

Universidade de Aveiro

Python vs. Java

Ana Almeida

Pedro Alagoa

Relatório no âmbito de Laboratórios de Informática

Fevereiro, 2014
Aveiro

Relatório - Python vs. Java

Ana Almeida e Pedro Alagoa

Fevereiro 2014, Universidade de Aveiro

Conteúdo

References	1
Lista de Tabelas	3
Lista de Figuras	3
0.1 Resumo	4
1 Introdução	5
2 Preparação	6
2.1 Máquina e sistema operativo utilizados	6
2.2 Instalação das aplicações necessárias	6
3 Procedimento	7
4 Análise de Resultados	8
4.1 Exercício 1	8
4.2 Exercício 2	10
4.3 Exercício 3	11
4.4 Exercício 4	12
4.5 Exercício 5	13
4.6 Exercício 6	14
4.7 Exercício 7	15
4.8 Análise Geral Adaptado de Udemmy[1]	17
4.8.1 Tipos estáticos vs. tipos dinâmicas	17
4.8.2 Chavetas vs. Indentação	17
4.8.3 Portabilidade vs Velocidade	18
5 Conclusão	19
Glossário	19
A Exercício 1	22
A.1 Java	22
A.2 Python	23
B Exercício 2	24
B.1 Java	24
B.2 Python	25

C	Exercício 3	26
C.1	Java	26
C.2	Python	26
D	Exercício 4	28
D.1	Java	28
D.2	Python	29
E	Exercício 5	30
E.1	Java	30
E.2	Python	30
F	Exercício 6	32
F.1	Java	32
F.2	Python	32
G	Exercício 7	34
G.1	Java	34
G.2	Python	35

Lista de Figuras

3.1	Zen of Python	7
4.1	Execução do Exercício 1	8
4.2	Execução do Exercício 2	10
4.3	Execução do Exercício 3	11
4.4	Execução do Exercício 4	12
4.5	Execução do Exercício 5	13
4.6	Execução do Exercício 6	14
4.7	Execução do Exercício 7	15

0.1 Resumo

No seguimento da matéria que tem vindo a ser leccionada nas aulas de Laboratórios de Informática, nomeadamente Python, e com o intuito de explorar as potencialidades desta linguagem de programação, em especial quando comparada a outras linguagens de programação, como o Java, foi proposta a resolução de um guião de Programação constituído por sete exercícios que deveriam ser resolvidos nas duas linguagens e posteriormente comparar as implementações. Os códigos escritos quer em Python quer em Java foram comparados a vários níveis, como funções, sintaxe e indentação, sempre no sentido de avaliar qual a linguagem mais simples para realizar determinada instrução. Os códigos dos exercícios e as conclusões retiradas são apresentados mais detalhadamente no capítulo Análise de Resultados.

Após estabelecidas as comparações concluiu-se que Python é, no geral, uma linguagem mais simples de aprender, principalmente do ponto de vista de principiantes, pois é uma linguagem mais explícita, intuitiva, com um código mais legível e que se assemelha mais à linguagem corrente do dia a dia do que o Java.

Capítulo 1

Introdução

Este relatório tem como objectivo principal dar a conhecer melhor as funcionalidades da linguagem Python. Para isso, foram realizados os exercícios do Guião 1 de Programação 2, utilizando-se Python e também Java, de modo a servir de comparação. O 'Zen of Python' deve servir como base na comparação destas duas implementações.

Trata-se de um tema importante, pois permite saber de uma forma mais detalhada quais são os aspectos mais vantajosos e também os mais inconvenientes quando implementamos Python.

Realizaram-se 7 exercícios com cada linguagem de programação e foram comparados os seus códigos, tendo em conta aspectos como as suas funções, sintaxe e indentação.

Por fim, este relatório conclui com um comentário final aos resultados obtidos após a sua análise.

Capítulo 2





Preparação

2.1 Máquina e sistema operativo utilizados

Os exercícios foram resolvidos em dois computadores portáteis distintos. As máquinas utilizadas foram o Toshiba Satellite P50-A-125 e o Toshiba Satellite A200-2C5.

Os sistemas operativos utilizados foram o Windows 8 64 bit e o Windows 7 64 bit, respectivamente.

2.2 Instalação das aplicações necessárias

- Python[2] v2.7.6 
- JDK[3] 6 Update 35 
- Notepad ++ 
- Geany 

O *download* dos programas foi realizado através dos respectivos sites oficiais.

