Brindavan College of Engineering

Dwarakanagar, Bagalur Main Road, Bengaluru-560063
Department of Information Science and Engineering

Fifth Semester B.E. Degree Examination
Application Development Using Python[18CS55]
Model Question Paper Solution 2020-21

Module 1

Q1

a) List the salient features of python programming language.

Solution

- Easy to **code**: **Python** is a high-level **programming language**. ...
- Free and Open Source: ...
- Object-Oriented **Language**: ...
- GUI **Programming** Support: ...
- High-Level **Language**: ...
- Extensible **feature**: ...
- **Python** is Portable **language**: ...
- **Python** is Integrated **language**:

b) What are the different flow control statements supports in python .Explain any 3 with an suitable example program and flow chart.

Solution:

1.10.1 If statement
1.10.2 else statement
1.10.3 elif statement
1.01.4 while loop statements
1.10.5 an annoying while loop
1.10.6 break statements
1.10.7 continue statements
1.10.8 for loos and range function

1.10.1 If statement

Here if is the keyword
This is also known as simple if statement
It always checks only for the true conditions
In Python, if statement consists of the following:

- The if keyword
- A condition (that is, an expression that evaluates to True or False)
- A colon
- Starting on the next line, an indented block of code (called the if clause)

Example

If name == ‘Alice’:

print(‘Hi, Alicej’)

1.10.2 else statement

- An if clause can optionally be followed by an else statement.
- The else clause is executed only when the if statement’s condition is False.

An else statement always consists of the following:

- The else keyword
- A colon
- Starting on the next line, an indented block of code (called the else clause)

```
if name == 'Alice':
print('Hi, Alice.')
else:
print('Hello, stranger.')
```

1.10. 3 elif Statements

- The elif statement is an “else if” statement that always follows an if or another elif statement. It provides another condition that is checked only if any of the previous conditions were False.

In code, an elif statement always consists of the following:

- The elif keyword
- A condition (that is, an expression that evaluates to True or False)
- A colon
- Starting on the next line, an indented block of code (called the elif clause)

Let’s add an elif to the name checker to see this statement in action.

```
if name == 'Alice':
    print('Hi, Alice.')
elif age < 12:
    print('You are not Alice, kiddo.')
```

1C) Write a python program to calculate the area of circle, rectangular and triangle. print the results.

Solution

```
# Write a python program to calculate the
#area of  circle, rectangular and triangle. print the results.
import math
def area_cir():
    print('Enter the radius of a circle')
    rad = float(input())
    return (math.pi*rad*rad)
def area_rect():
    print('Enter Length of a Rectangle')
    l = float(input())
    print('Enter  Breadth of a Rectangle')
    b=float(input())
    return l*b
def area_triangle():
    print('Enter the Base   of the triangle')
    b = float(input())
    print('Enter the Height of the triangle')
    h=float(input())
    return 0.5*b*h
area_circle = area_cir()
print('Area of Circle is:'+str(area_circle))
area_tri = area_triangle()
print('Area of Triangle is:'+str(area_tri))
area_rectangle = area_rect()
print('Area of Rectangle is:'+str(area_rectangle))

""" Output

Enter the radius of a circle
2.5
Area of Circle is:19.634954084936208
Enter the Base   of the triangle
3
Enter the Height of the triangle
3.4
Area of Triangle is:5.1
Enter Length of a Rectangle
4
Enter  Breadth of a Rectangle
4
Area of Rectangle is:16.0
"""
```

2 a) What is a function? How to define a function in python? Write a program using function to find out the given string is palindrome or not.

Solutions:

Function are Subroutines or sub program which is defined to perform a specific task

To Define a Function in Python
A function should start using def keyword
Followed by a valid function name
Ending with colon(☺
Ex: def add() :

 # Python Program to check whether the given string is palindrome or not

```
str = input('Enter the First String to Check for Pallindrome:')
print(str)
def reverse(string):
    string = "".join(reversed(string))
    return string
rev_str=reverse(str)
def Check_Pallindrome(str, rev_str):
    if(str == rev_str):
        return 1
    else:
        return 0

check_pal = Check_Pallindrome(str,rev_str)
if(check_pal == 0):
    print('String is not a Pallindroome!')
else:
    print('String is a Pallindrome!')
```

Output

```
Enter the First String to Check for Pallindrome:121
121
String is a Pallindrome!
Enter the First String to Check for Pallindrome:123
123
String is not a Pallindroome!
```

2 b) What is local and global scope of variable in python .Explain the different scenarios with an example snippet.

