Python 2 to Python 3, New Features

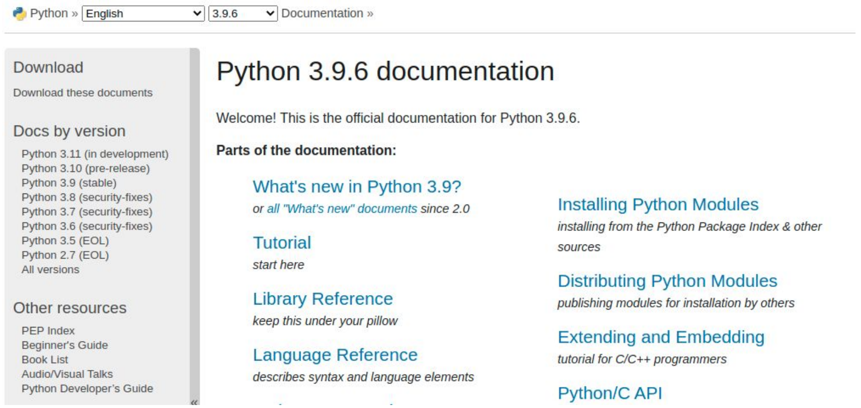
CONTET

CONTET

- Python 2 to 3, An Overview
    - Python in 2021
    - Why Python 2 -> 3 incompatible?
    - What’s new?
    - Python 3 version difference
- Migration Strategy

Python in 2021

- Current → 3.11, dev / 3.10, pre / 3.9, stable
- 2.0 release in 2000, 2.7 in 2010, and 2.x was sunseted in January 1, 2020
- 3.0 started in 2006
- 3.5 and 2.7 marked as EOL, 3.6 and above are recommended



Python » English 3.9.6 Documentation »

Download

Download these documents

Docs by version

- Python 3.11 (in development)
- Python 3.10 (pre-release)
- Python 3.9 (stable)
- Python 3.8 (security-fixes)
- Python 3.7 (security-fixes)
- Python 3.6 (security-fixes)
- Python 3.5 (EOL)
- Python 2.7 (EOL)
- All versions

Other resources

- PEP Index
- Beginner's Guide
- Book List
- Audio/Visual Talks
- Python Developer's Guide

Python 3.9.6 documentation

Welcome! This is the official documentation for Python 3.9.6.

Parts of the documentation:

What's new in Python 3.9?
or all "What's new" documents since 2.0

Tutorial
start here

Library Reference
keep this under your pillow

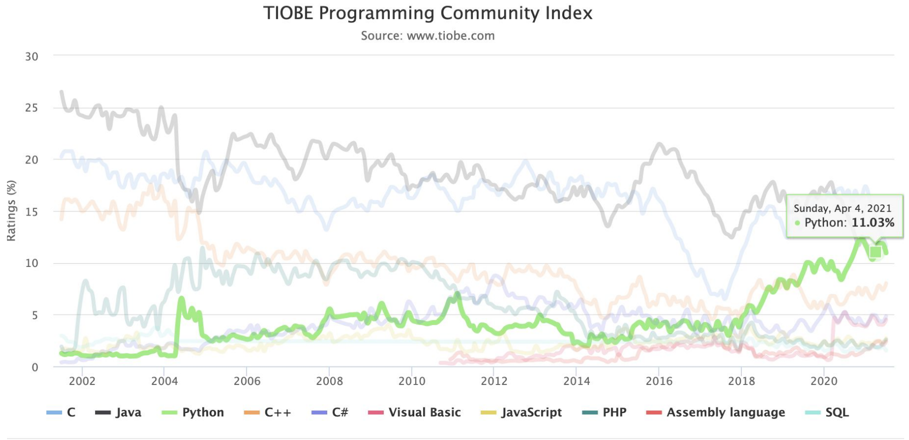
Language Reference
describes syntax and language elements

Installing Python Modules
installing from the Python Package Index & other sources

Distributing Python Modules
publishing modules for installation by others

Extending and Embedding
tutorial for C/C++ programmers

Python/C API



Python Doc
Sunsetting Python 2

Why Python 2 -> 3 incompatible?