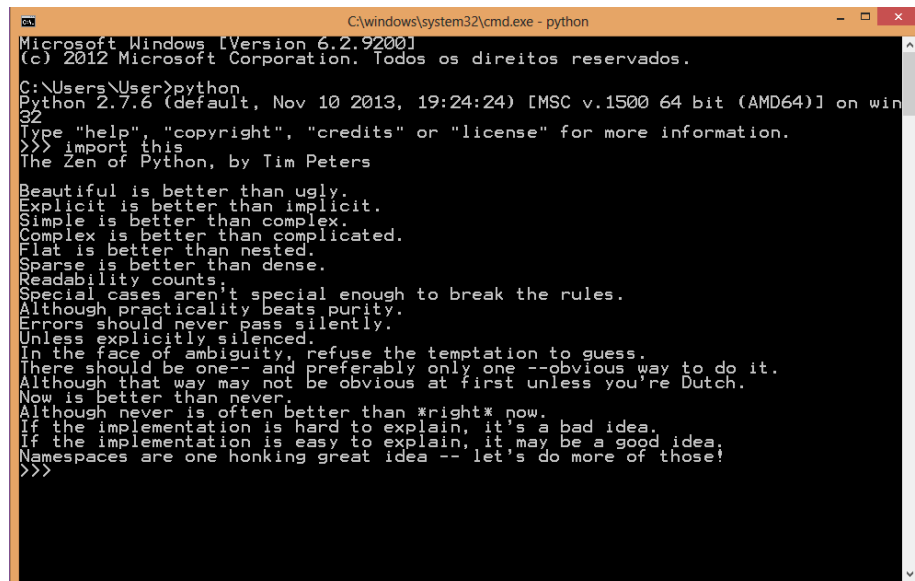
Capítulo 3

Procedimento

Depois de instalados e configurados os programas, procedeu-se à resolução dos exercícios do guião. O editor de texto utilizado para Java foi o Geany e o utilizado para Python foi o Notepad++.

Antes dos exercícios serem resolvidos com a linguagem Python, foram revistas as regras do 'Zen of Python'. Esta lista, que é como um código de conduta para quando estamos a utilizar esta linguagem, pode ser acedida ao importar o módulo 'this', na consola do Python.

Também foram revistos conceitos básicos de Python, tendo como base o Guião 12[4] de LABI.



```
C:\windows\system32\cmd.exe - python
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. Todos os direitos reservados.

C:\Users\User>python
Python 2.7.6 (default, Nov 10 2013, 19:24:24) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Figura 3.1: Zen of Python

Ao longo deste relatório, vamos aprofundar algumas destas regras, principalmente quando compararmos Python a Java.

Capítulo 4

Análise de Resultados

4.1 Exercício 1

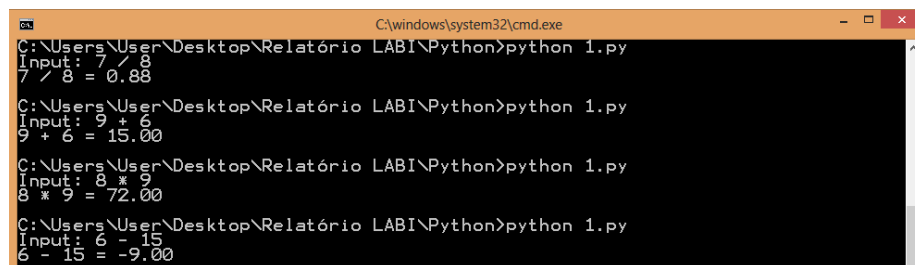
Crie uma calculadora simples que leia (do dispositivo de entrada) operações matemáticas como

$12.3 + 7.2$

e escreva o resultado respectivo (19.5 neste exemplo).

As operações serão sempre do género 'número' 'operador' 'número' com as três partes separadas por espaços ou em linhas diferentes. Implemente as quatro operações básicas.

Note que o operador é uma palavra (string) que contém apenas um símbolo. Se for introduzido um operador inválido, deve escrever uma mensagem apropriada para o dispositivo de saída de erros (System.err).