Solution:

1.18 Local and Global Scope

Local	Global
• Variables which are used within the function are call local variables it is	• Variables are assigned outside the function are called global variables. It

known as local scope	is known as global scope
• Local scope is created whenever a function is called. Any variables assigned in this function exist within the local scope • Code in the global scope cannot use any local variables.	• Global variable is created outside all the functions. Exists throughout the program • Code in the local scope can use any global variables

Different scenarios

1.18.1 Local Scopes Cannot Use Variables in Other Local Scopes
1.18.2 Global Variables Can Be Read from a Local Scope
1.18. 3 Local and Global Variables with the Same Name
1.18.4 The global Statement

1.18.1 Local Scopes Cannot Use Variables in Other Local Scopes

- A new local scope is created whenever a function is called, including when a function is called from another function. Consider this program:

```
def spam():
❶eggs = 99
❷bacon()
❸print(eggs)
def bacon():
ham = 101
❹eggs = 0
❺spam()
```

- When the program starts, the spam() function is called ❺, and a local scope is created.
- The local variable eggs ❶ is set to 99.
- Then the bacon() function is called ❷, and a second local scope is created.
- Multiple local scopes can exist at the same time. In this new local scope, the local variable ham is set to 101, and a local variable eggs — which is different from the one in spam()'s local scope — is also created ❹ and set to 0.
- When bacon() returns, the local scope for that call is destroyed. The program

execution continues in the spam() function to print the value of eggs ❸, and since the local scope for the call to spam() still exists here, the eggs variable is set to 99. This is what the program prints. The upshot is that local variables in one function are completely separate from the local variables in another function.

1.18.2 Global Variables Can Be Read from a Local Scope

Consider the following program:

```
def spam():
print(eggs)
eggs = 42
spam()
print(eggs)
```

- Since there is no parameter named eggs or any code that assigns eggs a value in the spam() function, when eggs is used in spam(),
- Python considers it a reference to the global variable eggs.
- This is why 42 is printed when the previous program is run.

1.18. 3 Local and Global Variables with the Same Name

```
def spam():
❶eggs = 'spam local'
print(eggs) # prints 'spam local'
def bacon():
❷eggs = 'bacon local'
print(eggs) # prints 'bacon local'
spam()
print(eggs) # prints 'bacon local'
❸eggs = 'global'
bacon()
print(eggs) # prints 'global'
```

When you run this program, it outputs the following:

```
bacon local
spam local
bacon local
```

global

There are actually three different variables in this program, but confusingly they are all named eggs.

The variables are as follows:

❶A variable named eggs that exists in a local scope when spam() is called.

❷A variable named eggs that exists in a local scope when bacon() is called.

❸A variable named eggs that exists in the global scope.

Since these three separate variables all have the same name, it can be confusing to keep track of which one is being used at any given time. This is why you should avoid using the same variable name in different scopes.

1.18.4 The global Statement

- If you need to modify a global variable from within a function, use the global statement.
- If you have a line such as global eggs at the top of a function, it tells Python, "In this function, eggs refers to the global variable, so don't create a local variable with this name." For example, type the following code into the file editor and save it as *sameName2.py*:

```
def spam():
❶global eggs
❷eggs = 'spam'
eggs = 'global'
spam()
print(eggs)
```

- When you run this program, the final print() call will output this:
- Spam Because eggs is declared global at the top of spam() ❶, when eggs is set to 'spam' ❷, this assignment is done to the globally scoped spam.
- No local spam variable is created.
- There are four rules to tell whether a variable is in a local scope or global scope:
    1. If a variable is being used in the global scope (that is, outside of all functions), then it is always a global variable.
    2. If there is a global statement for that variable in a function, it is a global variable.
    3. 3. Otherwise, if the variable is used in an assignment statement in the function, it is a local variable.
    4. 4. But if the variable is not used in an assignment statement, it is a global

variable.

To get a better feel for these rules, here's an example program. Type the following code into the file editor and save it as *sameName3.py*:

```
def spam():
❶global eggs
eggs = 'spam' # this is the global
def bacon():
❷eggs = 'bacon' # this is a local
def ham():
❸print(eggs) # this is the global
eggs = 42 # this is the global
spam()
print(eggs)
```

- In the spam() function, eggs is the global eggs variable, because there's a global statement for eggs at the beginning of the function ❶.
- In bacon(), eggs is a local variable, because there's an assignment statement for it in that function ❷. In ham() ❸, eggs is the global variable, because there is no assignment statement or global statement for it in that function.
- If you run *sameName3.py*, the output will look like this:
- spam

2 c Write a python program to create a function called collatz() which reads as parameter named number. If the number is even it should print and return number//2 and if the number is odd then it should print and return 3*number+1. The function should keep calling on that number until the function returns a value 1.

Solution:

```
num = int(input('Enter the Number'))

def collatz(num):
    if(num==1):
        print('Number is:'+str(num))
        return 1
    elif(num%2 == 0):
        print('Number is even:'+str(num)) #6
        #num=num/2
        return(collatz(num/2))
```

```
        else:
            print('Number is odd:'+str(num))
            #num= (3*num+1)
            return(collatz(3*num+1))

    collatz(num)
```

Output 1:

```
Enter the Number 8
Number is even:8
Number is even:4.0
Number is even:2.0
Number is:1.0
```

Output 2:

```
Enter the Number6
Number is even:6
Number is odd:3.0
Number is even:10.0
Number is odd:5.0
Number is even:16.0
Number is even:8.0
Number is even:4.0
Number is even:2.0
Number is:1.0
```

Module 2

3 a) What is list? Explain the concept of slicing and indexing with proper examples.

