Assignment No-2.

Module-2

1. What is list? Explain append(), insert() and remove() methods with examples.

Ans A list is a value that contains multiple values in an ordered sequence. A list begins with an opening square bracket and ends with a closing square bracket

Ex: >>> [1, 2, 3]
[1, 2, 3]
>>> ['cat', 'rat', 'bat']
['cat', 'rat', 'bat']

To add new values to a list, use the append() and insert() methods.

append() - The append() method call adds the argument to the end of the list

Ex:
```
>>> spam=['cat', 'dog', 'bat']
>>> spam.append('rat')
>>> spam
['cat', 'dog', 'bat', 'rat']
```

insert() - The insert() method can insert a value at any index in the list. The first argument to insert() is the index of the new value, and the second argument is the new value to be inserted.

Ex:
```
>>>spam= ['cat', 'dog', 'bat']
>>> spam.insert(1, 'rat')
>>> spam
['cat', 'rat', 'dog', 'bat']
```

remove() - The remove() method is passed the value to

be removed from the list it is called on

Ex: >>> spam = ['cat', 'bat', 'rat', 'dog']
>>> spam.remove('bat')
>>> spam
['cat', 'rat', 'dog']

If the value appears multiple times in the list, only first instance of the value will be removed.

Ex: >>> spam = ['cat', 'bat', 'rat', 'cat', 'hat', 'cat']
>>> spam.remove('cat')
>>> spam
['bat', 'rat', 'cat', 'hat', 'cat']

How is a tuple different from a list which function is used to convert list to tuple

The tuple Data Type

The tuple data type is almost identical to the list data type, except in two ways.

Tuples are typed with parentheses '()'

```
>>> eggs = ('hello', 42, 0.5)
>>> eggs[0]
'hello'
>>> eggs[1:3]
(42, 0.5)
>>> len(eggs)
3
```

Tuples cannot have their values modified, appended or removed.

```
>>> eggs = ('hello', 42, 0.5)
>>> eggs[1] = 99
TypeError: 'tuple' object does not support item assignment
```

tuple() function is used convert list into tuple

Ex: >>> tuple(['rat', 'rat'])
('rat', 'rat')
>>> tuple([1, 2, 3])
(1, 2, 3)
>>> tuple('hello')
('h', 'e', 'l', 'l', 'o')

3. Compare and contrast lists and tuples.

Ans

| Lists | Tuples. |
|---|---|
| Similarities | |
| a. The two data structures are both sequence types that store collections of items | |
| b. Items of any data type can be stored in them | |
| c. Items can be accessed by their index | |
| Differences: | |
| a. It is mutable | It is immutable |
| b. The implication of iterations is time-consuming in the list | Implications of iteratio are much faster in tu |
| c. Operations like insertion and deletion are better performed | Elements can be acces better |
| d. Consumes more memory | Consumes less memory |
| e. Many built-in methods are available | Does not have many b -in methods. |

4. Explain the concept of tuple-comparison. How is implemented in sort() function? Discuss

Ans Comparing tuples-

The comparison operators work with tuples and

other sequences. Python starts by comparing the first element from each sequence. If they are equal, it goes on to next element, and so on until it finds elements that differ.

Ex: >>>(2, 3, 4)>(2, 2, 4)
True
>>> (2, 3, 4)>(2, 5, 4)
False
>>> (2, 3, 4)>(1, 5, 4)
True
>>> (2, 3, 4)>(2, 3, 5)
False
>>> (2, 3, 4)>(2, 3, 4)
False

The sort function works the same way. It sorts primarily by first element, but in case of a tie, it sorts by second element, and so on

Ex: >>> txt='a quick brown fox jumps right over the lazy dog'
>>> t = list()
>>> words = txt.split()
>>> for word in words:
t.append((len(word), word))
>>> t.sort(reverse=True)
>>> t
[(5, 'right'), (5, 'quick'), (5, 'jumps'), (5, 'brown'), (4, 'over'), (4, 'lazy'), (3, 'the'), (3, 'fox'), (3, 'dog'), (1, 'a')]