- Change text model *(main)*

“abcd”

- Python 2 → bytes representing 97, 98, 99, and 100(ASCII) || string consisting of “abcd”
- Python 3 → string consisting of “abcd”
  “Text and binary data in Python 2 are a mess”

  - *“UnicodeDecodeError or UnicodeEncodeError in 2.x almost never points you to the code that is broken”*, while for *UnicodeError* in 3.x it points closely
  - encoding operations may raise decoding errors (vice-versa) and no exception throws in Python 2 (e.g. two 8-bit strings with data in different text encodings concatenated)
  - *“as far as is practical, always require users to opt in to behaviours that pose a significant risk of silently corrupting data in non-ASCII compatible encodings”* (guiding philosophy of text model in Python 3)

Python 3 Q & A
Why Python 3 exists

Other Notable Changes

- Drop deprecated features (or something have superior alternatives)
- Reduce the number of statements
  - e.g. *print*, *exec* (accept keyword arguments)
- Replace concrete list and dict objects with more memory efficient alternatives
  - *" many of Python's core APIs were designed before the introduction of the iterator protocol"*
- Renaming modules to be more PEP 8 compliant and to automatically use C accelerators when available
  - *"Using the API compatible C accelerators means end users no longer need to know about and explicitly request the accelerated variant"*
- … ...

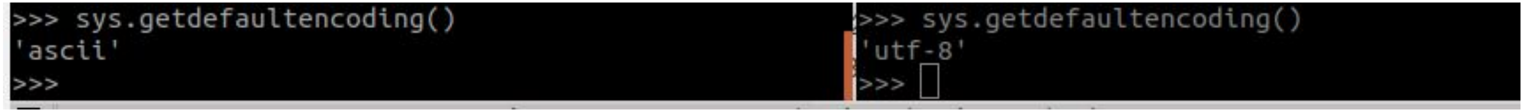
Python 3 Q & A
PEP 399 Pure Python/C Accelerator Module Compatibility Requirements

What’s new?

- Unicode
- Views & Iterators Instead of Lists
- Syntax Change
    - New Style Classes
    - Typing Hints
    - Advanced Unpacking
    - Chained Exception
    - Advanced String Formatting
    - … …

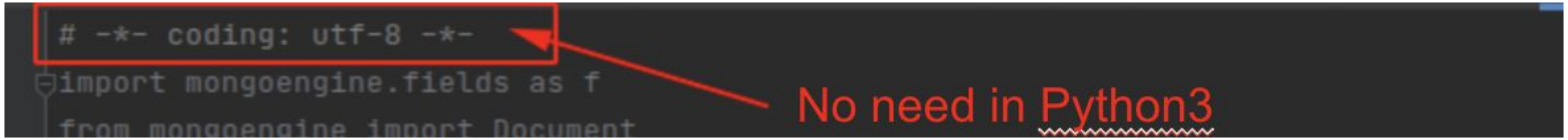
What’s New In Python 3.0 (by Guido, author of Python)
What’s New in Python (full list of Pyhon3.x version)

Unicode



```
>>> sys.getdefaultencoding()
'ascii'
>>>
```

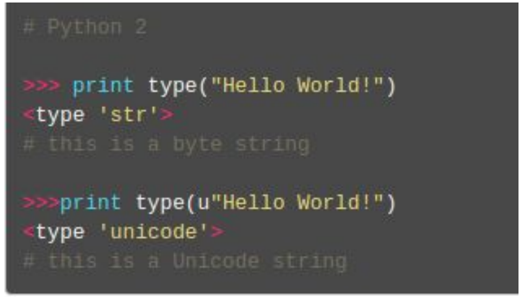
```
>>> sys.getdefaultencoding()
'utf-8'
>>>
```



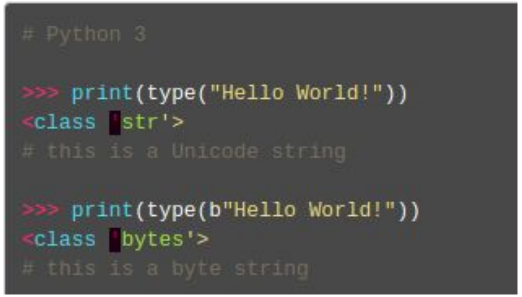
```
# -*- coding: utf-8 -*-
import mongoengine.fields as f
from mongoengine import Document
```