Solution:

- A ***list*** is a value that contains multiple values in an ordered sequence.

Slicing and indexing

- A slice is typed between square brackets, like an index, but it has two integers separated by a colon. Notice the difference between indexes and slices.

  spam[2] is a list with an index (one integer).
  spam[1:4] is a list with a slice (two integers)

- In a slice, the first integer is the index where the slice starts.
- The second integer is the index where the slice ends.
- A slice goes up to, but will not include, the value at the second index.
- Slice evaluates to a new list value.

- Example

```
>>> spam = ['cat', 'bat', 'rat', 'elephant']
>>> spam[0:4]
['cat', 'bat', 'rat', 'elephant']
>>> spam[1:3]
['bat', 'rat']
>>> spam[0:-1]
['cat', 'bat', 'rat']
```

- As a shortcut, you can leave out one or both of the indexes on either side of the colon in the slice. Leaving out the first index is the same as using 0, or the beginning of the list.
- Leaving out the second index is the same as using the length of the list, which will slice to the end of the list. Enter the following into the interactive shell:

```
>>> spam = ['cat', 'bat', 'rat', 'elephant']
>>> spam[:2]
['cat', 'bat']
>>> spam[1:]
['bat', 'rat', 'elephant']
>>> spam[:]
['cat', 'bat', 'rat', 'elephant']
```

Changing Values in a List with Indexes

- For example, spam[1] = 'aardvark' means "Assign the value at index 1 in the list spam to the string 'aardvark'." Enter the following into the interactive shell:

```
>>> spam = ['cat', 'bat', 'rat', 'elephant']

>>> spam[1] = 'aardvark'
>>> spam
['cat', 'aardvark', 'rat', 'elephant']
>>> spam[2] = spam[1]
>>> spam
['cat', 'aardvark', 'aardvark', 'elephant']
>>> spam[-1] = 12345
>>> spam
['cat', 'aardvark', 'aardvark', 12345]
```

Q 3 b)

For a given list num=[45,22,14,65,97,72], write a python program to replace all the integers divisible by 3 with "ppp" and all integers divisible by 5 with "qqq" and replace all the integers divisible by both 3 and 5 with "pppqqq" and display the output.

Solution

```
list_num = [45,22,14,65,97,72]
print('Original list is'+str(list_num))
for i in range(len(list_num)):
   num= int(list_num[i])
   #print(num)
```

```
    if((num % 3==0) and (num % 5==0)):
        list_num[i] = 'pppqqq'
    elif(num%3 == 0):
        list_num[i]='ppp'
    elif (num%5==0):
        list_num[i]='qqq'


print('Modification list is'+str(list_num))
```

Output
Original list is[45, 22, 14, 65, 97, 72]
Modification list is['pppqqq', 22, 14, 'qqq', 97, 'ppp']

3 c) What are the different methods supports in python List. Illustrate all the methods with an example.
Solution:

Method Name	Description	Example

Index()	Finding a Value in a List with the index() Method	>>> **spam = ['hello', 'hi', 'howdy', 'he**
		>>> **spam.index('hello')**
		0 >>> **spam.index('heyas')**
		3

Append()	To add new values to a list, use the append()	>>> **spam = ['cat', 'dog', 'bat']**
		>>> **spam.append('moose')**
		>>> **spam**
		['cat', 'dog', 'bat', 'moose']

Insert()	The insert() method can insert a value at any index in the list.	>>> **spam = ['cat', 'dog', 'bat']**
		>>> **spam.insert(1, 'chicken'**)
		>>> **spam**
		['cat', 'chicken', 'dog', 'bat']

Remove()	The remove() method is passed the value to be removed from the list it is called on.	>>> **spam = ['cat', 'bat', 'rat', 'elepha**
		>>> **spam.remove('bat')**
		>>> **spam**
		['cat', 'rat', 'elephant']

Sort()	Lists of number values or lists of strings can be sorted with the sort() method	>>> **spam = [2, 5, 3.14, 1, -7]**
		>>> **spam.sort()**
		>>> **spam**
		[-7, 1, 2, 3.14, 5]

	If you need to sort the values in regular alphabetical order, pass str. lower for the key keyword argument in the sort() method call.	>>> **spam = ['a', 'z', 'A', 'Z']**
		>>> **spam.sort(key=str.lower)**
		>>> **spam**
		['a', 'A', 'z', 'Z']
		This causes the sort() function to trea items in the list as if they were lower without actually changing the values

Q4 a) What is dictionary? Illustrate with an example python program the usage of nested dictionary.

Solution:

- Like a list, ***a dictionary*** is a collection of many values.
- But unlike indexes for lists, indexes for dictionaries can use many different data types, not just integers.
- ***Indexes for dictionaries are called keys***, and a key with its associated value is called a *key-value pair*.
- In code, a dictionary is typed with braces, {}.