5. Consider the list scores = [5, 4, 7, 3, 6, 2, 1] and write the python code to perform the following operations

Ans scores = [5, 4, 7, 3, 6, 2, 1]

a. insert an element 9 at the beginning of the list

Ans
```
>>> scores.insert(0,9)
>>> scores
[9,5,4,7,3,6,2,1]
```

b. insert an element 8 at the index position 3 of the list

Ans
```
>>> scores.insert(3,8)
>>> scores
[9,5,4,8,7,3,6,2,1]
```

c. insert an element 7 at the end of the list

Ans
```
>>> scores.append(7)
>>> scores
[9,5,4,8,7,3,6,2,1,7]
```

g. Delete an element at the beginning of the list

Ans
```
>>> del scores[0]
>>> scores
[5,4,8,7,3,6,2,1,7]
```

e. Delete an element at the index position 3

Ans
```
>>> del scores[3]
>>> scores
[5,4,8,3,6,2,1,7]
```

d. Delete all the elements of the list

Ans
```
>>> del scores[0:8]
>>> scores.clear()
>>> scores
[ ]
```

6. Write a program to demonstrate Magic 8 ball game using a list

Ans
```
>>>import random
>>>messages = ['It is certain',
```

'It is decidely so',
'Yes definitely',
'Reply lazy try again',
'Ask again later',
'concentrate and ask again',
'My reply is no',
'outlook not so good',
'very doubtful']

>>>print(messages[random.randint(0, len(messages)-1)])

output:
yes definitely.

7. what are mutable and immutable datatypes?

Ans String:
A string is immutable. It cannot be changed. Trying to reassign a single character in a string results in a TypeError error.

```
>>>name = 'Zophie a cat'
>>>name[7]= 'the'
TypeError: 'str' object does not support item assignment
```

The proper way to mutate a string is to use slicing and concatenation to build a new string by copying from parts of the old string.

```
>>>name = 'Zophie a cat'
>>>newName = name[0:7] + 'the' + name[8:12]
>>> name
'Zophie a cat'
>>> newName
'Zophie the cat'
```

List:
A list value is a mutable data type. It can have values added, removed or changed.

```
>>> eggs = [1, 2, 3]
>>> eggs = [4, 5, 6]
>>> eggs
[4, 5, 6]
```

Tuple:

Tuple are immutable datatypes. Tuples cannot have their values modified, appended or removed

```
>>> eggs = (1, 2, 3)
>>> eggs[1] = 99
Type Error: 'tuple' object does not support item assignment
```

8. Explain the difference between copy() and deepcopy() methods

Ans copy() - It can be used to make a duplicate copy of a mutable value like a list or dictionary, not just a copy of a reference

Ex:
```
>>> import copy
>>> spam = ['A', 'B', 'C', 'D']
>>> id(spam)
44684232
>>> cheese = copy.copy(spam)
>>> id(cheese)
4446667
>>> cheese == spam
True
>>> id(cheese) == id(spam)
False
>>> cheese[1] = 'Hai'
>>> cheese
['A', 'Hai', 'C', 'D']
```

```
>>>spam
['A', 'B', 'C', 'D']
>>> id(cheese)
4446667
```

deepcopy() - The deepcopy() function will copy the value and references.

Ex:
```
>>> import copy
>>> spam = ['A', 'B', 'C', 'D']
>>> id(spam)
44684232
>>> cheese = copy.deepcopy(spam)
>>> id(cheese)
44684232
>>> cheese == spam
True
>>> id(cheese) == id(spam)
True
>>> cheese[1] = 'Hai'
>>> cheese
['A', 'Hai', 'C', 'D']
>>> id(cheese)
44684232
>>> spam
['A', 'Hai', 'C', 'D']
```

>>> scores

O/P:

[ ]

## Chapter-2

1. Define a Dictionary and discuss the methods get(), iitem(), keys() and values()

Ans **Dictionary:** A dictionary is a collection of many values. 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. A dictionary is typed with braces { }

Ex: >>> mycat = {'size': 'fat', 'color': 'gray', 'disposition': 'loud'}

**The get() Method:**

Dictionaries have a get() method that takes two arguments:

1. The key of the value to retrieve and
2. A fallback value to return if that key does not exist

Ex: >>> picnicitems = {'apples': 5, 'cups': 2}

>>> 'I am bringing' + str(picnicitems.get('cups', 0)) + 'cups.'

'I am bringing 2 cups.'

>>> 'I am bringing' + str(picnicitems.get('eggs', 0)) + 'eggs.'

'I am bringing 0 eggs.'

