Chapter 4 Using Python Libraries

Ex 4.1 Solutions

1. Which operator is used in Python to import all modules from packages?

(a) . operator
(b) operator
(c) symbol
(d) , operator

(b) * operator

2. Which file must be part of the folder containing Python module files to make it importable python package?

(a) init.py
(b) _ setup_.py
(c) _init_.py
(d) setup.py

(c) __init__.py

3. In python which is the correct method to load a module math?

(a) include math
(b) import math
(c) #<include> math.h
(d) using math

(b) import math

4. Which is the correct command to load just the tempc method from a module called usable?

(a) import usable, tempc
(b) import tempc from usable
(c) from usable import tempc
(d) import tempc

(c) from usable import tempc

5. What is the extension of Python library modules?

(a) .mod (b) .1ib (c) .code (d) .py

(d) .py

Back Exercise Part A

3. What is a package? How is a package different from module?

A package is a collection of Python modules under a common namespace, created by placing different modules on a single directory along with special files (such as __init__ .py). It is different from a module as it is a collection of modules rather than the module itself.
e.g.
pkg folder
__init__.PY
modl . py
mod2. PY

4. What is a library? Write procedure to create own library in Python.

A python library is a reusable chunk of code that is used in program/script using import command. A package is a library if it is installable or gets attached to site-packages folder of python installation. The line between a package and a python library is quite blurred and both these terms are often used interchangeably. Procedure to create a python library is as follows:
i) Create the directory structure having folders with names of packages and sub packages.
ii) Create __init__.py files in package and sub package folders.
iii) Associate it with python installation.

5. What is the use of file __init__.py in a package even when it is empty?

In a directory structure, in order for a folder (containing different modules i.e., .py files) to be recognized as a package, the file __init__.py must also be stored in the folder, even if the file is empty.

6. What is the importance of site-packages folder of Python installation?

site-packages is the target directory of manually built python packages. When you build and install python packages from source (using distutils, probably by executing python setup . py install), you will find the installed modules in site-packages by default. site-packages is by default part of the python search path, so modules installed there can be imported easily afterwards.

7. How are following import statements different?

(a) import X
This imports the module and to use any of the objects in this module, we will need to use the module name as prefix. For instance if the module has a function called func(). It will be called as X. func()
(b) from X import *
This imports all the objects from the module X. We will not need to use the name prefix with any of the objects. e.g. func() # can be used to call the func() function from the module
(c) from X import a, b, c
Only the objects a, b & c have been imported from the X module. We cannot call any other object even with the name prefix.

8. What is PYTHON PATH variable? What is its significance?

PYTHON PATH variable is a list of directories that the python interpreter will look in when importing modules. Its significance is that only those modules which exist in the directories of PYTHON PATH variable can be imported in a programme. To see the PYTHON PATH variable

```
import sys
print (sys.path) # gives the PYTHON PATH variable
```

9. In which order Python looks for the function/module names used by you.

The python interpreter first searches for a built- in module with the name of the imported module. If not found, it then searches for the file in a list of directories called PYTHON PATH. The PYTHON PATH can be seen by printing sys.path

10. What is the usage of help () and dir () functions.

The help() function in python can be used to see the documentation or docstrings of the module passed as argument. For instance help(math) prints something like this

```
# This is only a portion of the output given from help(math)
Help on module math:
NAME
        math
MODULE REFERENCE
        https://docs.python.org/3.7/library/math
```

The following documentation is automatically generated from the Python source files. It may be incomplete, incorrect or include features that are considered implementation detail and may vary between Python implementations. When in doubt, consult the module reference at the location listed above.

```
DESCRIPTION
This module is always available. mathematical functions defined by It provides access
the C standard.
```

The Python dir () function, with arguments, tries to return the list of valid attributes for that object and without with the argument and without an argument dir () returns the list of names in the current local scope For instance dir ('a') will return

```
__add__, __eq__, __getnewargs__, __iter__, __new__, __setattr__, __class__,
__format__, __gt__, __reduce__, __sizeof__, __contains__, __ge__, __hash__, __len__,
__delattr__, __getattribute__, __It__, __init__, __mod__, __repr__, __subclasshook__,
__dir__, __getitem__, __init__, __subclass__, __mul__, __rmod__, __doc__, __ne__,
__rmul__, __reduce__ex__, __str__,
' capitalize' , `casefold', 'center', 'count', 'encode', 'endswith' , ' expandtabs' , 'find', 'format',
'format map', 'index', 'isalnum', 'isalpha' , ' Isascll', ' isdecimal ' isprintable', 'isspace',
'istitle','isupper' , 'join', 'lstrip', 'maketrans', 'partition' , `replace', 'rfind', 'rindex' , 'isdigit',
'isidentifier , 'islower', 'isnumeric' , 'ljust', 'lower', 'rjust', ' rpartition• , ' rsplit', ' rstrip', ' split',
'splitlines', ' startswith' , 'strip', ' swapcase' ,'title', 'translate', 'upper' , 'zfill']
```

