

# **Laboratórios de Algoritmia II**

Apresentação e introdução ao Python

# Programa

- Introdução ao Python
- Técnicas genéricas de resolução de problemas
  - Algoritmos de grafos
  - Programação dinâmica
  - Pesquisa exaustiva

# Docentes

- Alcino Cunha
  - [alcino@di.uminho.pt](mailto:alcino@di.uminho.pt)
  - Atendimento 2<sup>a</sup> ou 4<sup>a</sup>, das 9h às 11h, por marcação
- Hugo Pacheco
  - [hpacheco@di.uminho.pt](mailto:hpacheco@di.uminho.pt)
  - Atendimento 4<sup>a</sup>, das 9h às 11h, por marcação

# Avaliação

- Avaliação em 5 torneios de programação
  - 19 Fev, 11 Mar, 1 Abr, 29 Abr, 6 Mai
  - 2 problemas por torneio
  - Nota proporcional aos casos de teste com sucesso
  - Sistema [codeboard.io](https://codeboard.io), linguagem Python 3
  - Realizados nos laboratórios de computadores do CP2
  - Sem acesso à internet, consulta apenas de material escrito
- Não há exame, nem torneios de recuperação!

# Funcionamento

- Para cada um dos 4 tópicos do programa teremos
  - 1 aula teórica (comum aos 2 turnos, DI A2)
  - 1 aula de treino (DI 0.08 e DI 1.09)
  - 1 torneio de avaliação (CP2), em grupos de 2 alunos
- Mais 1 torneio individual que abrange todo o programa

# Python

- Multi-paradigma
  - Imperativo, orientado a objectos, funcional
- Interpretada
- Dinamicamente tipada
- Ênfase na legibilidade do código
  - Sintaxe concisa, blocos controlados por indentação
- Excelente repertório de tipos pré-definidos
  - Tuplos, listas, conjuntos, dicionários

# Python

<https://docs.python.org/3/tutorial/index.html>

# Controlo de fluxo

```
# condicional
```

```
if expr:
```

```
    stmt
```

```
elif expr:
```

```
    stmt
```

```
else:
```

```
    stmt
```

```
# ciclo while
```

```
while expr:
```

```
    stmt
```

```
# ciclo for
```

```
for var in expr:
```

```
    stmt
```



# Números

```
>>> 4+10
```

```
14
```

```
>>> (4+10)-3*5
```

```
-1
```

```
>>> 2 ** 1024
```

```
1797693134862315907729305190789024733617976978942306572734300811577326
```

```
7580550096313270847732240753602112011387987139335765878976881441662249
```

```
2847430639474124377767893424865485276302219601246094119453082952085005
```

```
7688381506823424628814739131105408272371633505106845862982399472459384
```

```
79716304835356329624224137216
```

```
>>> 40/3
```

```
13.333333333333334
```

```
>>> 40//3
```

```
13
```

```
>>> 40 % 3
```

```
1
```

```
>>> int(3.6)
```

```
3
```

```
>>> round(3.6)
```

```
4
```

# Factorial

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n*fact(n-1)
```

```
def fact(n):  
    r = 1  
    while n != 0:  
        r *= n  
        n -= 1  
    return r
```

```
def fact(n):  
    r = 1  
    for n in range(1,n+1):  
        r *= n  
    return r
```

# Strings

```
>>> s = "ola " + 'mundo'
>>> s
'ola mundo'
>>> s[0]
'o'
>>> s[0:3]
'ola'
>>> s[-1]
'o'
>>> len(s)
9
>>> s*2
'ola mundoola mundo'
>>> s[2]='e'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

# Listas

```
>>> l = [1,2,3,'a'] + [True,6]
>>> l
[1, 2, 3, 'a', True, 6]
>>> l[3]
'a'
>>> l[-3:]
['a', True, 6]
>>> len(l)
6
>>> s = l
>>> s[3] = 'b'
>>> l
[1, 2, 3, 'b', True, 6]
>>> l[3:5] = [4,5]
>>> [n+1 for n in l if n>3]
[5, 6, 7]
```