```
C:\windows\system32\cmd.exe
C:\Users\User\Desktop\Relatório LABI\Python>python 1.py
Input: 7 / 8
7 / 8 = 0.88
C:\Users\User\Desktop\Relatório LABI\Python>python 1.py
Input: 9 + 6
9 + 6 = 15.00
C:\Users\User\Desktop\Relatório LABI\Python>python 1.py
Input: 8 * 9
8 * 9 = 72.00
C:\Users\User\Desktop\Relatório LABI\Python>python 1.py
Input: 6 - 15
6 - 15 = -9.00
```

Figura 4.1: Execução do Exercício 1

O código de Java pode ser consultado na [Seção A.1](#) e o código de Python na [Seção A.2](#).

Neste exercício podemos observar um grande contraste entre Java e Python. Ao contrário do Java, em que é necessário importar classes quando se pretende utilizar scanners, ler variáveis do terminal, em Python, não implica a importação de módulos.

Java - `import java.util.Scanner; Scanner faug = new Scanner(System.in);`

Python - Não é necessário importar módulos nem criar scanners.

A declaração de variáveis é bastante mais simples em Python.

Java - `String conta = new String();conta=faug.nextLine();`

Python - `line = raw_input("Input: ")`

Na maior parte das linguagens de programação existe o tipo 'char'. Em Python, isso não acontece, existindo apenas strings. Isto segue a máxima "Special cases aren't special enough to break the rules", isto é, casos especiais não são especiais o suficiente para quebrar as regras.

Na maioria das linguagens de programação, existe a funcionalidade 'switch', para quando queremos testar vários valores para uma variável. Em Python, esta funcionalidade não existe. Em detrimento do 'switch', o Python oferece Dicionários.

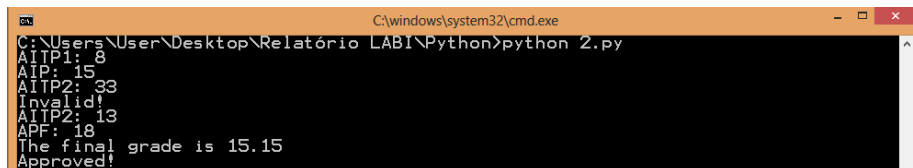
Java - `switch(op) { case '+' ... }`

Python - `d = {'+': ... }`

Aliado ao dicionário está a funcionalidade 'lambda', do Python. O comando lambda é usado para criar novos objetos função e retorná-los em tempo de execução. É uma das funcionalidades mais úteis que o Python pode oferecer.

4.2 Exercício 2

Escreva um programa que determine a nota na época normal de um aluno de Programação 2 (2013 / 2014) e que indique se o aluno está aprovado ou reprovado. Para esse fim o programa deve pedir as 4 notas necessárias (AITP1, AIP, AITP2) e APF), calcular e apresentar a nota final.



```
C:\windows\system32\cmd.exe
C:\Users\User\Desktop\Relatório LABI\Python>python 2.py
AITP1: 8
AIP: 15
AITP2: 33
Invalid!
AITP2: 13
APF: 18
The final grade is 15.15
Approved!
```

Figura 4.2: Execução do Exercício 2

O código de Java pode ser consultado na [Seção B.1](#) e o código de Python na [Seção B.2](#).

Este exercício é baseado em ciclos de repetição e lógica condicional. Podemos observar que a maneira de representar 'ou' e 'e' são diferentes nas duas linguagens.

Java - '||' e '&&'

Python - 'or' e 'and'

A regra "[Explicit is better than implicit](#):" está presente neste aspecto, apesar de também se referir a situações como esta:

Implícito:

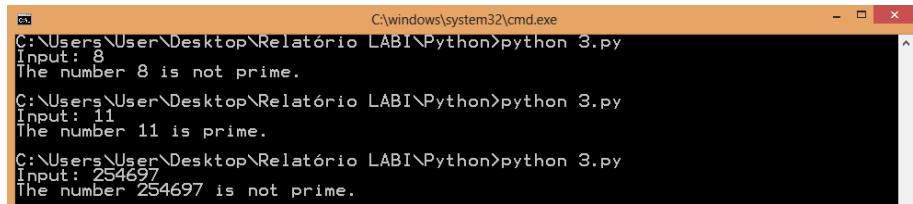
```
from os import *
print getcwd()
}
```

vs. Explícito

```
import os
print os.getcwd()
}
```

4.3 Exercício 3

Escreva um programa que indique se um número (inteiro positivo) é primo.



```
C:\windows\system32\cmd.exe
C:\Users\User\Desktop\Relatório LABI\Python>python 3.py
Input: 8
The number 8 is not prime.
C:\Users\User\Desktop\Relatório LABI\Python>python 3.py
Input: 11
The number 11 is prime.
C:\Users\User\Desktop\Relatório LABI\Python>python 3.py
Input: 254697
The number 254697 is not prime.
```

Figura 4.3: Execução do Exercício 3

O código de Java pode ser consultado na [Seção C.1](#) e o código de Python na [Seção C.2](#).

Neste exemplo é possível verificar que o código em Python é muito mais limpo e legível. Isto deve-se ao facto de não ser necessária a escrita de símbolos como '{' e ';;'.

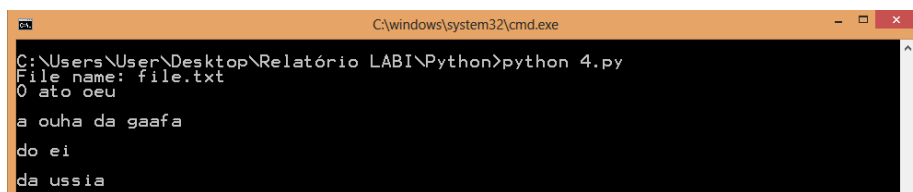
Apesar de estar presente em todos os exercícios, aqui a máxima "[Beautiful is better than ugly](#) ." é bastante evidente, bem como a regra "[Readability counts](#) .:"

4.4 Exercício 4

Na terra do Alberto Alexandre (localmente conhecido por Auexande Aubeto), o dialecto local é semelhante ao português com duas excepções:

- Não dizem os Rs;
- Trocam os Ls por Us.

Implemente um tradutor de português para o dialecto do Alberto. Por exemplo “lar doce lar” deve ser traduzido para “ua doce ua”. A tradução deve ser feita linha a linha, até que surja uma linha vazia.