There are three dictionary methods that will return a list like values of the dictionary's keys, values, or both keys and values: keys(), values(), & items()

**keys():** It will return key value

Ex; >>> spam = {'color': 'red', 'age': 42}

```
for k in spam.keys:
        print(k)
```

o/p: 'color'
'age'

values(): It will return values

Ex: >>> spam = {'color': 'red', 'age': 42}
```
for v in spam.values():
        print(v)
```

o/p: red
42

items(): It will return keys and values

Ex: >>> spam = {'color': 'red', 'age': 42}
```
for k,v in spam.items():
        print(i)
```

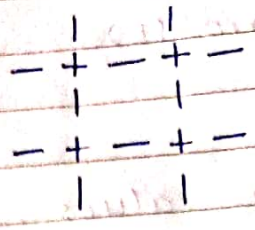
o/p: ('color': 'red')
('age': 42)

---

2. Create a function to print out a blank tic-tac-toe board

Ans
```
theBoard = {'Top_L': ' ', 'Top_M': ' ', 'Top_R': ' ',
            'Mid_L': ' ', 'Mid_M': ' ', 'Mid_R': ' ',
            'Low_L': ' ', 'Low_M': ' ', 'Low_R': ' '}
print(theBoard['Top_L'] + '|' + theBoard['Top_M'] + '|' +
      theBoard['Top_R'])
print('-+-+-')
print(theBoard['Mid_L'] + '|' + theBoard['Mid_M'] + '|' + theBoard
      ['Mid_R'])
print('-+-+-')
print(theBoard['Low_L'] + '|' + theBoard['Low_M'] + '|' + theBoard
      ['Low_R'])
```

output:



3. Explain the concept of pretty printing w sample code.

Ans. Pretty printing is helpful when we want a cleaner display of the items in a dictionary Importing pprint module will provide acces the pprint() and pformat() functions that u pretty print a dictionary's values.

Ex:
```
import pprint
message = 'The sun rises in the east'
count = { }
for i in message:
        count.setdefault(i,0)
        count[i] = count[i] + 1
pprint.pprint(count)
```

o/p:
```
{' ': 5,
 'T': 1,
 'a': 1,
 'e': 4,
 'h': 2,
 'i': 1,
 'n': 2,
 'r': 1,
 's': 4,
 't': 2,
 'u': 1}
```

2. write a program for storing and retrieving friends birthdays using dictionaries.

ans
```
birthday={'Alice':'Jan01', 'Bob':'Dec20', 'Carol':'April16'}
while True:
    print('Enter a name: (blank to quit)')
    name=input()
    if name=='':
        break
    if name in birthday:
        print(birthday[name]+'is the birthday of'+name)
    else:
        print('I do not have birthday info for'+name)
        print('what is their birthday')
        bday=input()
        birthday[name]=bday
        print('database updated')
```

O/P:

Enter a name(blank to quit)
Carol
April 16 is the birthday of Carol
Enter a name(blank to quit)
Sam
I do not have birthday infor for Sam what is their birthday
Feb28
database updated
Enter a name(blank to quit)
Sam
Feb28 is the birthday of Sam
Enter a name(blank to quit)

5. Differentiate the methods get() and setdefault() with sample code.

Ans The get() Method -

Dictionaries have get() method that takes two arguments

- The key of the value to retrieve and
- A fallback value to return if that key does not exist

Ex: >>> picnicItems = {'apples': 5, 'cups': 2}
>>> 'I am bringing' + str(picnicItems.get('cups', 0)) + 'cups.'
'I am bringing 2 cups'
>>> 'I am bringing' + str(picnicItems.get('eggs', 0)) + 'eggs.'
'I am bringing 0 eggs'

The setdefault() Method -

setdefault() takes 2 arguments

- The first argument is the key to check for and
- The second argument is the value to set at that key if the key does not exist. If the key does exist, the setdefault() method returns the key's value

Ex: >>> spam = {'name': 'Pooka', 'age': 5}
>>> spam.setdefault('color', 'black')
'black'
>>> spam
{'color': 'black', 'age': 5, 'name': 'Pooka'}
>>> spam.setdefault('color', 'white')
'black'
>>> spam
{'color': 'black', 'age': 5, 'name': 'Pooka'}

# Chapter 3

1. Discuss about the methods with examples

(a) upper() and lower()

Ans The upper() and lower() string methods return a new string where all the letters in the original string have been converted to uppercase or lowercase, respectively.

```
Ex: >>> spam = 'Hello world!'
    >>> spam = spam.upper()
    >>> spam
    'HELLO WORLD!'
    >>> spam = spam.lower()
    >>> spam
    'hello world!'
```

