Assignment :- 1

Date : / / 20

Assignment-1

Title :- Write Python code for Word count and find occurrence of word from file.

Problem Statement :- To pyton code for word count and find occurrence of word from file.

Pre-Lab : A basic understanding of computer programming terminologies. A basic understanding of any of the programming languages will help in understanding the Pyton Programming and datascience concepts.

Theory :-

What is Python? Python is a powerful modern computer Programming ~~lan~~ language. It bears some similarities to Fortran, one of the earliest programming languages, but it is much more powerful than fortran. Python allows you to use variable without declaring them (i.e. it determines types implicitly), and it relies on indentation as a control structure.

Python Commands :-

Comments In a Python command, anything a # symbol is a comment.

For example

Print "Hello world" # this is silly

Pooja

Date : / / 20

Comments are not part of the command, but rather intended as documentation for anyone reading the code. Multiline comments are also possible, and are enclosed by triple double-quote symbols:
""" This is an example of a long comments that goes on and on. """

comments:
In a Python command, anything after a # symbol is a comment. For example:
print "Hello World" # this is silly.
Comments are not part of the command, but rather intended as documentation for anyone reding the code. Multiple comments are also possible, and are enclosed by triple double-quote symbols.
""" This is an example of a long comment.
that goes on
and on
and on. """

3.2 Numbers and other data types.

Python recognizes several different types of data. for instance, 23 and -75 are integres, while 5.0 and -23.09 are floats or floating point numbers. The type float is (roughly) the same as a real number in mathematics. The number 12345878901 is a long integer; Python prints it with an "L" appended to the end.

Pooja

Date : / / 20

Usually the type of a piece of data is determined implicitly

3.2.1 The type function

To see the type of some data, use python's builtin type function:

```
>>> type (-75)
<type 'int'>
>>> type (5.0)
<type 'float'>
>>> type (12345678901)
<type 'long'>
```

Another useful data type is complex, used for complex numbers. for example:

```
>>> 2j
2j
>>> 2j -1
(-1+2j)
>>> complex (2,3)
(2+3j)
>>> type (-1+2j)
```

Notice that python uses j for the complex unit (such that $j^2 = -1$) just as physicists do, instead of the letter i preferred by mathematicians.

Pooja

Date : / / 20

3.2.2 Strings

other useful data types are strings (short for "character strings"): for example "Hello World!" strings are sequences of characters enclosed in single or double quotes:

```
>>> "This is a string"
This string
>>> "This is a string, loo' This
is a string, loo'
>>> type ("This is a string")<
type 'str'>
```

String are an example of sequence type.

3.2.3 Lists and tuples

other important sequence type used in python include lists and tuples. A sequence type is formed by putting together some other type in a sequence. Here is how we form lists and tuples:

```
>>> [1,3,4,1,6]
[1,3,4,1,6]
>>> type ([1,3,4,1,6])
 < type 'list'>
>>> (1,3,2)
(1,3,2)
>>> type (1,3,2))
< type 'tuple'>
```

Pooja

Notice that lists are enclosed in square bracket while tuples do not need to be homogeneous; the components can be of different types:

```
>>> [1, 2, "Hello", (1, 2)]
[1, 2, 'Hello', (1, 2)]
```

Here we created a list containing foure components: two integers, a string and atuple. Note that components of lists may be other lists, and so on:

```
>>> [1, 2, [1, 2], [1, [1, 2]], [5], [1, 2,
[1, 2], [1, [1, 2]], 5]
```

By nesting lists within lists in this way, we can build up complicated structures.
Sequence types such as lists, tuple, and string are always ordered, as opposed to a set in mathematics, which is always unordered.

32.4 The range function.

The range function is often used to creat lists of integers. It has three forms. In the simplest form, range (n) produces a list of all numbers 0, 1, 2 ..... n-1 starting with 0 and ending with n-1 for instance.

```
>>> range (17)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
```

Pooj

You can also specify an optional starting point and an increment, which may be negative. for instance we have

```
>> range (1,10)
[1,2,3,4,5,6,7,8,9]
>>> range (-6,0)
[-6,-5,-4,-3,-2,-1]
>>> range (1,10,2)
[1,3,5,7,9)
>>> range (10,0,-2) [10,8,
6,4,2]
```

Note the use of a negative increment in the last example.

3.2.5 Boolen values

Finally, we should mention the boolen type. this is a value which is either true or false.

```
>>> True
True
>>> type (True)
<type 'bool'>
>>> False False
>>> type (False)
<type 'bool'>
```

Boolen types are used is making decisions

Pooja

Python expressions are not commands, but rather form part of a command. an expression is anything which produces a value. Examples of expressions are 2+2, 2**10. f((x-1)/(x+1)). Note that in order for python to make sense of the last one, the variable x must have a value assigned and f should be a previously defined function.

Operations:

The common bing binary operators for arithmetic are + for addition, - for subtraction, * for multi-plication and / for division. As already mentioned, python use ** for exponentiation. Integet division is performed so that the result is always another integer (the integer quotient):

```
>>> 25/3
8
>>> 5/2
 2
```

This is a wrinkle that you will always have to keep in mind when working with python. To get a more accurate answer, use the float type:

```
>>>  25.0/3
8.333333333333333
>>>  5/2.0
 2.5
```

Pooja

If Just one of the operands is of type float, then the result will be of type float. Here is another example of this pitfall:

```
>>> 2 ** (1/2)
```

where we wanted to compute the square root of 2 as the exponent produced a result of 0 because of integer division, is:
1 power of 2, but the division in the A correct way to do this computation

```
>>> 2 ** 0.5
1.4142135623730951
```

Another useful operator is % which is read as "mod". This gives the reminder of an integer division, as in

```
>>> 5 % 2
1
>>> 25 % 3
1
```

Variable and Assignment

An assignment statement in python has the form variable = expression. This has the following effect. first the expression on the right hand side is evaluated, then the result is assigned to the variable. After the assignment, the variable becomes a name for the result.