No need in Python3

- *String*
  - Python2 --> *text* and *bytes*, silently converted
  - Python3 -- > incompatible, any attempt to mix text and data raises *TypeError*
  - *u"..."* does nothing in Python3
  - *b"..."* does nothing in Python2



```
# Python 2

>>> print type("Hello World!")
<type 'str'>
# this is a byte string

>>>print type(u"Hello World!")
<type 'unicode'>
# this is a Unicode string
```

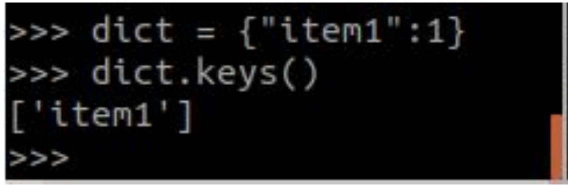


```
# Python 3

>>> print(type("Hello World!"))
<class 'str'>
# this is a Unicode string

>>> print(type(b"Hello World!"))
<class 'bytes'>
# this is a byte string
```

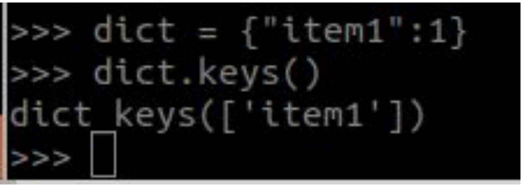
Pragmatic Unicode (PyCon 2012, more details)
The Conservative Python 3 Porting Guide - String
Strings, Bytes, and Unicode in Python 2 and 3

Views & Iterators Instead of Lists

- dict method return 'views' instead of lists (*"views are simply like a window on the keys and values (or items) of a dictionary", more lightweight*)

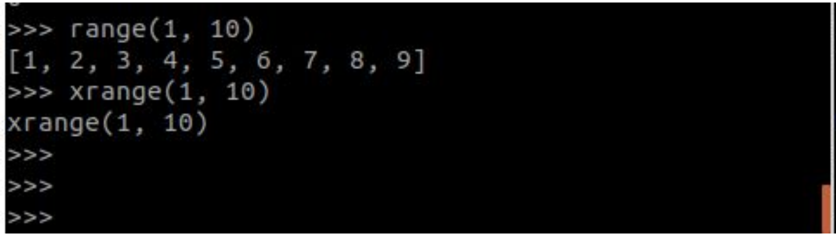


```
>>> dict = {"item1":1}
>>> dict.keys()
['item1']
>>>
```

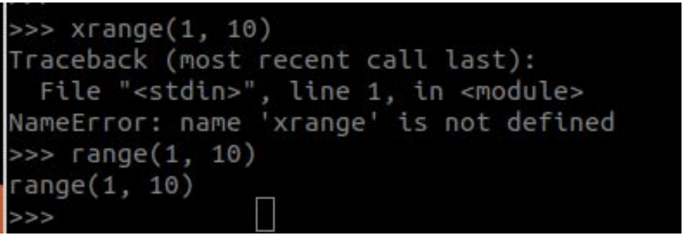


```
>>> dict = {"item1":1}
>>> dict.keys()
dict_keys(['item1'])
>>>
```

- *dict.iterkeys()*, *dict.iteritems()* and *dict.itervalues()* no longer supported (*.items()*)
- *xrange()* is abandoned, use *range()*



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



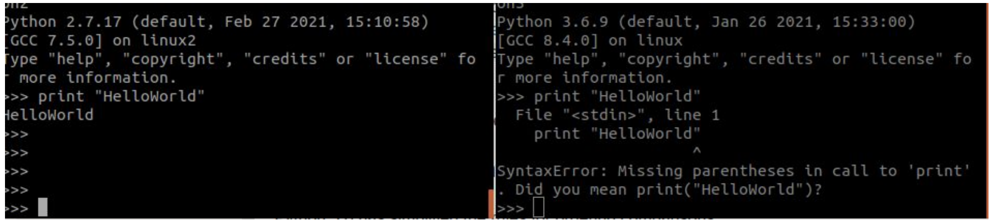
```
>>> xrange(1, 10)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'xrange' is not defined
>>> range(1, 10)
range(1, 10)
>>>
```

- *zip()* returns an iterator in Python 3, returns a list of tuples in Python 2

What's New In Python 3.0
What are dictionary view objects?
PEP 469 -- Migration of dict iteration code to Python 3 (Migration guide)

