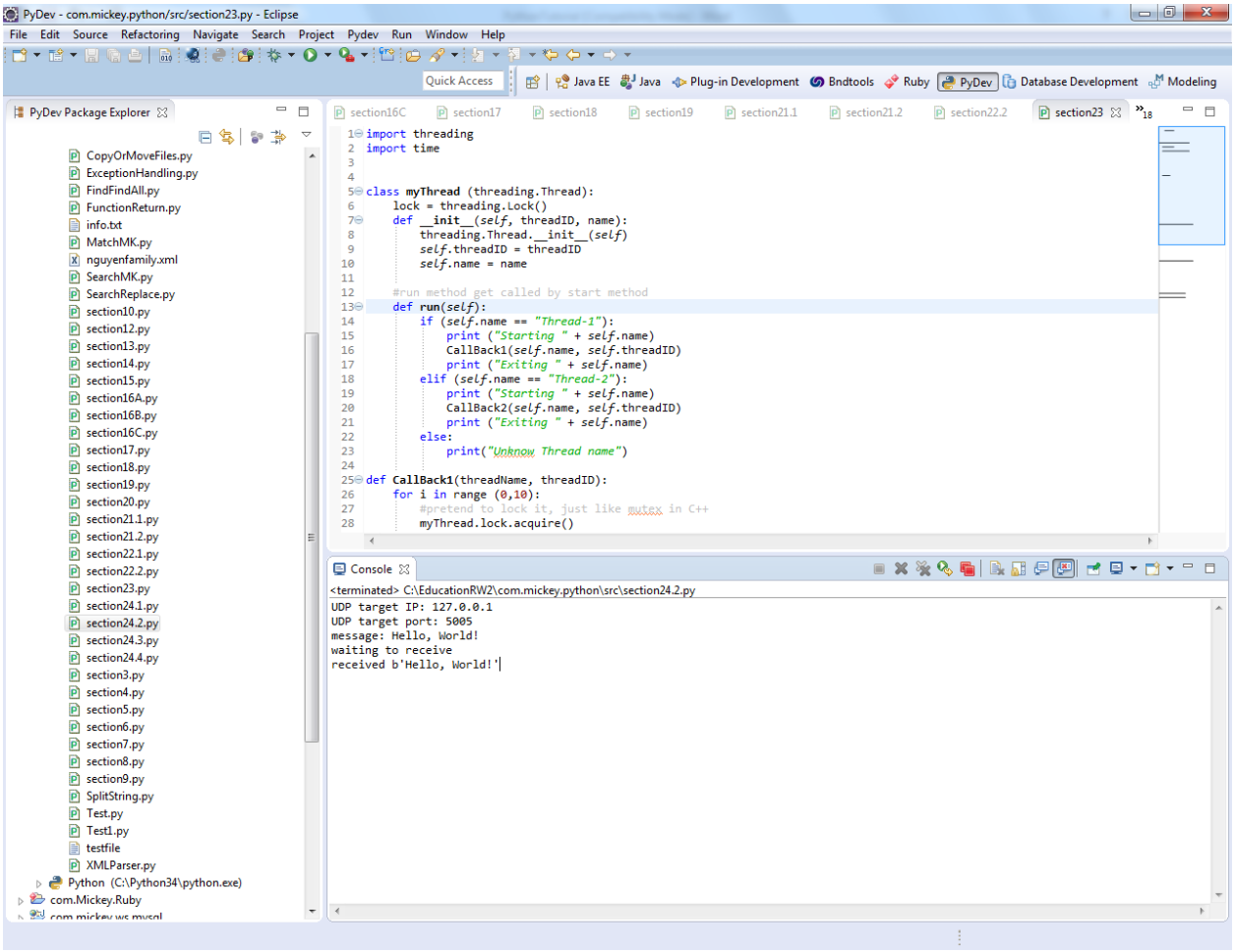
24.2 UDP Client

section24.2.py

```
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005
MESSAGE = "Hello, World!"


print ("UDP target IP:", UDP_IP)
print ("UDP target port:", UDP_PORT)
print ("message:", MESSAGE)

sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP
#python 3 has to cast str to bytes before to send
sock.sendto(bytes(MESSAGE, 'UTF-8'), (UDP_IP, UDP_PORT))

print ("waiting to receive")
data, server = sock.recvfrom(1024) # buffer size is 1024 bytes
print ("received %s" % data)

sock.close() #close the socket
```

To test it, start the server first, then start the client.

Output:



24.3 TCP server

section24.3.py

```
import socket               # Import socket module

s = socket.socket()         # Create a socket object
host = socket.gethostname() # Get local machine name
port = 12345                # Reserve a port for your service.
s.bind((host, port))        # Bind to the port

s.listen(5)                 # Now wait for client connection.
while True:
    data, addr = s.accept()     # Establish connection with client.
    print ("Got connection from", addr)
    #python 3 has to cast str to bytes before to send
    data.send(bytes("Thank you for connecting", 'UTF-8'))
    s.close()                   # Close the connection
```

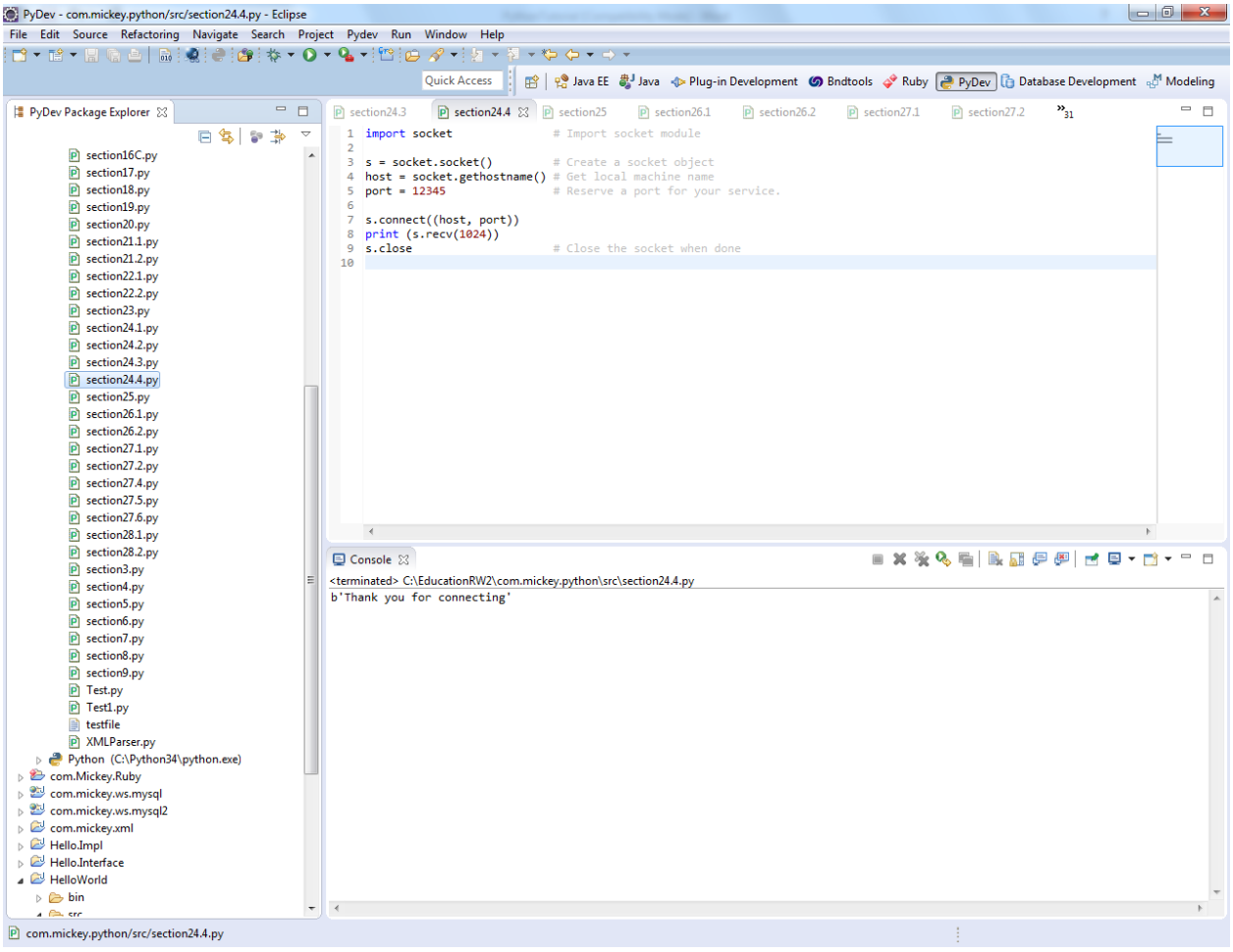
24.4 TCP Client

section24.4.py

```
import socket               # Import socket module

s = socket.socket()         # Create a socket object
host = socket.gethostname() # Get local machine name
port = 12345                # Reserve a port for your service.

s.connect((host, port))
print (s.recv(1024))
s.close                     # Close the socket when done
```

