Python 2.X vs 3.X

A LITTLE HISTORY OF PYTHON 2 VS 3

- Python 2.0 was first released in 2000. Its latest version, 2.7, was released in 2010.
- Python 3.0 was released in 2008
- In 2016, 71.9% of projects used Python 2.7, but by 2017, it had fallen to 63.7%..

Obvious ..

- **1. PYTHON 2 IS LEGACY, PYTHON 3 IS THE FUTURE.**
- **2. PYTHON 2 AND PYTHON 3 HAVE DIFFERENT (SOMETIMES INCOMPATIBLE) LIBRARIES**
- **3. THERE IS BETTER UNICODE SUPPORT IN PYTHON 3** *IN PYTHON 3, TEXT STRINGS ARE UNICODE BY DEFAULT. IN PYTHON 2, STRINGS ARE STORED AS ASCII BY DEFAULT–YOU HAVE TO ADD A "U" IF YOU WANT TO STORE STRINGS AS UNICODE IN PYTHON 2.X.*
- **4. PYTHON 3 HAS IMPROVED INTEGER DIVISION**
- **5. THE TWO VERSIONS HAVE DIFFERENT PRINT STATEMENT SYNTAXES**

Important differences

- Division operator
- print function
- Unicode
- xrange
- Error Handling
- _future_ module

Division operator in Python 2.7

```
Python 2.7.16 (v2.7.16:413a49145e, Mar  2 2019, 14:32:10)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print 5/2
2
>>> print -5/2
-3
```

```
Python 3.7.2 (v3.7.2:9a3ffc0492, Dec 24 2018, 02:44:43)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print (5/2)
2.5
>>> print(-5/2)
-2.5
```

```
Python 2.7.16 (v2.7.16:413a49145e, Mar  2 2019, 14:32:10)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print 5/2
2
>>> print -5/2
-3
>>> print 5.0/2
2.5
>>> print -5.0/2
-2.5
>>> print 5//2
2
>>> print -5//2
-3
>>> print 5.0//2
2.0
>>> print -5.0//2
-3.0
>>>
```

```
Python 3.7.2 (v3.7.2:9a3ffc0492, Dec 24 2018, 02:44:43)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print (5/2)
2.5
>>> print(-5/2)
-2.5
>>> print (5.0/2)
2.5
>>> print(-5.0/2)
-2.5
>>> print (5//2)
2
>>> print (-5//2)
-3
>>> print(5.0//2)
2.0
>>> print (-5.0//2)
-3.0
>>>
```

print function

```
Python 2.7.16 (v2.7.16:413a49145e, Mar  2 2019, 14:32:10)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print 'DigiiMento Education'
DigiiMento Education
>>> print ('DigiiMento Education')
DigiiMento Education
>>> print ("DigiiMento Education")
DigiiMento Education
```

```
Python 3.7.2 (v3.7.2:9a3ffc0492, Dec 24 2018, 02:44:43)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print 'DigiiMento Education'
SyntaxError: Missing parentheses in call to 'print'. Did you mean print('DigiiMento Ed
ucation')?
>>> print ('DigiiMento Education')
DigiiMento Education
>>> print ("DigiiMento Education")
DigiiMento Education
```

Print Single and Multiple variable in Python

```
>>> #Code 1
>>> print 1
1
>>> #Code 2
>>> print (1)
1
>>> #Code 3
>>> print 1,2
1 2
>>> #Code 4
>>> print (1,2)
(1, 2)
```

*There is no difference between code 1 and code 2 in case of single variable in **Python 2.X**, but in case of multiple variables, variable with brackets -() is treated as "tuple".*

For multiple variable:

- "print variable" prints the variables without any brackets '()' and splitted by a space
- "print(variable)" prints the variables with brackets '()' and splitted by a coma ',' so it's **treated as a tuple.**

In Python 3.0, the print statement is changed to print() function. Below are equivalent codes in Python 3.0.

```
>>> #Code 1
>>> print (1)
1
>>> #Code 2
>>> print ((1))
1
>>> #Code 3
>>> print (1,2)
1 2
>>> #Code 4
>>> print ((1,2))
(1, 2)
```

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

```
Python 2.7.16 (v2.7.16:413a49145e, Mar  2 2019, 14:32:10)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print (type('default String'))
<type 'str'>
>>> print (type(b'this is a String'))
<type 'str'>
```

```
Python 3.7.2 (v3.7.2:9a3ffc0492, Dec 24 2018, 02:44:43)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print (type('default String'))
<class 'str'>
>>> print (type(b'This is a String'))
<class 'bytes'>
```

xrange:

xrange() of Python 2.x doesn’t exist in Python 3.x. In Python 2.x, range returns a list i.e. range(3) returns [0, 1, 2] while xrange returns a xrange object i. e., xrange(3) returns iterator object which work similar to Java iterator and generates number when needed.

```
Python 2.7.16 (v2.7.16:413a49145e, Mar  2 2019, 14:32:10)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> for x in xrange(1, 5):
    print(x)


1
2
3
4
>>> for x in xrange(1, 5):
    print(x),


1 2 3 4
```

```
Python 3.7.2 (v3.7.2:9a3ffc0492, Dec 24 2018, 02:44:43)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> for x in xrange(1, 5):
    print(x)


Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    for x in xrange(1, 5):
NameError: name 'xrange' is not defined
>>> for x in xrange(1, 5):
    print(x),


Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    for x in xrange(1, 5):
NameError: name 'xrange' is not defined
```

Error Handling:

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

```
Python 2.7.16 (v2.7.16:413a49145e, Mar  2 2019, 14:32:10)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> try:
    trying_to_check_error
except NameError, err:
    print err, 'Error Caused'


name 'trying_to_check_error' is not defined Error Caused
>>> try:
    trying_to_check_error
except NameError as err:
    print err, 'Error Caused'


name 'trying_to_check_error' is not defined Error Caused
```

```
Python 3.7.2 (v3.7.2:9a3ffc0492, Dec 24 2018, 02:44:43)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> try:
    trying_to_check_error
except NameError, err:
    print (err, 'Error Caused')

SyntaxError: invalid syntax
>>> try:
    trying_to_check_error
except NameError as err:
    print (err, 'Error Caused' )


name 'trying_to_check_error' is not defined Error Caused
```