Advanced Python Topics Course @ Samsung

Amit Kotlovski

- amit@amitkot.com
- http://amitkot.com (http://amitkot.com)

Course Materials

https://tinyurl.com/python-topics-3 (https://tinyurl.com/python-topics-3)

Tools

Interpreter

- http://python.org (http://python.org) (Download latest Python 2 or 3)

IDE

- PyCharm (https://www.jetbrains.com/pycharm/ (https://www.jetbrains.com/pycharm/))
- Vim (https://www.vim.org/ (https://www.vim.org/))
- Anything but Windows Notepad

In [1]:

```
def foo(a, b):
    return a + b
```

In [2]:

```
foo(34, 792)
```

Out[2]:

```
826
```

In [3]:

```
foo('cat', 'dog')
```

Out[3]:

```
'catdog'
```

In [6]:

```
d = {'first': 'cat', 'second': 'dog'}
d = [45, 67, 'dfsadfsd']
```

In [7]:

```
for key in d:
    print(key)
```

```
45
67
dfsadfsd
```

In [8]:

```
foo(78, '1')
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-8-d68eeacacc92> in <module>()
----> 1 foo(78, '1')

<ipython-input-1-89a95242c6a7> in foo(a, b)
      1 def foo(a, b):
----> 2     return a + b

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [9]:

```
foo(str(78), '1')
```

Out[9]:

```
'781'
```

In [10]:

```
1, 2, 3
```

Out[10]:

```
(1, 2, 3)
```

In [11]:

```
def foo():
    return 23, 'banana'
```

In [12]:

```
foo()
```

Out[12]:

```
(23, 'banana')
```

In [13]:

```
res = foo()
```

In [14]:

```
print(res)
```

```
(23, 'banana')
```

In [15]:

```
a, b = foo()
```

In [16]:

```
a
```

Out[16]:

```
23
```

In [17]:

```
b
```

Out[17]:

```
'banana'
```

In [18]:

```
a, b = (99, 100)
```

In [19]:

```
a
```

Out[19]:

```
99
```

In [20]:

```
b
```

Out[20]:

```
100
```

In [21]:

```
c = a, b
```

In [22]:

```
c
```

Out[22]:

```
(99, 100)
```

In [23]:

```
a, b = b, a
```

In [24]:

```
dict(a=34, b=67, v=900)
```

Out[24]:

```
{'a': 34, 'b': 67, 'v': 900}
```

In [25]:

```
c = 'banana'
```

In [26]:

```
id(c)
```

Out[26]:

```
4404276392
```

In [27]:

```
c2 = c.upper()
```

In [28]:

```
c2
```

Out[28]:

```
'BANANA'
```

In [29]:

```
id(c2)
```

Out[29]:

```
4404906616
```

In [33]:

```
a = 'cat'
b = a
print('a', id(a))
print('b', id(b))
a = 'dog'
print(id(a))
```

```
a 4370498984
b 4370498984
4404907400
```

In [34]:

```
a
```

Out[34]:

```
'dog'
```

In [36]:

```
f"what a nice {a}"
```

Out[36]:

```
'what a nice dog'
```

In [37]:

```
d = [1, 2, 3]
```

In [38]:

```
di = iter(d)
```

In [39]:

```
next(di)
```

Out[39]:

```
1
```

In [40]:

```
next(di)
```

Out[40]:

```
2
```

In [41]:

```
next(di)
```

Out[41]:

```
3
```

In [42]:

```
next(di)
```

```
----------------------------------------------------------------------
-------
StopIteration                           Traceback (most recent cal
l last)
<ipython-input-42-6c6a363d385f> in <module>()
----> 1 next(di)

StopIteration:
```

In [43]:

```
d = {1:'1', 2: '2'}
2 in d
```

Out[43]:

```
True
```

In [44]:

```
if 2 in d:
    print('hi')
```

```
hi
```

In [45]:

```
d
```

Out[45]:

```
{1: '1', 2: '2'}
```

In [50]:

```
def check_if_something():
    return {1: '1', 2: '2'}

d = check_if_something()
if d:
    for k, v in d.items():
        print('hi: ', k, v)
```

```
hi:  1 1
hi:  2 2
```

In [52]:

```
from collections import namedtuple


Quad = namedtuple('Quad', 'a b c')
```

In [53]:

```
Quad(a=12, b=45, c=34)
```

Out[53]:

```
Quad(a=12, b=45, c=34)
```

In [54]:

```
Quad = namedtuple('Quad', ['a', 'b', 'c'])
```

In [55]:

```
Quad(a=12, b=45, c=34)
```

Out[55]:

```
Quad(a=12, b=45, c=34)
```

In [56]:

```
Quad(1, 2)
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-56-30fbbed23e51> in <module>()
----> 1 Quad(1, 2)

TypeError: __new__() missing 1 required positional argument: 'c'
```

In [57]:

```
v = Quad(1, 2, 3)
```

In [59]:

```
print(v)
```

```
Quad(a=1, b=2, c=3)
```

In [60]:

```
v.a
```

Out[60]:

```
1
```

In [61]:

```
help(v)
```

```
Help on Quad in module __main__ object:

class Quad(builtins.tuple)
 |  Quad(a, b, c)
 |
 |  Method resolution order:
 |      Quad
 |      builtins.tuple
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __getnewargs__(self)
 |      Return self as a plain tuple.  Used by copy and pickle.
 |
 |  __repr__(self)
 |      Return a nicely formatted representation string
 |
 |  _asdict(self)
 |      Return a new OrderedDict which maps field names to their val
ues.
 |
 |  _replace(_self, **kwds)
 |      Return a new Quad object replacing specified fields with new
values
 |
 |  ----------------------------------------------------------------
------
 |  Class methods defined here:
 |
 |  _make(iterable, new=<built-in method __new__ of type object at 0
x10447a010>, len=<built-in function len>) from builtins.type
 |      Make a new Quad object from a sequence or iterable
 |
 |  ----------------------------------------------------------------
------
 |  Static methods defined here:
 |
 |  __new__(_cls, a, b, c)
 |      Create new instance of Quad(a, b, c)
 |
 |  ----------------------------------------------------------------
------
 |  Data descriptors defined here:
 |
 |  a
 |      Alias for field number 0
 |
 |  b
 |      Alias for field number 1
 |
 |  c
 |      Alias for field number 2
 |
 |  ----------------------------------------------------------------
------
 |  Data and other attributes defined here:
 |
 |  _fields = ('a', 'b', 'c')
 |
 |  _source = "from builtins import property as _property, tupl..._i
temget...
 |
 |  ------------------------------------------------------------------
------
 |  Methods inherited from builtins.tuple:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, key, /)
 |      Return self[key].
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __hash__(self, /)
 |      Return hash(self).
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mul__(self, value, /)
 |      Return self*value.n
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __rmul__(self, value, /)
 |      Return self*value.
 |
 |  count(...)
 |      T.count(value) -> integer -- return number of occurrences of
value
 |
 |  index(...)
 |      T.index(value, [start, [stop]]) -> integer -- return first i
ndex of value.
```

```
 |      Raises ValueError if the value is not present.
```

In [62]:

```
def foo(func):
    return func()
```

In [63]:

```
foo(list)
```

Out[63]:

```
[]
```

In [64]:

```
foo([])
```

```
---------------------------------------------------------------------
------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-64-542a79b56cff> in <module>()
----> 1 foo([])

<ipython-input-62-b54098fde56c> in foo(func)
      1 def foo(func):
----> 2     return func()

TypeError: 'list' object is not callable
```

In [65]:

```
foo(tuple)
```

Out[65]:

```
()
```

In [66]:

```
int()
```

Out[66]:

```
0
```

In [67]:

```
int(1)
```

Out[67]:

```
1
```

In [68]:

```
def tasty():
    return 'tasty'
```

In [69]:

```
foo(tasty)
```

Out[69]:

```
'tasty'
```

In [70]:

```
foo(lambda: 'tasty')
```

Out[70]:

```
'tasty'
```

In [71]:

```
def call_with_x(func, x):
    return func(x)
```

In [72]:

```
def x_2(x):
    return x**2

call_with_x(x_2, 9)
```

Out[72]:

```
81
```

In [73]:

```
call_with_x(lambda x: x**2, 4)
```

Out[73]:

```
16
```

In [74]:

```
from collections import deque

q = deque([1, 2, 3])
```

In [76]:

```
for i in q:
    print(i)
```

```
1
2
3
```

In [77]:

```
q
```

Out[77]:

```
deque([1, 2, 3])
```

In [78]:

```
q
```

Out[78]:

```
deque([1, 2, 3])
```

In [79]:

```
help(q)
```

```
Help on deque object:

class deque(builtins.object)
 |  deque([iterable[, maxlen]]) --> deque object
 |
 |  A list-like sequence optimized for data accesses near its endpoi
nts.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __bool__(self, /)
 |      self != 0
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __copy__(...)
 |      Return a shallow copy of a deque.
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, key, /)
 |      Return self[key].
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __iadd__(self, value, /)
 |      Implement self+=value.
 |
 |  __imul__(self, value, /)
 |      Implement self*=value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signatur
e.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mul__(self, value, /)
 |      Return self*value.n
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate
signature.
 |
 |  __reduce__(...)
 |      Return state information for pickling.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __reversed__(...)
 |      D.__reversed__() -- return a reverse iterator over the deque
 |
 |  __rmul__(self, value, /)
 |      Return self*value.
 |
 |  __setitem__(self, key, value, /)
 |      Set self[key] to value.
 |
 |  __sizeof__(...)
 |      D.__sizeof__() -- size of D in memory, in bytes
 |
 |  append(...)
 |      Add an element to the right side of the deque.
 |
 |  appendleft(...)
 |      Add an element to the left side of the deque.
 |
 |  clear(...)
 |      Remove all elements from the deque.
 |
 |  copy(...)
 |      Return a shallow copy of a deque.
 |
 |  count(...)
 |      D.count(value) -> integer -- return number of occurrences of
value
 |
 |  extend(...)
 |      Extend the right side of the deque with elements from the it
erable
 |
 |  extendleft(...)
 |      Extend the left side of the deque with elements from the ite
rable
 |
 |  index(...)
 |      D.index(value, [start, [stop]]) -> integer -- return first i
ndex of value.
 |      Raises ValueError if the value is not present.
 |
 |  insert(...)
 |      D.insert(index, object) -- insert object before index
 |
 |  pop(...)
 |      Remove and return the rightmost element.
 |
 |  popleft(...)
 |      Remove and return the leftmost element.
 |
 |  remove(...)
 |      D.remove(value) -- remove first occurrence of value.
 |
 |  reverse(...)
 |      D.reverse() -- reverse *IN PLACE*
 |
 |  rotate(...)
 |      Rotate the deque n steps to the right (default n=1).  If n i
s negative, rotates left.
 |
 |  ----------------------------------------------------------------
------
 |  Data descriptors defined here:
 |
 |  maxlen
 |      maximum size of a deque or None if unbounded
 |
 |  ----------------------------------------------------------------
------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

In [81]:

```
q = Quad(1, 2, 3)
```

In [82]:

```
q[0]
```

Out[82]:

```
1
```

In [83]:

```
def is_comment_line(line):
    """Checks if provided line is a comment.

    The function checks if the provided line is a comment
    by inspecting it's characters to see wether it starts
    with a a "#" char.

    :param line: the line to be checked
    :return: true if the line is indeed a comment, false otherwise
    """
    return line.startswith("#")
```

In [84]:

```
help(is_comment_line)
```

```
Help on function is_comment_line in module __main__:

is_comment_line(line)
    Checks if provided line is a comment.
    
    The function checks if the provided line is a comment
    by inspecting it's characters to see wether it starts
    with a a "#" char.
    
    :param line: the line to be checked
    :return: true if the line is indeed a comment, false otherwise
```

In [85]:

```
is_comment_line?
```

In [86]:

```
is_comment_line.__doc__
```

Out[86]:

```
'Checks if provided line is a comment.\n    \n    The function check
s if the provided line is a comment \n    by inspecting it\'s charac
ters to see wether it starts \n    with a a "#" char.\n    \n    :pa
ram line: the line to be checked\n    :return: true if the line is i
ndeed a comment, false otherwise\n    '
```

In [ ]:

```
a = 'cat'
b = a
print(b)
```

In [2]:

```
print(3)
```

```
3
```

In [3]:

```
'A'.islower()
```

Out[3]:

```
False
```

In [4]:

```
'a'.islower()
```

Out[4]:

```
True
```

In [5]:

```
str.islower('a')
```

Out[5]:

```
True
```

In [6]:

```
data = [(1, 300), (60, 5), (3000, 9)]
sorted(data, key=lambda x: x[1])
```