(b) isupper() and islower()

The isupper() and islower() methods will return a Boolean True value if the string has at least one letter and all the letters are uppercase or lowercase, respectively. otherwise the method returns False

```
Ex: >>> spam = 'Hello world!'
    >>> spam.islower()
    False
    >>> 'HELLO'.isupper()
    True
    >>> 'abc12345'.islower()
    True
    >>> '12345'.islower()
    False
    >>> '12345'.isupper()
    False
```

(c) isalnum() and isspace()

Ans isalnum() returns True if the string consists only of letters and numbers and is not blank. isspace() returns True if the string consists of words that begin w spaces, tabs and newlines and is not blank.

Ex:
```
>>> 'hello123'.isalnum()
True
>>> 'hello'.isalnum()
True
>>> '  '.isspace()
True
>>> '  '.isspace()
True
```

(d) startswith() and endswith()

Ans The startswith() and endswith() methods return True if the string value they are called on begins or ends with the string passed to the method, otherwise they return False.

Ex:
```
>>> 'Hello world!'.startswith('Hello')
True
>>> 'Hello world!'.endswith('world!')
True
>>> 'abc123'.startswith('abcdef')
False
>>> 'abc123'.endswith('12'
False
>>> text = 'Good Morning'
>>> text.startswith('Good')
True.
```

(e) join() and split()

Ans The join() method is called on a string, gets passed a list of strings and returns a string. The returned string is the concatenation of each string in the passed-in list.

```
Ex: >>> ', '.join(['cats', 'rats', 'bats'])
    'cats, rats, bats'
    >>> ' '.join(['My', 'name', 'is', 'simon'])
    'My name is simon'
    >>> 'ABC'.join(['My', 'name', 'is', 'Alice'])
    'MyABCnameABCisABCAlice'
```

The split() method is called on a string value and returns a list of string.

```
Ex: >>> 'My name is simon'.split()
    ['My', 'name', 'is', 'simon']
    >>> 'MyABCnameABCisABCsimon'.split('ABC')
    ['My', 'name', 'is', 'simon']
    >>> 'My name is simon'.split('m')
    ['My na', 'e is si', 'on']
```

---

2. Discuss the following methods

(a) partition()

Ans The partition() method is called on a string value and returns a list.

```
Ex: >>> usns = [1, 2, 3, 4, ... 100]
    >>> usns.partition(50)
    ('1, 2, 3, 4, ... 49', '50', '51, 52, 53. . . 100')
    >>> text = 'GoodMorning'
    >>> text.partition('M')
    ('Good', 'M', 'orning')
    >>> text.partition('o')
    ('G', 'o', 'odMorning')
```

(b) rjust(), ljust() and center()

Ans The rjust(), ljust(), center() string methods return a padded version of the string they are called on with spaces inserted to justify the text.

Ex: def printPicnic(itemsDict, leftwidth, rightwidth):
    print('PICNIC ITEMS'.center(leftwidth+rightwidth, '*'))
    for k, v in itemsDict.items():
        print(k.ljust(leftwidth, '*') + str(v).rjust(rightwidth))
picnicItems = {'sandwiches': 4, 'apples': 12, 'cups': 4, 'cookies': 8000}
printPicnic(picnicItems, 12, 5)
printPicnic(picnicItems, 20, 6)

O/P:

```
---PICNIC ITEMS--
sandwiches..   4
apples.....   12
cups.......   4
cookies.....  8000
-----PICNIC ITEMS-----
sandwiches..........  4
apples..............  12
cups................  4
cookies.............  8000
```

Ex: >>> 'Good'.rjust(10)
'      Good'
>>> 'Good'.ljust(10)
'Good      '
>>> 'Good'.center(10)
'   Good   '
>>> 'Good'.center(11, '*')
'***Good****'

(c) strip()

Ans The strip() string method will return a new string without any whitespace characters at the beginning or end

The lstrip() and rstrip() method will remove white spaces characters from the left and right ends respectively

Ex: >>> spam: '  HaiEveryone  '.

```
>>> spam.strip()
'HaiEveryone'
>>> spam.rstrip()
'  HaiEveryone'
>>> spam.lstrip()
'HaiEveryone  '
>>> text = '123ABC345CAB10099ACB'
>>> text.strip(CAB()
'12334510099
```

(d) ord()

Ans ord() function returns the unicode from a given character.

```
Ex: >>> value = ord('A')
>>> value
65
```

(e) Chor()

Ans Chr() function is used to get a string representing of a character which points to a unicode code integer.

```
Ex: >>> text = chr(65)
>>> text
A
```

3. Explain about pyperclip module. How it can be used to add bullets to wiki markup.

Ans The pyperclip module has copy() and paste() functions that can send text to and receive text from your computer's clipboard

Ex: >>> import pyperclip
>>> pyperclip.copy('Hello world!')
>>> pyperclip.paste()
'Hello world!'