Nested Dictionary Example:

```
allGuests = {'Alice': {'apples': 5, 'pretzels': 12},
'Bob': {'ham sandwiches': 3, 'apples': 2},
'Carol': {'cups': 3, 'apple pies': 1}}
def totalBrought(guests, item):
numBrought = 0
❶ for k, v in guests.items():
❷ numBrought = numBrought + v.get(item, 0)
return numBrought
print('Number of things being brought:')
print(' - Apples ' + str(totalBrought(allGuests, 'apples')))
print(' - Cups ' + str(totalBrought(allGuests, 'cups')))
print(' - Cakes ' + str(totalBrought(allGuests, 'cakes')))
print(' - Ham Sandwiches ' + str(totalBrought(allGuests, 'ham sandwiches')))
print(' - Apple Pies ' + str(totalBrought(allGuests, 'apple pies')))
```

- Inside the totalBrought() function, the for loop iterates over the key-value pairs in guests ❶. Inside the loop, the string of the guest's name is assigned to k, and the dictionary of picnic items they're bringing is assigned to v. If the item parameter exists as a key in this dictionary, it's value (the quantity) is added to numBrought ❷. If it does not exist as a key, the get() method returns 0 to be added to numBrought.

4b) List out all the useful string methods which supports in python. Explain with an example for each method

Solution:

The upper(), lower(), isupper(), and islower() String Methods

```
>>> spam = spam.upper()
```

```
>>> spam
'HELLO WORLD!'
>>> spam = spam.lower()
>>> spam
'hello
 world!'
```

2.25.2 The isX String Methods

- **isalpha()** returns True if the string consists only of letters and is not blank.
- **isalnum()** returns True if the string consists only of letters and numbers and is not blank.
- **isdecimal()** returns True if the string consists only of numeric characters and is not blank.
- **isspace()** returns True if the string consists only of spaces, tabs, and new-lines and is not blank.
- **istitle()** returns True if the string consists only of words that begin with an uppercase letter followed by only lowercase letters.

```
>>> 'hello'.isalpha()
True
>>> 'hello123'.isalpha()
False
>>> 'hello123'.isalnum()
True
>>> 'hello'.isalnum()
True
>>> '123'.isdecimal()
True
>>> ' '.isspace()
True
>>> 'This Is Title Case'.istitle()
True
>>> 'This Is Title Case 123'.istitle()
True
>>> 'This Is not Title Case'.istitle()
False
>>> 'This Is NOT Title Case Either'.istitle()
```

False

c) What are the different steps in project Adding Bullets to Wiki Markup.
Solution:

Step 1: Copy and Paste from the Clipboard

Step 2: Separate the Lines of Text and Add the Star

Step 3: Join the Modified Lines

Module 3

5 a) What are regular expression? What are the different steps to be follow to use a regular expression in python.

Solution:

Definition

A sequence of symbols and characters expressing a string or pattern to be searched for within a longer piece of text.

- Import the regex module with import re.
- Create a Regex object with the re.compile() function. (Remember to use a raw string.)
- Pass the string you want to search into the Regex object's search() method. This returns a Match object. Call the Match object's group() method to return a string of the actual matched text.

5 b) List out what are the different character classes and its representation also regular expression symbol and its meaning.

Solution:

Shorthand character class	Represents
\d	Any numeric digit from 0 to 9.
\D	Any character that is *not* a numeric digit from 0 to 9.
\w	Any letter, numeric digit, or the underscore character. (Think of this as matching "word" characters.)
\W	Any character that is *not* a letter, numeric digit, or the underscore character.
\s	Any space, tab, or newline character. (Think of this as matching "space" characters.)
\S	Any character that is *not* a space, tab, or newline.

5c) Write a python program to create phone number and email address by using regular expression.

Create a Regex for Phone Numbers

1. import pyperclip, re
2. phoneRegex = re.compile(r'''(

```
    (\d{3}|\(\d{3}\))? # area code
    (\s|-|\.)? # separator
    (\d{3}) # first 3 digits
    (\s|-|\.) # separator
    (\d{4}) # last 4 digits
    (\s*(ext|x|ext.)\s*(\d{2,5}))? # extension
    )''', re.VERBOSE)

# Step 2: Create a Regex for Email Addresses

emailRegex = re.compile(r'''(
❶ [a-zA-Z0-9._%+-]+ # username
❷ @ # @ symbol
❸ [a-zA-Z0-9.-]+ # domain name
(\.[a-zA-Z]{2,4}) # dot-something
)''', re.VERBOSE)
```

Q6a) What are the key properties of a file? Explain in detail file reading/writing process wit an example of python program.
Solution:

- A file has two key properties: a *filename* (usually written as one word) and a *path*.
- The path specifies the location of a file on the computer.
- There are three steps to reading or writing files in Python.
    - Call the open() function to return a File object.
    - Call the read() or write() method on the File object.
    - Close the file by calling the close() method on the File object.

1 Opening Files with the open() Function

- helloFile = open('C:\\Users*your_home_folder*\\hello.txt')

2. Reading the Contents of Files

- helloContent = helloFile.read()

3. Writing to Files

```
>>> baconFile = open('bacon.txt', 'w')
>>> baconFile.write('Hello world!\n')
>>> baconFile.close()
>>> baconFile = open('bacon.txt')
>>> content = baconFile.read()
>>> baconFile.close()
>>> print(content)
Hello world!
```

6b) Explain in briefly, What are the different methods of file operations supports in python shutil module.