```
C:\windows\system32\cmd.exe
C:\Users\User\Desktop\Relatório LABIN\Python>python 4.py
File name: file.txt
O ato oeu
a ouha da gaafa
do ei
da ussia
```

Figura 4.4: Execução do Exercício 4

O código de Java pode ser consultado na [Seção D.1](#) e o código de Python na [Seção D.2](#).

Este exercício é o melhor exemplo em termos da extensão do código. Em Java, implementar uma funcionalidade que troque caracteres implica escrever várias linhas de código, incluindo lógica condicional. Em Python, é mais simples.

Java - `if(c == 'r' || c == 'R')`

Python - `string.replace("X","Y")`

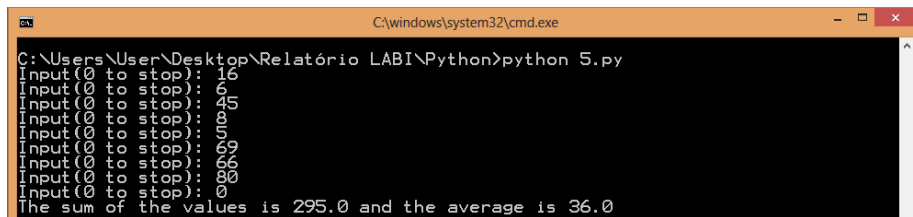
Uma das grandes utilidades do Python é ter inúmeras funções predefinidas prontas a utilizar, como é o exemplo da função `replace`.

Para além disto, para utilizar ficheiros, em Java é necessário utilizar o comando `java.io.*`.

Em Python, não é necessário importar módulos para ler ou escrever em ficheiros.

4.5 Exercício 5

Escreva um programa que leia uma lista de números e imprima a sua soma e a sua média. O fim da lista é indicado pela leitura do número zero, que não deve ser considerado parte da lista. (Note que se a lista for vazia, a soma será zero, mas a média não pode ser calculada.)



```
C:\windows\system32\cmd.exe
C:\Users\User\Desktop\Relatório LABI\Python>python 5.py
Input(0 to stop): 16
Input(0 to stop): 6
Input(0 to stop): 45
Input(0 to stop): 8
Input(0 to stop): 5
Input(0 to stop): 69
Input(0 to stop): 55
Input(0 to stop): 0
Input(0 to stop): 0
The sum of the values is 295.0 and the average is 36.0
```

Figura 4.5: Execução do Exercício 5

O código de Java pode ser consultado na [Seção E.1](#) e o código de Python na [Seção E.2](#).

O facto de, em Python, a indentação influenciar o funcionamento do código, faz com que o este se torne mais legível, uma vez que o utilizador é forçado a utilizar uma indentação fixa.

A regra "[Beautiful is better than ugly](#) ." está novamente presente.

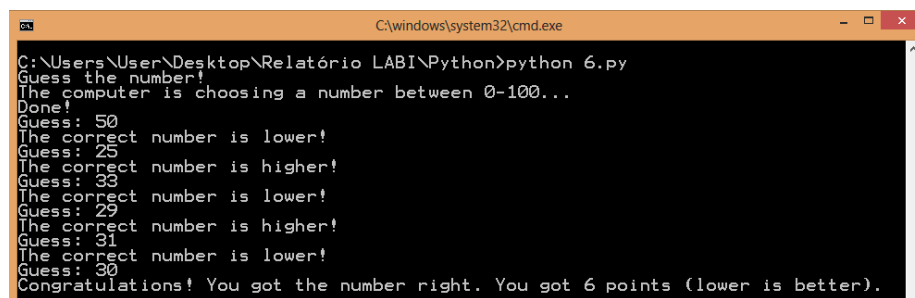
4.6 Exercício 6

Escreva um programa que implemente o jogo “Adivinha o número!”.

Neste jogo, o programa deve escolher um número aleatório no intervalo [0; 100], dando depois a possibilidade de o utilizador ir tentando descobrir o número escolhido.

Para cada tentativa, o programa deve indicar se o número escolhido é maior, menor ou igual à tentativa feita.

O jogo termina quando o número correcto for indicado, sendo a pontuação, do jogador o número de tentativas feito (portanto o valor 1 será pontuação máxima).