Start the server first, then start the client

Output:



25 Exception Handling

EXCEPTION NAME	DESCRIPTION
Exception	Base class for all exceptions
StopIteration	Raised when the next() method of an iterator does not point to any object.
SystemExit	Raised by the sys.exit() function.
StandardError	Base class for all built-in exceptions except StopIteration and SystemExit.
ArithmeticError	Base class for all errors that occur for numeric calculation.
OverflowError	Raised when a calculation exceeds maximum limit for a numeric type.
FloatingPointError	Raised when a floating point calculation fails.
ZeroDivisonError	Raised when division or modulo by zero takes place for all numeric types.
AssertionError	Raised in case of failure of the Assert statement.
AttributeError	Raised in case of failure of attribute reference or assignment.
EOFError	Raised when there is no input from either the raw_input() or input() function and the end of file is reached.
ImportError	Raised when an import statement fails.
KeyboardInterrupt	Raised when the user interrupts program execution,

	usually by pressing Ctrl+c.
LookupError	Base class for all lookup errors.
IndexError	Raised when an index is not found in a sequence.
KeyError	Raised when the specified key is not found in the dictionary.
NameError	Raised when an identifier is not found in the local or global namespace.
UnboundLocalError	Raised when trying to access a local variable in a function or method but no value has been assigned to it.
EnvironmentError	Base class for all exceptions that occur outside the Python environment.
IOError	Raised when an input/ output operation fails, such as the print statement or the open() function when trying to open a file that does not exist.
IOError	Raised for operating system-related errors.
SyntaxError	Raised when there is an error in Python syntax.
IndentationError	Raised when indentation is not specified properly.
SystemError	Raised when the interpreter finds an internal problem, but when this error is encountered the Python interpreter does not exit.
SystemExit	Raised when Python interpreter is quit by using the sys.exit() function. If not handled in the code, causes the interpreter to exit.
Raised when Python interpreter is quit by using the sys.exit() function. If not handled in the code, causes the interpreter to exit.	Raised when an operation or function is attempted that is invalid for the specified data type.
ValueError	Raised when the built-in function for a data type has the valid type of arguments, but the arguments have invalid

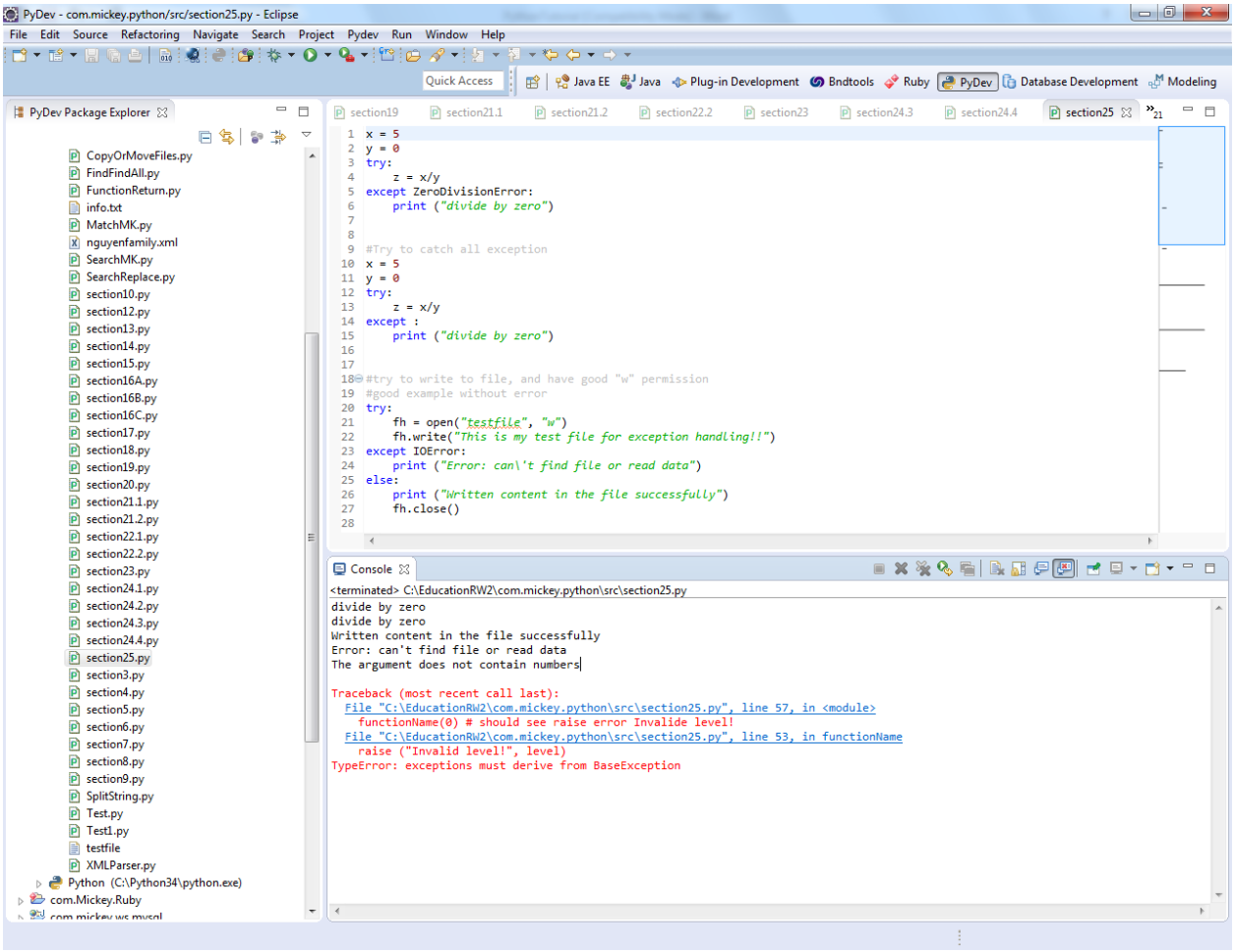
	values specified.
RuntimeError	Raised when a generated error does not fall into any category.
NotImplementedError	Raised when an abstract method that needs to be implemented in an inherited class is not actually implemented.

section25.py

```
x = 5
y = 0
try:
    z = x/y
except ZeroDivisionError:
    print ("divide by zero")


#Try to catch all exception
x = 5
y = 0
try:
    z = x/y
except :
    print ("divide by zero")


#try to write to file, and have good "w" permission
#good example without error
try:
    fh = open("testfile", "w")
    fh.write("This is my test file for exception handling!!")
except IOError:
    print ("Error: can\'t find file or read data")
else:
    print ("Written content in the file successfully")
    fh.close()

#try to write to file and don't have "w" permission
try:
    fh = open("testfile", "r")
    fh.write("This is my test file for exception handling!!")
except IOError:
    print ("Error: can\'t find file or read data")
else:
    print ("Written content in the file successfully")


#Another example of try catch_warnings# Define a function here.
def temp_convert(var):
    try:
```

```
        return int(var)
    except ValueError:
        print ("The argument does not contain numbers\n")

# Call above function here.
temp_convert("xyz")


#Example of raise
def functionName( level ):
    if level < 1:
        raise ("Invalid level!", level)
        # The code below to this would not be executed
        # if we raise the exception

functionName(0) # should see raise error Invalide level!

#or try this example
#example or raise
def mistake(name):
    # Raise an example with custom string.
    raise Exception(name + " caused exception")

# Call method.
mistake("Voorheesville")
```

