



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Lecture 4: Data structures

Introduction to programming (LT2111)

Markus Forsberg
Språkbanken
University of Gothenburg

2011-09-27



Assignment 2

GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

- ▶ You will work with a readability measure in assignment 2.
- ▶ You will only be presented with the top level function. It is your job to define the functions that you need to solve the problem.
- ▶ Hint: keep the functions short, do not try to do too much in a function.
- ▶ The assignment has two optional tasks. Solve the non-optional part first, then, if you have the time, go for the optional tasks.
- ▶ Deadline: week 40, Friday, 23.59 CET



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Note on programming

- ▶ For the overwhelmed...
- ▶ If you know:
 - ▶ basic types (e.g., numbers, strings, lists) and how to perform calculations on them;
 - ▶ how to input and output data;
 - ▶ conditionals;
 - ▶ loops;
 - ▶ functions;
 - ▶ and modules.
- ▶ Then you know the basics; you have the tools to solve almost any computational problem.
- ▶ However, everything else you learn in this course will help you become a better programmer.



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Repetition: Values and expressions

- ▶ **Values** are the basic things we are working with, such as number, lists, sets, and strings.
- ▶ Every value has a **type**. If we know a value's type, then we know what we can do with the value (e.g., numbers can be added, strings can be concatenated).
- ▶ **Expressions** denotes a value, possibly after some **computation**. A value is (trivially) an expression.
- ▶ We use **variables** to give values names.



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Repetition: Conditionals

- ▶ We use conditionals to control the flow of execution.
- ▶ In Python, we use if-statements with **conditions**.
- ▶ A condition is an expression that computes to a value of type `bool` (`True`, `False`).

```
if 'house' in word:  
    print "word contains the string 'house'"  
elif 'dog' in word:  
    print "word contains the 'dog', but not 'house'"  
else:  
    print "word contains neither 'house' nor 'dog'"
```



Repetition: Loops

GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

- ▶ We use `for` to loop over something, and `while` to loop over a condition.

```
for (m,n) in [(m,m**2) for m in range(1,11)]:  
    print m,n
```

```
print 'Enter a number: ',  
n = raw_input()  
while not(n.isdigit()):  
    print "'%s' is not a number" % n  
    print 'Enter a number: ',  
    n = raw_input()
```



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Repetition: Index and Slicing

- ▶ We use `indexing` to access an element in a sequence (e.g, lists).
- ▶ We use `slicing` to access parts of a sequence.
- ▶ The first position is at number 0.

```
>>> sentence = ['Alice', 'drinks', 'coffee']
```

```
>>> sentence[0][0]  
'A'
```

```
>>> sentence[1][0:5] # position 0-4  
'drink'
```



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Repetition: Functions

- ▶ A **function** consists of a **header** with a name and **parameters**, and a body of code having one or more **return** statements.
- ▶ A **procedure** has no return statement. Python returns a dummy value `None` of type `NoneType` for procedures.
- ▶ If you end up with a `None` value you probably misused a procedure as a function.

```
def ask_for_a_number(name):  
    print 'Hello, %s!' % name  
    print 'Enter a number: ',  
    n = raw_input()  
    while not(n.isdigit()):  
        print "'%s' is not a number" % n  
        print 'Enter a number: ',  
        n = raw_input()  
    return int(n)
```



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Repetition: Modules

- ▶ We use modules to group things together, e.g., related functions.
- ▶ A module is a file `NAME.py`, where the name of the module is `NAME`.
- ▶ Modules provides a new **namespace**, which allow us to reuse names used outside the module.
- ▶ Names in a module is accessed with the **dot notation**: `MODULE.name`.



