ASSIGNMENT 7

Aim: PACKAGES AND REGULAR EXPRESSION IN PYTHON

THEORY:

PACKAGES:

- A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and subpackages and sub-subpackages.
- Modules that are related to each other are mainly put in the same package.
- When a module from an external package is required in a program, that package can be imported and its modules can be put to use.
- As application program grows larger in size with a lot of modules, develoer places similar modules in one package and different modules in different packages which makes a project (program) easy to manage and conceptually clear.
- We can also reuse our code using packages.
- A Python module may contain several classes, functions, variables, etc. whereas a Python package can contains several modules.

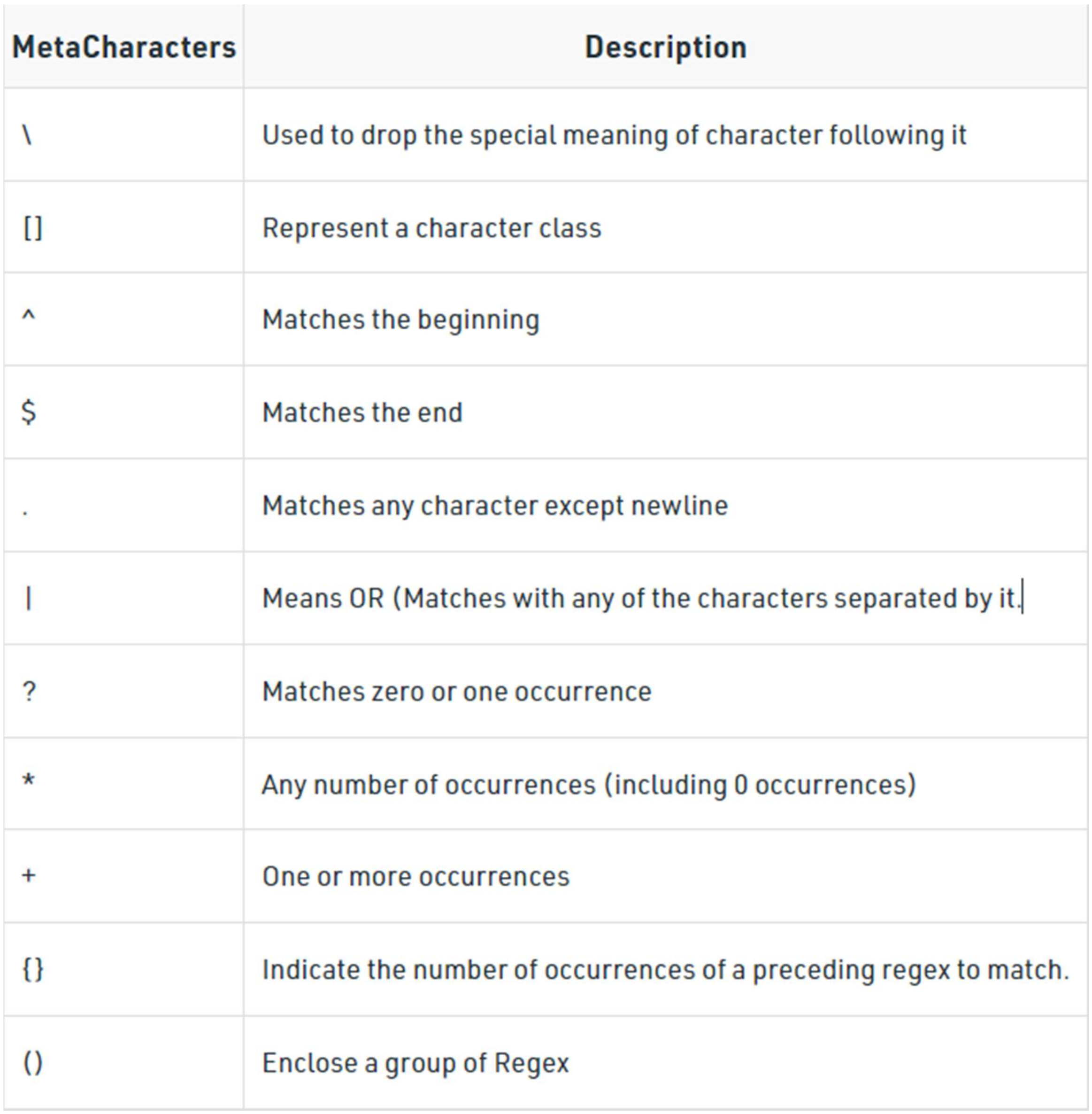
- __init__.py helps the Python interpreter to recognise the folder as package.
- It also specifies the resources to be imported from the modules.
- If the __init__.py is empty this means that all the functions of the modules will be imported.