```
C:\windows\system32\cmd.exe
C:\Users\User\Desktop\Relatório LABI\Python>python 6.py
Guess the number!
The computer is choosing a number between 0-100...
Done!
Guess: 50
The correct number is lower!
Guess: 25
The correct number is higher!
Guess: 33
The correct number is lower!
Guess: 29
The correct number is higher!
Guess: 31
The correct number is lower!
Guess: 30
Congratulations! You got the number right. You got 6 points (lower is better).
```

Figura 4.6: Execução do Exercício 6

O código de Java pode ser consultado na [Seção F.1](#) e o código de Python na [Seção F.2](#).

Apesar de, em Java, também existir um comando que gera um número aleatório, este é muito mais simples e perceptível em Python.

Java - `n=(int)(Math.random()*(100+1));`

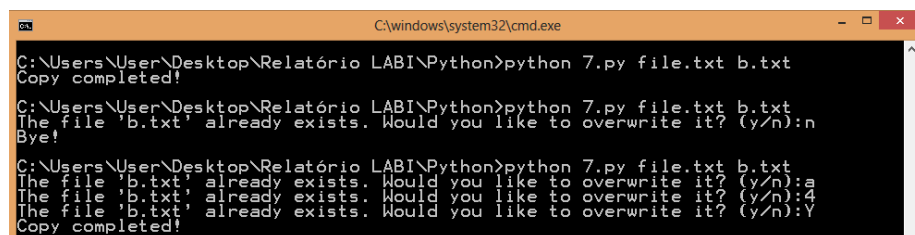
Python - `num = randint(1,100)`

Este aspecto segue a regra "[Simple is better than complex](#) :."

4.7 Exercício 7

Crie um programa que copie um ficheiro de texto para outro. Os nomes dos dois ficheiros envolvidos devem ser dados como argumentos na linha de comandos. Assim a execução do programa com os argumentos Texto1.txt Texto2.txt deve criar um ficheiro Texto2.txt com um conteúdo igual ao do ficheiro Texto1.txt.

Nota: Tente fazer com que o programa tenha alguma robustez, não só detectando a existência do ficheiro original (apresentando uma mensagem de erro quando este não existe), como também a possível existência do ficheiro destino (neste caso fará sentido perguntar ao utilizador se de facto deseja destruir esse ficheiro). Deve também verificar se sobre os ficheiros se podem realizar as operações de leitura e escrita, e se algum deles é um directório (caso em que deve ser apresentada uma mensagem de erro).



