Differences between python 2 and python 3

**Division operator**

print 7 / 5

print -7 / 5

Output in Python 2.x is:

1

-2

Output in Python 3.x is:

1.4

-1.4

**Print Function**

print 'Hello, people'      # Python 3.x doesn't support

print('Hope You like these facts')

'''

Output in Python 2.x :

Hello, people

Hope You like these facts

Output in Python 3.x :

File "a.py", line 1

    print 'Hello, people'

                       ^

SyntaxError: invalid syntax

**Unicode**

In Python 2, implicit str type is ASCII. But in Python 3.x implicit str type is Unicode.

|  |
| --- |
| print(type('default string '))  print(type(b'string with b '))    Output in Python 2.x (Bytes is same as str)  <type 'str'>  <type 'str'>    Output in Python 3.x (Bytes and str are different)  <class 'str'>  <class 'bytes'> |

Python 2.x also supports Unicode

|  |
| --- |
| print(type('default string '))  print(type(u'string with b '))    '''  Output in Python 2.x (Unicode and str are different)  <type 'str'>  <type 'unicode'>    Output in Python 3.x (Unicode and str are same)  <class 'str'>  <class 'str'> |

**Error handling**

There is a small change in error handling in both versions. In python 3.x, ‘as’ keyword is required.

|  |
| --- |
| try:      trying\_to\_check\_error  except NameError, err:      print err, 'Error Caused'   # Would not work in Python 3.x    '''  Output in Python 2.x:  name 'trying\_to\_check\_error' is not defined Error Caused    Output in Python 3.x :  File "a.py", line 3      except NameError, err:                      ^  SyntaxError: invalid syntax  ''' |

Run on IDE

|  |
| --- |
| try:       trying\_to\_check\_error  except NameError as err: # 'as' is needed in Python 3.x       print (err, 'Error Caused')    '''  Output in Python 2.x:  (NameError("name 'trying\_to\_check\_error' is not defined",), 'Error Caused')    Output in Python 3.x :  name 'trying\_to\_check\_error' is not defined Error Caused |