Add bullets to wikiMarkup

bullet.py

```
import pyperclip
text = pyperclip.paste()
lines = text.split('\n')
for i in range(len(lines)):
        lines[i] = '* ' + lines[i]
text = '\n'.join(lines)
pyperclip.copy(text)
```

---

4. Write a simple python code to store 6 account passwords using pyperclip and sys module and to retrieve the password to clipboard by passing command line arguments

Ans Password Locker.

```
PASSWORDS = { 'email': 'hai123', 'blog': 'hello123',
              'facebook': bye123', 'twitter': 'good456',
              'instagram: 'Moni78', 'bank' : 'Moni98' }
import sys, pyperclip
if (len(sys.argv < 2)
print('usage: python password.py [account] - copy
      account password')
```

```
        sys.exit()
    account = sys.argv[1]

if account in PASSWORDS:
    pyperclip.copy(PASSWORDS[account])
    print('Password for' + account + 'copied to
          clipboard.')
else:
    print('There is no account named' + account)
```

output: python pswd.py
usage: python password.py [account] - copy account password

python password.py facebook

python password.py luggage
There is no account named twitter.

---

5. What are escape characters. How to use them inside the programs. explain with example.

Ans If you need to use both single quotes and double quotes in the string. you'll need to use escape characters.

→ An escape character consists of a backslash(\) followed by the character you want to add to the string

Ex: >>> spam = 'say hi to Bob's mother.'
    say hi to Bob's mother.

>>> print('Hello there!\n How are you: \n I'm doing fine.")
Hello there:
How are you:
I'm doing fine.

→ The different special escape characters can be used in a program

| Escape character | Prints as |
|---|---|
| \' | Single quote |
| \" | Double quote |
| \t | Tab |
| \n | Newline (line break) |
| \\ | Backslash |

# MODULE - 3 [CHAPTER - 1]

1. Discuss about search() and findall functions of re module?

Ans: Regex objects have search() method and findall() method. while search() will return a match object of the first matched text in the searched string, the findall method will return the string of every match in the searched string.

search() method:

```
>>> phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
>>> mo = phoneNumRegex.search('Cell: 415-555-9999 Work:
                                212-555-0000')
>>> mo.group()
'415-555-9999'
```

findall() method:

```
>>> phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
>>> phoneNumRegex.findall('Cell: 415-555-9999' Work:
                                212-555-0000')
['415-555-9999', '212-555-0000']
```

---

2. what are regular expressions? Describe question mark, star, plus and dot Regex symbols with suitable python code snippet?

Ans. Regular expressions:

Regular expressions called regexes for short, are descriptions for a pattern of text

Ex: \d in a regex stands for a digit character - that is any single number 0 to 9

a. Question mark(?) - optional matching.

The ? character flags the group that precedes it is an optional part of the pattern.

Ex: >>> batRegex = re.compile(r'Bat(wo)?man')
>>> mo1 = batRegex.search('The Adventures of Batman')
>>> mo1.group()
'Batman'
>>> mo2 = batRegex.search('The Adventures of Batwoman')
>>> mo2.group()
'Batwoman'

b. star(*) - Matching zero or more
The * means the group that precedes the star can occur any number of times in the text. It can be completely absent or repeated over and over again.
Ex: >>> batRegex = re.compile(r'Bat(wo)*man')
>>> mo1 = batRegex.search('The Batman')
>>> mo1.group()
'Batman'
>>> mo2 = batRegex.search('The Batwoman')
>>> mo2.group()
'Batwoman'
>>> mo3 = batRegex.search('The Batwowowowoman')
>>> mo3.group()
'Batwowowowoman'

c. plus(+) - Matching one or more
The + means the group preceding a plus must appear atleast once. It is not optional
Ex: >>> batRegex = re.compile(r'Bat(wo)+man')
>>> mo1 = batRegex.search('The Batwoman')
>>> mo1.group()
'Batwoman'
>>> mo2 = batRegex.search('The Batman')

```
>>> mo2 == None
True
```

d. dot(.)