Solution:

- The shutil (or shell utilities) module has functions to let you copy, move, rename, and delete files in your Python programs.
- **Copying Files and Folders**

```
>>> import shutil, os
>>> os.chdir('C:\\')
❶ >>> shutil.copy('C:\\spam.txt', 'C:\\delicious')
'C:\\delicious\\spam.txt'
❷ >>> shutil.copy('eggs.txt', 'C:\\delicious\\eggs2.txt')
'C:\\delicious\\eggs2.txt'
```

- **Moving and Renaming Files and Folders**

```
>>> import shutil
>>> shutil.move('C:\\bacon.txt', 'C:\\eggs')
'C:\\eggs\\bacon.txt'
```

- **Permanently Deleting Files and Folders**

```
import os
for filename in os.listdir():
if filename.endswith('.rxt'):
os.unlink(filename)
```

6c) Write a python program to create a folder PYTHON and under the hierarchy 3 files file1,file2 and file3.write the content in file1 as "VTU" and in file2 as "UNIVERSITY" and file3 content should be by opening and merge of file1 and file2. Check out the necessary condition before write file3.

Solution:

```
import os
os.makedirs('C:\\PYTHON')

str_file1 ='VTU'
str_file2 ='UNIVERSITY'

#Writing the contents to files
def writefile():
file1 = open('C:\\PYTHON\\file1.txt','w')
file2 = open('C:\\PYTHON\\file2.txt','w')
file1.write(str_file1)
file2.write(str_file2)
file1.close()
file2.close()
#reading the contents of the file and merge
def read_merge_file():
```

```
15. file1 = open('C:\\PYTHON\\file1.txt')#Here file is opened in read mode
16. readcontent1 = file1.read()
17. file2 = open('C:\\PYTHON\\file2.txt')#Here file is opened in read mode
18. readcontent2 = file2.read()
19. #merging the contents
20. if(os.path.isdir('C:\\PYTHON')):
21. file3 = open('C:\\PYTHON\\file3.txt','w')
22. file3.write(readcontent1)
23. file3.write ('\n')
24. file3.write(readcontent2)
25. file1.close()
26. file2.close()
27. file3.close()
28. print('Merging files completed....')
29. else:
30. print('Directory Not Found or Path Mismatch')
31. writefile()
32. read_merge_file()
```

Module – 4

7a) What is a class? How to define class in python? How to initiate a class and how the class members are accessed?

Solution:
Definition:
A programmer-defined type is also called a **class**.

Defining a Class

```
class Point:
    """Represents a point in 2-D space."""
```

- The header indicates that the new class is called Point.
- The body is a doc-string that explains what the class is for.
- You can define variables and methods inside a class definition.

```
class Student:
    """
    attributes:name, usn
    """
s1 = Student()
```

```
s1.name ='Raj'
s1.usn = '1BO18IS001'
print(s1.name)
print(s1.usn)
```

Using dot (.) operator the members of the class are accessed.

7b) Explain init and str method with an example python program.

4.13 The init method

- The init method (short for "initialization") is a special method that gets invoked when an object is instantiated.
- Its full name is **_ _init_ _** (two underscore characters, followed by init, and then two more underscores).
- An init method for the Time class might look like this:

```
# inside class Time:
def __init__(self, hour=0, minute=0, second=0):
self.hour = hour
self.minute = minute
self.second = second
```

- It is common for the parameters of __init__ to have the same names as the attributes. The statement
  - self.hour = hour

- stores the value of the parameter hour as an attribute of self.
- The parameters are optional, so if you call Time with no arguments, you get the default values.

```
>>> time = Time()
>>> time.print_time()
00:00:00
If you provide one argument, it overrides hour:
>>> time = Time (9)
>>> time.print_time()
09:00:00
```

- If you provide two arguments, they override hour and minute.
  - >>> time = Time(9, 45)
  - >>> time.print_time()
  - 09:45:00

- And if you provide three arguments, they override all three default values

- As an exercise, write an init method for the Point class that takes x and y as optional parameters and assigns them to the corresponding attributes.

4.14 The __str__ method

- __str__ is a special method, like __init__, that is supposed to return a string representation of an object.
- For example, here is a str method for Time objects:
- # inside class Time:

```
def __str__(self):
return '%.2d:%.2d:%.2d' % (self.hour, self.minute, self.second)
```

- When you print an object, Python invokes the str method:

```
>>> time = Time(9, 45)
>>> print(time)
09:45:00
```

- When I write a new class, I almost always start by writing __init__, which makes it easier to instantiate objects, and __str__, which is useful for debugging.

As an exercise, write a str method for the Point class.

- Create a Point object and print it.

7c)Write a python program that uses datetime module within a class, takes a birthday as input and prints the age and the number of days, hours, minutes and second.