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Comments and documentation

- ▶ Comments and documentation is written for humans.
- ▶ They are not only for others, but also for the programmer herself.
- ▶ We use documentation string to add information to the module interface (accessed by `help(MODULE)`).

```
def ask_for_a_number(name):  
    """Ask a user 'name' to enter a number"""  
    print 'Hello, %s!' % name  
    print 'Enter a number: ',  
    n = raw_input()  
    while not(n.isdigit()): # user must input a number.  
        print "'%s' is not a number" % n  
        print 'Enter a number: ',  
        n = raw_input()  
    return int(n)
```



Data structure

GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

- ▶ A **data structure** is a container for data, supporting a number of operations that can be performed on it.
- ▶ What data structure to use depends on the data and what we want to do with it.
- ▶ Today: **lists**, **sets**, **tuples**, and **dictionaries**.



Set

GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

- ▶ Set: unordered collection (no indexing) of unique elements (no duplication).

```
>>> s = set()
>>> s.add('sun')
>>> s.add('shining')
>>> s.add('sun')
>>> s
set(['sun', 'shining'])
```

```
>> s[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object does not support indexing
```

```
>>> list(s)
['sun', 'shining']
```



Set operations

GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

```
>>> s = set(['the', 'sun', 'and', 'the', 'moon'])
```

```
>>> for word in s:  
    print word,
```

```
and sun the moon
```

```
>>> s.remove('the')  
>>> s  
set(['and', 'sun', 'moon'])
```

```
>>> s.update(['tree', 'cloud'])  
>>> s  
set(['and', 'sun', 'tree', 'moon', 'cloud'])
```



Tuples

GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

- ▶ We use tuples to group values together.
- ▶ Tuples are immutable, otherwise they work like lists.
- ▶ Why not use lists instead?
 - ▶ A tuple is a more efficient representation.
 - ▶ A tuple works as documentation, and guards against mistakes.

```
>>> t = ('tree', 'branch')
>>> type(t)
<type 'tuple'>
```

```
>>> for (x,y) in [(m,n) for m in range(10)
                  for n in range(10)]:
...     print x,y
0 0
0 1
...
```



List

GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

- ▶ A list is a sequence of values.
- ▶ A list is mutable.
- ▶ We can create a list with `list(ITERATOR)`.

```
>>> s = set([1,2,1,2])
```

```
>>> type(s)
<type 'set'>
```

```
>>> list(s)
[1, 2]
```

```
>>> type(list(s))
<type 'list'>
```



Built-in List operations

GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

```
>>> sentence = ['The', 'sun', 'is', 'shining']
```

```
>>> sentence.append('.')
```

```
>>> sentence
```

```
['The', 'sun', 'is', 'shining', '.']
```

```
>>> sentence.extend(sentence)
```

```
>>> sentence
```

```
['The', 'sun', 'is', 'shining', 'The',  
 'sun', 'is', 'shining']
```

```
>>> sentence.sort()
```

```
>>> sentence
```

```
['The', 'is', 'shining', 'sun']
```

```
>>> del sentence[0]
```

```
>>> sentence
```

```
['sun', 'is', 'shining']
```



Built-in List operations

GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

```
>>> sentence.pop()  
'shining'  
>>> sentence  
['The', 'sun', 'is']
```

```
>>> sentence.remove('sun')  
>>> sentence  
['The', 'is', 'shining']
```

```
>>> [1,2] + [3,4]  
[1,2,3,4]
```

```
>>> [1,2]*3  
[1, 2, 1, 2, 1, 2]
```



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Objects

- ▶ Everything in Python is an **object**.
- ▶ Testing if `name1` is the same object as `name2` is done with `name1 is name2`.
- ▶ If they refer to the same object or not, matter only if the object is mutable.

```
>>> x = sentence # Note! Not a copy (aliasing).
>>> x is sentence
True
>>> x[0] = 'Moon'
>>> sentence
['Moon', 'sun', 'is', 'shining']
```

```
>>> x = list(sentence) # shallow copy
>>> x is sentence
False
>>> x = sentence[:] # same as above
```



List processing

GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

- ▶ **Map:** change the elements of the list.

```
>>> sentence = ['The', 'sun', 'is', 'shining', '.']  
>>> u_sentence = [token.upper() for token in sentence]  
>>> u_sentence  
['THE', 'SUN', 'IS', 'SHINING', '.']
```