Pooja

Executing the assignment produces no output, its purpose is to make the association between the variable and its value.

```
>>> x = 2+2
>>> print x
4
```

In the example above, the assignment statement sets x to 4 producing no output. If we want to see the value of x, we must print it. If we execute another assignment to x then the privious value is lost.

```
>>> x = 380.5
>>> print x 380.5
>>> y = 2*x
>>> print y 761.0
```

Remember. A single = is used for assignment, the double == is used to test for equality.

In mathematics the equation x = x+1 is nonsense, it has no solution. In compter science, the statement x = x+1 is useful. Its purpose is to add 1 to x, and reassign the result to x. In short, x is incremented by 1.

```
>>> x = 10
>>> x = x+1
>>> print x
11
>>> x = x+1
>>> print x
11
```

Pooja

Decisions

The if-else is used to make choices in python code. This is a compound statement. The simplest form is

```
if condition :
    action -1
else :
    action -2
```

The indentation is required. Note that the else and its action are optional. The action action -1 and action -2 may consist of many statement, they must all be indented the same amount. Th
of course. If the condition evaluates to true then action -1 is executed, otherwise action -2 is exe-cuted. In either case execution continues with the statement after the if-else. for example, the code

```
x = 1
if x > 0 :
    print "friday is wonderful"
else :
    print "Monday sucks"
print "Have a good weekend"
```

results in the output

```
Friday is wonderful
Have a good weekend
```

Pooja

Python provides two looping commands: for and while
These are compound commands.
for loop:
The syntax of a for loop is
for itemin list:
 actions
n = 10
while n > 0:
 print n.
  n = n - 1
else:
print "blastoff"

has the same effect (if produces identical output).
break, continue and pass

The break statement like in C, breaks out of the smallest enclosing for or while loop. The continue statement, also borrowed from C, continues with the next iteration of the loop. The pass statement does nothing.
Here is an example of the use of a break statement and an else clause in a loop

```
for n in range (2, 10):
    for x in range (2, n):
        if n % x = 0
        print n, "equals: x ", n/x
        break
```

Pooja

Lists

As already mentioned a list is a finite sequence of items & one could use the range fun to create lists of integers.

In python, lists not required to be homogeneous.

```
a=[2,"Jack",45,"23 wentworth Ave"]
```

is a perfectly valid list consisting of two integers & two strings, one can refer to the entire list using the identifier a or to the i-th item in the list.

```
>>>a= [2,'Jack',45,'23 wentworth Ave']
>>>a
[2,'Jack',45,'23 wentworth Ave']
>>> a[0]
2
>>>a['i']
'Jack'
>>> a[2]
45
>>> a[3]
'23 wentworth Ave'
```

- length of a list; empty list:-

Every list has a length the no. of item in the list, obtained using the len fun.

```
>>> x=[9,4,900,-45]
>>> len(x)
4
```

Of special importance is the empty list of length 0.

```
>>> x=[]
>>> len(x)
0
```

Pooja

Sublists are obtained by slicing, which works analogously to the range fun discussed before if x is an existing list, then x is the object consiting of all items
start ≤ i < end
of course, we must remeber that indexing item always starts at 0 in Python

```
>>>x
[0,2,4,6,8,10,12,14,16,18]
>>>x[2:5]
[4,6,8]
```

- Joining two lists :-
two existing lists may be concatenet together to make a longest list, using the + operator:

```
>>> [2,3,6,10] + [4,0,0,5,0]
[2,3,6,10,4,0,0,5,0]
```

- list methods :-
if x is the name of an existing list, we can append an item to the end of the list using x.append(item)

e.g →
```
>>>x = [3,6,8,9]
>>>x.append(999)
>>>x
[3,6,8,9,999]
```

e.g →
```
>>>x = ['a','c','3','d','7']
>>> x.insert(0,100)
>>>x
[100,'a','c','3','d','7']
```

Pooja

- Strings:-
  A string in Python is a sequence of characters I some sense strings are similar to lists, however, there are imp difference, one major difference is that Python string are immutable, meaning that we are not allowed to change individual parts of them as we could for a list

```
>>> x = 'gobbletygook'
>>> x[2]
'b'
>>> x[5]
'e'
>>> x[5] = 's'
Traceback (most recent call last);
  file "<stdin>", line1, in<module>
TypeError: 'str' object does not support item
  assignment
```

- There are many string methods for manipulating strings, document in the python library Reference manual [2] for eg, you can capitalize an existing string x using x.capitalize(); this return a new copy of the string in which the first character has been capitalized.

```
>>> a = 'gobbletygook is refreshing'
>>> a.capitalize()
'Gobbletygook is refreshing'
```

- conclusion:- Thus the Python2 Programing Tool is installed & exercised Basic syntax, Datatypes, variables operators, vectors, lists, matrices, Dataframe---

Pooja