Solution:

```
import datetime

rules = {0: "Monday",
        1: "Tuesday",
        2: "Wednesday",
        3: "Thursday",
        4: "Friday",
        5: "Saturday",
        6: "Sunday"}


class Time(object):
   now = datetime.datetime.now()

   def __init__(self, year=1, month=1, day=1, hour=0, minute=0, second=0):
      self.date = datetime.datetime(year, month, day, hour, minute, second)

today = Time().now
birthday = Time(1983, 5, 26).date


def day_of_week():
```

```
    return "1) Today is %s" % rules[today.weekday()]


def birthday_stats(birthday):
    age = today.year - birthday.year
    if (birthday.month == today.month) and (birthday.day <= today.day):
        pass
    elif birthday.month < today.month:
        pass
    else:
        age -= 1

    birthday_ = Time(today.year, birthday.month, birthday.day).date
    till_birthday = str(birthday_ - today).split()

    if len(till_birthday) > 1:
        days = int(till_birthday[0])
        time = till_birthday[2].split(":")
    else:
        days = 365
        time = till_birthday[0].split(":")

    hours = time[0]
    mins = time[1]
    secs = time[2][:2]

    if (days < 0) and (days != 365):
        days = 365 + days
    elif (days == 365):
        days = 0
    else:
        days = abs(days)

    print ("2) You are %s years old; %sd:%sh:%sm:%ss until your next birthday."% (age, days,
hours, mins, secs))

print (day_of_week())
birthday_stats(birthday)
```

8a) What is a pure function? Illustrate with an example python program

Solution:

A function that does not modify any of the objects it receives as arguments. Most pure functions are fruitful.

- Here is a simple prototype of **add_time:**

```
def add_time(t1, t2):
    sum = Time()
    sum.hour = t1.hour + t2.hour
    sum.minute = t1.minute + t2.minute
    sum.second = t1.second + t2.second
    return sum
```

- The function creates a new Time object, initializes its attributes, and returns a reference to the new object.
- This is called a **pure function** because it does not modify any of the objects passed to it as arguments and it has no effect.

8b) Define polymorphism. Demonstrate polymorphism with function to find histogram to count the numbers of times each letters appears in a word and in sentence.

Solution:

Polymorphism is a technique where the same methods which can perform many tasks.

```
# Python3 code to demonstrate
# each occurrence frequency using
# naive method
# initializing string
test_str = "GeeksforGeeks"
# using naive method to get count
# of each element in string
all_freq = {}
for i in test_str:
   if i in all_freq:
      all_freq[i] += 1
   else:
      all_freq[i] = 1
# printing result
print ("Count of all characters in GeeksforGeeks is :\n "+str(all_freq))
```

8 C)What is type based dispatch? Illustrate with python program.

- The following is a version of __add__ that checks the type of other and invokes either add_time or increment:

```
# inside class Time:
def __add__(self, other):
        if isinstance(other, Time):
                return self.add_time(other)
        else:
        return self.increment(other)
def add_time(self, other):
```

```
            seconds = self.time_to_int() + other.time_to_int()
            return int_to_time(seconds)
    def increment(self, seconds):
            seconds += self.time_to_int()
            return int_to_time(seconds)
```

- The built-in function isinstance takes a value and a class object, and returns True if the value is an instance of the class.
- If other is a Time object, __add__ invokes add_time. Otherwise it assumes that the parameter is a number and invokes increment. This operation is called a **type-based dispatch** because it dispatches the computation to different methods based on the type of the arguments.
- Here are examples that use the + operator with different types

```
>>> start = Time(9, 45)
>>> duration = Time(1, 35)
>>> print(start + duration)
11:20:00
>>> print(start + 1337)
10:07:17
```

Unfortunately, this implementation of addition is not commutative. If the integer is the
first operand, you get

```
>>> print(1337 + start)
```

Module 5

9 a) What is web scraping? how to download files from web, check the error and save the downloaded files to hard drive with request module in python.

Solution:

- *Web scraping* is the term for using a program to download and process content from the Web. For example, Google runs many web scraping programs to index web pages for its search engine.

```
>>> import requests
❶>>>res=
requests.get('http://www.gutenberg.org/cache/epub/1112/pg1112.
txt')
>>> type(res)
<class 'requests.models.Response'>
❷ >>> res.status_code == requests.codes.ok
True
>>> len(res.text)
178981
>>> print(res.text[:250])
```

Checking for Errors

A simpler way to check for success is to call the raise_for_status() method on the Response object

```
>>> res = requests.get('http://inventwithpython.com/page_that_does_not_exist')
>>> res.raise_for_status()
```

To write the web page to a file, you can use a for loop with the Response object's

1. **iter_content() method.**
2. **>>> import requests**
3. **>>> res = requests.get('http://www.gutenberg.org/cache/epub/1112/pg1112.txt')**
4. **>>> res.raise_for_status()**
5. **>>> playFile = open('RomeoAndJuliet.txt', 'wb')**
6. **>>> for chunk in res.iter_content(100000):**
7. **playFile.write(chunk)**
8. **100000**
9. **78981**
10. **>>> playFile.close()**

- The iter_content() method returns "chunks" of the content on each iteration through the loop. Each chunk is of the *bytes* data type, and you get to specify how many bytes each chunk will contain.
- One hundred thousand bytes is generally a good size, so pass 100000 as the argument to iter_content().
- The file *RomeoAndJuliet.txt* will now exist in the current working directory. Note that while the filename on the website was *pg1112.txt*, the file on your hard drive has a different filename.

