Fundamentos da Programação

Cadeias de carateres revisitadas

Aula 9

Alberto Abad, Tagus Park, IST, 2021-22

Cadeias de Carateres Revisitadas

- Em Python, as cadeias de carateres (`str`) são um tipo estruturado **imutável** correspondente a uma **sequência** de carateres individuais.
- São definidas de acordo com a sintaxe BNF:

```
<cadeia de carateres> ::=  '<caráter>*' | "<caráter>*" | """<caráter>*"""
```

- A sequência de carateres com 0 caracteres, ou vazia, é representada por `''` ou `""` .
- Nota-se que `"""` em Python é também utilizada para documentação:

In [7]:
```
# ola, bom dia
# dsad
# dassda
def soma(t):
    """
    Recebe um tuplo que retorna a soma dos seus elementos.
    tuple --> int
    Author: Alberto
    """

    resultado = 0
    #t is a tuple
    for x in t:
        resultado = resultado + x

    return resultado


help(soma)
```

```
Help on function soma in module __main__:

soma(t)
    Recebe um tuplo que retorna a soma dos seus elementos.
    tuple --> int
    Author: Alberto
```

Cadeias de Carateres: Operações e Funções _built-in_

Operação	*Tipo dos argumentos*	*Valor*
s_1 + s_2	Cadeias de carateres	A concatenação das cadeias de carateres s_1 e s_2.
s * i	Cadeia de carateres e inteiro	A repetição i vezes da cadeia de carateres s.
$s[i_1:i_2]$	Cadeia de carateres e inteiros	A sub-cadeia de carateres de s entre os índices i_1 e $i_2 - 1$.
e `in` s	Cadeias de carateres	`True` se e pertence à cadeia de carateres s; `False` em caso contrário.
e `not in` s	Cadeias de carateres	A negação do resultado da operação e `in` s.
`len`(s)	Cadeia de carateres	O número de elementos da cadeia de carateres s.
`str`(a)	Universal	Transforma o seu argumento numa cadeia de carateres.

Cadeias de Carateres: Operações e Funções _built-in_

```
>>> f = 'Fundamentos'
>>> p = 'Programacao'

>>> f + ' da ' + p
'Fundamentos da Programacao'
>>> f*3
'FundamentosFundamentosFundamentos`

>>> 'c' in p
True
>>> 'c' in f
False

>>> len(p)
11
>>> str(9+8)
'17'
>>> str((9,8,20))
'(9, 8, 20)'
>>> eval('f + p')
'FundamentosProgramacao'
```

In [ ]:

Cadeias de Carateres: Indexação e _slicing_

- Tal como os tuplos, as *strings* são sequências e podemos aceder aos seus elementos de forma idêntica:

```
>>> fp='Fundamentos da Programacao'

>>> fp[0]
'F'
>>> fp[15:]
'Programacao'

>>> fp[:11]
'Fundamentos'
>>> fp[-3:]
'cao'
>>> fp[::2]
'Fnaetsd rgaaa'
```

In [ ]:

Cadeias de Carateres: Exemplo 1

Exemplo *simbolos_comum*

- Escrever funçao que recebe duas strings e retorna os símbolos (carateres) comuns
- (Opcional) Alterar para não mostrar repetidos

In [12]:
```
def simbolos_comum(s1, s2):
    res = ''
    for car in s1:
        if car in s2:
            res = res + car

    return res

f1 = 'Fundamentos da programação'
f2 = 'Algebra linear'
simbolos_comum(f1, f2)

#Question1: alterar para não mostrar repetidos
```

Out[12]:
```
'naen a rgraa'
```

Representação Interna de Caracteres

- Os carateres são representados dentro do computador associados a um código numérico.
- O Python utiliza o código **UTF-8** (o código ASCII está contido neste)
- Funções built-in relacionas:
  - `ord` : devolve o código numérico (unicode) de um caráter
  - `chr` : devolve o string correspondente a um código numérico (unicode)

```
>>> ord('A')
65
>>> ord('a')
97
>>> chr(97)
'a'
```

```
In [3]: ord("😂")
```