- Importing Modules from a Package:
  syntax: import package_name.module_name

REGULAR EXPRESSIONS:

- A Regular Expressions (RegEx) is a special sequence of characters that uses a search pattern to find a string or set of strings.
- It can detect the presence or absence of a text by matching it with a particular pattern, and also can split a pattern into one or more sub-patterns.
- Python provides a re module that supports the use of regex in Python whose primary function is to offer a search, where it takes a regular expression and a string.
- The basic syntax of a regular expression pattern in Python starts with a raw string (r) followed by the pattern itself.

- Metacharacters:
  To understand the RE analogy, metacharacters are useful, important, and will be used in functions of module re.



MetaCharacters	Description
\	Used to drop the special meaning of character following it
[]	Represent a character class
^	Matches the beginning
$	Matches the end
.	Matches any character except newline
|	Means OR (Matches with any of the characters separated by it.
?	Matches zero or one occurrence
*	Any number of occurrences (including 0 occurrences)
+	One or more occurrences
{}	Indicate the number of occurrences of a preceding regex to match.
()	Enclose a group of Regex

- Special sequences:
  Special sequences do not match for the actual character in the string instead it tells the specific location in the search string where the match must occur. It makes it easier to write commonly used patterns.

Special Sequence	Description	Examples	
\A	Matches if the string begins with the given character	\Afor	for geeks
			for the world
\b	Matches if the word begins or ends with the given character. \b(string) will check for the beginning of the word and (string)\b will check for the ending of the word.	\bge	geeks
			get
\B	It is the opposite of the \b i.e. the string should not start or end with the given regex.	\Bge	together
			forge
\d	Matches any decimal digit, this is equivalent to the set class [0-9]	\d	123
			gee1
\D	Matches any non-digit character, this is equivalent to the set class [^0-9]	\D	geeks
			geek1
\s	Matches any whitespace character.	\s	gee ks
			a bc a
\S	Matches any non-whitespace character	\S	a bd
			abcd
\w	Matches any alphanumeric character, this is equivalent to the class [a-zA-Z0-9_].	\w	123
			geeKs4
\W	Matches any non-alphanumeric character.	\W	>$
			gee<>
\Z	Matches if the string ends with the given regex	ab\Z	abcdab
			abababab

- Regex Module:
  - Python has a module named re that is used for regular expressions in Python.
  - We can import this module by using the import statement.
  - re.findall() - Return all non-overlapping matches of pattern in string, as a list of strings.
  - re.compile() - Regular expressions are compiled into pattern objects, which have methods for various operations such as searching for pattern matches or performing string substitutions.