The . character in a regex is called a wildcard and will match any character except the newline

Ex: >>> atRegex = re.compile(r'.at')
>>> atRegex.findall('The cat in the hat sat on the flat mat.')
['cat', 'hat', 'sat', 'lat', 'mat']

---

3. write a program that reads a string with five character which starts with 'a' and ends with 'z'. print search successful if pattern matches string.

Ans

```
import re
message = 'abcde adcdz'
def func(text):
    pattern = re.compile(r'a...z')
    mo = pattern.search(text)
    if(mo == None):
        print('search not found')
    else: print('search successful')
func(message)
```

o/p: Search successful

---

4. Describe about pipe, caret, dollar signs character in re?

Ans

pipe(|) - Matching Multiple Groups

The | character is called a pipe. It is used to match one of many expression

Ex:

```
>>> heroRegex = re.compile(r'Batman|Tina Fey')
```

```
>>> mo1 = heroRegex.search('Batman and Tina Fey.')
>>> mo1.group()
'Batman'
>>> mo2 = heroRegex.search('Tina Fey and Batman.')
>>> mo2.group()
'Tina Fey'
```

→ The caret (^) and Dollar ($) sign characters:

The caret (^) symbol can be used at the start of a regex to indicate that a match must occur at the beginning of the searched text.

The Dollar Sign ($) can be used at the end of the regex to indicate that string must end with this regex pattern.

Ex:

```
>>> beginswithhello = re.compile(r'^Hello')
>>> beginswithhello.search('Hello world!')
<_sre.SRE_MATCH object; span=(0,5), match='Hello'>
>>> beginswithhello.search('He said hello.') == None
True
```

Ex:

```
>>> Regex = re.compile(r'\d$')
>>> Regex.search('Your number is 42')
<sre.SRE_Match object; span=(16,17), match='2'>
>>> Regex.search('Forty two') == None
True
```

---

5. How to create a regex objects, what is the method used for pattern matching and displaying text? Discuss briefly about the advantages of regex?

Ans All the regex functions in python are in the re module. Enter the following into the interactive shell to import this module

```
>>> import re
```

* To create a regex object re.compile() is used. passing a string value representing your regular expression to re.compile() returns a Regex pattern object (or simply, a regex object).

Ex: To create a regex object that matches the phonenumber pattern

```
>>> phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
```

Now the phoneNumRegex variable contains a Regex object.

* A Regex objects search() method searches the string it is passed for any matches to the regex. The search() method will return None if the regex pattern is not found in the string. If the pattern is found, the search() method returns a match object. Match objects have a group() method that will return the actual matched text from the searched string

```
>>> phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
>>> mo = phoneNumRegex.search('My number is 415-555-4242.')
>>> print('phone number found:' +mo.group())
phone number found: 415-555-4242
```

→ Advantages of regex:

1. Regex is very feasible
2. It is fast processing
3. Language independent
4. More lines of code for matching without Regex but using Regex we can write the code in single line.

6. what is the advantage of Braces in regex. explain with example. Differentiate greedy and non-greedy matching.

Ans If you have a group that you want to repeat a specific number of times, follow the group in your regex with a number in curly backets. It will match the exact number of times in the given string. you can specify a range by writing a minimum, a comma and a maximum in between the curly brockets.

Ex:

```
>>> haRegex = re.compile(r'(Ha){3}')
>>> mo1 = haRegex.search('HaHaHa')
>>> mo1.group()
'HaHaHa'
>>> mo2 = haRegex.search('Ha')
>>> mo2 == None
True
```

→ ** Greedy and Non greedy matching.

* Python's Regexs are greedy by default, which means that in ambiguous situations they will match the longest string possible

* The non-greedy version of the curly bracket, which matches the shortest string possible, has the closing curly bracket followed by a question mark

Ex:

```
>>> greedy = re.compile(r'(Ha){3,5}')
>>> mo1 = greedy.search('HaHaHaHaHa')
>>> mo1.group()
'HaHaHaHaHa'
>>> nongreedy = re.compile(r'(Ha){3,5}?')
>>> mo2 = greedy.search('HaHaHaHaHa')
```

```
>>> mo2.group()
'HaHaHa'
```

7. write a short note python character classes and also explain how to create your own character classes.

