Escuela Politécnica Superior

UNIVERSIDAD AUTONOMA DE MADRID

Introduction to Programming with Python

II.2. Basic elements

Oscar Delgado
oscar.delgado@uam.es

Loops - for

Starts with 0, not 1

for *variable* in range(*number of times to repeat*):
statements to be repeated

Nothing special about variable name. It's a convention to use the letters i, j, k for loops.

colon

Loops - for

```
for i in range(3):
        num = eval(input('Enter a number: '))
        print ('The square of your number is', num*num)
print('The loop is now done.')
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Loops – **range** function

range(*start*, *end*, *increment*)

Statement	Values generated
`range(10)`	`0,1,2,3,4,5,6,7,8,9`
`range(1,10)`	`1,2,3,4,5,6,7,8,9`
`range(3,7)`	`3,4,5,6`
`range(2,15,3)`	`2,5,8,11,14`
`range(9,2,-1)`	`9,8,7,6,5,4,3`

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

while loop

Suitable when we don't know ahead of time exactly how many times it has to be repeated

```
while <<condition>>:
    # statements to be repeated go here
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

while loop

```
for i in range(10):
    num = eval(input('Enter number: '))
    if num<0:
        break
```

```
i=0
num=1
while i<10 and num>0:
    num = eval(input('Enter a number: '))
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Math operators

Operator	Description
+	addition
–	subtraction
*	multiplication
/	division
**	exponentiation
//	integer division
%	modulo (remainder)

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Random numbers

Module import

```
from random import randint

x = randint(1,10)
print('A random number between 1 and 10: ', x)
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Random numbers

Module import

```
from random import randint

x = randint(1,10)
print('A random number between 1 and 10: ', x)
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Math functions

from math import ...

sin, cos, tan, exp, log, log10, factorial, sqrt, floor, ceil, pi, e

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Conditionals

```
if boolean_expression:
        <<true_branch>>
else:
        <<false branch>>
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Conditionals

```
from random import randint

num = randint(1,10)
guess = eval(input('Enter your guess: '))
if guess==num:
    print('You got it!')
else:
    print('Sorry. The number is ', num)
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Conditional operators

`if x>3:`	if x is greater than 3
`if x>=3:`	if x is greater than or equal to 3
`if x==3:`	if x is 3
`if x!=3:`	if x is not 3
`if x==3 and y ==5:`	if x is 3 and y is 5
`if x==3 or y==5:`	if x is 3 or y is 5

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Common mistakes

The equality operator needs TWO equal signs

Incorrect	Correct
`if x=1:`	`if x==1:`

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Strings

Data type for representing text

```
s = 'Hello'
t = "Hello"
m = """This is a long string that is
spread across two lines."""
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Strings

Empty string The empty string '' is the string equivalent of the number 0, with nothing in it.

Length To get the length of a string (how many characters it has), use the built-in function len. For example, len('Hello') is 5.

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Strings – Concatenation and repetition

Expression	Result
`'AB'+'cd'`	`'ABcd'`
`'A'+'7'+'B'`	`'A7B'`
`'Hi'*4`	`'HiHiHiHi'`

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Strings – The **in** operator

The **in** operator is used to tell if a string contains something.

```
if 'a' in string:
        print('Your string contains the letter a.')

if ';' not in string:
        print('Your string does not contain any semicolons.')
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Strings – Indexing

T ves some examples of indexing the s

Remember that first character is s[0], not s[1]

S ement	Result	Description
s[0]	P	first character of s
s[1]	y	second character of s
s[-1]	n	last character of s
s [illegible]	o	second-to-last character of s

Negative indices count backwards from the end of the string.

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Strings – Slices

A *slice* is used to pick out part of a string

index: 0 1 2 3 4 5 6 7 8 9
letters: a b c d e f g h i j

Code	Result	Description
`s[2:5]`	`cde`	characters at indices 2, 3, 4
`s[ :5]`	`abcde`	first five characters
`s[5: ]`	`fghij`	characters from index 5 to the end
`s[-2: ]`	`ij`	last two characters
`s[ : ]`	`abcdefghij`	entire string
`s[1:7:2]`	`bdf`	characters from index 1 to 6, by twos
`s[ : :-1]`	`jihgfedcba`	a negative step reverses the string

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Strings are immutable

Python strings are **immutable**, which means we can't modify any part of them.