Example:

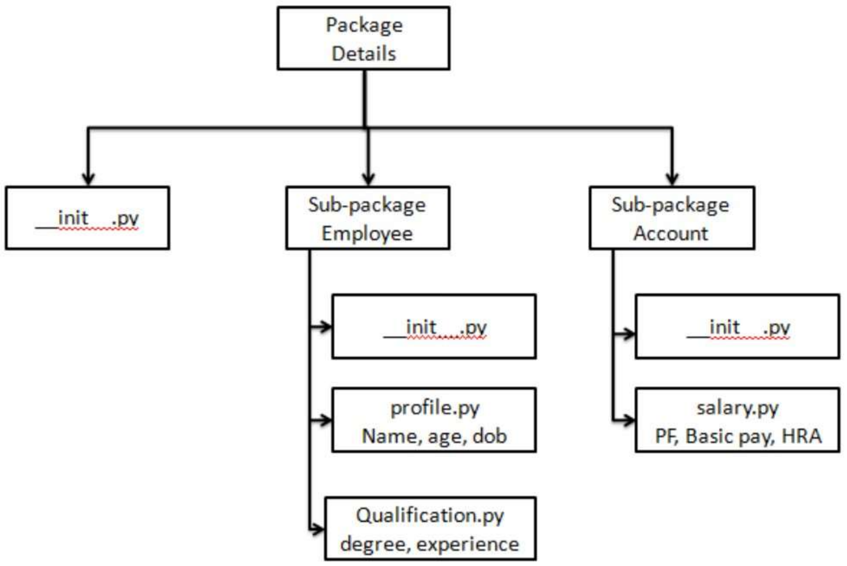
```
import re

# Define the pattern
pattern = r"\d+"

# Define the input string
text = "The price of the item is rs10.99"

# Compile the regular expression
regex = re.compile(pattern)

# Find all matches
matches = regex.findall(text)

# Print the results
print(matches)
```

Output:

```
['10', '99']
```

1.

Python program to demonstrate use of packages.


Package Details
__init__.py
Sub-package Employee
Sub-package Account
__init__.py
profile.py Name, age, dob
Qualification.py degree, experience
__init__.py
salary.py PF, Basic pay, HRA

Create a file to import all the packages , calculate the salary and display all the details of the Employee
Salary = Basic +H.R.A – P.F.

CODE:

```
# salary.py
class Salary:
  def__init_(self, basic_pay, HRA, PF):
     self.basic_pay = basic_pay
     self.HRA = HRA
     self.PF = PF
     self.salary= self.basic_pay+self.HRA-self.PF

# profile.py
class Profile:
  def__init_(self, name, age, dob):
     self.name = name
     self.age = age
     self.dob = dob

# qualification.py
class Qualification:
  def__init_(self, degree, qualification):
     self.degree = degree
     self.qualification = qualification
```

```
# main.py
from my_package.Account.salary import Salary
from my_package.Employee.qualification import qualification
from my_package.Employee.profile import Profile

n = int(input("Enter number of Employees: "))
emp_list = []
for i in range(n):
   name = input(f"Enter Name of Employee {i+1}: ")
   age = input(f"Enter Age of Employee {i+1}: ")
   dob = input(f"Enter Birth year of Employee {i+1}: ")
   degree = input(f"Enter Degree of Employee {i+1}: ")
   experience = input(f"Enter Experience of Employee {i+1}: ")
   basic = int(input(f"Enter Basic Pay of Employee {i+1}: "))
   hra = int(input(f"Enter HRA of Employee {i+1}: "))
   pf = int(input(f"Enter PF of Employee {i+1}: "))

   emp_prof = Profile(name, age, dob)
   emp_q = Qualification(degree, experience)
   emp_s = Salary(pf, basic, hra)
   new_employee = [emp_prof, emp_q, emp_s]
   emp_list.append(new_employee)
   print()
   print("Name\tAge \tDob \tDegree  Exp \tBasic  HRA \tPF  \tSalary")print('-'*75)
for emp in emp_list:
   print(emp[0].name, emp[0].age, emp[0].dob,
emp[1].degree,emp[1].qualification,emp[2].basic_pay, emp[2].HRA, emp[2].PF,
emp[2].salary, sep='\t\t')
```