13. Why should the from import statement be avoided to import objects?

from <module> import <object> should be avoided because it can lead to name clashes. For instance, if the programme uses a object name which is also being imported, the programme will not be able to use that imported object. e.g. from math import log

```
def log (x):
    return x*x*x
print (log(10)) # it will return 1000 rather than returning actual log
```

14. What do you understand by standard library of Python?

The Python Standard Library is a collection of script modules accessible to a Python program to simplify the programming process and removing the need to rewrite commonly used commands. They can be used by 'calling/importing' them at the beginning of a script. Some commonly used modules of Python standard library are math, cmath, random, statistics etc.

15. Explain the difference between import and from import statements, with examples.

When we use import <module> command, all the defined functions and variables created in the module are now available to the program that imported it. The objects can be called by using the module 's name as prefix before them. e.g.
import math
print (math.sin(3 . 14)) # prefix math is being used
When we use from <module> import statements, only the asked objects are imported to the programme. No prefix is required to use them. e.g.
from math import sin
print (sin(3.14))

Back Exercise Part B

1. Create module tempConversion.py as given in Fig. 8.2 in the chapter. If you invoke the module with two different types of import statements, how would the function call statement for imported module's functions be affected?

```
# tempConversion.py
"""Conversion functions between fahrenheit and centrigrade"""

# Functions
def to_centigrade(x):
"""Returns: x converted to centigrade"""
        return 5*(x-32)/9.0

def to_fahrenheit(x):
"""Returns: x converted to fahrenheit"""
        return 9*x/5.0 + 32

# Constants
FREEZING_C = 0.0 # water freezing temp.(in celsius)
FREEZING_F = 32.0 # water freezing temp.(in fahrenheit)

# programme
import tempConversion
print(tempConversion.to_centigrade(10)) # prefix is used in function call

from tempConversion import to_centigrade
print(to_centigrade(1)) # no prefix needed
```

2. A function checkMain( ) defined in module Allchecks.py is being used in two different programs. In 1 as

Allchecks.checkMain(3, 'A')

and in program 2 as

checkMain(4, 'Z)

Why are these two function-call statements different from one another when the function being invoked is just the same ?

In program 1, the import statement must have been
import Allchecks
while in program 2, it must have been
from Allchecks import checkMain

3. Given below is semi-complete code of a module basic.py :

```
    def square(x):
      return mul(x, x)


#
"""Module for basic calculations! “/”"
def square (x) :
"""Returns squa re of a number"" "
        return mul(x, x)
def mul (x, y):
"""'Returns Product of t wo numbers"""
        return x*y
def div (x, y):
""" Returns float (x)/y""
        return float (x)/y
def fdiv (x, y) :
“/”” Returns x//y"""
        return x//y
def floordiv (x, y):
        return fdiv(x, y)
```

6. Suppose that after. we import the random module, we define the following function called diff in a Python session:

```
def diff():
      x= random. random() - random. random ()
      return (x)
```

What would be the result if you now evaluate?

```
y = diff()
print(y)
```

At the Python prompt? Give reasons for your answers.

```
import random
def diff():
        x = random. random() - random. random( )
        return (x)
y = diff()
print (y)
```

The output will be between -1 & 1 because random. random() returns a value between 0 and 1.

5. Import the above module basics.py and write statements for the following:

(a) Compute square of 19.23
(b) Compute floor division of 1000.01 with 100.23
(c) Compute product of 3, 4 and 5. (Hint. use a function multiple times).
(d) What is the difference between
basics. div (100, 0) and basics.div(0, 100)?

```
import basics
a) basics. square(19.23)
b) basics. floordiv(IOOO.OI, 100.23)
c) basics.mul(basic.mul(3, 4), 5)
d) basics. div (100, 0) will return error because divisor is 0 where as
basics.div(0, 100) will return 0 without any error]
```

4. After importing the above module, some of its functions are executed as per following statements. Find errors, if any:

(a) square(print 3)
(b) basic . div()
(c) basic.floordiv(7 7)
(d) div(100, 0)
(e) basic.mul(3, 5)
(f) print(basic .square(3.5))
(g) z = basic.div(13, 3)

```
a) square(print 3)
# function expects an integer not another function name, from basic import square should be used
b) basic.div()
# function expects 2 arguments, none given
c) basic. floordiv(7.0, 7)
# No error
d) div(100, 0)
# division by 0 will raise error, from basic import div should be used
e) basic.muI(3, 5)
# No error
f) print(basic.square(3.5))
# No error
z = basic.div(13, 3)
#No error
```

7. What are the possible outcome(s) executed from the following code? Also specify the maximum and minimum values that can be assigned to variable NUMBER.

```
import random
STRING = " CBSEONLINE "
NUMBER = random. randint(0, 3)
N=9
while STRING[N] != 'L':
        print(STRING[N]+ STRING [NUMBER] + '#' , end =' ')
        NUMBER = NUMBER + 1
```

(i) ES#NE#IO# (ii) LE#NO#ON# (iii) NS#1E#LO# (iv) EC#NB#IS#

The 1st character would be 'E' , the 4th character would be 'N' and the 7th character would be 'I' . The 2nd, 5th & 8th characters would be contiguous
sequence of STRING. Hence the possible outcomes are
(i) ES#NE#IO# or (iv) EC#NB#IS#
The white loop will run three times and then STRING [N] will become 'L' .
Hence maximum value of NUMBER can be 3+3=6. Its minimum value can be 0 after being assigned from the randint function.

8. Consider the following code:

```
import random
print (int ( 20 + random. random () * 5 ) , end = ' `)
print (int ( 20 + random. random () * 5 ) , end = ' ` )
print (int ( 20 + random. random () * 5 ) , end = ' `)
print (int ( 20 + random . random () * 5 ) )
```

Find the suggested output options (1) to (iv). Also, write the least value and highest value that can be generated
(I) 20 22 24 25
(II) 22 23 24 25
(III) 23 24 23 24
(iv) 21 21 21 21

The least value that can be generated i s 20 when random() returns 0 . The highest value that can be generated is almost 25 when random() returns a value close to 1 .
So, the only two possible suggested outputs are:
iii) 23 24 23 24 and iv) 21 21 21 2 11

9. Consider the following code:

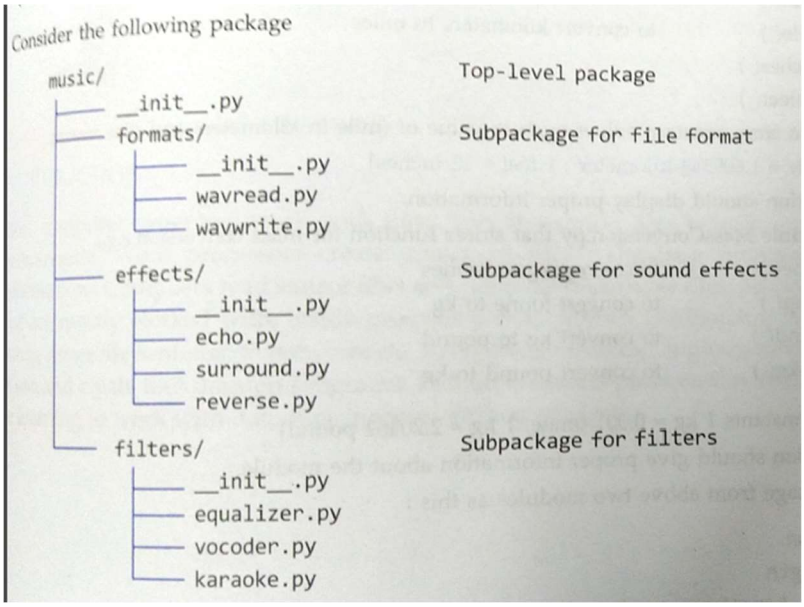
```
import random
print(100 + random. randint(5, 10), end = " " )
print(100 + random. randint(5, 10), end = " " )
print (100 * random. randint(5, 10), end = " " )
print(100 + random. randint(5, 10) )
```

Find the suggested output options ((I) to (IV). Also, write the least value and highest value that Can be generated.
(i) 102 105 104 105
(ii) 110 103 104 105
(iii) 105 107 105 110
(iv) 110 105 105 110

The least value can be 105 when randint returns 5 and the highest value can be 110 when randint returns 10.

Suggested options (iii) & (iv) are possible because they have all the numbers in the range 105 to 110.

10. Consider the following package



Each of the above modules contain functions play( ), writefile( ) and readfile()
(a) If the module wavwrite is imported using command import music.formats.wavwrite. How will you invoke its writefile( ) function? Write command for it.
(b) If the module wavwrite is imported using command from music.formats import wavwrite. How will you invoke its writefile( ) function? Write Command for it.