Syntax Change

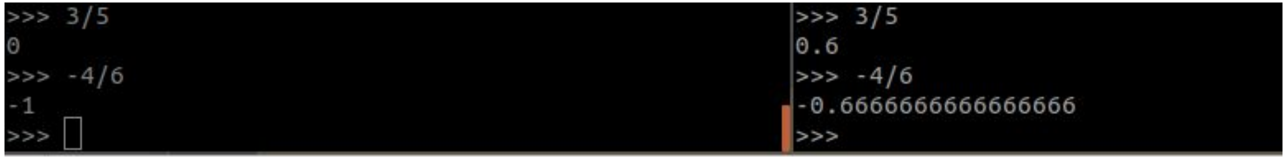
- *print* is a function in py3, which is a statement in py2



```
ython 2.7.17 (default, Feb 27 2021, 15:10:58)
[GCC 7.5.0] on linux2
ype "help", "copyright", "credits" or "license" fo
 more information.
>> print "HelloWorld"
elloWorld
>>
>>
>>
>>
>>
```

```
Python 3.6.9 (default, Jan 26 2021, 15:33:00)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" fo
r more information.
>>> print "HelloWorld"
  File "<stdin>", line 1
    print "HelloWorld"
                     ^
SyntaxError: Missing parentheses in call to 'print'
. Did you mean print("HelloWorld")?
>>>
```

- Integer Division (you can use *'division'* for compatibility)



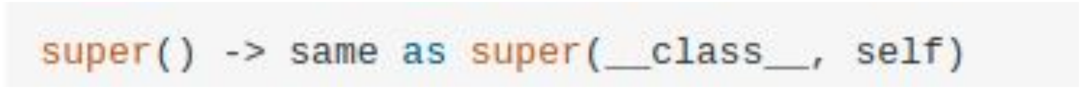
```
>>> 3/5
0
>>> -4/6
-1
>>>
```

```
>>> 3/5
0.6
>>> -4/6
-0.6666666666666666
>>>
```

[What's New In Python 3.0](#)
[10 awesome features of Python that you can't use because you refuse to upgrade to Python 3](#)

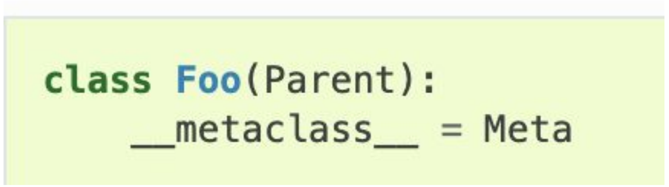
New Style Classes

- *super()* without argument format in Python 3

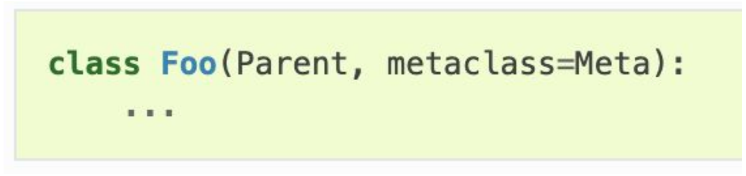


```
super() -> same as super(__class__, self)
```

- In Python 3, all classes are new-style: *object* is the default superclass
- metaclass is specified with a keyword argument in Python 3



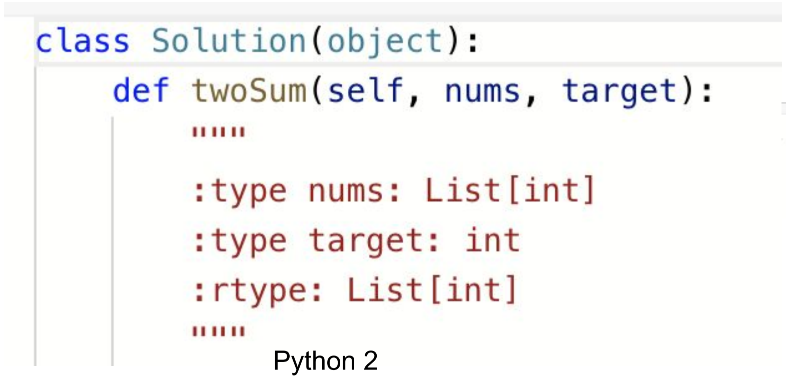
```
class Foo(Parent):
    __metaclass__ = Meta
```



```
class Foo(Parent, metaclass=Meta):
    ...
```