```
C:\windows\system32\cmd.exe
C:\Users\User\Desktop\Relatório LABI\Python>python 7.py file.txt b.txt
Copy completed!
C:\Users\User\Desktop\Relatório LABI\Python>python 7.py file.txt b.txt
The file 'b.txt' already exists. Would you like to overwrite it? (y/n):n
Bye!
C:\Users\User\Desktop\Relatório LABI\Python>python 7.py file.txt b.txt
The file 'b.txt' already exists. Would you like to overwrite it? (y/n):a
The file 'b.txt' already exists. Would you like to overwrite it? (y/n):4
The file 'b.txt' already exists. Would you like to overwrite it? (y/n):Y
Copy completed!
```

Figura 4.7: Execução do Exercício 7

O código de Java pode ser consultado na [Seção G.1](#) e o código de Python na [Seção G.2](#).

Neste exercício observam-se grandes diferenças no que toca a manipulação de ficheiros.

Criação de uma variável do tipo File:

Java - `File fin = new File(args[0]);`

Python - `file_in = open(name_in, "r")`

Como se pode constatar, a principal diferença é o facto de, em Python, se poder definir se se quer utilizar esse ficheiro para ler, escrever ou ambos.

Em Java é necessário criar scanners associados a um ficheiro para os poder ler, e printwriters para escrever em ficheiros. Em Python, abrir, ler e escrever em ficheiros pode ser feito através de comandos muito mais simples, sem haver a necessidade de criar scanners ou printwriters.

Ler de um ficheiro:

Java - `String a = sc.nextLine()`

Python - `line q= file_in.readline()`

Escrever num ficheiro:

Java - `PrintWriter pw = new PrintWriter(fout); pw.println(a);`

Python - `file_out.write(line)`

4.8 Análise Geral Adaptado de Udemmy[1]

4.8.1 Tipos estáticos vs. tipos dinâmicas

1. **Java** - As variáveis em Java têm de ter o seu tipo definido desde a sua declaração, não podendo ser mudado ao longo do programa.
2. **Python** - As variáveis têm tipos dinâmicos, isto é, não é necessário definir um tipo de variável e este pode mudar ao longo do programa.

Variáveis com tipos dinâmicos torna a tarefa de programar mais fácil, principalmente para programadores principiantes. Porém, tipos estáticos reduzem o risco de ocorrência de erros, pois, em Python, existe o risco de, ao cometer erros de ortografia, declarar uma variável nova, o que irá influenciar o funcionamento do programa.

4.8.2 Chavetas vs. Indentação

1. **Java** - Utiliza chavetas para definir os blocos de código.
2. **Python** - Utiliza a indentação para definir os blocos de código.

Como foi referido no [Seção 4.5](#), o facto do Python forçar o utilizador a usar indentação no seu código torna-o mais legível e melhor esteticamente. ("Beautiful is better than ugly :."/"Readability counts :.")

4.8.3 Portabilidade vs Velocidade

1. **Java** - A maioria dos sistemas suporta a execução de aplicações em Java. Porém, num ambiente virtual, é mais lento que Python.
2. **Python** - Para executar códigos em Python é necessário o sistema ter um compilador que converta o código em Python para outro código que o sistema consiga interpretar. Apesar disto, a sua velocidade de execução é bastante alta

Mais uma vez, "[Explicit is better than implicit](#) .", como é mostrado na funcionalidade de escrever ou ler através das funções acima. Em Java, com a utilização de um scanner, não é tão evidente que se está a efectuar, por exemplo, uma leitura.

Capítulo 5

Conclusão

Depois de comparar os códigos dos dois exercícios, concluiu-se que ao escolher uma implementação em Python, o código resultante será mais legível e conciso.

Python é mais acessível a programadores principiantes, uma vez que as variáveis possuem tipos dinâmicos e as funções existentes são explícitas e intuitivas.

Também tem a vantagem de ser rápido a executar. A legibilidade e simplicidade do código é a principal característica que distingue esta linguagem das outras.

Glossário

Beautiful is better than ugly : Bonito é melhor que feio.. [11](#), [13](#), [17](#)

Explicit is better than implicit : Explícito é melhor que implícito.. [10](#), [18](#)

Readability counts : A legibilidade conta.. [11](#), [17](#)

Simple is better than complex : Simples é melhor que complexo.. [14](#)

Bibliografia

- [1] Kasia Mikoluk. Python vs. Java. <https://www.udemy.com/blog/python-vs-java/>. [Acedido em 22 de Fevereiro de 2014].
- [2] Python. Python Documentation. <http://www.python.org/doc/>. [Acedido em 22 de Fevereiro de 2014].
- [3] Oracle. Java Documentation. <http://docs.oracle.com/javase/7/docs/api/>. [Acedido em 22 de Fevereiro de 2014].
- [4] João Paulo Barraca. Guião 12 - Python. [Acedido em 20 de Fevereiro de 2014].

Apêndice A

Exercício 1

A.1 Java

```
import java.util.Scanner;
import java.lang.Object;

public class ex1 {

    public static void main (String args[]) {
        Scanner faug = new Scanner(System.in);
        String conta = new String();
        System.out.println();
        System.out.print("Input:");
        conta = faug.nextLine();
        String[] parts = conta.split(" ");
        char op = parts[1].charAt(0);
        switch(op) {
case '+': System.out.printf("%s = %.2f\n", conta,
        Double.parseDouble(parts[0]) +
        Double.parseDouble(parts[2]));
        break;
case '-': System.out.printf("%s = %.2f\n", conta,
        Double.parseDouble(parts[0]) -
        Double.parseDouble(parts[2]));
        break;
case '*': System.out.printf("%s = %.2f\n", conta,
        Double.parseDouble(parts[0]) *
        Double.parseDouble(parts[2]));
        break;
case '/': System.out.printf("%s = %.2f\n", conta,
        Double.parseDouble(parts[0]) /
        Double.parseDouble(parts[2]));
        break;
default: System.out.println("Invalid input!");
        }
    }
}
```


A.2 Python

```
import sys

def default():
    sys.exit("Invalid input!")

