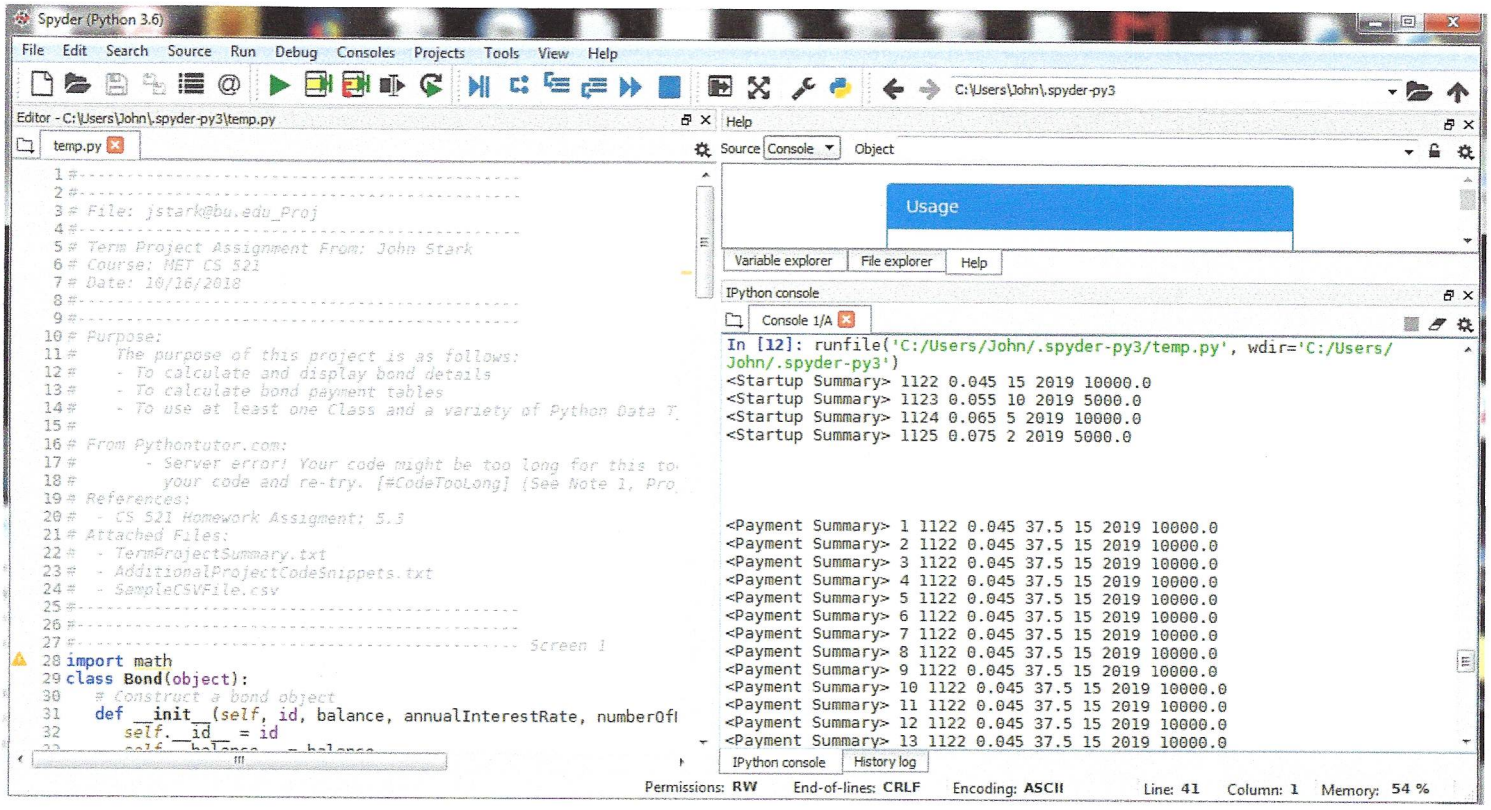
Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\John\.spyder-py3
Editor - C:\Users\John\.spyder-py3\temp.py
temp.py
1 #---
2 #---
3 # File: jstark@bu.edu_Proj
4 #---
5 # Term Project Assignment From: John Stark
6 # Course: MET CS 521
7 # Date: 10/16/2018
8 #---
9 #---
10 # Purpose:
11 #   The purpose of this project is as follows:
12 #   - To calculate and display bond details
13 #   - To calculate bond payment tables
14 #   - To use at least one Class and a variety of Python Data T
15 #
16 # From Pythontutor.com:
17 #   - Server error! Your code might be too long for this to
18 #   your code and re-try. [#CodeTooLong] (See Note 1, Pro
19 # References:
20 #   - CS 521 Homework Assigment: 5.3
21 # Attached Files:
22 #   - TermProjectSummary.txt
23 #   - AdditionalProjectCodeSnippets.txt
24 #   - SampleCSVFile.csv
25 #---
26 #---
27 #-- Screen 1
28 import math
29 class Bond(object):
30   # Construct a bond object
31   def __init__(self, id, balance, annualInterestRate, numberOfI
32   self.__id__ = id
Help
Source Console Object
Usage
Variable explorer File explorer Help
IPython console
Console 1/A
In [12]: runfile('C:/Users/John/.spyder-py3/temp.py', wdir='C:/Users/
John/.spyder-py3')
<Startup Summary> 1122 0.045 15 2019 10000.0
<Startup Summary> 1123 0.055 10 2019 5000.0
<Startup Summary> 1124 0.065 5 2019 10000.0
<Startup Summary> 1125 0.075 2 2019 5000.0
<Payment Summary> 1 1122 0.045 37.5 15 2019 10000.0
<Payment Summary> 2 1122 0.045 37.5 15 2019 10000.0
<Payment Summary> 3 1122 0.045 37.5 15 2019 10000.0
<Payment Summary> 4 1122 0.045 37.5 15 2019 10000.0
<Payment Summary> 5 1122 0.045 37.5 15 2019 10000.0
<Payment Summary> 6 1122 0.045 37.5 15 2019 10000.0
<Payment Summary> 7 1122 0.045 37.5 15 2019 10000.0
<Payment Summary> 8 1122 0.045 37.5 15 2019 10000.0
<Payment Summary> 9 1122 0.045 37.5 15 2019 10000.0
<Payment Summary> 10 1122 0.045 37.5 15 2019 10000.0
<Payment Summary> 11 1122 0.045 37.5 15 2019 10000.0
<Payment Summary> 12 1122 0.045 37.5 15 2019 10000.0
<Payment Summary> 13 1122 0.045 37.5 15 2019 10000.0
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 41 Column: 1 Memory: 54 %

```
#---------------------------------------------
#---------------------------------------------
# File: jstark@bu.edu_Proj
#---------------------------------------------
# Term Project Assignment From: John Stark
# Course: MET CS 521
# Date: 10/20/2018
#---------------------------------------------
#---------------------------------------------
# Purpose:
#    The purpose of this project is as follows:
#    - To calculate and display bond details
#    - To calculate bond payment tables
#    - To use at least one Class and a variety of Python Data Types
#
# From Pythontutor.com:
#       - Server error! Your code might be too long for this tool. Shorten
#         your code and re-try. [#CodeTooLong] (See Note 1, Project Summary)
# References:
#       - CS 521 Homework Assigment: 5.3
# Attached Files:
#       - TermProjectSummary.txt
#       - AdditionalProjectCodeSnippets.txt
#       - SampleCSVFile.csv
#---------------------------------------------
#---------------------------------------------
#--------------------------------------------- Screen 1
import math
class Bond(object):
        # Construct a bond object
        def __init__(self, id, balance, annualInterestRate, numberOfPeriods,
maturityDate, faceValue):
                self.__id__ = id
                self.__balance__ = balance
                self.__annualInterestRate__ = annualInterestRate
                self.__numberOfPeriods__ = numberOfPeriods
                self.__maturityDate__ = maturityDate
                self.__faceValue__ = faceValue
#---------------------------- Accessing the Bond Object, basic inputs -- Screen 2
        def getId(self):
                return self.__id__
        def __str__(self):
                return '{} {}'.format('Bond: ', self.__id__)
        def getBalance(self):
                return self.__balance__
        def getAnnualInterestRate(self):
                return self.__annualInterestRate__
        def getNumberOfPeriods(self):
                return self.__numberOfPeriods__
        def getMaturityDate(self):
                return self.__maturityDate__
        def getfaceValue(self):
                return self.__faceValue__
#---------------------------- Accessing the Bond Object, per month -- Screen 3
        def getMonthlyInterestRate(self):
                return self.__annualInterestRate__ / 12
        def getMonthlyInterest (self):
                return ((self.__annualInterestRate__/12) * self.__faceValue__)
#---------------------------- Accessing the Bond Object, during setup -- Screen 4
        def setBalance(self, balance):
                # This is for setting up a Clients initial balance
                # during opening of account
                self.__balance__ = balance
                return
#-------------- Accessing the Bond Object, Service associated Clients Cash Account
