A nano intro to Python
(just to start working with NLTK…)

Ing. Roberto Tedesco, PhD
roberto.tedesco@polimi.it

arcslab
adaptable, relational and cognitive software environments

NLP – AA 20-21

Brief description of Python progs

- Official web site: `https://www.python.org`
- Object oriented
- Typed objects but un-typed variables
- Type constraints are not checked at compile time
- Whitespace indentation
  - An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block
  - Spaces or tab, but not the two at the same time
  - The “continuation operator”: ‘\’
- Functions and methods

Some Python hints: string and main

- String literals

```
a= “ciao”
a = 'ciao'
a = “ciao\n”
a = “”” ciao this is
an example with new lines “””
```

- No `main()`: code is executed from the beginning
- But, if functions are defined:

```
if __name__ == '__main__':
    my_entry_point()
```

  launches the function `my_entry_point()`
- Otherwise, the file is just a library of functions

Some Python hints: if

- The `If` statement:

```
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

- The ternary conditional operator (as the "C" `x = cond ? a : b`)

```
a = idx - 10 if idx >= 10 else 0
```

Some Python hints: loops

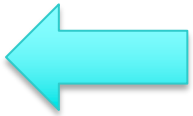
- Loops:

```
c = 0
while (c < 9):
  print('Count:', c)
  c = c + 1
```

```
primes = [2, 3, 5, 7]
for prime in primes:
  print(prime)
```

- NB: do-while does not exist, but…

```
i = 1
while True:
  print(i)
  i = i + 1
  if (i > 3):
    break
```



```
// C-style do-while
int i = 1;
do {
  printf("%d\n", i);
  i = i + 1;
} while(i <= 3);
```

Some Python hints: collections; output

- Tuples, lists, sets, dictionaries (i.e., hashtables)

```
t = ()
t = ('ciao', 1, 2.343)
l = []
l = ['a','f', 'a’]
s = set()
s = {'a','f’,'a'}
```
no duplicates→ `{'a','f'}`

```
d = {}
d = d['pippo'] = 3
```
contains→ `{'pippo': 3}`

- Formatted output

```
print (“Example:%f3.2;%2d” % (59.058,1))
```

generates: `Example: 59.06; 1`

https://www.python-course.eu/python3_formatted_output.php

Some Python hints: comprehensions

- List comprehension

  $S = [f(x) : x \in X \ \wedge \text{condition}(x) \text{ holds}]$

```
s = [v for v in 'ABCDABCD' if v not in 'CB']
print(s)  # generates ['A', 'D', 'A', 'D']
```

- Set comprehension

  $S = \{f(x) : x \in X \wedge \text{condition}(x) \text{ holds}\}$

```
s = {v for v in 'ABCDABCD' if v not in 'CB'}
print(s)  # generates ['A', 'D']
```

- Dictionary comprehension

```
d = {n: n**2 for n in range(5)}
```

  variable `d` contains:

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

Some Python hints: for-in; generators

- For.. in

```
list1 = [5, 6, 8, 3]
for a in list1:
    print(a)
```

- List comprehensions generates **all the element**
  - Memory consumption!
- Generators return an object that will only generate the items **when needed**

```
doubles_list = [2 * n for n in range(100000)]
doubles_gen = (2 * n for n in range(100000))
for a in doubles_list:
    print(a)
for a in doubles_gen:
    print(a)
```

Some Python hints: functions

- Functions with optional return

```
def name(parameter1, parameter2):
    .......
    return xyz
```

- Optional arguments (i.e.: with default values) and named argument

```
def open_file(host,filename="default.txt"):
    .......

open_file("111.111.111.111")
open_file("111.111.111.111","pippo.txt")
open_file(filename="pippo.txt", \
          host="111.111.111.111")
```

Some Python hints: functions

- Some NLTK functions and methods make use of the Python `*args` and `**kwargs` idioms to allow arbitrary number of arguments to functions and methods
- The `*args` will give you all parameters as a tuple:

```
def foo(*args):
    for a in args:
        print(a)

>>> foo(1,2)
1
2
```

Some Python hints: functions

- The `**kwargs` will give you all keyword arguments, as a dictionary:

```
def foo(x, **kwargs):
    for a in kwargs:
        print(a, kwargs[a])
```

formal parameters should be defined before `**kwargs`

```
>>> foo('pippo', name='one', age=27)
age 27
name one
```

Some Python hints: object oriented programming

- Classes

```
class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def length(self):
        a=math.sqrt(self.x**2+self.y**2)
        return a
```

- New objects

```
v = Vector(3,5)
```

- Object methods

```
l = v.length()
```

- Class methods

```
Path.cwd()
```

Integer division

- In Python 2.7, the `/` operator performs integer division if inputs are integers
- If you want float division just use this special import at the beginning of the file:

```
from __future__ import division
```

- For Python 3.x, this is not required, as the `/` operator performs float division in any case

Character encoding

- Python 3.x works with Unicode by default
- Python 2.7 provides some support to Unicode, but it is not the default way of coding characters
- For details, see:
  `https://docs.python.org/3/howto/unicode.html`

NLTK

- NLTK, is a suite of libraries and programs for both symbolic and statistical NLP
- Based on the Python programming language
- It provides:
  - interfaces to over 50 corpora and lexical resources
  - a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning
  - wrappers for industrial-strength NLP libraries
- Available for Windows, OS X, and Linux

NLTK installation

- Current version: NLTK 3
- Install Python (version 2.7 or 3.x)
- Install NLTK:
  see: http://www.nltk.org/install.html
- Install NLTK Data:
  see: http://www.nltk.org/data.html
- The API:
  see: http://www.nltk.org/api/nltk.html
- The on-line book (updated for NLTK3, Python3):
  see: http://www.nltk.org/book/