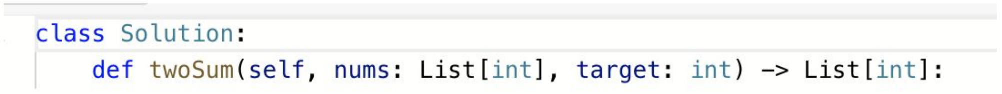
[Python 3 Porting Guide -- Classes]

Type Hint



```
class Solution(object):
    def twoSum(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: List[int]
        """
```

Python 2



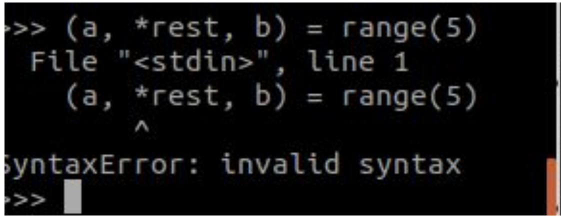
```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
```

Python 3

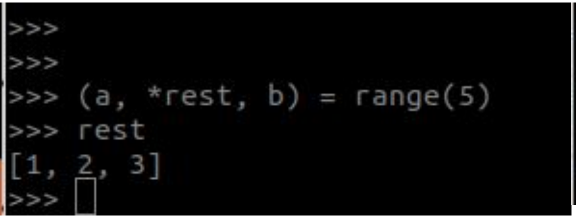
- formal solution to statically indicate the type of a value within your Python code (since 3.5+)
- Help catch certain errors, document your code, build and maintain a cleaner architecture
- take developer time and effort to add

typing — Support for type hints
PEP 484 -- Type Hints

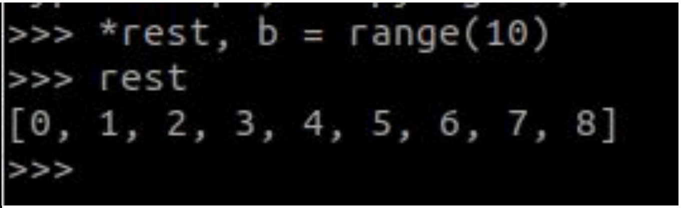
Advanced Unpacking



```
>> (a, *rest, b) = range(5)
 File "<stdin>", line 1
   (a, *rest, b) = range(5)
       ^
yntaxError: invalid syntax
>>
```



```
>>>
>>>
>>> (a, *rest, b) = range(5)
>>> rest
[1, 2, 3]
>>>
```



```
>>> *rest, b = range(10)
>>> rest
[0, 1, 2, 3, 4, 5, 6, 7, 8]
>>>
```

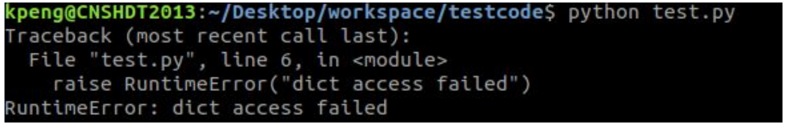
- e.g. read first & last line of a file

```
>>> with open("using_python_to_profit") as f:
...     first, *_, last = f.readlines()  # Warning: this puts the whole file contents in memory!
>>> first
'Step 1: Use Python 3\n'
>>> last
'Step 10: Profit!\n'
```

Chained Exception

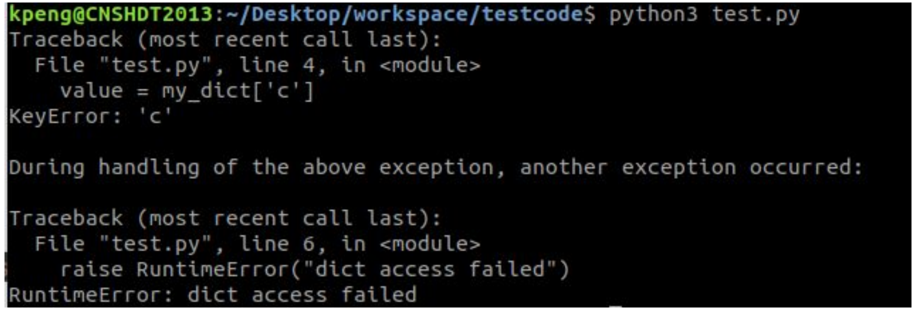
```
my_dict = {'a': 1, 'b': 2}
try:
    value = my_dict['c']
except KeyError:
    raise RuntimeError("dict access failed")
```