(a)
```
import music . formats .wavwrite
# To invoke the writefile() function
music.formats,wavwrite.writefile( )
```
(b)
```
from music,fo rmats import wavwrite
# to invoke the writefile() function
wavwrite .write file()
```

11. What are the possible outcome(s) executed from the following code? Also specify the maximum and minimum values that can be assigned to variable PICKER.

```
import random
PICK = random. randint( 0 , 3)
CITY = ( "DELHI " , "MUMBAI" , "CHENNAI" , "KOLKATA" (
for I in CITY :
for J in range ( 1 , PICK)
print (I, end = " " )
print ()
```

(i) DELHIDELHI MUMBAIMUMBAI CHENNAICHENNAI KOLKATAKOLKATA	(ii) DELHI DELHIMUMBAI DELHIMUMBAICHENNAI
(iii)DELHI MUMBAI CHENNAI KOLKATA	(iv) DELHI MUMBAIMUMBAI KOLKATAKOLKATAKOLKATA

```
import random
PICK = random. randint( 0 , 3)
CITY = [ "DELHI ", "MUMBAI", "CHENNAI", "KOLKATA"]
for I in CITY :
for J in range ( 1 , PICK)
print (I, end = " ")
print ()
```

The minimum value can be 0 and the maximum value can be 3, lower limit of range is 1. So, when PICK is 0 nothing gets printed, when pick is 1 nothing gets printed, when PICK is 2 everything gets printed once, when PICK is 3 everything gets printed twice. So (i) & (iv) are the possible outcomes.

Back Exercise Part C

1. Create a module lengthconversion.py that stores functions for various lengths conversion e.g.,

miletokm( ) to convert miles to kilometer
kmtomile( ) to convert kilometers to miles
feettoinches( )
inchestofeet( )

It should also store Constant values such as value of (mile in kilometers and vice versa).
[1 mile = 1.609314 kilometer; feet = 12 inches]

```
# lengthconversion.py
"""Length conversion module"""
def miletokm (mile) :
"""To conve rt miles to kms"""
        return mile*1.609
def kmtomile ( km) :
"""To convert kms to miles"""
        return km*0.62
```

```
def feettoinches (feet):
"""To convert feet to inches"""
        return feet* 12
def inchestofeet (inch):
"To convert inches to feets"""
        return inch*0.0833
# Constants
MILE = '1.609344 KM'
FEET = '12 INCHES'
```

help(lengthconversion) displays

```
NAME
        pbrpython - Length conversion module
FUNCTIONS
        feettotnches( feet)
                To convert feet to inches
        inchestofeet(tnch)
                TO convert inches to feets
        kmtomtte(km)
                To convert kms to miles
        miletokm(mile)
                To convert miles to kms
DATA
        FEET = '12 INCHES'
        MILE - '1.609344 KM'
FILE
```

2. Create a module MassConversion.py that stores function for mass conversion e.g.,

A kgtotonne( )	to convert kg to tonnes
tonnetokg( )	to convert tonne to kg
kgtopound( )	to convert kg to pound
poundtokg( )	to convert pound to kg

(Also store constants 1 kg 0.001 tonne. 1 kg - 2.20462 pound)

```
# MassConversion.py
"""Mass conversion module"""
def kgtotonne (kg):
"""To convert kgs to tonnes ......
        return kg*0.001
def tonnetokg (tonne):
"""To convert tonnes to kgs"""
        return tonne*1000
def kgtopound (kg):
```

"""To convert kgs to pounds"""
        return kg*2.20462
def poundtokg (pound):
"To convert pounds to kgs"""
        return pound*0.453592
# Constants
KG = '0 .0001 TDNNE5'
POUND = '0.453952 KG5'

help(MassConversion) displays

NAME
        pbrpython - Mass conversion module
FUNCTIONS
        kgtopound(kg)
                To convert kgs to pounds
        kgtotonne(kg)
                To convert kgs to tonnes
        poundtokg(pound)
                To convert pounds to kgs
        tonnetokg ()
                To convert tonnes to kgs
DATA
        KG - '0.0001 TONNES'
        POUND '0 .453952 KGS '
FILE
:

3. Create a package from above two modules as this:

Conversion

Length

	Lengthconversion. py

Mass

	Massconversion.py

Make sure that above package meets the requirements of being a Python package. Also, you should be able to import above package and/or its modules using import command.

The directory structure would be same, just add the __init__ . py files
Conversion
__init__ . py
Length
        __init__ . py
        Lengthconversion.py
Mass
        __init__ . py
        Massconversion.py