```

```
-- Screen 5
        def withdraw (self, wdraw):
                # This is for Client withdrawals
                self.__balance__ = self.balance - wdraw
                return
        def deposit(self, depos):
                # This is for Client deposits
                self.__balance__ = self.__balance__ + depos
                return
#----------------------------- End of Bond Object ------------------ Screen 6
def DisplayStartup (bondList):
        # Display bond list
        # Instantiate the bondList elements one bond at a time
        # and display its initial specifics at startup time
        for bond in ( bondList ):
                id = bond.getId()
                apr = bond.getAnnualInterestRate()
                nop = bond.getNumberOfPeriods()
                md = bond.getMaturityDate()
                fv = bond.getfaceValue()
                print ( "<Startup Summary>", id, apr, nop, md, fv )
        print("\n")
        return
def DisplayBondPayments (bondList, bondPaymentsList):
        # Display bond payments list, for all the bonds
        #Instantiate the bondList elements one at a time
        #
        for bond in ( bondList ):
                id = bond.getId()
                apr = bond.getAnnualInterestRate()
                nop = bond.getNumberOfPeriods()
                md = bond.getMaturityDate()
                fv = bond.getfaceValue()
                intr = fv * (apr/12)
                print("\n")
                paymentsList = ""
                for i in range ( 1, nop+1):
                        paymentsList = paymentsList + "'self.i', intr, \
                        'bond.getMonthlyInterest'"
                        print ("<Payment Summary>",i, id, apr, intr, nop, md, fv )

        print("\n")
        return
def printAccountSummary( bondList ):
        # Display the Client's cash account summary
        for bond in ( bondList ):
                id = bond.getId()
                bal = bond.getBalance()
                print ("<Client Cash Summary>",id, bal)
        print("\n")
        return
#-------------- Main -------------- -- Screen 7
# Instantiate the Bond objects
bond1 = Bond (1122, 20000, 0.045, 15, 2019, 10000.00)
bond2 = Bond (1123, 30000, 0.055, 10, 2019, 5000.00)
bond3 = Bond (1124, 20000, 0.065, 5, 2019, 10000.00)
bond4 = Bond (1125, 30000, 0.075, 2, 2019, 5000.00)
#
bondList = [] # Create a bond object list
bondList.append( bond1 )
bondList.append ( bond2 )
bondList.append( bond3 )
bondList.append ( bond4 )
# Display the initial state of the bond list
# Assume only one bond per Client
```

```
DisplayStartup (bondList)
#-------- Use the Bond List to generate principle/interest tables
paymentsList = [] # Create a payments list
# For each bond, display the payments schedule
DisplayBondPayments (bondList, paymentsList)
#-------------- End --------------
```

```
#---------------------------------------------
#---------------------------------------------
# File: TermProjectSummary.txt
#---------------------------------------------
# Term Project Summary From: John Stark
# Course: MET CS 521
# Date: 10/20/2018
#---------------------------------------------
#---------------------------------------------

    Screen 1 shows the Bond Object Constructor.  The
internal variables are name mangled.

    Screens 2 and 3 show the getter routines with
information being restriced in the Bond Object.

    Screen 4 contains a setter routine for the
internal balance variable.  You would expect to see
setter routines here for id, annual intrest rate,
number of periods, maturity date and face value.
They have been left out for reasons discussed in
Summary Note 1 below.

    Screen 5 contains functions for servicing
withdrawals and deposits.

    Screen 6 begins where the Bond Object ends.
Screen 6 contains 3 display functions (outside
the bond object) which use the getter functions to
access information in the Bond Object.

    In Screen 7 the instantiation of 4 Bond Objects
are hard coded for reasons discussed in Notes 1
and 2 below.

    I have attached a file "AdditionalProjectCodeSnippets.txt"
which has some keyboard input samples filling in the
various data types discussed in class.  There are also
some code snippets for the zip function, datetime/days to
maturity, pickle, formatting, input, and data types.

#---------------------------------------------
# Summary Notes:
#---------------------------------------------

Note 1:
*******

    I have Spyder 3 installed on my system but it
has been running slow and is not opening files
correctly as shown in Note 2 below.

    I have also been using pythontutor.com, but
there is a limit there of about 130 lines or so.

    I have provided a file "SampleCSVFile.csv"
for the same four bonds that are hard coded as
a workaround.  I have also been using single spacing
omitting the comments at the top, and omitting
keyboard inputs for purposes of working with
pythontutor.com.

Note 2:
*******
    It may be necessary to install Spyder on a
```