# Listas

```
>>> l = list(range(5))
>>> l.append(5)
>>> l.extend(range(7,10))
>>> l.insert(6,6)
>>> l
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> l.pop()
9
>>> l.pop(2)
2
>>> del l[1]
>>> l
[0, 3, 4, 5, 6, 7, 8]
>>> l.reverse()
>>> l.sort()
>>> l
[0, 3, 4, 5, 6, 7, 8]
```

# Maior palavra

```
def maior(texto):  
    pals = texto.split()  
    m = pals[0]  
    for p in pals:  
        if len(p)>len(m):  
            m = p  
    return m  
  
def maior(s):  
    return max(s.split(),key=len)
```

# Tuplos

```
>>> t = ('Alcino', 10885)
>>> t[0]
'Alcino'
>>> len(t)
2
>>> t[0] = 'Manuel'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> x, y = t
>>> x
'Alcino'
>>> y
10885
```

# Pauta ordenada

```
la2 = [  
    ('Pedro', [5, 4, 3, 2, 5]),  
    ('Maria', [3, 5, 5, 2, 1]),  
    ('Carla', [5, 5, 5, 5, 3]),  
    ('Jose', [2, 3, 4, 3, 4])  
]  
  
# notas crescentes, nome crescente  
def ordena(pauta):  
    final = [(nome, 20*sum(notas)/25) for (nome, notas) in pauta]  
    final.sort(key = lambda t: (t[1], t[0]))  
    return final  
  
# notas decrescentes, nome crescente  
def ordena(pauta):  
    final = [(nome, 20*sum(notas)/25) for (nome, notas) in pauta]  
    final.sort(key = lambda t: t[0])  
    final.sort(key = lambda t: t[1], reverse=True)  
    return final
```



# Conjuntos

```
>>> a = {'Pedro', 'Jose', 'Maria'}
>>> b = set()
>>> b.add('Maria')
>>> b.add('Carla')
>>> len(a)
3
>>> 'Maria' in a
True
>>> a | b
{'Maria', 'Pedro', 'Jose', 'Carla'}
>>> a & b
{'Maria'}
>>> a - b
{'Pedro', 'Jose'}
>>> a ^ b
{'Pedro', 'Carla', 'Jose'}
>>> a <= b
False
>>> {x for x in a if x in b}
{'Maria'}
```

# Palavras diferentes

```
def diferentes(texto):  
    difs = []  
    for p in texto.split():  
        if p not in difs:  
            difs.append(p)  
    difs.sort()  
    return difs  
  
def diferentes(texto):  
    return list(sorted(set(texto.split())))
```

# Dicionários

```
>>> d = {'Pedro':10, 'Maria':15, 'Jose':16}
>>> d['Maria']
15
>>> del d['Jose']
>>> d
{'Pedro': 10, 'Maria': 15}
>>> 'Pedro' in d
True
>>> list(d)
['Pedro', 'Maria']
>>> list(d.items())
[('Pedro', 10), ('Maria', 15)]
```

# Junta notas

```
la2a = [('Pedro', 5), ('Maria', 3), ('Carla', 5),  
        ('Jose', 2), ('Pedro', 4), ('Maria', 5),  
        ('Carla', 5), ('Jose', 3), ('Pedro', 3),  
        ('Maria', 5), ('Carla', 5), ('Jose', 4),  
        ('Pedro', 2), ('Maria', 2), ('Carla', 5),  
        ('Jose', 3), ('Pedro', 5), ('Maria', 1),  
        ('Carla', 3), ('Jose', 4)]
```

```
def junta(notas):  
    pauta = {}  
    for nome,nota in notas:  
        if nome not in pauta:  
            pauta[nome] = []  
        pauta[nome].append(nota)  
    return list(pauta.items())
```

# codeboard.io

- O username deve ser `1a2_a99999`
- Um projecto por treino (aberto em permanência)
- Um projecto por torneio
- Um módulo diferente por cada problema
- O comando `Run` executa o `main( )` em `src/main.py`
- O comando `Test` executa os testes em `test/testes.py`
- O comando `Submit` submete a solução
- As soluções submetidas são pontuadas com base em testes escondidos