Ans character class is shorthand for the regular expression. There are many shorthand character classes.

| shorthand character class | Represents |
|---|---|
| \d | - Any numeric digit 0 to 9 |
| \D | - Any character that is not a numeric digit 0 to 9 |
| \w | - Any letter, numeric digit, or the underscore character |
| \W | - Any character that is not a letter, numeric digit, or the underscore character |
| \s | - Any space, tab or newline character |
| \S | - Any character that is not a space, tab or newline |

Ex:

```
>>> xmasRegex = re.compile(r'\d+\s\w+')
>>> xmasRegex.findall('12 drummers, 11 pipers, 10 lords')
['12 drummers', '11 pipers', '10 lords']
```

→ Making your own character class:

* we can define our own character class using square brackets.

Ex: >>> Regex = re.compile(r'[aeiouAEIOU]')
>>> Regex.findall('Today is Monday')
['o', 'a', 'i', 'o', 'a']

* we can also include ranges of letters or numbers by using a hyphen.

* Inside the square brackets, the normal regular expression symbols are not interpreted as such
Ex: character class [0-5.] will match digits 0 to 5 and a period

* By placing a caret character(^) just after the character class's opening bracket, we make a negated character class

Ex: Regex = re.compile(r'[^aeiouAEIOU]')
Regex.findall('Today is Monday')
['T','d','y','s','M','n','d','y']

---

8. write a short note on

(i) re.DOTALL

Ans re.DOTALL will match all the characters, including the newline character.

Ex:

>>> Regex = re.compile('.*', re.DOTALL)
>>> Regex.search('Serve the public trust.\nProtect the innocent.\n').group()
'Serve the public trust.\nProtect the innocent.\n'

(ii) re.I

Ans By passing re.IGNORECASE or re.I as the second argument to re.compile() we can make the regex case-insensitive ie they can be uppercase or lowercase

Ex: >>> Regex = re.compile(r'Robo', re.I)
>>> Regex.search('Robo ROBO RoBO robO').group()
'Robo'
>>> Regex.findall('Robo ROBO RoBo robo')
['Robo', 'ROBO', 'RoBo', 'robo']

(iii) re.VERBOSE

The VERBOSE flag of the regex package allows the user to write regular expressions that can look nicer and are more readable.

Ex: regex = re.compile(r'^([a-z0-9_\.-]+)@([0-9a-z\.-]+)\.([a-z\.]{2,6})$', re.IGNORCASE)

using VERBOSE

```
regex = re.compile(r'''
                    ^([a-z0-9_\.-]+)
                    @
                    ([0-9a-z\.-]+)
                    \.
                    ([a-z]{2,6})$
                    ''', re.VERBOSE| re.I)
```

(iv) sub() method

Regular expressions can not only find text patterns but can also substitute new text in place of those patterns. The sub() method for Regex objects is passed two arguments. The first argument is string to replace any matches. The second is the string for the regular expression. The sub() method returns a string with the substitutions applied.

Ex:

```
>>> Regex = re.compile(r'Agent \w+')
>>> Regex.sub('Censored', 'Agent Alice died.')
'Censored died'
>>> Regex = re.compile(r'Agent (\w)\w*')
>>> Regex.sub(r'\1****', 'Agent Alice told Agent Carol that Agent Eve was a double agent.')
'A**** told C**** that E**** was a double agent'
```

9. Write the code in python that extracts phone numbers and email address from a wikipage that contains many phone Numbers and emails using pyperclip and re modules.

Ans.
```
import re, pyperclip
PhoneRegex = re.compile(r'''(
    (\d{3}|\(\d{3}\))?
    (\s|-|\.)?
    \d{3}
    (\s|-|\.)?
    \d{4}
    (\s*(ext|x|ext.)\s*\d{2,5})?
    )''', re.VERBOSE)
EmailRegex = re.compile(r'''(
    [a-zA-Z0-9_%+-]+
    @
    [a-zA-Z0-9.-]+
    (\.[a-zA-Z]{2,4})
    )''', re.VERBOSE)
text = str(pyperclip.paste())
matches = []
for groups in PhoneRegex.findall(text):
    phoneNum = '-'.join([groups[1], groups[3],
                         groups[5]])
    if groups[8] != '':
        phoneNum = 'x' + groups[8]
    matches.append(phoneNum)
for groups in emailRegex.findall(text):
    matches.append(groups[0])
if len(matches) > 0:
    pyperclip.copy('\n'.join(matches))
    print('copied to clipboard:')
```

```
    print('\n'.join(matches))
else:
    print('No phonenum/emailid found')
```