new computer to obtain a working file open.

File opening issue:

```
            ^
SyntaxError: (unicode error) 'unicodeescape' codec can't
 decode bytes in position 2-3: truncated \UXXXXXXXX escape
```

```
#---------------------------------------------
#---------------------------------------------
# AdditionalProjectCodeSnippets.txt
#---------------------------------------------
# Term Project Assignment From: John Stark
# Course: MET CS 521
# Date: 10/20/2018
#---------------------------------------------
# The purpose of these code snippets
# is to load data into the various Python
# data types:
#---------------------------------------------
#---------------------------------------------
# Build a String
inp = input ("Build a string:")
tokens = inp.split()
String1 ="eval (x) for x in tokens"
print ("String1= ", String1)

#---------------------------------

# Build a List
inp = input ("Build a List:")
tokens = inp.split()
List1 = [eval (x) for x in tokens]
print ("List1 =", List1)

#---------------------------------

# Build a Tuple
inp = input ("Build a Tuple:")
tokens = inp.split()
#Build a tuple
Tuple1=(eval (x) for x in tokens)
print ("Tuple1= ", Tuple1)

#---------------------------------

# Build a Set
inp = input ("Build a Set:")
tokens = inp.split()
#Build a set
Set1 ={eval (x) for x in tokens}
print ("Set1= ", Set1)

#---------------------------------

# Portable Filename Construction
# Read a file into a list
# Module 3
#
import os
fileName = os.path.join('C:\' , 'Users', 'John', 'Desktop')
print (fileName)
textFile = open (fileName, "r")
lines = textFile.readlines()
textFile,close()
print (lines)

#---------------------------------

# Enumerate
# Module 3
x_list = x [1, 5, 10, 15, 20, 25, 30, 2, 3, 4]
for i, elem in enumerate(x_list):
```

```
     if (elem > 15):
         print (i, elem)
```