```
s[4]='X'
```

```
s = s[:4] + 'X' + s[5:]
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Strings methods

Method	Description
`lower()`	returns a string with every letter of the original in lowercase
`upper()`	returns a string with every letter of the original in uppercase
`replace(x,y)`	returns a string with every occurrence of `x` replaced by `y`
`count(x)`	counts the number of occurrences of `x` in the string
`index(x)`	returns the location of the first occurrence of `x`
`isalpha()`	returns **`True`** if every character of the string is a letter

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Strings methods - Examples

Statement	Description
`print(s.count(' '))`	prints the number of spaces in the string
`s = s.upper()`	changes the string to all caps
`s = s.replace('Hi','Hello')`	replaces each `'Hi'` in `s` with `'Hello'`
`print(s.index('a'))`	prints location of the first `'a'` in `s`

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Counting

The key to counting is to use a variable to keep the count

```
count = 0
for i in range(10):
    num = eval(input('Enter a number: '))
    if num>10:
        count=count+1
print('There are', count, 'numbers greater than 10.')
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Flag variables

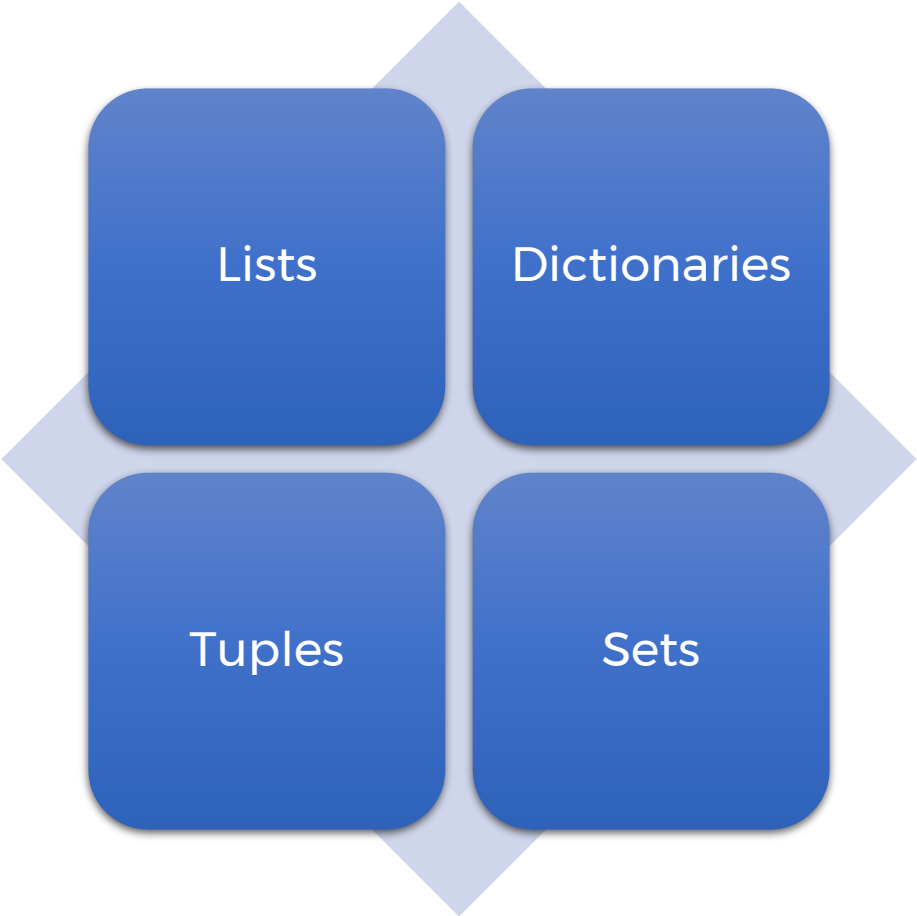
A flag variable can be used to let one part of your program know when something happens in another part of it.

```
num = eval(input('Enter number: '))

flag = 0
for i in range(2,num):
    if num%i==0:
        flag = 1

if flag==1:
    print('Not prime')
else:
    print('Prime')
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Data structures



Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Lists

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Lists

Ordered set of objects

Creating lists Here is a simple list:

```
L = [1,2,3]
```

The empty list The empty list is []. It is the list equivalent of 0 or ''

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Lists

Printing lists You can use the **print** function to print the entire contents of a list.

```
L = [1,2,3]
print(L)
```

Data types Lists can contain all kinds of things, even other lists. For example, the following is a valid list:

```
[1, 2.718, 'abc', [5,6,7]]
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Similarities to strings

len The number of items in L is given by len(L)

in The in operator tells you if a list contains something.