Output:



26 Run Java Code

26.1 Calling Java JAR runnable

```
package PackageA;

import javax.swing.JOptionPane; //import class JOptionPane

public class cHelloWorld {

	public static void main(String[] args) {
		JOptionPane.showMessageDialog (null,"Hello from Mickey Nguyen \nWhat are you doing today? " );

	}

}
```

Generate helloworld.jar from java code above

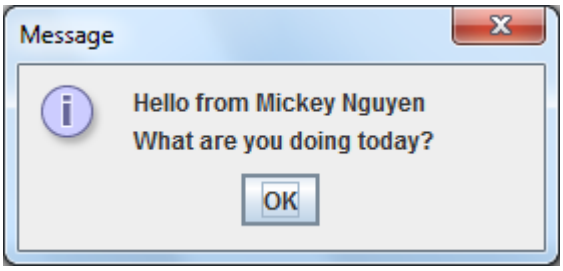
Python script:

section26.1.py

```
import subprocess
#I copy helloworld.jar to under src folder.  You can have a full path to where it
locates
subprocess.call(['java', '-jar','helloworld.jar'])
```

Then run python script use run as Jython option.

Output:



26.2 Calling Java method

Need to download a third party SW (Pyjnius/Jnius, JCC, javabridge, Jpype and Py4j.

) acting as a wrapper to be as a bridge between Python and Java.

Instead download third party software, you can write normal python script (file.py) and run it as Jython option.

If you want to import Java class into python script, you need to compile Javafile (module) first. Then you can import those Java module into python script. Make sure set the path correct, or simply copy those javafile.class to where your python script locates.

Here is example:

Java file MickeySayHello.Java

```
import javax.swing.JOptionPane; //import class JOptionPane
public class MickeySayHello {
	public void Hello()
	{
		JOptionPane.showMessageDialog (null,"Hello from Mickey Nguyen \nHow are
you doing today? " );
	}
```

```
}
```

Compile this Java file on your Java project first. It should produce MickeySayHello.class

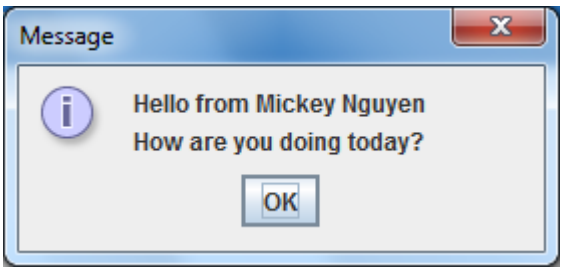
Copy MickeyHello.class to same location(under src) of your python script so you don't have to set path for it.

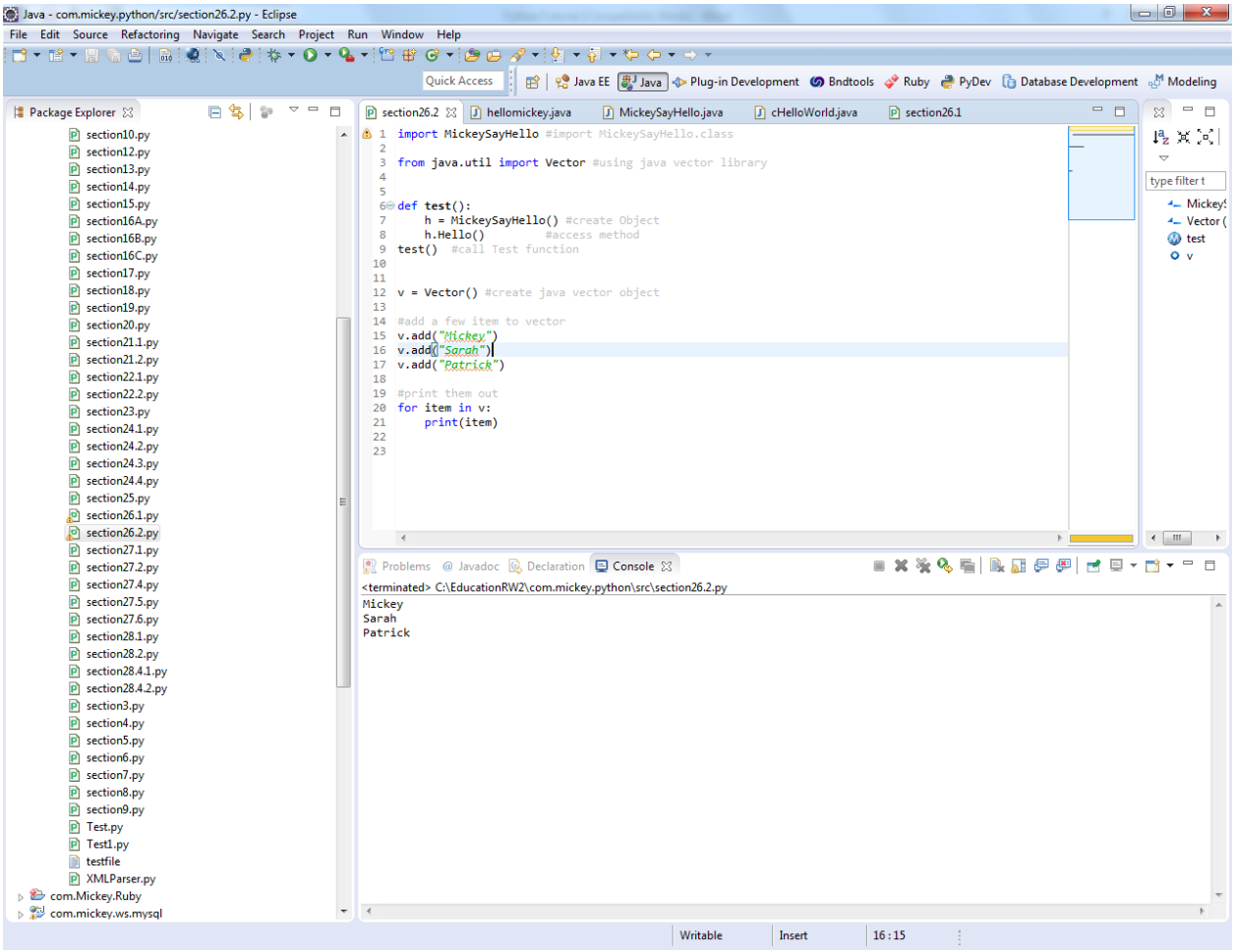
Python script section26.2.py, I am showing you how to use Java class and Java built in libraries such as vector,list,map,dictionary,…

```
import MickeySayHello #import MickeySayHello.class

from java.util import Vector #using java vector library


def test():
    h = MickeySayHello() #create Object
    h.Hello()         #access method
test()  #call Test function


v = Vector() #create java vector object

#add a few item to vector
v.add("Mickey")
v.add("Sarah")
v.add("Patrick")

#print them out
for item in v:
    print(item)
```

Then run python script use run as Jython option.

Output:





27 Regular Expression (Regex)

Regular Expression Modifiers: Option Flags

Regular expression literals may include an optional modifier to control various aspects of matching. The modifiers are specified as an optional flag. You can provide multiple modifiers using exclusive OR (|), as shown previously and may be represented by one of these −

Modifier	Description
re.I	Performs case-insensitive matching.

re.L	Interprets words according to the current locale. This interpretation affects the alphabetic group (\w and \W), as well as word boundary behavior (\b and \B).
re.M	Makes $ match the end of a line (not just the end of the string) and makes ^ match the start of any line (not just the start of the string).
re.S	Makes a period (dot) match any character, including a newline.
re.U	Interprets letters according to the Unicode character set. This flag affects the behavior of \w, \W, \b, \B.
re.X	Permits "cuter" regular expression syntax. It ignores whitespace (except inside a set [] or when escaped by a backslash) and treats unescaped # as a comment marker.

Regular Expression Patterns

Except for control characters, **(+ ? . * ^ $ ( ) [ ] { } | \)**, all characters match themselves. You can escape a control character by preceding it with a backslash.

