Introduction to the Python programming language

Laszlo SZATHMARY

University of Debrecen
Faculty of Informatics

Lab #4
- what is evaluated as False; string buffer
- tuple data type
- list comprehension
- control structures
- functions

(last update: 2016-10-26 [yyyy-mm-dd])

2016-2017, 1st semester

What is evaluated as False?

```
False
None
0     # and: 0.0
''    # or: ""
[]
()
{}
```

empty sequences

empty set

```
>>> bool(None)
False
>>> bool([])
False
>>> bool([1,2,3])
True
>>> bool("py")
True
```

Anything else is considered to to be True.

HW: XOR.

http://docs.python.org/2/library/stdtypes.html#truth-value-testing

String buffer

Example: consider the natural numbers from 1 to 15 (included) and write them down one after the other. Let the result be a string: „`123456789101112131415`".

String buffer

Example: consider the natural numbers from 1 to 15 (included) and write them down one after the other. Let the result be a string: „123456789101112131415".

Naïve approach:

```
5 res = ""
6 for i in range(1, 15+1):
7     res += str(i)
8 #
9 print res
```

Using a string buffer:

```
5 parts = []
6 for i in range(1, 15+1):
7     parts.append(str(i))
8 #
9 res = "".join(parts)
10 print res
```

tuple

```
 4  >>> a = (1, 2, 3)
 5  >>> a[0]
 6  1
 7  >>> a[0] = 5
 8  Traceback (most recent call last):
 9    File "<stdin>", line 1, in <module>
10  TypeError: 'tuple' object does not support item assignment
11  >>>
12  >>> m = ('Total Recall', 1990, 7.5)
13  >>> m
14  ('Total Recall', 1990, 7.5)
15  >>> len(m)
16  3
17  >>> m[:2]
18  ('Total Recall', 1990)
19  >>>
20  >>> (x, y) = (1, 2)
21  >>> x
22  1
23  >>> y
24  2
25  >>> single = ('hi',)
26  >>> single
27  ('hi',)
```

immutable
(read-only)

parallel assignment
(parentheses are
often optional)

tuple with one element
(Notice the special syntax!)

HW: tuple02.py

```
 4   def get_movie_info():
 5       # contact database,
 6       # fetch info from a website, etc.
 7       return ('Total Recall', 1990, 7.5)
 8
 9
10   def main():
11       (title, year, score) = get_movie_info()
12       print 'Title:', title
13       print 'Year:', year
14       print 'Score:', score
```

using this method, a function can return **multiple values**

```
 1   >>> a = 3
 2   >>> b = 9
 3   >>> a
 4   3
 5   >>> b
 6   9
 7   >>> a, b = b, a
 8   >>> a
 9   9
10   >>> b
11   3
12   >>>
```

Swap two variables.
No temp variable is needed :)

List comprehension: a compact way for building lists

```
1   >>> nums = [1, 2, 3, 4]
2   >>> squares = [n*n for n in nums]
3   >>> squares
4   [1, 4, 9, 16]
```

```
1   >>> nums = [1, 2, 3, 4]
2   >>> squares = []
3   >>> for n in nums:
4   ...     squares.append(n*n)
5   ...
6   >>> squares
7   [1, 4, 9, 16]
```

Generally:

[*expr* for *var* in *list*]

Optional „if":

```
1   >>> nums = [8, 3, 2, 1, 5, 9, 2]
2   >>> small = [n for n in nums if n <= 2]
3   >>> small
4   [2, 1, 2]
5   >>>
```

it keeps only those elements that satisy the condition

7

Exercise

Solve the following exercises with list comprehensions.