9 b) Explain in details how to parse HTML with the BeautifulSoup.

Solution:

- Beautiful Soup is a module for extracting information from an HTML page (and is much better for this purpose than regular expressions).
- The BeautifulSoup module's name is bs4 (for Beautiful Soup, version 4).

Example.html

```
<!-- This is the example.html example file. -->
<html><head><title>The Website Title</title></head>
<body>
<p>Download my <strong>Python</strong> book from <a href="http://
inventwithpython.com">my website</a>.</p>
<p class="slogan">Learn Python the easy way!</p>
<p>By <span id="author">Al Sweigart</span></p>
</body></html>
```

Creating a BeautifulSoup Object from HTML

```
1. >>> import requests, bs4
2. >>> exampleFile = open('example.html')
3. >>> exampleSoup = bs4.BeautifulSoup(exampleFile)
4. >>> type(exampleSoup)
5. <class 'bs4.BeautifulSoup'>
```

9 c) How to work with Excel spreadsheet in python. Explain briefly.

- Excel is a popular and powerful spreadsheet application for Windows.
- The openpyxl module allows your Python programs to read and modify Excel spreadsheet files.

Opening Excel Documents with OpenPyXL

```
1. >>> import openpyxl
2. >>> wb = openpyxl.load_workbook('example.xlsx')
3. >>> type(wb)
```

Getting Sheets from the Workbook

```
1. >>> import openpyxl
2. >>> wb = openpyxl.load_workbook('example.xlsx')
3. >>> wb.get_sheet_names()
   ['Sheet1', 'Sheet2', 'Sheet3']
4. >>> sheet = wb.get_sheet_by_name('Sheet3')
5. >>> sheet
   <Worksheet "Sheet3">
6. >>> type(sheet) <class 'openpyxl.worksheet.worksheet.Worksheet'>
7. >>> sheet.title
   'Sheet3'
8. >>> anotherSheet = wb.get_active_sheet()
9. >>> anotherSheet
   <Worksheet "Sheet1">
```

Getting Cells from the Sheets

```
1. >>> import openpyxl
2. >>> wb = openpyxl.load_workbook('example.xlsx')
3. >>> sheet = wb.get_sheet_by_name('Sheet1')
4. >>> sheet['A1']
   <Cell Sheet1.A1>
5. >>> sheet['A1'].value
   datetime.datetime(2015, 4, 5, 13, 34, 2)
6. >>> c = sheet['B1']
7. >>> c.value
```

'Apples'

10 a) How to work with PDF document in python. Explain with extracting text, decrypting, creating copying pages, encrypting PDF.s

Solution:

PDF stands for *Portable Document Format* and uses the *.pdf* file extension

Extracting Text from PDFs

1. **>>> import PyPDF2**
2. **>>> pdfFileObj = open('meetingminutes.pdf', 'rb')**
3. **>>> pdfReader = PyPDF2.PdfFileReader(pdfFileObj)**
4. ❶ >>> pdfReader.numPages
5. 19
6. ❷ **>>> pageObj = pdfReader.getPage(0)**
7. ❸ **>>> pageObj.extractText()**

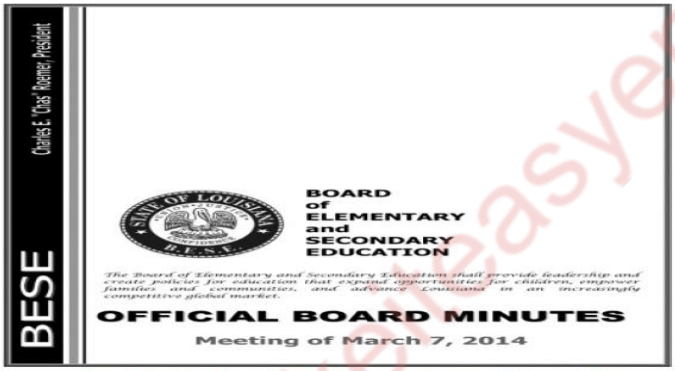

Figure 13-1. The PDF page that we will be extracting text from

5.29 Decrypting PDFs

1. >>> **import PyPDF2**
2. >>> **pdfReader = PyPDF2.PdfFileReader(open('encrypted.pdf', 'rb'))**
3. ❶ >>> **pdfReader.isEncrypted**
4. True
5. >>> **pdfReader.getPage(0)**

- ❷ Traceback (most recent call last):
- File "<pyshell#173>", line 1, in <module>
- pdfReader.getPage()
- *--snip--*
- File "C:\Python34\lib\site-packages\PyPDF2\pdf.py", line 1173, in getObject raise utils.PdfReadError("file has not been decrypted") PyPDF2.utils.PdfReadError: file has not been decrypted

❸ >>> **pdfReader.decrypt('rosebud')**
1

- >>> **pageObj = pdfReader.getPage(0)**

Creating PDFs

- PyPDF2's counterpart to PdfFileReader objects is PdfFileWriter objects, which can create new PDF files. But PyPDF2 cannot write arbitrary text to a PDF like Python can do with plaintext files. Instead, PyPDF2's PDF-writing capabilities are limited to copying pages from other PDFs, rotating pages, overlaying pages, and encrypting files.

