



Biotecnologia

Introduction to Programming with Python

Prof. Dr. Wedson Gomes

Prof. Dr. Mozart Marins



Tutorial Outline

- interactive "shell"
- basic types: numbers, strings
- list
- variables
- control structures
- functions & procedures
- files & standard library



Código em Python:

```
print "Hello World!"
```

Código em Java:

```
public class Hello  
{  
    public static void main(String[] args) {  
        System.out.printf("Hello World!");  
    }  
}
```



Interactive “Shell”

- Great for learning the language
- Great for experimenting with the library
- Great for testing your own modules
- Two variations: IDLE (GUI),
python (command line)
- Type statements or expressions at prompt:

```
>>> print "Hello, world"
```

```
Hello, world
```


```
>>> x = 12**2
```

```
>>> x/2
```

```
72
```

```
>>> # this is a comment
```

Interactive “Shell”

The Python logo, a green snake with a yellow underbelly, is positioned to the left of the code input.


```
>>> print “hello world”
```

```
hello world
```

The Python logo, a green snake with a yellow underbelly, is positioned to the left of the code input.


```
>>> 2+2
```

```
4
```

The Python logo, a green snake with a yellow underbelly, is positioned to the left of the code input.


```
>>> 52*345
```

```
17940
```

The Python logo, a green snake with a yellow underbelly, is positioned to the left of the code input.

```
>>> 2.0 / 3.0
```

```
0.6666666666666666
```

The Python logo, a green snake with a yellow underbelly, is positioned to the left of the code input.

```
>>> a=1
```


The Python logo, a green snake with a yellow underbelly, is positioned to the left of the code input.

```
>>> b=2
```

The Python logo, a green snake with a yellow underbelly, is positioned to the left of the code input.

```
>>> a+b
```

```
3
```

The Python logo, a green snake with a yellow underbelly, is positioned to the left of the code input.

```
>>> for a in range(5):
```

```
    print a,
```

```
0 1 2 3 4
```

```
>>>
```

Data Type

- Numéricos:
 - Integer (1,2,3,-1,-2,0,7,234,...)
 - Float (1.23,0.3566,3.53e+63)
 - Complex (2+3j,2j,5+27j)
 - Boolean (True, False)
 - Estruturados:
 - Strings ('ATGCCCAATTG')
 - Listas ([1 ,5 , 0.2, 'A', 'xxx'])
 - Tuplas ((1, 2, 3))
 - Conjuntos (set(['A','T','G','C']))
 - Dicionários ({ 'A' : 'Ala', 'V' : 'Val', 'I' : 'Ile', 'L' : 'Leu' })
- Ordenados (sequenciais)
- Não Ordenados

Strings

- "hello"+"world" "helloworld" # concatenation
- "hello"*3 "hellohellohello" # repetition
- "hello"[0] "h" # indexing
- "hello"[-1] "o" # (from end)
- "hello"[1:4] "ell" # slicing
- len("hello") 5 # size
- "hello" < "jello" 1 # comparison
- "e" in "hello" 1 # search
- "escapes: \n etc, \033 etc, \if etc"
- 'single quotes' ""triple quotes"" r"raw strings"



Strings

```
>>> seq1='ATGGGCA'
>>> seq2='AATTAAAT'
>>> seq1 + seq2
'ATGGGCAAATTAAAT'
>>> poly_alanine='A'*100
>>> len(poly_alanine)
100
>>> poly_alanine
'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
>>> seq1[0]
'A'
>>> seq1[3:6]
'GGC'
```




Strings (operations)

```
>>> seq1
'ATGGGCA'
>>> 'A' in seq1
True
>>> 'X' in seq1
False
>>> len(seq1)
7
>>> min(seq1)
'A'
>>> max(seq1)
'T'
>>> seq1*3
'ATGGGCAATGGGCAATGGGCA'
```



Strings (Methods)

Método	Descrição
<code>str.capitalize()</code>	Primeira letra para maiúscula
<code>str.count(s)</code>	Conta número de ocorrências de s
<code>str.index(s,[start, [end]])</code>	Retorna a posição de s
<code>str.replace(s,r)</code>	Substitui s por r
<code>str.split(sep)</code>	Separa numa lista usando separador sep
<code>str.lower()</code>	Converte para minúsculas
<code>str.strip()</code>	Remove espaços circundantes
<code>str.isalpha()</code>	True se str é apenas alfabético
<code>str.upper()</code>	Converte para maiúsculas



Strings (Methods)

```
>>> myseq = 'ATTGGCCAAACCG'
>>> myseq.count('A')
4
>>> myseq.count('CC')
2
>>> myseq.replace('A','U')
'UTTGGCCUUUCCG'
>>> myseq.capitalize()
'Attggccaaaccg'
>>> myseq.lower()
'attggccaaaccg'
>>> myseq.index('AAA')
7
>>> '193.136.227.168'.split('.')
['193', '136', '227', '168']
```

Lists

- Flexible arrays, *not* Lisp-like linked lists
 - `a = [99, "bottles of beer", ["on", "the", "wall"]]`
- Same operators as for strings
 - `a+b`, `a*3`, `a[0]`, `a[-1]`, `a[1:]`, `len(a)`
- Item and slice assignment
 - `a[0] = 98`
 - `a[1:2] = ["bottles", "of", "beer"]`
→ `[98, "bottles", "of", "beer", ["on", "the", "wall"]]`
 - `del a[-1] #` → `[98, "bottles", "of", "beer"]`