Following table lists the regular expression syntax that is available in Python −

Pattern	Description
^	Matches beginning of line.
$	Matches end of line.
.	Matches any single character except newline. Using m option allows it to match newline as well.
[...]	Matches any single character in brackets.
[^...]	Matches any single character not in brackets
re*	Matches 0 or more occurrences of preceding expression.
re+	Matches 1 or more occurrence of preceding expression.

re?	Matches 0 or 1 occurrence of preceding expression.
re{ n}	Matches exactly n number of occurrences of preceding expression.
re{ n,}	Matches n or more occurrences of preceding expression.
re{ n, m}	Matches at least n and at most m occurrences of preceding expression.
a| b	Matches either a or b.
(re)	Groups regular expressions and remembers matched text.
(?imx)	Temporarily toggles on i, m, or x options within a regular expression. If in parentheses, only that area is affected.
(?-imx)	Temporarily toggles off i, m, or x options within a regular expression. If in parentheses, only that area is affected.
(?: re)	Groups regular expressions without remembering matched text.
(?imx: re)	Temporarily toggles on i, m, or x options within parentheses.
(?-imx: re)	Temporarily toggles off i, m, or x options within parentheses.
(?#...)	Comment.
(?= re)	Specifies position using a pattern. Doesn't have a range.
(?! re)	Specifies position using pattern negation. Doesn't have a range.
(?> re)	Matches independent pattern without backtracking.
\w	Matches word characters.
\W	Matches nonword characters.
\s	Matches whitespace. Equivalent to [\t\n\r\f].

\S	Matches nonwhitespace.
\d	Matches digits. Equivalent to [0-9].
\D	Matches nondigits.
\A	Matches beginning of string.
\Z	Matches end of string. If a newline exists, it matches just before newline.
\z	Matches end of string.
\G	Matches point where last match finished.
\b	Matches word boundaries when outside brackets. Matches backspace (0x08) when inside brackets.
\B	Matches nonword boundaries.
\n, \t, etc.	Matches newlines, carriage returns, tabs, etc.
\1...\9	Matches nth grouped subexpression.
\10	Matches nth grouped subexpression if it matched already. Otherwise refers to the octal representation of a character code.

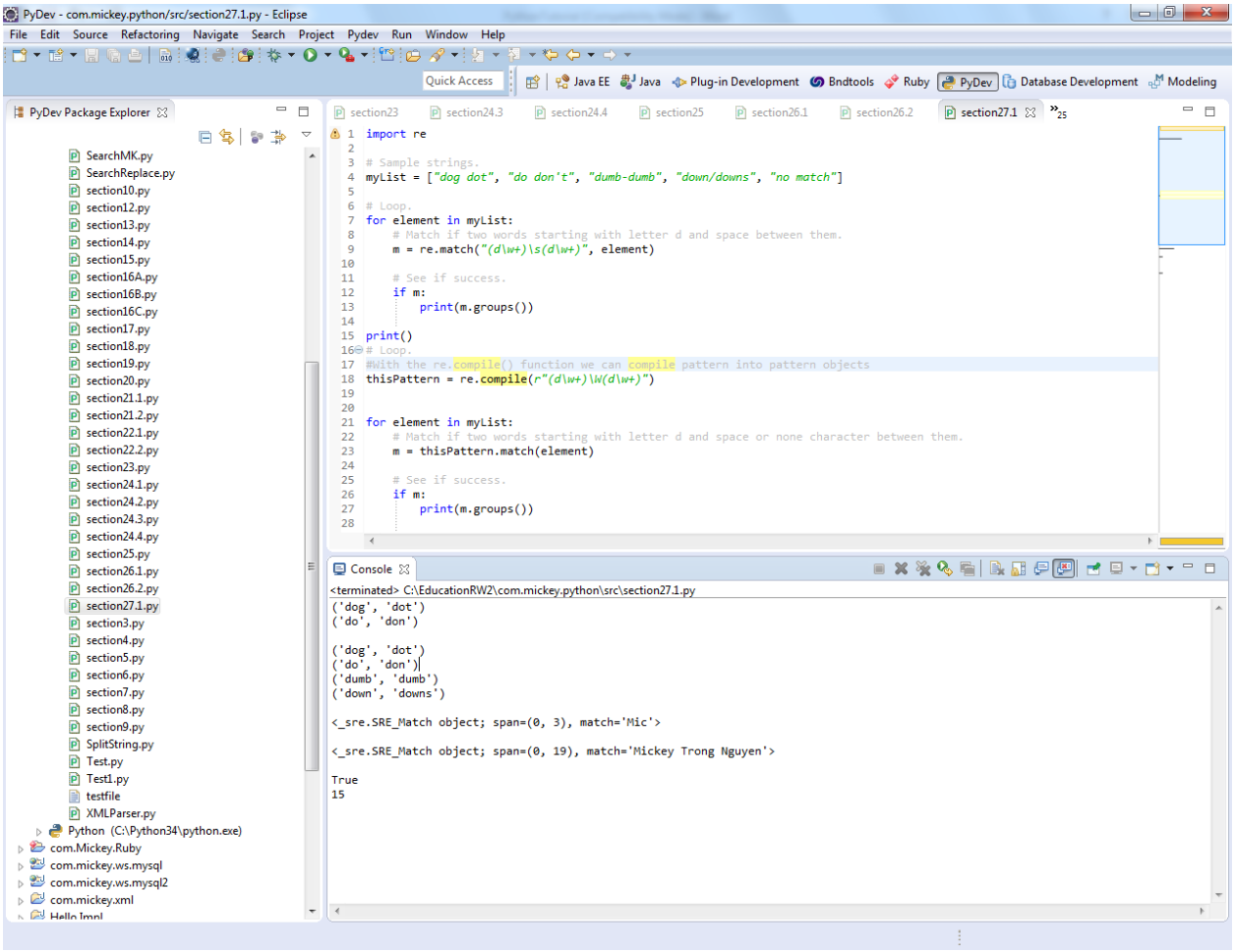
27.1 Match

section27.1.py

```
import re

# Sample strings.
myList = ["dog dot", "do don't", "dumb-dumb", "down/downs", "no match"]
```

```
# Loop.
for element in myList:
    # Match if two words starting with letter d and space between them.
    m = re.match("(d\w+)\s(d\w+)", element)

    # See if success.
    if m:
        print(m.groups())

print()
# Loop.
#With the re.compile() function we can compile pattern into pattern objects
thisPattern = re.compile(r"(d\w+)\W(d\w+)")


for element in myList:
    # Match if two words starting with letter d and space or none character between
them.
    m = thisPattern.match(element)

    # See if success.
    if m:
        print(m.groups())


#match a specific substring in big string
name = "Mickey Trong Nguyen is a software engineer for many years"
#at begin of string is easy
m = re.match("^Mic",name)
print()
print(m)
#at middle of string is harder
m = re.match(".*uyen",name)
print()
print(m)


#another way to find substring in big string
print()
print ("uyen" in name)
print (name.index("uyen"))
```

output:



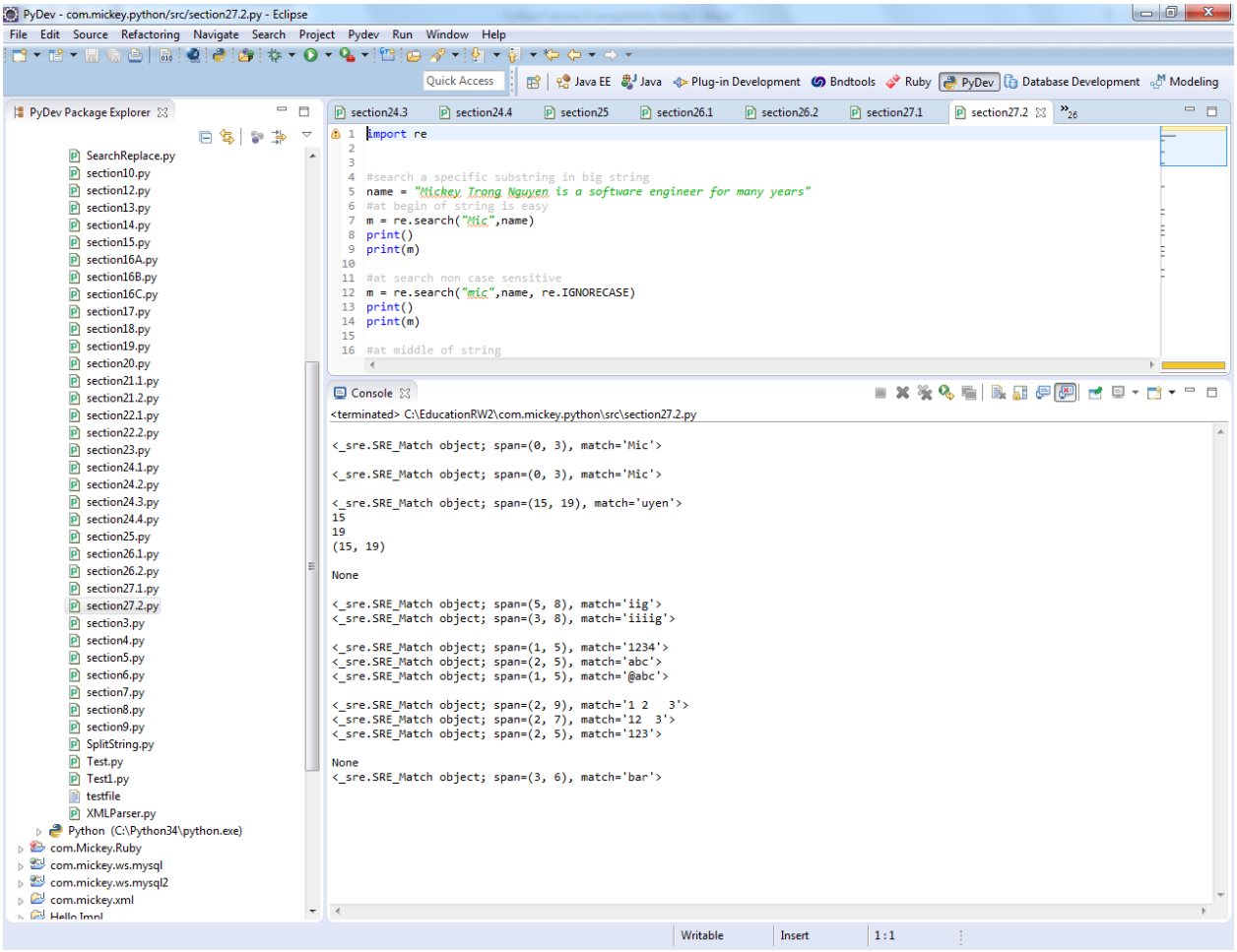
27.2 Search

section27.2.py

```
import re


#search a specific substring in big string
name = "Mickey Trong Nguyen is a software engineer for many years"
#at begin of string is easy
m = re.search("Mic",name)
print()
print(m)

#at search non case sensitive
m = re.search("mic",name, re.IGNORECASE)
print()
print(m)

#at middle of string
m = re.search("uyen",name)
```

```
print()
print(m)
if (m):
    print(m.start())#match begin
    print(m.end()) #match end
    print(m.span())#begin and end


#at middle of string, could not find it because case sensitive
m = re.search("nguyen",name)
print()
print(m)
if (m):
    print(m.start())#match begin
    print(m.end()) #match end
    print(m.span())#begin and end

#another example
print()
m = re.search(r'..g', 'piiiiiig')
print (m)
m = re.search(r'....g', 'piiiiiig')
print (m)

#another example
print()
m = re.search(r'\d\d\d\d', 'p12345g')
print(m)
m = re.search(r'\w\w\w', '@@abcdef!!')
print(m)
m = re.search(r'\W\w\w\w', '@@abcdef!!')
print(m)

#another example
print()
m = re.search(r'\d\s*\d\s*\d', 'xx1 2   3xx')
print(m)
m = re.search(r'\d\s*\d\s*\d', 'xx12  3xx')
print(m)
m = re.search(r'\d\s*\d\s*\d', 'xx123xx')
print(m)

#another example
print()
## ^ = matches the start of string, so this fails:
m = re.search(r'^b\w+', 'foobar')
print(m)
## but without the ^ it succeeds:
m = re.search(r'b\w+', 'foobar')
print(m)
```

output:



27.3 re.match vs re.search

re.match attempts to match a pattern **at the beginning of the string**. re.search attempts to match the pattern **throughout the string** until it finds a match.

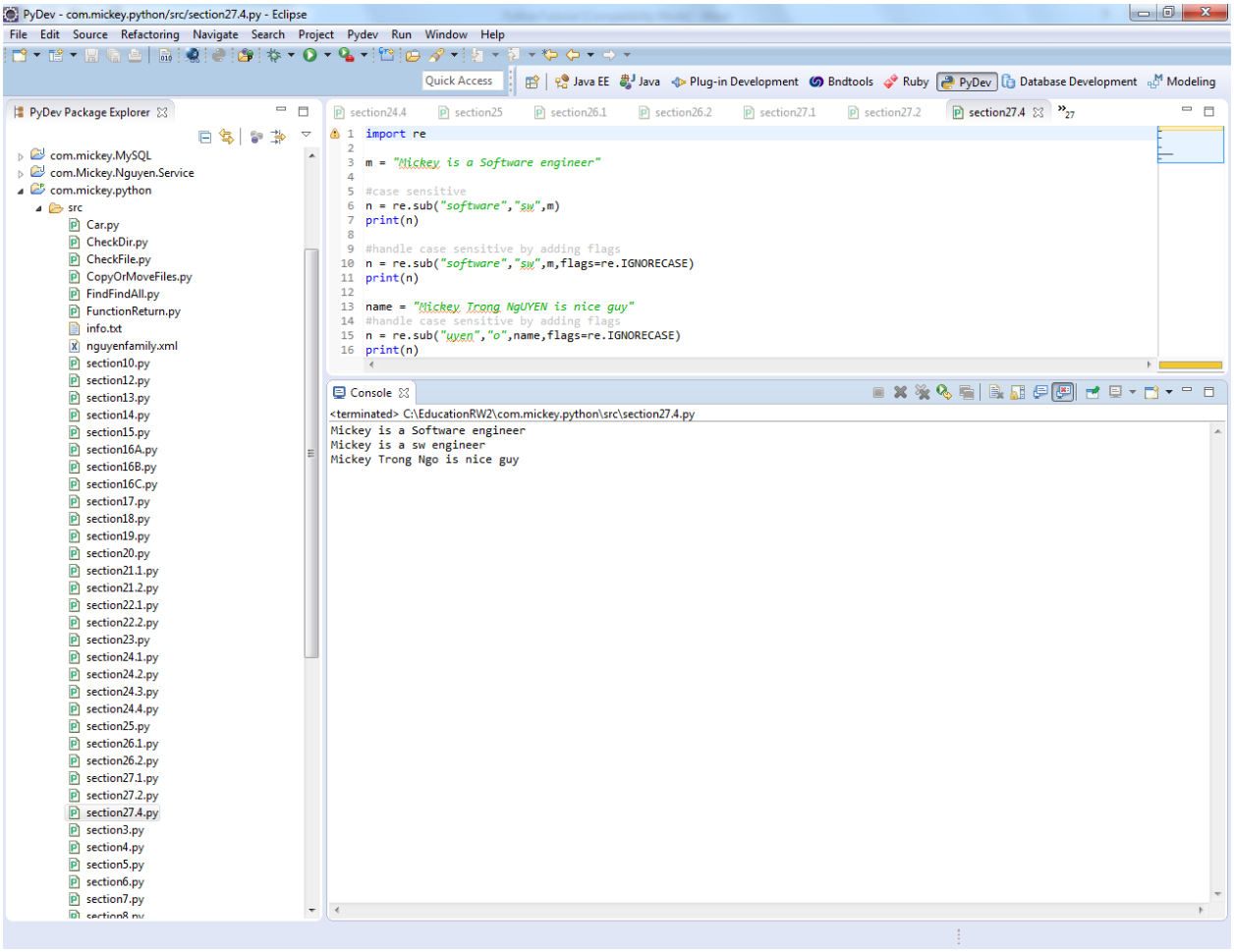
27.4 Search and Replace

section27.4.py

```
import re

m = "Mickey is a Software engineer"
```

```
#case sensitive
n = re.sub("software","sw",m)
print(n)

#handle case sensitive by adding flags
n = re.sub("software","sw",m,flags=re.IGNORECASE)
print(n)

name = "Mickey Trong NgUYEN is nice guy"
#handle case sensitive by adding flags
n = re.sub("uyen","o",name,flags=re.IGNORECASE)
print(n)
```

output:



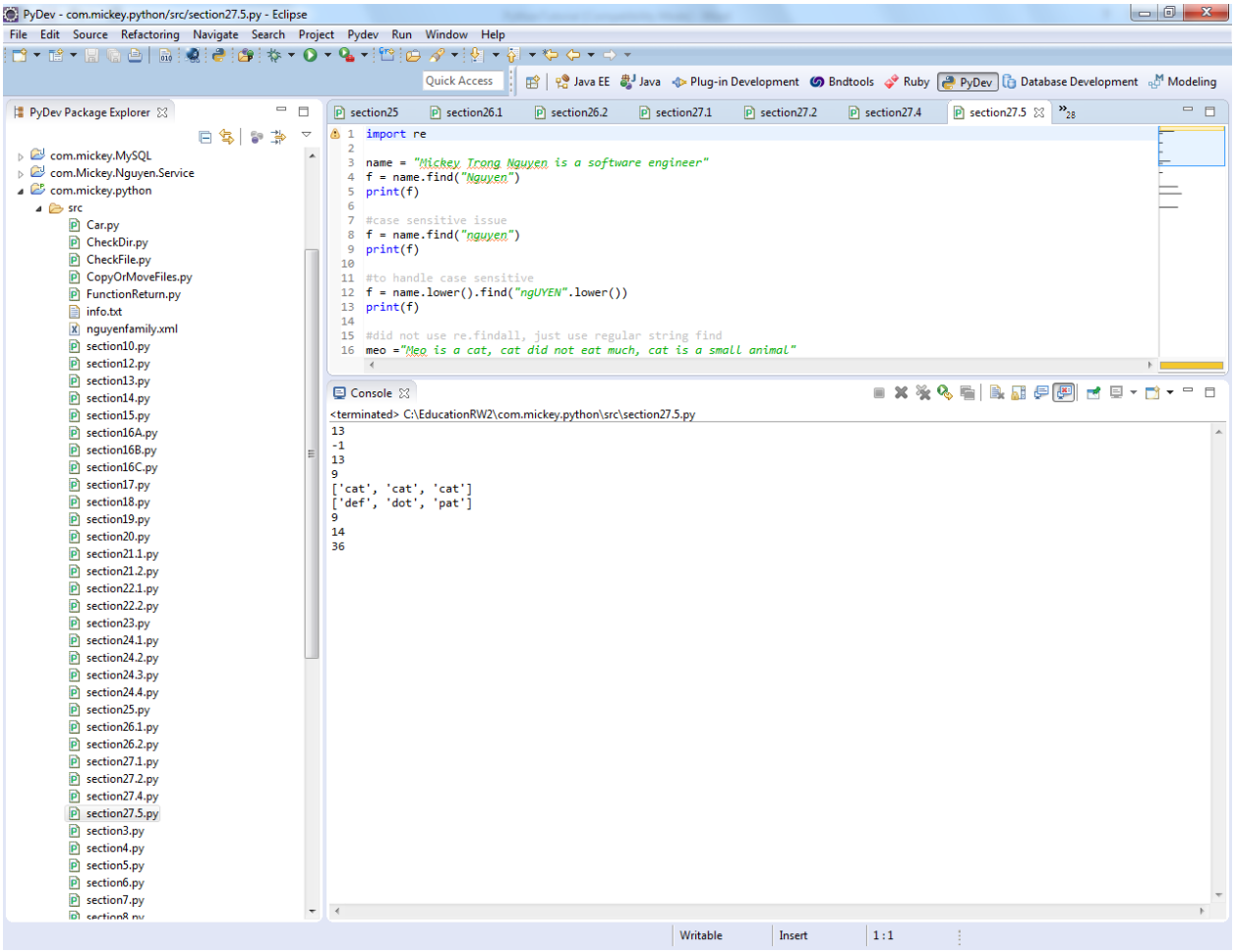
27.5 find() sub-string, re.findall (regex), re.finditer (index of all matched)

section27.5.py

```
import re
```

```
name = "Mickey Trong Nguyen is a software engineer"
f = name.find("Nguyen")
print(f)

#case sensitive issue
f = name.find("nguyen")
print(f)

#to handle case sensitive
f = name.lower().find("ngUYEN".lower())
print(f)

#did not use re.findall, just use regular string find
meo ="Meo is a cat, cat did not eat much, cat is a small animal"
f = meo.find("cat")
print(f)

#use re.findall
f = re.findall("cat",meo)
print(f)

#another example of find multiple things in one shot

# Input.
value = "abc 123 def 456 dot map pat"

# Find all words starting with d or p.
myList = re.findall("[dp]\w+", value)
# Print result.
print(myList)


#use re.finditer
regex = re.compile("cat", re.IGNORECASE)

for m in regex.finditer(meo):
    print(m.start())
```

Output:



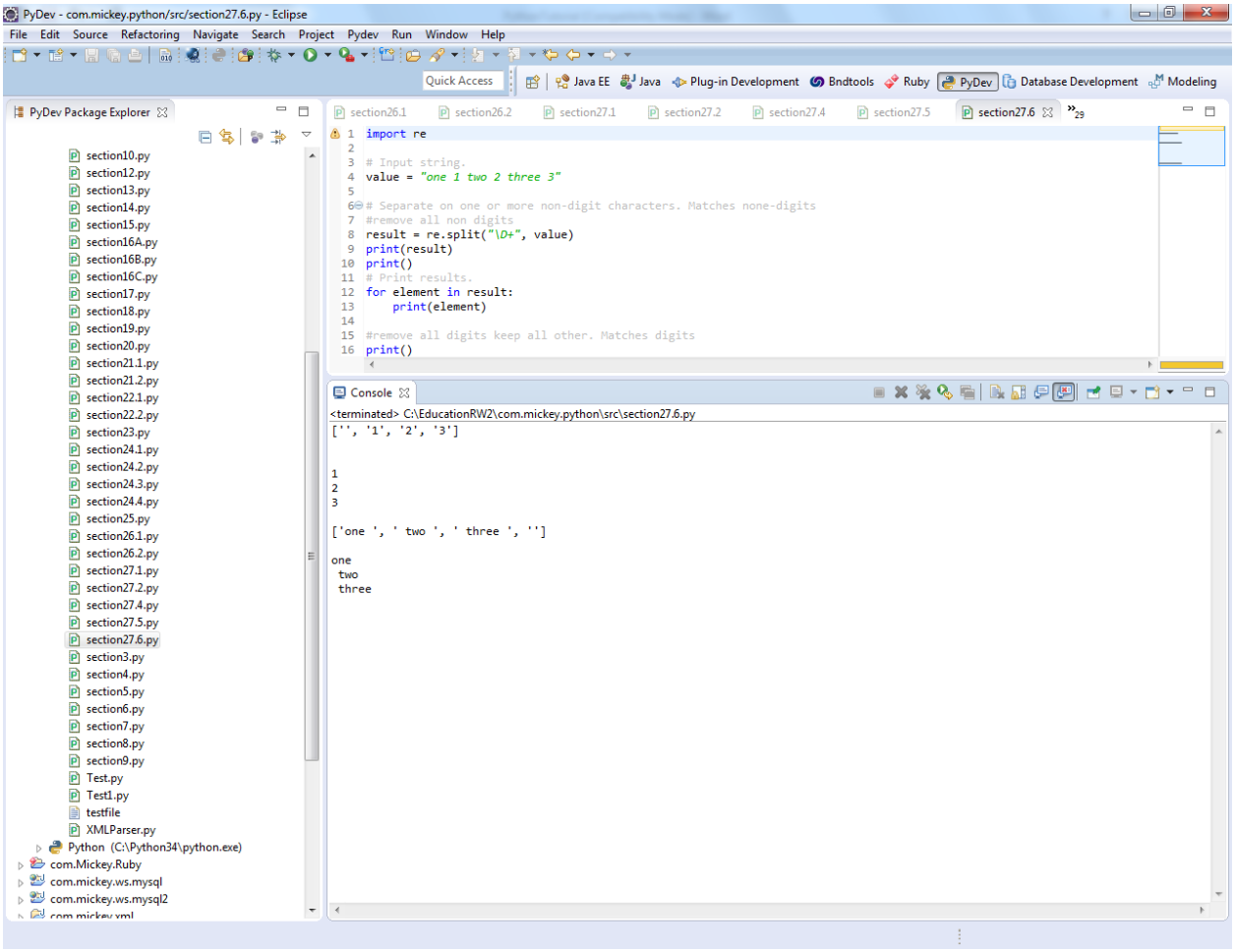
27.6 Spliting the string

section27.6.py

```
import re

# Input string.
value = "one 1 two 2 three 3"

# Separate on one or more non-digit characters. Matches none-digits
#remove all non digits
result = re.split("\D+", value)
print(result)
print()
# Print results.
for element in result:
    print(element)

#remove all digits keep all other. Matches digits
print()
result = re.split("\d+", value)
```

```
print(result)
print()
# Print results.
for element in result:
    print(element)
```

Output:



28 Some useful examples of Python script

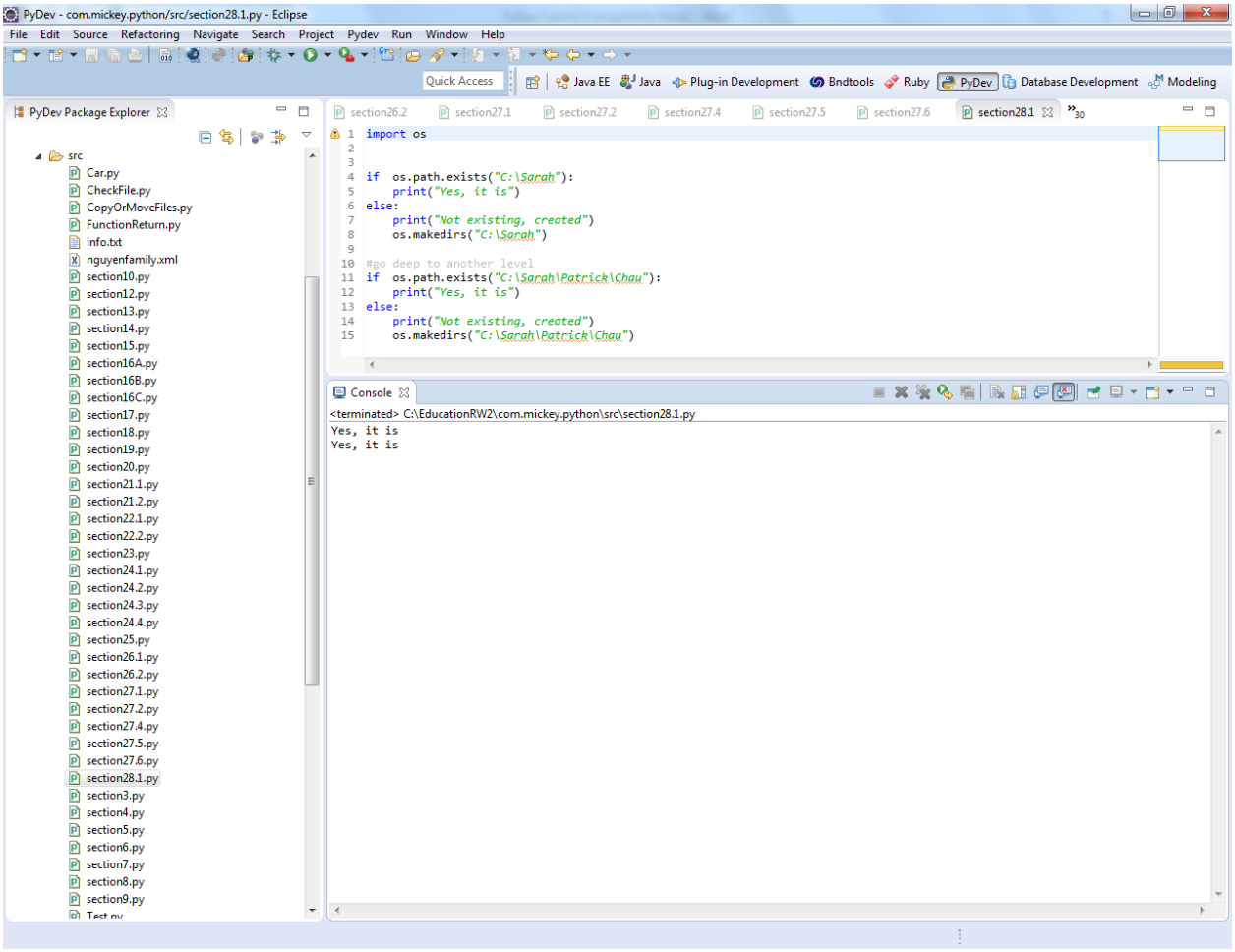
28.1 Directory exist? Or created it

section28.1.py

```
import os
```

```
if  os.path.exists("C:\Sarah"):
    print("Yes, it is")
else:
    print("Not existing, created")
    os.makedirs("C:\Sarah")

#go deep to another level
if  os.path.exists("C:\Sarah\Patrick\Chau"):
    print("Yes, it is")
else:
    print("Not existing, created")
    os.makedirs("C:\Sarah\Patrick\Chau")
```

Output:

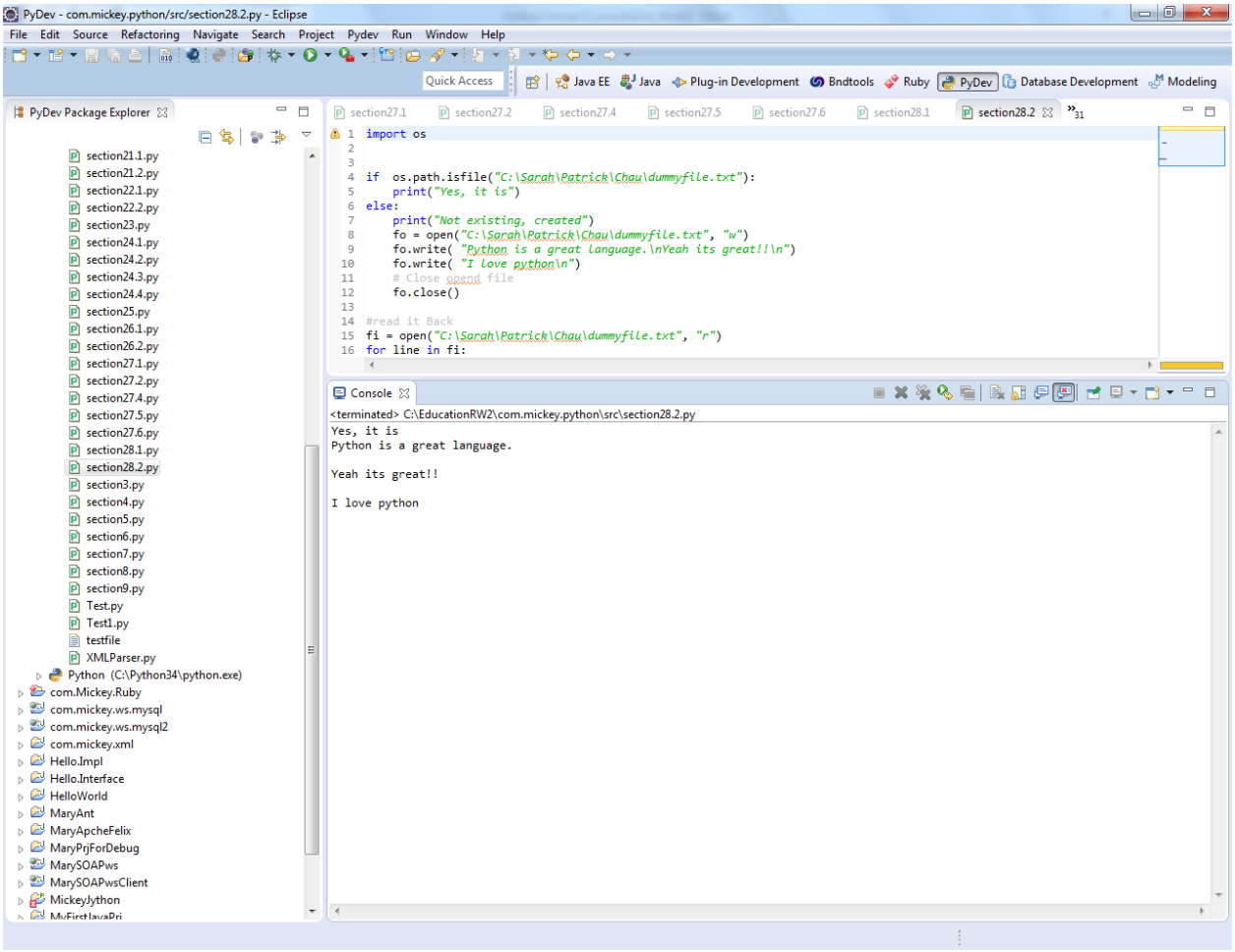


28.2 File exist? Or Create it

section28.2.py

```
import os


if  os.path.isfile("C:\Sarah\Patrick\Chau\dummyfile.txt"):
    print("Yes, it is")
else:
    print("Not existing, created")
    fo = open("C:\Sarah\Patrick\Chau\dummyfile.txt", "w")
    fo.write( "Python is a great language.\nYeah its great!!\n")
    fo.write( "I love python\n")
    # Close opend file
    fo.close()

#read it Back
fi = open("C:\Sarah\Patrick\Chau\dummyfile.txt", "r")
for line in fi:
    print (line,)
fi.close()
```

Output:



28.3 Copy,move file between folders(directories)

Using these libraries

```
import shutil
import os
```

shutil.**copy**(*src*, *dst*, ***, *follow_symlinks=True*)

shutil.**copyfile**(*src*, *dst*, ***, *follow_symlinks=True)*

shutil.**move**(*src*, *dst*)

more and more

28.4 Run Java (JAR) and using Jython to acces Java code

28.4.1 Calling Java JAR runnable

Rerun section 26.1

28.4.2 Calling Java method

Rerun section 26.2

29 XML Processing

Copy nguyenfamily.xml to under src folder

```
<family nguyen="family members">
<member name="Mickey Nguyen">
   <profession>SW Engineer</profession>
   <age>50</age>
   <gender>male</gender>
</member>
<member name="Chau Nguyen">
   <profession>SQA Engineer</profession>
   <age>20</age>
   <gender>female</gender>
</member>
<member name="Patrick Nguyen">
   <profession>nhoc con</profession>
   <age>4</age>
   <gender>male</gender>
</member>
<member name="Sarah Nguyen">
   <profession>student</profession>
   <age>8</age>
   <gender>female</gender>
</member>

</family>
```

Here is Python code to parse xml

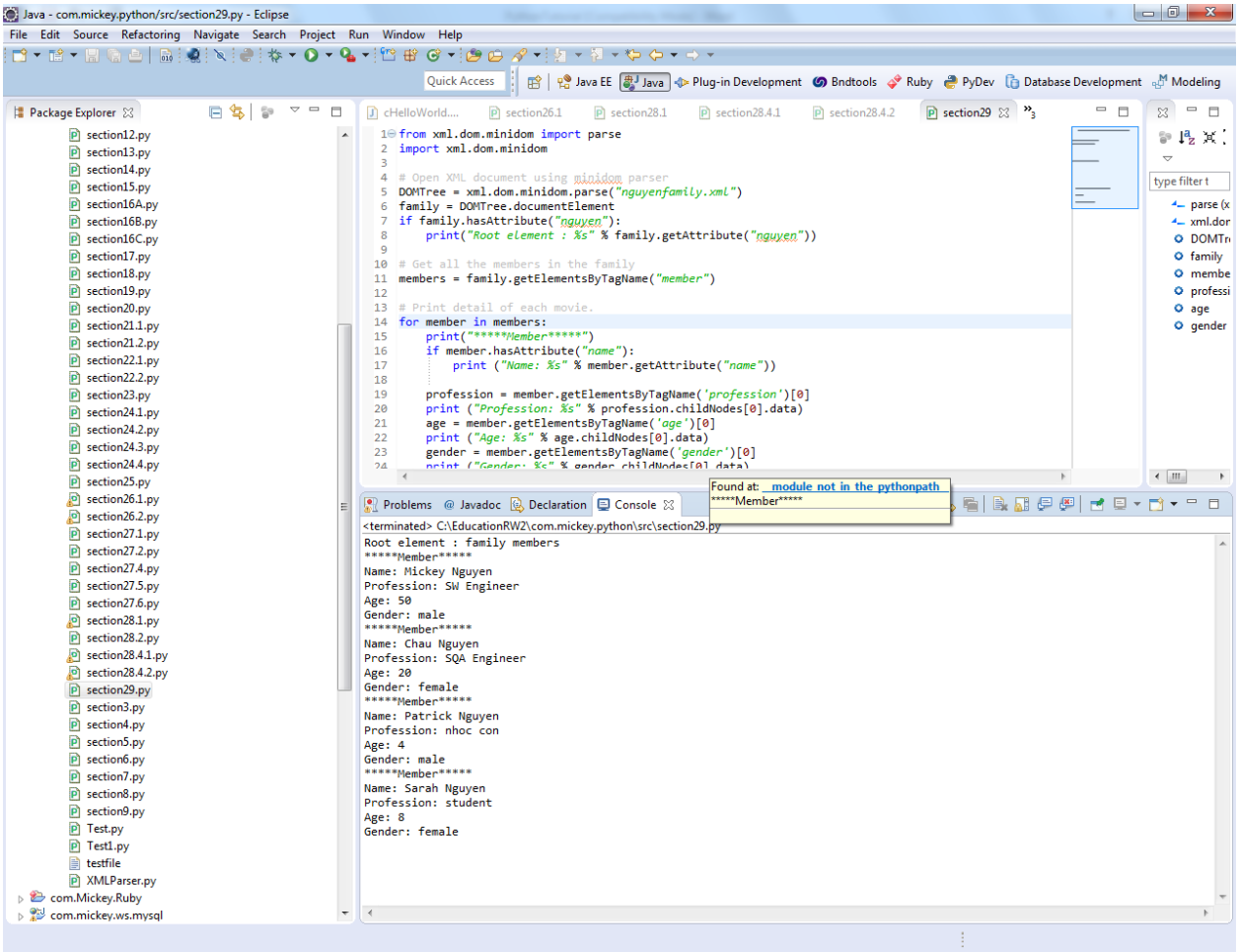
section29.py

```
from xml.dom.minidom import parse
import xml.dom.minidom

# Open XML document using minidom parser
DOMTree = xml.dom.minidom.parse("nguyenfamily.xml")
family = DOMTree.documentElement
if family.hasAttribute("nguyen"):
    print("Root element : %s" % family.getAttribute("nguyen"))

# Get all the members in the family
members = family.getElementsByTagName("member")
```

```
# Print detail of each movie.
for member in members:
    print("*****Member*****")
    if member.hasAttribute("name"):
        print ("Name: %s" % member.getAttribute("name"))

    profession = member.getElementsByTagName('profession')[0]
    print ("Profession: %s" % profession.childNodes[0].data)
    age = member.getElementsByTagName('age')[0]
    print ("Age: %s" % age.childNodes[0].data)
    gender = member.getElementsByTagName('gender')[0]
    print ("Gender: %s" % gender.childNodes[0].data)
```

Run it as python (not Jython)

Output:



30 CGI (Common Getway Interface) programming

TBD

31 Database Access

TBD

32 GUI Programming

TBD