- Python 2



```
kpeng@CNSHDT2013:~/Desktop/workspace/testcode$ python test.py
Traceback (most recent call last):
  File "test.py", line 6, in <module>
    raise RuntimeError("dict access failed")
RuntimeError: dict access failed
```

- Python 3



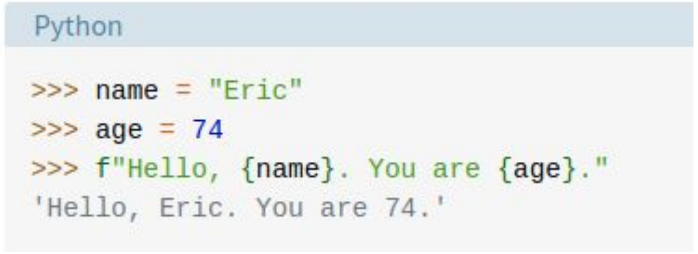
```
kpeng@CNSHDT2013:~/Desktop/workspace/testcode$ python3 test.py
Traceback (most recent call last):
  File "test.py", line 4, in <module>
    value = my_dict['c']
KeyError: 'c'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "test.py", line 6, in <module>
    raise RuntimeError("dict access failed")
RuntimeError: dict access failed
```

Python 3 shows whole chain of exception while Python 2 may lose the original traceback

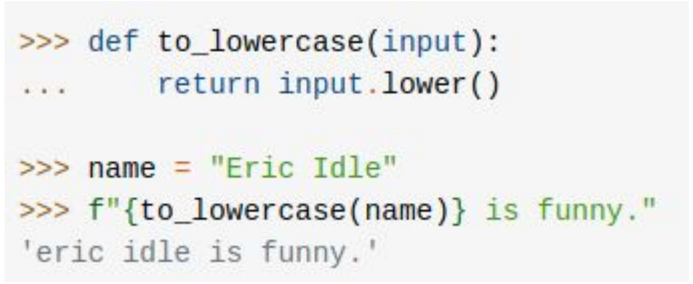
PEP 3134 -- Exception Chaining and Embedded Tracebacks

Advanced String Formatting

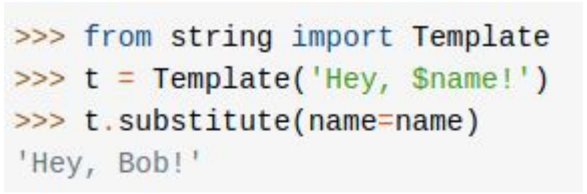
- % operator e.g. *Hello %s' % name*
- .format() e.g. *'Hello, {}'.format(name)*
- String Interpolation / f-Strings (Python 3.6+ Supported, more Pythonic)



```
Python
>>> name = "Eric"
>>> age = 74
>>> f"Hello, {name}. You are {age}."
'Hello, Eric. You are 74.'
```



```
>>> def to_lowercase(input):
...     return input.lower()

>>> name = "Eric Idle"
>>> f"{to_lowercase(name)} is funny."
'eric idle is funny.'
```

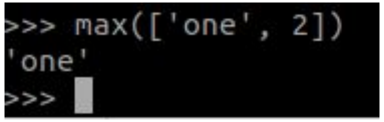
- Template Strings (handling format strings generated from user input, safer)
    - *from string import Template*
    - 

```
>>> from string import Template
>>> t = Template('Hey, $name!')
>>> t.substitute(name=name)
'Hey, Bob!'
```

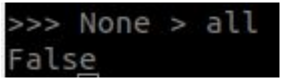
[Python String Formatting Best Practices]
[Python 3's f-Strings: An Improved String Formatting Syntax (Guide)]
[Be Careful with Python's New-Style String Format]

No more comparison of everything to everything *("Explicit is better than implicit")*

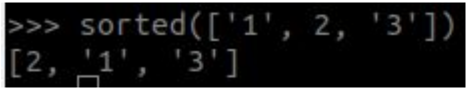
- Python 2



```
>>> max(['one', 2])
'one'
>>>
```