```
if 2 in L:
    print('Your list contains the number 2.')
if 0 not in L:
    print('Your list has no zeroes.')
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Similarities to strings

Indexing and slicing — These work exactly as with strings. For example, L[0] is the first item of the list L and L[:3] gives the first three items.

index and **count** — These methods work the same as they do for strings.

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Similarities to strings

+ and * — The + operator adds one list to the end of another. The * operator repeats a list. Here are some examples:

Expression	Result
`[7,8]+[3,4,5]`	`[7,8,3,4,5]`
`[7,8]*3`	`[7,8,7,8,7,8]`
`[0]*5`	`[0,0,0,0,0]`

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Similarities to strings

Looping — The same types of loops that work for strings also work for lists. The following example prints out the items of a list, one-by-one, on separate lines.

```
for item in L:
    print(item)
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

List methods

Method	Description
`append(x)`	adds `x` to the end of the list
`sort()`	sorts the list
`count(x)`	returns the number of times `x` occurs in the list
`index(x)`	returns the location of the first occurrence of `x`
`reverse()`	reverses the list
`remove(x)`	removes first occurrence of `x` from the list
`pop(p)`	removes the item at index `p` and returns its value
`insert(p,x)`	inserts `x` at index `p` of the list

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

List methods - Examples

Example 1 Write a program that generates a list L of 50 random numbers between 1 and 100

```
from random import randint
L = []
for i in range(50):
    L.append(randint(1,100))
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

split()

The **split** method returns a list of the words of a string. It assumes that words are separated by *whitespace*, which can be either spaces, tabs or newline characters, by default, or any other character specified.

```
s = 'Hi! This is a test.'
print(s.split())
```

```
s = '1-800-271-8281'
print(s.split('-'))
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

join()

The **join** method is in some sense the opposite of split. It is a string method that takes a list of strings and joins them together into a single string.

Operation	Result
`' '.join(L)`	`A B C`
`''.join(L)`	`ABC`
`', '.join(L)`	`A, B, C`
`'***'.join(L)`	`A***B***C`

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Multi-dimensional lists

In Python, one way to create a two-dimensional list is to create a list whose items are themselves lists

```
L = [[1,2,3],
     [4,5,6],
     [7,8,9]]
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Multi-dimensional lists

Indexing We use two indices to access individual items. To get the entry in row r, column c, use the following:

```
L[r][c]
```

Printing One easy option is to use the **pprint** function of the pprint module.

```
from pprint import pprint
pprint(L)
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Reading lists

How to read an undetermined number of integers, letters or characters from keyboard or a file into a list

eval() The string needs to be exactly formatted as a list '[x,x,x]' (odd for a user typing)

```
L = eval(input('Enter a list: '))
print('The first element is ', L[0])
```

```
Enter a list: [5,7,9]
The first element is 5
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Reading lists

split() Any separator can be used (more natural for user)

```
L = list(input("Enter some numbers separated by
commas")).split(",")
print(L)
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Removing elements

del() If we know position of the element to be removed:

```
L=[1,2,3]
del(L[1])
# Now, L=[1,3]
```

remove() If we know the value of the element but not its position:

```
L=[1,2,3]
L.remove(1)
# Now, L=[2,3]
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

From strings to lists and viceversa

String > List Usually with **split()** function:

```
"1:2:3".split(":")
# ['1','2','3']
```

List > String One easy option is to use the **pprint** function of the pprint module.

```
":".join(['1','2','3'])
# "1:2:3"
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Dictionaries

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Dictionaries

Non-ordered set of (key,value) pairs

Creating dictionaries Here is a simple dictionary:

```
d = {'January': 31, 'February': 28}
```

Keys
Must be of immutable type (strings, numbers) and unique

Value
Any type and value are allowed

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Updating dictionaries

Adding/changing elements `d['January'] = 30`

Removing elements `del(d['January'])`

Empty dictionary `d={}`

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Working with dictionaries

in operator Works like usual in strings and lists:

```
letter = input('Enter a letter: ')
if letter in d:
    print('The value is', d[letter])
else:
    print('Not in dictionary')
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

Working with dictionaries

Looping Works like usual in strings and lists:

Print all keys

```
for key in d:
    print(key)
```

Print all values

```
for key in d:
    print(d[key])
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID

List of keys and values

To obtain a list of all keys or values

Dicts are non-ordered!!

```
d = {'A':1, 'B':3}
list(d.keys())
```

```
['A','B']
```

```
d = {'A':1, 'B':3}
list(d.values())
```

```
[1,3]
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTONOMA DE MADRID