OUTPUT:

```
Enter number of Employees: 2
Enter Name of Employee 1: YASH
Enter Age of Employee 1: 19
Enter Birth year of Employee 1: 2003
Enter Degree of Employee 1: BTECH
Enter Experience of Employee 1: 4
Enter Basic Pay of Employee 1: 5000
Enter HRA of Employee 1: 3000
Enter PF of Employee 1: 2000

Enter Name of Employee 2: RYAN
Enter Age of Employee 2: 20
Enter Birth year of Employee 2: 2002
Enter Degree of Employee 2: B.E
Enter Experience of Employee 2: 5
Enter Basic Pay of Employee 2: 6000
Enter HRA of Employee 2: 3000
Enter PF of Employee 2: 2000


Name   Age   Dob    Degree     Exp     Basic   HRA        PF     Salary
-------------------------------------------------------------------------

YASH    19   2003   BTECH        4     5000   3000        2000   6000

RYAN    20   2002   B.E          5     6000   3000        2000   7000


Process finished with exit code 0
```

2.

Python program to demonstrate use of regular expression

- Create a string with the name of cities in India separated by spaces.
- Find all cities ending with “ai”
- Find all cities starting with “Mu” or “Ma”
- print name of cities with ‘u’ as second letter and ‘a’ as second last letter

CODE:

```
import re
cities = "Mumbai Delhi Shanghai Dubai Kolkata Jaipur Ahmedabad Pune Hyderabad Surat
Lucknow"

# Find all cities ending with "ai"
pattern1 = re.compile(r"\w+ai\b")
ai_cities = pattern1.findall(cities)
print("Cities ending with 'ai':",
ai_cities)

# Find all cities starting with "Mu" or "Ma"
pattern2 = re.compile(r"\b(Mu|Ma)\w+")
ma_mu_cities = pattern2.findall(cities)
print("Cities starting with 'Mu' or 'Ma':", ma_mu_cities)

# Print names of cities with 'u' as second letter and 'a' as second last letter
pattern3 = re.compile(r"\b\w[u]\w*[a]\b")
ua_cities = pattern3.findall(cities)
print("Cities with 'u' as second letter and 'a' as second last letter:", ua_cities)
```

OUTPUT:

```
Cities ending with 'ai': ['Mumbai', 'Shanghai', 'Dubai']Cities
starting with 'Mu' or 'Ma': ['Mu']
ties with 'u' as second letter and 'a' as second last letter: [‘Mumbai’, ’Dubai’, ‘Surat’]
```

3.

Python program to demonstrate use of regular expression

- Create a **phone list** using file **(surname name number)**
- Find all the entries of phone book with surname as **“Rao”** and first name

starting with **‘J’ or ‘K’**

CODE:

```
import re
# Open the file containing the phone list and read the contents
with open("phonelist.rtf", "w") as file:
   # Write the name and number to the file
   file.write("Singh King 1234567890\n")
   file.write("Rao Krish 0987654321\n")
   file.write("Rao Jatin 4567891237\n")
   file.write("Rao Harsh 7766554412\n")

# Open the file in read mode
with open("phonelist.rtf", "r") as file:
   # Read the contents of the file into a string
   contents = file.read()

# Print the contents of the file
print(contents)
with open("phonelist.rtf", "r") as file:
   phonelist = file.read()

# Find all entries of phone book with surname as "Rao" and first name starting with "J" or "K"
pattern = re.compile(r"Rao\s+(J\w+|K\w+)\s+(\d{10})")
matches = pattern.findall(phonelist)

# Print the matching entries
print("Entries of phone book with surname as 'Rao' and first name starting with 'J' or 'K':")
for match in matches:
   print(match[0], match[1])
```

OUTPUT:

```
/Users/yash/PycharmProjects/pythonProject/venv/bin/python
/Users/yash/PycharmProjects/pythonProject/exp7.py Singh
King 1234567890
Rao Krish 0987654321
Rao Jatin 4567891237
Rao Harsh 7766554412

Entries of phone book with surname as 'Rao' and first name starting with 'J' or 'K':Krish
0987654321
Jatin 4567891237

Process finished with exit code 0
```