
∇ueron.△i

DOOH(Digital Out of Home) Advertising

VUERON is working to:

- Bring about an innovative and cost-effective digital system of advisement flow.
- Provide an easily accessible platform for masses.
- Bridge the gaps in the chain between advertisers and buyers.
- Overcome the barriers of conventional marketing.
- Empower advertisers to reach their exact audiences in a better manner.

∇ueron.Δi

Example: Ad of a Lipstick

Any guesses where do we ***jump*** in???

VUERON SYSTEMS PVT. LTD.

	https://www.vueron.ai
	https://www.linkedin.com/company/vueron
	http://www.facebook.com/vueron
	http://www.twitter.com/vueron_ai
	https://instagram.com/vueron_ai

Vueron.Ai

Getting Started with Ubuntu

What is ubuntu?

Ubuntu is a free and open-source Linux based operating system. Most Linux users are familiar with the command line interface. We will go through some of the popular command line's used in Ubuntu.

Directory Listing

The directory listing command is used to list the directory contents.

Syntax:

ls –option directoryname

Another variant is “ls –l” command which lists the directory, but with more details on each line item.

Clearing the Screen

To clear the command line screen, we can use the clear command.

Syntax:

```
clear
```

Command Help

To get more information on a command, we can use the ‘man’ command.

Syntax:

```
man commandname
```

Whoami Command

This command is used to display who is the current logged on user.

Syntax:

```
whoami
```

Present Working Directory

This command will display the current working directory.

Syntax:

```
pwd
```

Change Directory

This command will change the working directory.

Syntax:

cd

Make Directory

This command will create folders and files.

Syntax:

```
mkdir directoryname
```

Echo command

This command will print its arguments back out again (hence the name).

Syntax:

```
echo
```

Sudo command

Sudo program allows regular users to run programs with the security privileges of the superuser or root.

Syntax:

```
sudo command_you_want_to_execute
```

Remove Files

The rm command removes files. Be careful with this command — rm doesn't ask you for confirmation.

Syntax:

```
rm filename
```

Moving and Re-naming files

The mv command moves a file to a new location. This is also the command you'll use to rename files.

Syntax:

mv oldfile newfile(For renaming file)

mv filename new_file_location(For moving file)

Copy Files

This command works the same way as the mv command, except it copies the original files instead of moving them.

Syntax:

```
cp sourcefile destinationfile
```

Vim

Vim is a highly configurable text editor for efficiently creating and changing any kind of text.

Installing vim

```
sudo apt-get install vim-nox
```

Vim Modes

- Command mode: This is the default mode in which Vim starts. We can enter editor commands in this mode.
- Insert mode: We can use this mode to enter/edit text. To switch from default command to insert mode press i key. Use Escape key to switch back to command mode from this mode.

Create new file

- Open Vim-

$ vim

- Type following command in Vim-

:edit message.txt

(It will reload file if, it exist already)

- Switch to insert mode-

i

(Enter some text)

- Switch back to command mode

Esc

- Save changes to file

:w

(Now message.txt file will be created.)

- Quit Vim

:q

Edit existing file

- Open file using Vim -

$ vim message.txt

- Switch to insert mode and enter some text there.

i

- Quit editor without saving changes

:q!

OR

- Save changes and quit editor using following command -

:wq!

∇ueron.Δi

Python Programming

Presentation Link: https://github.com/vueron/python3-workshop

What is Python ?

https://github.com/vueron/python3-workshop

Let’s do some mathematics

Let’s do some mathematics

Consider a function, f(x) = x+2

when x = 1, f(1)= 3

x = 3, f(3)= 5

x = 5, f(5)= 7

How can we implement this function in programming ?