Link: https://arato.inf.unideb.hu/szathmary.laszlo/pmwiki/index.php?n=EnPy.20120818d

Common operation: reverse a sequence

```
1  >>> s = 'Python rulz'
2  >>> s[::-1]
3  'zlur nohtyP'
4  >>>
5  >>> li = [23, 76, 84, 12]
6  >>> li[::-1]
7  [12, 84, 76, 23]
8  >>>
```

from the beginning to the end,
but with a reversed step
(it returns a new sequence)

Raising to a power; int and long types

```
1  >>> 2**3
2  8
3  >>> 2**64
4  18446744073709551616L
5  >>> 2**128
6  340282366920938463463374607431768211456L
7  >>>
8  >>> import sys
9  >>> sys.maxint
10 9223372036854775807
11 >>> a = sys.maxint + 1
12 >>> a
13 9223372036854775808L
14 >>> type(sys.maxint)
15 <type 'int'>
16 >>> type(a)
17 <type 'long'>
```

Python can handle arbitrarily
large integers
(only depends on the memory)

here: 64 bit int (2**63 - 1)

instead of an overflow
its type is changed
automatically from „int" to
„long"

9

if / elif / else

```python
 4  def main(num):
 5      if num < 0:
 6          print 'Negative number.'
 7      elif num == 0:
 8          print 'Zero.'
 9      elif num < 100:
10          print 'Less than 100.'
11      else:
12          print 'More than 100.'
```

Instead of „else if" we write „elif".

How to memorize that? The word „elif" is just as long as the word „else"…

for

```
 4   def main():
 5       li = ['alfa', 'beta', 'gamma']
 6
 7       for e in li:
 8           print e
 9
10       for index, e in enumerate(li):
11           print index, e
```

alfa
beta
gamma

0 alfa
1 beta
2 gamma

It's a common case that we also need the indexes of the elements.
Use the `enumerate()` function.

break / continue
pass

```python
 4  def main():
 5      cnt = 0
 6      while True:
 7          cnt += 1
 8          if cnt == 42:
 9              break
10      #
11      print cnt      # 42
12
13      li = ['ananas', 'banana', 'orange']
14      for e in li:
15          if e == 'banana':
16              continue
17          #
18          print e
19
20
21  def palindrome(s):
22      pass    # TODO...
```

42

ananas
orange

empty statement

docstring

Documentation should be one line:

```
4  def square(num):
5      """Return the square of a given number."""
6      return num ** 2
```

Or multiline. In this case the first line should be a short summary in one line. Then leave a line empty, and then you can detail the goal of the function, how it works, what are its side effects, etc.

```
 9  def square_v2(num):
10      """Return the square of a given number.
11
12      Calculate the square of the input number."""
13      return num * num
```

(see also: annex F)

<u>Document your programs! Get used to it!</u>

optional parameters / default parameter values

```
>>> def greet(name, greetings="Hello"):
...     print "{g}, {n}!".format(g=greetings, n=name)
...
>>> greet("Laszlo")
Hello, Laszlo!
>>>
>>> greet("Laszlo", greetings="Hola")
Hola, Laszlo!
>>>
>>> greet("Laszlo", "Bonjour")
Bonjour, Laszlo!
```

optional parameters (another example)

```
 4    def hello(name, repeat=1, postfix=''):
 5        for i in range(repeat):
 6            print name + postfix
 7
 8
 9    def main():
10        hello('Laci')
11        print '#' * 10
12        hello('Laci', repeat=3)
13        print '#' * 10
14        hello('Laci', postfix='!')
15        print '#' * 10
16        hello('Laci', repeat=3, postfix='!')
17        print '#' * 10
18        print '#' * 10
19        hello('Laci', 3)
20        print '#' * 10
21        hello('Laci', 3, '')
22        print '#' * 10
23        hello('Laci', '')       # error
```

Exercises

1. [20120818a] distance between two points (tuple)
2. [20120818d] list comprehensions (to complete)
3. [20120818e] multiples of 3 or 5 (PE #1)
   (this time with list comprehensions)
4. [20120920e] line break
5. [20130211b] diamond
6. [20120818f] sum square difference (PE #6)
7. [20130305a] XOR (Warning! Requires thinking!)