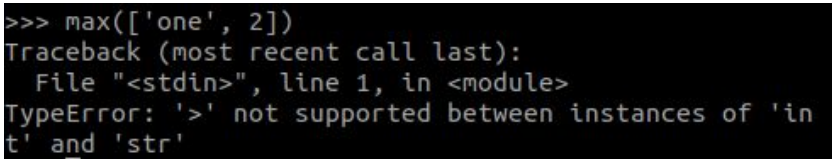


```
>>> None > all
False
```

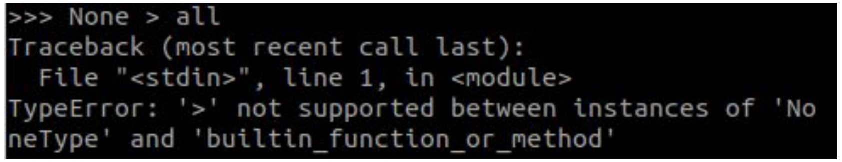


```
>>> sorted(['1', 2, '3'])
[2, '1', '3']
```

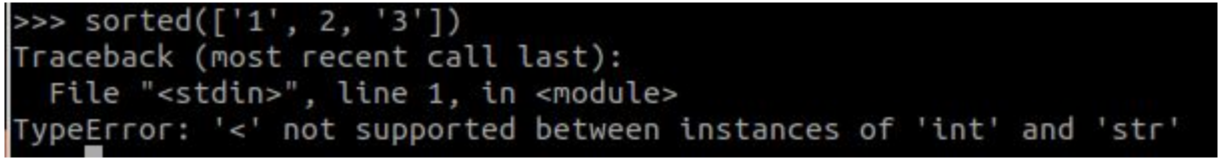
- Python 3



```
>>> max(['one', 2])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '>' not supported between instances of 'in
t' and 'str'
```



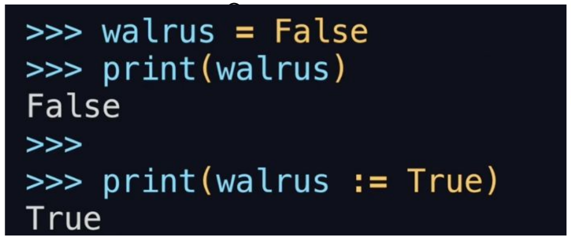
```
>>> None > all
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '>' not supported between instances of 'No
neType' and 'builtin_function_or_method'
```



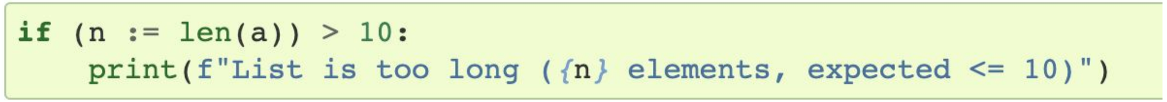
```
>>> sorted(['1', 2, '3'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '<' not supported between instances of 'int' and 'str'
```

Python 3 version difference

- 3.6 (released on December 23, 2016)
  - formatted string literals
- 3.7 (released on June 27, 2018)
  - *async* & *await* as reserved keyword (coroutines, asyncio …)
- 3.8 (released on October 14th, 2019)
  - assignment expressions *:= (The Walrus “海象” Operator)*
  - allow you to assign and return a value in the same expression



```
>>> walrus = False
>>> print(walrus)
False
>>>
>>> print(walrus := True)
True
```



```
if (n := len(a)) > 10:
    print(f"List is too long ({n} elements, expected <= 10)")
```

- 3.9 (released on October 5th, 2020)
  - New String Methods to Remove Prefixes and Suffixes
  - …

[What’s New in Python](#)
[PEP 602 -- Annual Release Cycle for Python](#)

CONTET

Overview

- Migration strategies
  - S1. Only supporting Python 3
  - S2. Separate branches for Python 2 and Python 3
  - S3. Converting to Python 3 with 2to3
  - Recommend. “fix code up until it works on Python 3, then support Python2.7”

- Tips
  - have a good test coverage (try to over 80%)
  - don’t regress
  - find your blocker dependencies
  - … ...

Porting Python 2 Code to Python 3
Supporting Python 3: An in-depth guide
10 Tips for Upgrading to Python 3

Library Support

- 2to3
  - end2end tool from Python 2 to Python 3, generate new code
  - a rule set for transformation
- future
  - modernize Python 2 code without changing the effect of the code
  - E.g. division, print_function()
- six (2*3=6 :))
  - a Python 2 and 3 compatibility library
  - *“ the goal of writing Python code that is compatible on both Python versions”*
- caniusepython3 (not active now)
  - figure out dependencies you need to transition to Python 3

python-future
2to3
six