Basics of a function

In computer science, a function can be of many types

S.NO	Input Parameter	Output Return
1	NO	VOID
2	NO	YES
3	YES	VOID
4	YES	YES

Syntax for a function

```
>>> def f( ):     # no input parameter
        -----
        -----      #  no return output

>>> def f( x ):    # input 'x'
        -----
        -----      # no return output
```

Syntax for a function

```
>>> def f(  ):     # no input parameter
        -----
        return res      #  return 'res'

>>> def f( x ):    # input 'x'
        -----
        return res      # return 'res'
```

Numbers

```
>>> value_1 = 11   # this is an integer
>>> value_2 = 7    # this is an integer
>>> value_3 = 1.0  # this is a float
>>> value_4 = 1/2  ## this is an NOT integer
                   ## in Python 3.6, But is an
                   ## integer in Python 2.7
```

Numbers

```
>>> value_5 = 1.0 / 2    # this is a float
>>> value_6 =  1 / 2.0   # this is a float
>>> value_7 =  1.0 / 2.0 # this is a float
>>> value_8 = 1.0 * 1    # this is a float
>>> value_9 = 1.0 + 1    # this is a float
```

What about these ?

```
>>>  value_10 = 1.0 – 1 # this is ?
>>>  value_11 = 1.0 * 0 # this is ?
```

Convert int to float

```
>>> value_12 = 21  # integer
>>> value_12_new = float( value_12 )
>>> type( value_12 )
>>> type( value_12_new )
```

Convert float to int

```
>>> value_13 = 12.0   # float
>>> value_13_new =  int( value_13 )
>>> type( value_13 )
>>> type( value_13_new )
```

Numbers, and Lists

```
>>> this_is_an_integer = 1
>>> this_is_a_float = 1.0
>>> list_of_integers= [ 1, 2, 3 ]
>>> list_of_floats=[ 1.0, 2.0, 3.0 ]
>>> list_of_numbers=[ 1, 2.0, 3/2 ]
```

Indexing in Lists

```
>>> a = [ 21, 22, 23, 24 ] # index starts with 0
>>> print( a ) # this function prints entire list a
>>> print( a[0] ) # element at zero index i.e 21
>>> print( a[1] ) # element at 1 index i.e. 22
>>> print( a[-1] ) # element at last index i.e. 24
```

List Operations

```
>>> a = [ 0 ]
>>> a = a.append(1)
>>> print(a)  # value of updated list ?
>>> b = [2, 3, 4]
>>> c = a.extend(b)
>>> print(c) # value of updated list ?
```

List of Lists

```
>>> list_a = [ 1, 2, 3.0 ]
>>> list_b = [ 7, 8.0, 0 ]
>>> list_a[ 0 ] = list_b        # what will happen ?
>>> print( list_a )
>>> print( list_a[0][0] )       # what will happen ?
>>> print( list_a[0][-1] )      # what will happen ?
```

Copying a List

```
>>> list_1 = [ 1, 2 ,3 ]
>>> list_2 = [ 7, 8 ,9 ]
>>> list_1[0] = list_2
>>> list_2[0] = -1
>>> print( list_1 )
```

Copying a List

```
>>> list_1 = [ 1, 2 ,3 ]
>>> list_2 = [ 7, 8 ,9 ]
>>> list_1[0] =list_2.copy( )      # copying the list
>>> list_2[0] = -1
>>> print( list_1 )
>>> print( list_2 )
```

Indexing in depth

```
>>> list_a = [ 91, 92, 93, 94, 95 ]
>>> new_list = list_a[ 1 : 3 ]  ## represent [ 1, 3)
                                ##     index values
>>> print( new_list )           # what will happen ?
>>> print( list_a )             # what will happen ?
>>> print( list_a[3:-1])        # what will happen ?
```

Copying a List

```
>>> list_1 = [ 1, 2 ,3 ]
>>> list_2 = [ 7, 8 ,9 ]
>>> list_1[0] =list_2[ : ]      # copying the list
>>> list_2[0] = -1
>>> print( list_1 )
>>> print( list_2 )
```

Length Function

```
>>> list_1 = [1, 2, 3, 4, 50 ]
>>> print (   len(  list_1  )    )          ## prints 5 as total no. Of
                                             ## elements in list_1 is 5

>>> list_2 = [ 9, 8 ]
>>> print (   len(  list_2  )    )
>>> list_3 = list_2.copy( )
>>> list_3[0] = list_1
>>> print ( len( list_3 ) )                  # what will happen ?
```

