Python Functions

Petr Svarny, 2020

Functions

- A block of organized, reusable code that is used to perform a single, related action

```
def function name(parameters):
    """function definition"""
    function block

    >>> def print_hello():
    ...     """prints Hello, world!"""
    ...     print('Hello, world!')
    >>> print_hello()
    Hello, world!
```

Function parameters

- Get function parameters - help(function_name)

```
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep: string inserted between values, default a space.
    end: string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

- Use question mark after function name
  - print?

Calling function without parameters

```
>>> def print_hi():
...     print('Hi friend')

>>> print_hi()
Hi friend

>>> print_hi('Miky')
---------------------------------------------------------------
Traceback (most recent call last)
<ipython-input-5-a41c06be9ee5> in <module>()
----> 1 print_hi('Miky')
TypeError: print_hi() takes 0 positional arguments but 1 was given
```

Calling function with parameters

```
>>> def print_hi(name):
...     print('Hi ' + name)

>>> print_hi('Miky')
Hi Miky

>>> print_hi('Miky', 'Anna')
Traceback (most recent call last):
File "<ipython-input-53-11fce89a9c9f>", line 1, in <module>
print_hi('Miky', 'Anna')
TypeError: print_hi() takes 1 positional argument but 2 were given
```

Implicit parameter value

```
>>> def print_hi(name='friend'):
...     print('Hi ' + name)

>>> print_hi('Anna')
Hi Anna

>>> print_hi()
Hi friend
```

Return value

- Function can return value using ***return***

```
>>> def return_sum(x,y):
...     c = x + y
...     return c

>>> res = return_sum(4,5)
>>> print(res)
9
```

```
>>> def print_sum(x,y):
...     c = x + y
...     print(c)

>>> res = print_sum(4,5)
9
>>> print(res)
None
```

Exercise

- Create function to divide first number by the second number (numbers are parameters)

- Create function that has list of numbers as an input and prints sum of all elements of the list

Anonymous (lambda) function

- One line functions
- You might want to use lambdas when you don't want to use a function twice in a program

```
lambda argument: manipulate(argument)
```

```
>>> a = [(1, 2), (4, 1), (9, 10), (13, -3)]
>>> a.sort(key=lambda x: x[1])
>>> print(a)
[(13, -3), (4, 1), (1, 2), (9, 10)]
```

Anonymous (lambda) function

```
>>> min = (lambda x, y: x if x < y else y)
>>> min(20, 45)
20
>>> min(10*5, 100/25)
25
```

Lambda versus standard functions

```
>>>vowel_or_not = (
. . .       lambda letter: print('vowel') if letter.lower() in
'aeiou' else
. . .       print('not_vowel') )
>>> vowel_or_not('A')
vowel

>>> def vowel_or_not2(letter):
. . .       if letter.lower() in 'aeiou':
. . .           print('vowel')
. . .       else:
. . .           print('not_vowel')
>>> vowel_or_not2('A')
vowel
```

Exercise

- Using lambda create function that
  - Takes as list as an input
  - If list length is more than 5, prints 'big list'
  - Else it prints 'small list'

```
>>> compare_list([1,2,3])
small list

>>> compare_list([1,2,3,4,5,6])
big list
```

Return multiple values

- It is possible to return multiple values

```
>>> def size_uk_to_eu_it(uk_size):
... eu_size = uk_size + 28
...     it_size = uk_size + 32
...     return eu_size, it_size

>>> eu_dress, it_dress = size_uk_to_eu_it(12)
>>> print(eu_dress)
40
>>> print(it_dress)
44
```

Return multiple values in one variable

- It is possible to return multiple values

```
>>> def size_uk_to_eu_it(uk_size):
...     eu_size = uk_size + 28
...     it_size = uk_size + 32
...     return eu_size, it_size

>>> dress = size_uk_to_eu_it(12)
>>> print(dress)
(40, 44)
```

Multiple return statements

```
>>> def check_x(x):
...     if x < 0:
...         return False
...     return True

>>> check_x(-5)
False

>>> check_x(1)
True
```

Referenced before assignment

```
>>> def check_x(x):
...     if x < 0:
...         x_abs = abs(x)
...     return x_abs

>>> check_x(-5)
5

>>> check_x(1)