line = raw_input("Input: ")
parts = line.split(" ")
part1 = float(parts[0])
op = parts[1];
part3 = float(parts[2])

d = {
    '+': lambda x,y: x+y,
    '-': lambda x,y: x-y,
    '*': lambda x,y: x*y,
    '/': lambda x,y: x/y
}

try:
    result = d[op] (part1, part3)
except KeyError:
    default()

print "%s = %.2f" % (line, result)
```

Apêndice B

Exercício 2

B.1 Java

```
import java.util.Scanner;

public class ex2{

    public static Scanner sc = new Scanner(System.in);
    public static void main (String args []){
        double aitp1, aip, aitp2, apf;
        while(true) {
            System.out.print("AITP1: ");
            aitp1=sc.nextDouble();
            if (aitp1<0 || aitp1>20){
                System.out.println("Invalid!");
                continue;
            }
            break;
        }
        while(true) {
            System.out.print("AIP: ");
            aip=sc.nextDouble();
            if (aip<0 || aip>20){
                System.out.println("Invalid!");
                continue;
            }
            break;
        }
        while(true) {
            System.out.print("AITP2: ");
            aitp2=sc.nextDouble();
            if (aitp2<0 || aitp2>20){
                System.out.println("Invalid!");
                continue;
            }
            break;
        }
        while(true) {
```

```

        System.out.print("APF: ");
        apf=sc.nextDouble();
        if (apf<0 || apf>20){
            System.out.println("Invalid!");
            continue;
        }
        break;
    }

    double media;
    media=(aitp1 + aitp2 + aip + apf)/4;
    System.out.println("The final grade is " + media);
    if (media>=9.5)
        System.out.println("Aproved!");
    else System.out.println("Reproved!");
}
}

```

B.2 Python

```

while True:
    aitp1 = input("AITP1: ")
    if (aitp1 >= 0) and (aitp1 <= 20) :
        break
    print "Invalid!"
while True:
    aip = input("AIP: ")
    if (aip >= 0) and (aip <= 20) :
        break
    print "Invalid!"
while True:
    aitp2 = input("AITP2: ")
    if (aitp2 >= 0) and (aitp2 <= 20) :
        break
    print "Invalid!"
while True:
    apf = input("APF: ")
    if (apf >= 0) and (apf <= 20) :
        break
    print "Invalid!"
final= aitp1*0.15 + aip*0.2 + aitp2*0.15 + apf*0.5
print "The final grade is %.2f" % (final)
if final >= 9.5 :
    print "Approved!"
else:
    print "Reproved!"

```

Apêndice C

Exercício 3

C.1 Java

```
import java.util.Scanner;

public class ex3 {

    public static void main (String args[]) {
        Scanner faug = new Scanner(System.in);
        System.out.print("Input: ");
        int n = faug.nextInt();
        boolean primo = true;
        for(int i=2; i<=n/2; i++) {
            if (n%i == 0) {
                primo = false;
            }
        }
        if(primo == true) {
            System.out.println("The number is prime.");
        }
        else {
            System.out.println("The number is not prime.");
        }
    }
}
```

C.2 Python

```
a = input("Input: ")

i = 2
prime = True
for i in range(2, a/2):
    if a%i == 0:
        prime = False

if prime == False:
```

```
        print "The number %d is not prime." %(a)

elif prime == True:
    print "The number %d is prime." %(a)
```

Apêndice D

Exercício 4

D.1 Java

```
import java.util.Scanner;
import java.io.*;

public class ex4
{
    public static Scanner sc = new Scanner (System.in);
    public static void main (String [] args) throws IOException
    {
        String f = new String();
        System.out.print("File name: ");
        f=sc.nextLine();
        File fin = new File(f);
        Scanner kb = new Scanner(fin);
        while (kb.hasNextLine())
        {
            String t = new String();
            t=kb.nextLine();
            if (t.length()!=0)
            {
                for(int i = 0 ; i < t.length() ; i++){
                    char c = t.charAt(i);
                    if(c == 'r' || c == 'R')
                    {
                        System.out.print("");
                    }
                    else if(c == 'l')
                    {
                        System.out.print("u");
                    }
                    else if(c == 'L')
                    {
                        System.out.print("U");
                    }
                    else
                    {

```

```

                                System.out.print(c);
                                }
                                }
                                }else System.exit(1);
                                System.out.println("");
                                }
                                kb.close();
                                }
}

```

D.2 Python

```

name = raw_input("File name: ")
f = open(name, 'r+')
while True:
    line = f.readline()
    if line == "":
        break
    new_line = line.replace("r", "")
    new_line = new_line.replace("R", "")
    new_line = new_line.replace("l", "u")
    new_line = new_line.replace("L", "u")
    print new_line
f.close()

```

Apêndice E

Exercício 5

E.1 Java

```
import java.util.Scanner;
public class ex5
{
    public static Scanner sc = new Scanner (System.in);
    public static void main (String[] args)
    {
        double n, x=0, c=0;
        do{
            System.out.println("Input(0 to stop): ");
            n=sc.nextDouble();
            if (n!=0)
            {
                x=x+n;
                c++;
            }
            else break;
        }while(n!=0);
        double m;
        if (c!=0)
        {
            m=x/c;
            System.out.printf("The sum of the values is %.1f
                               and the average is %.1f\n", x, c);
        }
        else
            System.out.print("The sum is 0 and is not possible
                               to determine the average.");
    }
}
```

E.2 Python