Copying Pages

1. >>> **import PyPDF2**
2. >>> **pdf1File = open('meetingminutes.pdf', 'rb')**
3. >>> **pdf2File = open('meetingminutes2.pdf', 'rb')**
4. ❶ >>> **pdf1Reader = PyPDF2.PdfFileReader(pdf1File)**
5. ❷ >>> **pdf2Reader = PyPDF2.PdfFileReader(pdf2File)**
6. ❸ >>> **pdfWriter = PyPDF2.PdfFileWriter()**
7. >>> **for pageNum in range(pdf1Reader.numPages):**
8. ❹ **pageObj = pdf1Reader.getPage(pageNum)**
9. ❺ **pdfWriter.addPage(pageObj)**
10. >>> **for pageNum in range(pdf2Reader.numPages):**
11. ❻ **pageObj = pdf2Reader.getPage(pageNum)**
12. ❼ **pdfWriter.addPage(pageObj)**
13. ❽ >>> **pdfOutputFile = open('combinedminutes.pdf', 'wb')**
14. >>> **pdfWriter.write(pdfOutputFile)**
15. >>> **pdfOutputFile.close()**
16. >>> **pdf1File.close()**
17. >>> **pdf2File.close()**

- Open both PDF files in read binary mode and store the two resulting File objects in pdf1File and pdf2File. Call PyPDF2.PdfFileReader() and pass it pdf1File to get a PdfFileReader object for *meetingminutes.pdf* ❶. Call it again and pass it pdf2File to get a PdfFileReader object for *meetingminutes2.pdf* ❷.
- Then create a new PdfFileWriter object, which represents a blank PDF document ❸. Next, copy all the pages from the two source PDFs and add them to the PdfFileWriter object. Get the Page object by calling getPage() on a PdfFileReader object ❹. Then pass that Page object to your PdfFileWriter's addPage() method ❺.
- These steps are done first for pdf1Reader and then again for pdf2Reader. When you're done copying pages, write a new PDF called *combinedminutes.pdf* by passing a File object to the PdfFileWriter's write() method

Encrypting PDFs

1. A PdfFileWriter object can also add encryption to a PDF document. Enter the following

2. into the interactive shell:
3. >>> **import PyPDF2**
4. >>> **pdfFile = open('meetingminutes.pdf', 'rb')**
5. >>> **pdfReader = PyPDF2.PdfFileReader(pdfFile)**
6. >>> **pdfWriter = PyPDF2.PdfFileWriter()**
7. >>> **for pageNum in range(pdfReader.numPages):**
8. **pdfWriter.addPage(pdfReader.getPage(pageNum))**
9. ❶ >>> **pdfWriter.encrypt('swordfish')**
10. >>> **resultPdf = open('encryptedminutes.pdf', 'wb')**
11. >>> **pdfWriter.write(resultPdf)**
12. >>> **resultPdf.close()**

- Before calling the write() method to save to a file, call the encrypt() method and pass it a password string ❶. PDFs can have a *user password* (allowing you to view the PDF) and an *owner password* (allowing you to set permissions for printing, commenting, extracting text, and other features). The user password and owner password are the first and second arguments to encrypt(), respectively. If only one string argument is passed to encrypt(), it will be used for both passwords.

10 b) What is CSV and JSON files? Explain with an example program the usage of json module in python.

Solution:

- Python also comes with the special csv and json modules, each providing functions to help you work with these file formats. CSV stands for "comma-separated values," and CSV files are simplified spreadsheets stored as plaintext files. Python's csv module makes it easy to parse CSV files.
- (JSON is short for JavaScript Object Notation.) You don't need to know the JavaScript programming language to use JSON files, but the JSON format is useful to know because it's used in many web applications.

Reading JSON with the loads() Function

1. >>> **stringOfJsonData = '{"name": "Zophie", "isCat": true,**
   "miceCaught": 0,
2. **"felineIQ": null}'**
3. >>> **import json**
4. >>> **jsonDataAsPythonValue = json.loads(stringOfJsonData)**
5. >>> **jsonDataAsPythonValue**

After you import the json module, you can call loads() and pass it a string of JSON data.

Note that JSON strings always use double quotes. It will return that data as a Python dictionary. Python dictionaries are not ordered, so the key-value pairs may appear in a

different order when you print jsonDataAsPythonValue

Writing JSON with the dumps() Function

```
1. >>> pythonValue = {'isCat': True, 'miceCaught': 0, 'name': 'Zophie',
2. 'felineIQ': None}
3. >>> import json
4. >>> stringOfJsonData = json.dumps(pythonValue)
5. >>> stringOfJsonData
6. '{"isCat": true, "felineIQ": null, "miceCaught": 0, "name": "Zophie" }'
```