UnboundLocalError: local variable 'x_abs' referenced before
assignment
```

Local versus global variables

- Functions create **local namespace** inside function body, which doesn't modify **global namespace**

```
>>> def count_apples():
...     apple = 10

>>> count_apples()
>>> print(apple)
Traceback (most recent call last):
File "<ipython-input-11-c7ad7df6ef38>", line 1, in <module>
print(apples)
NameError: name 'apple' is not defined
```

Local versus global variables

- Local variables of functions can't be accessed from outside, when the function call has finished

```
>>> apple = 5

>>> def count_apples():
...     apple = 10
...     return apple

>>> count_apples()
>>> print(apple)
5
```

Using function inside a function

```
def count_vowels(word):
    vowel_num = 0
    for item in word:
        if item.lower() in 'aeoiu':
            vowel_num += 1
    return vowel_num

def count_vowels_list(list_of_strings):
    vowel_counts = {}
    for item in list_of_strings:
        vowel_counts[item] = count_vowels(item)
    return vowel_counts
```

Defining and using a function inside a function

```
def count_vowels_list(list_of_strings):
    def count_vowels(word):
        vowel_num = 0
        for item in word:
            if item.lower() in 'aeoiu':
                vowel_num += 1
        return vowel_num

    vowel_counts = {}
    for item in list_of_strings:
    vowel_counts[item] = count_vowels(item)
        return vowel_counts
```

Exercise

- Write function that has one parameter (string) and returns
  - String
  - Number of uppercase letters
  - Number of lowercase letters
  - Hint: you can increase variable value in each iteration by using `"variable += 1"`
- Example:

```
>>> string_upper_lower(s)
String:Peter Piper picked a peck of pickled peppers. A peck of
pickled peppers Peter Piper picked.If Peter Piper picked a peck
of pickled peppers, where’s the peck of pickled peppers Peter
Piper picked?
Uppercase letters: 10
Lowercase letters: 148
```

Exercise

- Create a function with two parameters
  - Lunch cost
  - Meal voucher value
- Function will compute how much to pay in meal vouchers and how much remains to be paid in cash
- Example

```
>>> meal_vouchers(500, 74)
Lunch cost: 500 CZK
Pay in cash: 56 CZK
Pay in meal vouchers: 6 pcs, 74 CZK each
```

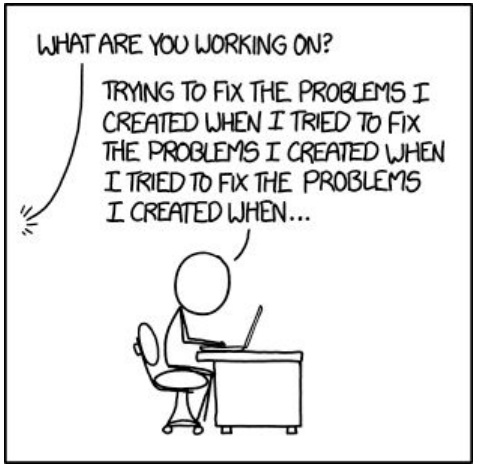
Recursion functions

- Function that calls itself
- “A human is someone whose mother is human”.

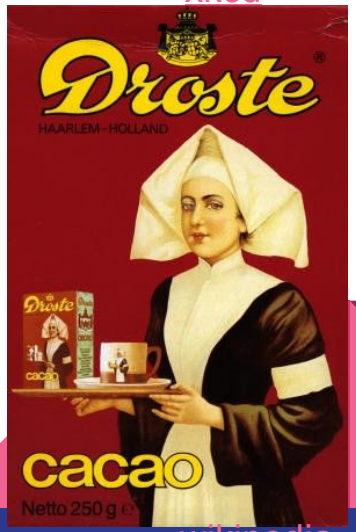
```
>>>  def count_down(n):
…       if n == 0:
…           print('countdown completed')
…       else:
…           print(n)
…           count_down(n-1)
>>> count_down(3)
3
2
1
countdown completed
```

Base call

Recursion step



xkcd



wikipedia

Exercise

- Using recursion, write function that computes factorial for positive integer

```
>>> compute_factorial(1)
1
>>> compute_factorial(3)
6
```