Range Function

```
>>> range_val_1 = range( 5 )  #  [ 0, 1, 2, 3, 4 ]
>>> range_val_2 = range ( 2, 5 ) # [ 2, 3, 4 ]
```

```
print("break time")
```

https://github.com/vueron/python3-workshop

Swapping

Let us say **a = 2**, and **b =7**, now swap(i.e interchange) the values of ***a*** and ***b***

```
>>>  a = 2
>>>  b = 7
>>>  c = a
>>>  a =  b
>>>  b = c
>>>  print(a)
>>>  print(b)
```

Relational Operators

S.NO	RELATION	SYMBOL
1	GREATER THAN	>
2	GREATER THAN OR EQUAL	> =
3	LESS THAN	<
4	LESS THAN OR EQUAL	< =
5	EQUAL TO	==
6	NOT	!
7	NOT EQUAL	! =

if-elif-else

```
value_a = 10
value_b = 7
if value_a > value_b:
    print(“value_a is greater than value_b”)
print(“completed”)
```

if-elif-else

```
value_a = 10
value_b = 7
if value_a > value_b:
    print(“value_a is greater than value_b”)
else:
    print(“value_a is less than or equal to value_b”)
print(“completed”)
```

if-elif-else

```
value_a = 10
value_b = 7
if value_a > value_b:
    print(“value_a is greater than value_b”)
elif value_a == value_b:
    print(“value_a is equal to value_b”)
else:
    print(“value_a is less than to value_b”)
print(“completed”)
```

Loops

1. **for** loop (element based)
2. **while** loop (condition based)

for loop

```
>>> for  elem in range(0,10):
        print(elem)
```

while loop

```
>>> a = 12
>>> b = 7
>>> while a > b :
        print(“a is greater than b”)
        a = a - 1
```

Sum of ***N*** natural numbers with loops

```
>>> sum = 0
>>> n = int( input(  ) )
>>> for elem in range( n+1 ):
        sum = sum + elem
>>> print(sum)
```

Sum of *N* natural numbers with loops

```
>>> cnt = 1
>>> sum = 0
>>> n = int( input(  ) )
>>> while cnt < = n:
        sum = sum + cnt
        cnt = cnt + 1
>>> print(sum)
```

Strings

```
>>> s_1 = “this is one string”
>>> s_2 = “this is also 1 string”
>>> s_3 = “this is ” + ”also 1 string”
>>> s_4 = “this is also ”+ str(1) + ” string”
```

Operation on Strings

```
>>> s_1 = “this is a string”
>>> length_of_s1 = len(s_1)
>>> print(length_of_s1)
>>> s_2 = s_1[ 1:4 ]
>>> print(s_2)
```

Split and Join

```
>>>  s_data = “abc @company.com”
>>>  s_split = s_data.split(“@”)
>>>  print(s_split)
```

Split and Join

```
>>>  s_data = “abc @company.com”
>>>  s_split = s_data.split(“@”)
>>>  s_data = “@”.join(s_split)
>>> print(s_data)
```

Tuples & Dictionary

```
>>> t1 =( 7,0,11 ) ## once defined, we can not
                   ##change length of a tuple
>>> print(t1)
>>> print(t1[0] )
```

Tuples

```
>>> t1 = ( 1, 2, 3 )
>>> list_a = [ 9, 10, 11 ]
>>> t1[0] = list_a.copy( ) # Error
>>> t1 = ( [1], 2, 3 )
>>> t1[0] = t1[0].append( 9 ) # it is possible !!!
>>> print( t1 )
```

Dictionary { }

```
>>> dict_a = { “fruit”: “apple”, “cards”: [ 1, 2, 3 ] }
>>> print( dict_a[“fruit”] )
>>> dict_a[ “cards” ] = [ 4, 5, 6, 7 ]
>>> print( dict_a[“cards”] )
>>> dict_a[ “cards” ].append( 10 )
>>> print( dict_a[“cards”] )
```

Presentation Link:

https://github.com/vueron/python3-workshop