- ▶ **Reduce:** reduce a list to a value.

```
>>> ns = [1,2,3,4,5]  
>>> sum(ns)  
15
```

- ▶ **Filter:** filter some of the elements in a list.

```
>>> ns = [1,2,3,4,5]  
>>> [n for n in ns if n % 2 == 0]  
[2, 4]
```

- ▶ **Combination of above:**

```
>>> sum([n**2 for n in ns if n % 2 == 0])  
20
```



Dictionaries

GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

- ▶ A dictionary is a data structure for associating keys with values (items), e.g., the keys could be names, and values phone numbers.

```
>>> d = {'Alice': '555-1212', 'Bob': '555-1213' }
```

```
>>> type(d)  
<type 'dict'>
```

```
>>> d['Alice']  
'555-1212'
```

```
>>> d['Alice'] = '555-1111'
```

```
>>> d['Sue'] = '555-0101'
```

```
>>> d  
{'Bob': '555-1213', 'Alice': '555-1111', 'Sue': '555-0101' }
```



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Dictionary operations

```
>>> for key in d:  
...     print key, d[key]  
Bob 555-1213  
Alice 555-1111  
Sue 555-0101  
  
>>> list(d)  
['Bob', 'Alice', 'Sue']  
  
>>> for (key,value) in d.items():  
...     print key,value  
Bob 555-1213  
Alice 555-1111  
Sue 555-0101
```



Dictionary and word frequency

GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

```
def word_frequency(words):  
    freq = {}  
    for word in words:  
        if word not in freq:  
            freq[word] = 1  
        else:  
            freq[word] = freq[word] + 1  
    return freq  
  
>>> word_frequency(['the', 'sun', 'the', 'moon'])  
{'sun': 1, 'the': 2, 'moon': 1}
```



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Dictionaries

- ▶ A dictionary is also called **hash table**.
- ▶ A key in a dictionary must be **hashable** (there must exist a function `hash` that transforms a key value to an integer).
- ▶ The fact that keys are hashable is what enables the data structure to be efficient.

```
>>> d = dict()
>>> d
{}
```

```
>>> d[[1,2,3]] = 'error'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

```
>>> hash('Alice')
85084331628275277
```



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Exception: try, except, raise

- ▶ **Exceptions** exist to enable treatment of things that 'should not happen', such as indexing outside a list.
- ▶ We use `try` and `except` to catch them. This allows us to do something meaningful with them.

```
>>> int('fd12')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: 'fd12'
```

```
def ask_for_a_number(name):
    print 'Hello, %s!' % name
    while True:
        print 'Enter a number: ',
        n = raw_input()
        try:
            return int(n)
        except ValueError:
            print "'%s' is not a number" % n
```



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Exercise: anagrams

- ▶ Two words are anagrams if you can rearrange the letters from one to spell the other.
- ▶ Given the file 'words.txt', and a word in the list, find all anagrams of that word.



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Exercise: Birthday paradox

- ▶ If there are 23 students in your class, what are the chances that two of you have the same birthday? You can estimate this probability by generating random samples of 23 birthdays and checking for matches.



GÖTEBORGS
UNIVERSITET

Språk
BANKEN

CLT

Solution: Birthday paradox

```
import random

def birthday():
    years = [365,365,365, 366]
    year = random.choice(years)
    return random.randint(1,year)

def birthdays(num_of_students):
    return [birthday() for _ in range(num_of_students)]

def match(list_of_birthdays):
    list_of_birthdays.sort()
    for i in range(len(list_of_birthdays)-1):
        if list_of_birthdays[i] == list_of_birthdays[i+1]:
            return True
    return False

def simulation(num_of_students, num_of_runs):
    num_of_matches = 0
    for _ in range(num_of_runs):
        if match(birthdays(num_of_students)):
            num_of_matches += 1
    return 100*float(num_of_matches)/num_of_runs
```