More List Operations

```
>>> a = range(5)      # [0,1,2,3,4]
>>> a.append(5)        # [0,1,2,3,4,5]
>>> a.pop()           # [0,1,2,3,4]
5
>>> a.insert(0, 42)    # [42,0,1,2,3,4]
>>> a.pop(0)          # [0,1,2,3,4]
5.5
>>> a.reverse()        # [4,3,2,1,0]
>>> a.sort()           # [0,1,2,3,4]
```



Variables

- No need to declare
- Need to assign (initialize)
 - use of uninitialized variable raises exception
- Not typed

```
if friendly: greeting = "hello world"
else: greeting = 12**2
print greeting
```
- ***Everything*** is a "variable":
 - Even functions, classes, modules

Variables



```
>>> 1aa = 12
```

```
SyntaxError: invalid syntax
```



```
>>> Ala+ = 10
```

```
SyntaxError: invalid syntax
```



```
>>> new_seq = 'ATTGTC'
```

```
>>> Ala = 10
```

```
>>> Ala+new_seq
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#345>", line 1, in <module>
```

```
    new_seq+Ala
```

```
TypeError: cannot concatenate 'str' and 'int' objects
```

```
>>> str(Ala)+new_seq
```

```
'10ATTGTC'
```




Reference Semantics

- Assignment manipulates references
 - $x = y$ **does not make a copy** of y
 - $x = y$ makes x **reference** the object y references
- Very useful; but beware!
- Example:

```
>>> a = [1, 2, 3]
>>> b = a
>>> a.append(4)
>>> print b
[1, 2, 3, 4]
```




Control Structures

if condition:

statements

[elif condition:

statements] ...

else:

statements

while condition:

statements

for var in sequence:

statements

break

continue



Grouping Indentation

In Python:

```
for i in range(20):  
    if i%3 == 0:  
        print i  
        if i%5 == 0:  
            print "Bingo!"  
    print "---"
```

In C:

```
for (i = 0; i < 20; i++)  
{  
    if (i%3 == 0) {  
        printf("%d\n", i);  
        if (i%5 == 0) {  
            printf("Bingo!\n"); }  
        }  
    printf("---\n");  
}
```

```
0  
Bingo!  
---  
---  
3  
---  
---  
6  
---  
---  
9  
---  
---  
12  
---  
---  
15  
Bingo!  
---  
---  
18  
---  
---
```



Functions, Procedures

```
def name(arg1, arg2, ...):  
    """documentation""" # optional doc string  
    statements
```

```
return          # from procedure
```

```
return expression  # from function
```



Instance Variable Rules

- On use via instance (`self.x`), search order:
 - (1) instance, (2) class, (3) base classes
 - this also works for method lookup
- On assignment via instance (`self.x = ...`):
 - always makes an instance variable
- Class variables "default" for instance variables
- But...!
 - mutable *class* variable: one copy *shared* by all
 - mutable *instance* variable: each instance its own



Modules

- Collection of stuff in *foo.py* file
 - functions, classes, variables
- Importing modules:
 - `import re; print re.match("[a-z]+", s)`
 - `from re import match; print match("[a-z]+", s)`
- Import with rename:
 - `import re as regex`
 - `from re import match as m`



File Objects

- `f = open(filename[, mode[, buffersize]])`
 - mode can be "r", "w", "a" (like C stdio); default "r"
 - append "b" for text translation mode
 - append "+" for read/write open
 - buffersize: 0=unbuffered; 1=line-buffered; buffered
- methods:
 - `read([nbytes])`, `readline()`, `readlines()`
 - `write(string)`, `writelines(list)`
 - `seek(pos[, how])`, `tell()`
 - `flush()`, `close()`
 - `fileno()`



Input / Output

- `raw_input()` - data type **string** / `input()`

```
>>> seq = raw_input("Introduza uma sequência: ")
Introduza uma sequência: ATTGGCCCGAA
>>> print seq
ATTGGCCCGAA
>>> n = raw_input("Introduza um número: ")
Introduza um número: 5
>>> print "o quadrado do número introduzido é ", n*n
>>> n = int(raw_input("Introduza um número: "))
Introduza um número: 5
>>> print "o quadrado do número introduzido é ", n*n
o quadrado do número introduzido é  25
```

Input / Output

```
>>> x=55.23456
>>> print "O valor de x é %5.2f" % x
O valor de x é 55.23
>>> print "O valor de x é %08.3f" % x
O valor de x é 0055.235
```

Forma geral de **print** formatado:

```
print "... %n.mc ... %n.mc ... " % (x,y,z,...)
```

FORMATOS

VARIÁVEIS



Python + GUI Interface

- IDE geany





python training exercise

1 - Write a program that translates a DNA sequence into mRNA

AAATTGCGCG = AAUUUGCGCG

2 - Write a program that calculates the GC content of a DNA sequence

```
gc_count = g_count + c_count
```

```
gc_content = (100.0*gc_count) / length
```