Test Case:

```
runfile('C:/Users/John/.spyder-py3/temp.py', wdir='C:/Users/John/.spyder-py3')
4 20
5 25
6 30
```

```
#---------------------------------
```

```
# Zip function Uses two lists to build a dictionary
# Module 4
#
x_values = ['Lion',' Tiger', 'Wolfhound']
x_keys = [3,4,5]
x_tuples = zip (x_keys, x_values)
x = dict (x_tuples)
print (x)
# Checking key membership
#Module 4
y = x[6]
```

Test Case:

```
{3: 'Lion', 4: ' Tiger', 5: 'Wolfhound'}
KeyError: 6
```

```
#---------------------------------
```

```
# Days to Maturity - two datetimes
#
from datetime import datetime
start = datetime(2018,1,1,0,0)
maturity = datetime(2019,1,1,0,0)
difference  = maturity - start
#
print ("Days to maturity: ", difference.days)
```

Test Run:

```
runfile('C:/Users/John/.spyder-py3/temp.py', wdir='C:/Users/John/.spyder-py3')
Days to maturity:  365
```

```
#---------------------------------
```

```
# Use pickle to copy an object to a file
# See Notes 1 & 2 in the Project Summary

import pickle
emp = {1:"Cat",2:"Dog",3:"Mouse",4:"Chipmunk",5:"Squirrel"}
pickling_on = open("c:\Users\John\temp.pickle","wb")
pickle.dump(emp, pickling_on)
pickling_on.close()
```

Test Case:

```
Traceback (most recent call last):

  File "C:\Users\John\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py",
line 2963, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)

  File "<ipython-input-13-b2ea6b67a0f4>", line 1, in <module>
    runfile('C:/Users/John/.spyder-py3/temp.py', wdir='C:/Users/John/.spyder-py3')

  File
```

```
"C:\Users\John\Anaconda3\lib\site-packages\spyder\utils\site\sitecustomize.py", line
705, in runfile
    execfile(filename, namespace)

  File
"C:\Users\John\Anaconda3\lib\site-packages\spyder\utils\site\sitecustomize.py", line
102, in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

  File "C:/Users/John/.spyder-py3/temp.py", line 3
    pickling_on = open("c:\Users\John\temp.pickle","wb")
                       ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position
2-3: truncated \UXXXXXXXX escape

#---------------------------------

# Old and New Style Formatting:
#
old = '%s %s' % ('Boston', 'MA')
new = '{} {}'.format('Burlington', 'VT')
print (old)
print (new)

Test Case:
Boston MA
Burlington VT

#---------------------------------

def __str__(self):
        return '{} {}'.format('Bond: ', self.__id__)

Test Cases:
print (bond1)
Bond:  1122

print (bond2)
Bond:  1123

print (bond3)
Bond:  1124

print (bond4)
Bond:  1125

#---------------------------------

with open ('example.csv') as csv file
readCSV = CSV.reader(CSVfile, delimiter=',')
        print csvfile
        for row in readCSV:
                print (row)


Test Case:

#---------------------------------
```

MET CS 521 Information Structures with Python

Prerequisite: MET CS 300. Or, Instructor's consent.
Delivery: Boston-Charles River Campus, Online
Program: MSSD Core, MSCIS Core option
Syllabus: CS521_C1_Fall 2017

Description: This course covers the concepts of the object-oriented approach to software design and development using the Python programming language. It includes a detailed discussion of programming concepts starting with the fundamentals of data types, control structures methods, classes, arrays and strings, and proceeding to advanced topics such as inheritance and polymorphism, creating user interfaces, exceptions and streams. Upon completion of this course students will be capable of applying software engineering principles to design and implement Python applications that can be used in conjunction with analytics and big data. (4 credits)