```
sum = 0
cont = 0
while True:
```



```

n = input("Input(0 to stop): ")
if n == 0:
    break
sum = sum + n
cont += 1
if cont==0:
    print "The sum is 0 and is not possible to determine the average."
else:
    av = sum/cont
    print "The sum of the values is %.1f and the average is %.1f" % (sum, av)

```

Apêndice F

Exercício 6

F.1 Java

```
import java.util.Scanner;

public class ex6
{
    public static Scanner sc = new Scanner (System.in);
    public static void main (String[] args)
    {
        System.out.println("Guess the number!");
        System.out.println("The computer is choosing a number
                            between 0-100...");

        int n;
        n=(int)(Math.random()*(100+1));
        System.out.println("Done!");
        int x, cont=0;
        do
        {
            System.out.print("Guess: ");
            x = sc.nextInt();
            if (x>n)
                System.out.println("The correct number is lower!");
            else if (x<n)
                System.out.println("The correct number
                                    is higher!");

            cont++;
        }while (x!=n);
        System.out.printf("Congratulations! You got the number right.
                            You got %d points (lower is better).: ",
                            cont);
    }
}
```

F.2 Python

```
from random import randint
```

```

num = randint(1,100)
cont = 0
print "Guess the number!"
print "The computer is choosing a number between 0-100..."
print "Done!"
while True:
    tmp = input("Guess: ")
    cont += 1
    if tmp == num:
        break
    if tmp > num:
        print "The correct number is lower!"
    if tmp < num:
        print "The correct number is higher!"
print "Congratulations! You got %d points (lower is better)." % (cont)

```

Apêndice G

Exercício 7

G.1 Java

```
import java.util.Scanner;
import java.io.*;
public class ex7{

    public static Scanner kb = new Scanner(System.in);
    public static void main (String args []) throws IOException{
        File fin = new File(args[0]);
        File fout = new File(args[1]);

        if (!fin.exists())
        {
            System.out.println("ERROR: input file does not exist!");
            System.exit(1);
        }
        if (fin.isDirectory())
        {
            System.out.println("ERROR: input file is a directory!");
            System.exit(2);
        }
        if (!fin.canRead())
        {
            System.out.println("ERROR: cannot read from input file!");
            System.exit(3);
        }

        String resp=new String();
        while(true) {
            if (fout.exists())
            {
                System.out.println("The file "+args[1]+"
                                   already exists. Would you like to
                                   overwrite it? (y/n)");
                resp=kb.nextLine();
            }
        }
    }
}
```

```

        char r=resp.charAt(0);
        if (r=='n' || r=='N')
        {
            System.out.println("Bye!");
            System.exit(1);
        }
        if (r == 'y' || r=='Y') {
            break;
        }
    }

    Scanner sc = new Scanner(fin);
    PrintWriter pw = new PrintWriter(fout);

    if (!fout.canWrite())
    {
        System.out.println("ERROR: cannot write in the output
            file!");
        System.exit(4);
    }

    while (sc.hasNext())
    {
        String a = sc.nextLine();
        pw.println(a);
    }
    pw.close();
    sc.close();
    System.out.println("Copy completed!");
}
}

```

G.2 Python

```

import sys
import os.path

name_in = sys.argv[1]
name_out = sys.argv[2]

#Permissions(file in)
if not os.path.exists(name_in):
    sys.exit("ERROR: file '%s' does not exist!" % name_in)
if os.path.isdir(name_in):
    sys.exit("ERROR: file '%s' is a directory!" % name_in)
if not os.path.isfile(name_in):
    sys.exit("ERROR: '%s' is not a file!" % name_in)
name_in_dir = os.path.dirname(os.path.realpath(name_in))
if not os.access(name_in_dir, os.R_OK):
    sys.exit("ERROR: Cannot read file.")

```

```

#Permissions(file_out)
if os.path.exists(name_out) and os.path.isfile(name_out):
    name_out_dir = os.path.dirname(os.path.realpath(name_out))
    while True:
        sub = raw_input("The file '%s' already exists. Overwrite it? (y/n):")
        if sub == 'n' or sub == 'N':
            print "Bye!"
            sys.exit()
        if sub == 'y' or sub == 'Y':
            break
    name_out_dir = os.path.dirname(os.path.realpath(name_out))
    if not os.access(name_out_dir, os.W_OK):
        sys.exit("It is not possible to write in this directory.")

file_in = open(name_in, "r")
file_out = open(name_out, "w")

#Copy
while True:
    line = file_in.readline()
    if line == '':
        break
    file_out.write(line)

file_in.close()
file_out.close()

print "Copy completed!"

```