[Chapter-2]

```
    print ('\n'.join(matches))
else:
    print ('No phonenum/email id found')
```

## [Chapter-2]

1. With examples explain the following os module functions

Ans (a) getcwd

Every program that runs on our computer has a current working directory, or cwd. Any filenames or paths that do not begin with the root folder are assumed to be under cwd. we can get the current working directory as a string value with the os.getcwd() function.

```
Ex: >>> import os
    >>> os.getcwd()
    'C:\\Python34'
```

(b) chdir

Using os.chdir() function we can change the current working directory

```
Ex: >>> import os
    >>> os.getcwd()
    'C:\\Python34'
    >>> os.chdir('C:\\Windows\\System32')
    >>> os.getcwd()
    'C:\\Windows\\System32'
```

(c) mkdir

Using os.mkdir() function we can create new directory

Ex:

```
>>> import os
>>> os.mkdir('C:\\Windows\\Student\\Python')
```
It creates a python directory.

(d) listdir

Calling os.listdir(path) will return a list of filenames strings for each file in the path argument.

Ex: >>> import os

```
>>> os.listdir('C:\\Windows\\System32')
['0409', '12520437.cpx', '12520850.cpx', '5U877.ax', ...
```

2. Explain with examples the following os.path module functions

(a) exits()

Calling os.path.exists(path) will return True if the file or folder referred to in the argument exists and will return False if it does not exist.

Ex: >>> import os

```
>>> os.path.exists('C:\\Windows')
True
>>> os.path.exists('C:\\Linux')
False
```

(b) isfile()

Calling os.path.isfile(path) will return True if the path argument exists and is a file and will return False otherwise

Ex: >>> import os

```
>>> os.path.isfile('C:\\Windows\\System32')
False
>>> os.path.isfile('C:\\Windows\\System32\\calc.exe')
True
```

(c) isdir()

calling os.path.isdir(path) will True if the path argument exists and is a folder and will return False otherwise

Ex: >>> import os

>>> os.path.isdir('C:\\Windows\\System32')

True

>>> os.path.isdir('C:\\Windows\\System32\\calc.exe')

False

(d) getsize()

calling os.path.getsize(path) will return the size in bytes of the file in the path argument

Ex: >>> import os

>>> os.path.getsize('C:\\Windows\\System32\\calc.exe')

776192

(e) split()

The split() string method will work to return a list of each part of the path. It will work on any OS if you pass it os.path.sep.

Ex: >>> calcFilepath.spilt(os.path.sep)

['C:', 'Windows', 'System32', 'calc.exe']

If you need a path's dir name and basename

(f) ~~dirname()~~ together, you can call os.path.split() to get a tuple value with these two string.

>>> calcFilepath = 'C:\\Windows\\System32\\calc.exe'

>>> os.path.split(calcFilepath)

('C:\\Windows\\System32', 'calc.exe')

(g) dirname()

calling os.path.dirname(path) will return a string of

everything that comes before the last slash in the path argument.

Ex: >>>import os

>>>path = 'C:\\Windows\\System32\\Calc.exe'

>>> os.path.dirname(path)

'C:\\Windows\\System32'

(g) abspath()

Calling os.path.abspath(path) will return a string of the absolute path of the argument. This is an easy way to convert a relative path into an absolute one.

Ex: >>>import os

>>> os.path.abspath('.')

'C:\\Python34'

>>> os.path.abspath('.\\Scripts')

'C:\\Python34\\Scripts'

(h) relpath()

Calling os.path.relpath(path, start) will return a string of a relative path from the start path to path. If start is not provided, the cwd is used as the start path.

Ex: >>> import os

>>> os.path.relpath('c:\\Windows', 'C:\\')

'Windows'

>>> os.path.relpath('c:\\Windows', 'c:\\Spam\\eggs')

'..\\..\\Windows'

(i) basename()

Calling os.path.basename(path) will return a string of everything that comes after the last slash in the

path argument

Ex: path = 'c:\\Windows\\System32\\calc.exe'

```
>>> os.path.basename(path)
'calc.exe'
```

(j) isabs()

Calling os.path.isabs(path) will return True if the argument is an absolute path and False if it is a relative path.

Ex: >>> os.path.isabs('.')

False

```
>>> os.path.isabs(os.path.abspath('.'))
True
```

3: Explain file open, file close, file read and file write concepts in python with example

Ans