tutorial_2_solutions

September 7, 2016

1 Solution to the exercises

1.1 Exercise: revision on type-casting

```
>>> s = 'a little string of size '
>>> 'This is ' + s + '.'
'This is a little string.'
>>> s + 26
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-12-1ffac3e9a4d8> in <module>()
----> 1 s + 26

TypeError: Can't convert 'int' object to str implicitly
```

Here you have two options: convert `s` to integers, which does not make sense (how would you do that? You could use ASCII but again, what is the point?). Instead, you want to convert the number (`int`) `25` into the string (`str`) `'25'`. This is done easily by type-casting `s`:

```
>>> s = 'a little string of size '
>>> 'This is ' + s + '.'
'This is a little string.'
>>> new_s = s + str(26)
>>> new_s
'a little string of size 26'
```

As a little bonus:

```
>>> len(new_s)
26
```

Isn't the world fantastic?

1.2 Exercise: divisions

You tried to divide by 0 and as you know (we hope), this is not possible and Python signals this by returning an accurately named `ZeroDivisionError`.

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-9-18ba57c3b8d9> in <module>()
      3     return a/b
      4
----> 5 divide(1, 0)

<ipython-input-9-18ba57c3b8d9> in divide(a, b)
      1 def divide(a, b):
      2     """divides a with b"""
----> 3     return a/b
      4
      5 divide(1, 0)

ZeroDivisionError: division by zero
```

This also tells you that the error occured while executing the instruction `divide(1, 0)`, inside the function `divide`. And in case you did not get it it specifies that the error occured because of a `division by zero`.

If you want your function to instead return `'NaN'`, you have to catch the `ZeroDivisionError`:

```
def divide(a, b):
    """Divideds a with b"""
    try:
        return a/b
    except ZeroDivisionError:
        return 'NaN'
```

And now, if we try to divide 1 by 0:

```
>>> divide(1, 0)
'NaN'
```

1.3 Exercise: reserved keywords

The problematic code is this:

```
>>> b = (1, 2, 3) # OK
>>> list = [4, 5, 6] # redefining list
>>> c = list([1, 2, 3]) # WRONG!
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-70ea19d80413> in <module>()
----> 1 c = list(b)

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

You simply called a variable `list` which is a reserved keyword defining … lists! While `list` originally converts an iterable in a list, Python lets you do this but `list` is now a list (mind bending, I know…) and no longer a keyword:

```
>>> list
[4, 5, 6]
```

The errors is appropriately a `TypeError` as it makes no sense to call a list, it is not a function (i.e. not callable).

You might have noticed though that the keyword `list` is not the only way to define a list and you can still do:

```
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]
>>> type(a)
list
```

Whaaaaaat?

1.4 Exercise: scope of a function

when you type

```
x = 10
```

you define a variable in the current environment. Print `locals()` to see a list of all the variables existing in your current (local) environment.

However, when you enter the function add, a new environement is created in which only the variable `a` exists:

```
x = 10

def add(a):
    print(locals())
    x = x + a
    print(x)
```

before crashing, your function will print the variables existing inside the function (only `a`).

When you define a new function

```
def add(a, b):
    c = a + b

add(2, 3)
print(c)
```

the variable `c` exists **only** inside of the function `add` and no longer exists as soon as you exit it! That is why you need to use the return statement to *return* a variable from the function :)

2 Start of a solution: open exercise

Choose a file

```
In [1]: filename = 'odyssey.txt'
```

We will consider the following as punctuation and ignore it:

```
In [2]: punctuation = '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~0123456789'
```

```
In [6]: def count_word(filename):
            f = open(filename, 'r')
            res = dict()
            for line in f.readlines():
                l = line.strip()
                # remove punctuation
                for char in punctuation:
                    l = l.replace(char, '')
                # Remove capital letter
                l = l.lower()
                try:
                    for word in l.split():
                        res[word] += 1
                except KeyError:
                    res[word] = 1
            return res
```

```
In [7]: word_frequency = count_word(filename)
```

List the 20 words most frequent words:

```
In [9]: sorted(word_frequency.items(), key=lambda item: item[1], reverse=True)[:20]
```

```
Out[9]: [('the', 6952),
         ('and', 5508),
         ('of', 3370),
         ('to', 2429),
         ('in', 1667),
         ('a', 1412),
         ('he', 1370),
         ('i', 1306),
         ('that', 1251),
         ('his', 1212),
         ('for', 1145),
         ('him', 1064),
         ('but', 866),
         ('with', 842),
         ('all', 812),
         ('my', 788),
         ('me', 786),
         ('thou', 743),
         ('they', 688),
         ('so', 687)]
```