Out[6]:

```
[(60, 5), (3000, 9), (1, 300)]
```

In [18]:

```
import functools
import operator

def calculate(exp):
    """
    >>> calculate("3 + 32 + 45 + 7")
    87
    >>> calculate("3+++ 32")
    35
    >>> calculate("+ 25 +++6 +")
    31
    """
    return functools.reduce(operator.add,
                            map(int,
                                filter(str.isdigit,
                                       map(str.strip,
                                           exp.split("+")))))
```

In [20]:

```
calculate('33 + 0')
```

Out[20]:

```
33
```

In [16]:

```
'3 '.isdigit()
```

Out[16]:

```
False
```

In [24]:

```python
def foo():
    i = 300
    def bar():
        return i + 32
    i = 500
    return bar

a = foo()
print(a())
```

```
532
```

In [25]:

```python
foo(aaaa=500)
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-25-28ba2ef7e235> in <module>()
----> 1 foo(aaaa=500)

TypeError: foo() got an unexpected keyword argument 'aaaa'
```

In [31]:

```python
def counter(fn):
    count = 0
    wrapper.count = 0

    def wrapper():
        count += 1
        res = fn()
        print("+++ fn call counter is {}".format(wrapper.count))
        return res
    return wrapper

def cookie():
    print("cookie")

cc = counter(cookie)
```

In [32]:

```
cc()
```

```
cookie

---------------------------------------------------------------------
-------
AttributeError                            Traceback (most recent cal
l last)
<ipython-input-32-f6b58c567c1a> in <module>()
----> 1 cc()

<ipython-input-31-8326e6b5f14a> in wrapper()
      5         count += 1
      6         res = fn()
----> 7         print("+++ fn call counter is {}".format(wrapper.cou
nt))
      8         return res
      9     #wrapper.count = 0

AttributeError: 'function' object has no attribute 'count'
```

In [52]:

```
class A:
    def __init__(self):
        self.__ver = 9


class B(A):
    def set_ver(self, value):
        self.__ver = value
        self.banana = 45

    def __repr__(self):
        return f"B(banana={self.banana})"

a = A()
```

In [49]:

```
a.__ver
```

```
---------------------------------------------------------------------
-------
AttributeError                            Traceback (most recent cal
l last)
<ipython-input-49-c2f4984d2431> in <module>()
----> 1 a.__ver

AttributeError: 'A' object has no attribute '__ver'
```

In [35]:

```
a._A__ver
```

Out[35]:

```
9
```

In [40]:

```
b = B()
```

In [41]:

```
b.set_ver(900)
```

In [42]:

```
dir(b)
```

Out[42]:

```
['_A__ver',
 '_B__ver',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 'set_ver']
```

In [43]:

```
b._B__ver
```

Out[43]:

```
900
```

In [45]:

```
b._A__ver = 700
```

In [46]:

```
b._A__ver
```

Out[46]:

```
700
```

In [47]:

```
print(b)
```

```
<__main__.B object at 0x104284668>
```

In [54]:

```
c = B()
c.set_ver(34)
print(c)
```

```
B(banana=45)
```

In [55]:

```
print(c)
```

```
B(banana=45)
```

In [63]:

```
class C:
    count = 0

    @staticmethod
    def foo():
        print(4)
```

In [64]:

```
c = C()
c.foo()
```

```
4
```

In [62]:

```
C.foo()
```

```
4
```

In [65]:

```
c.count
```

Out[65]:

```
0
```

In [66]:

```
C.count
```

Out[66]:

```
0
```

In [77]:

```
class Table(object):
    def __init__(self):
        self._color = None

    @property
    def color(self):
        return self._color

    @color.setter
    def color(self, value):
        if value == 'Yellow':
            raise ValueError('Evil color')
        self._color = value
```

In [75]:

```
t = Table()
```

In [76]:

```
t.color = 'Yellow'
```

```
---------------------------------------------------------------------
-------
ValueError                                Traceback (most recent cal
l last)
<ipython-input-76-46130d8b5080> in <module>()
----> 1 t.color = 'Yellow'

<ipython-input-74-bc7c5b793d42> in color(self, value)
     10     def color(self, value):
     11         if value == 'Yellow':
---> 12             raise ValueError('Evil color')
     13         self._color = value

ValueError: Evil color
```

In [70]:

```
t.color
```

Out[70]:

```
'Yellow'
```