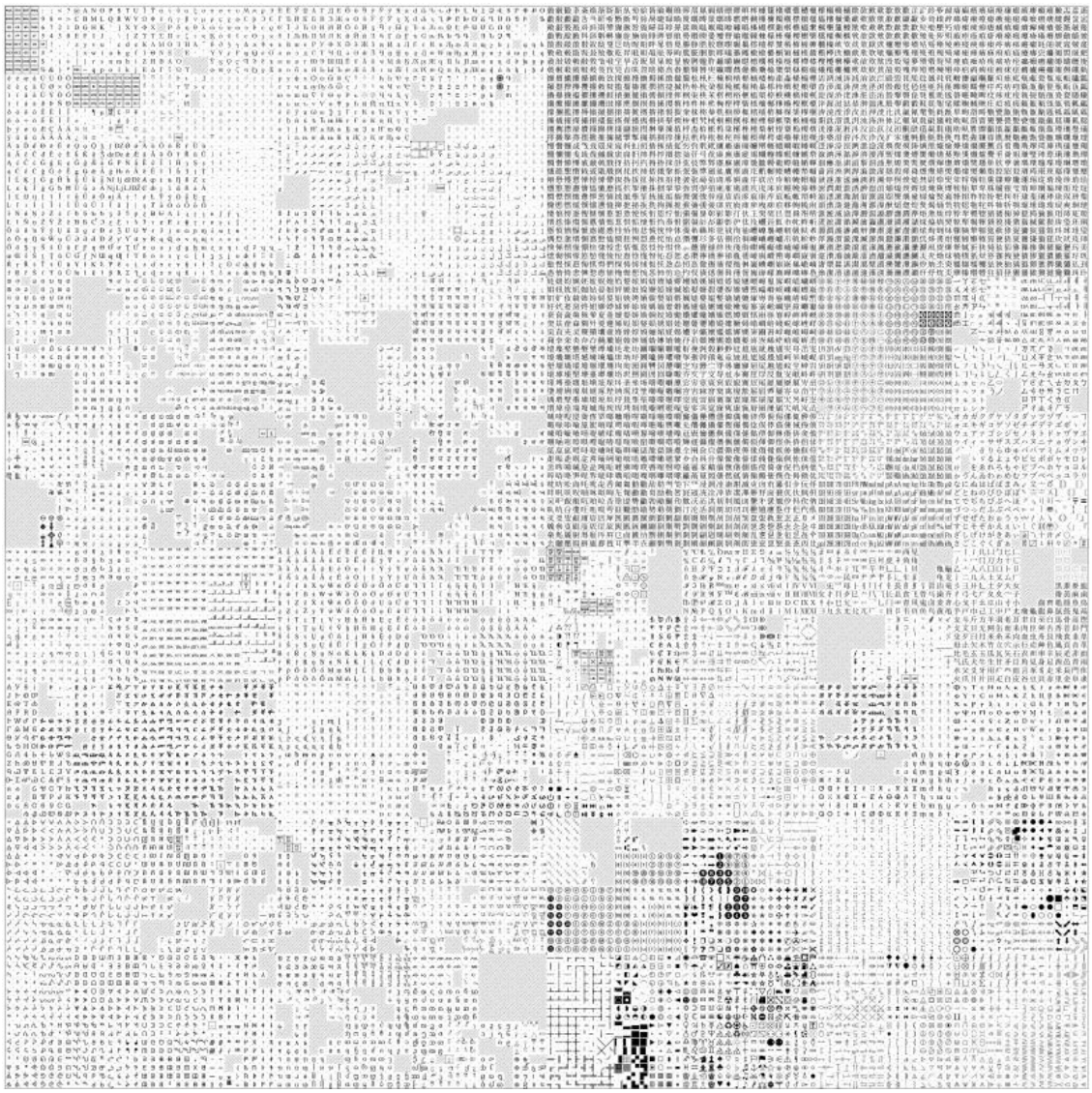
```
Out[3]: 128514
```

Código ASCII

American Standard Code for Information Interchange

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	@	96	60	`
1	1	*[START OF HEADING]*	33	21	!	65	41	**A**	97	61	**a**
2	2	*[START OF TEXT]*	34	22	"	66	42	**B**	98	62	**b**
3	3	*[END OF TEXT]*	35	23	#	67	43	**C**	99	63	**c**
4	4	*[END OF TRANSMISSION]*	36	24	$	68	44	**D**	100	64	**d**
5	5	*[ENQUIRY]*	37	25	%	69	45	**E**	101	65	**e**
6	6	*[ACKNOWLEDGE]*	38	26	&	70	46	**F**	102	66	**f**
7	7	*[BELL]*	39	27	'	71	47	**G**	103	67	**g**
8	8	*[BACKSPACE]*	40	28	(	72	48	**H**	104	68	**h**
9	9	*[HORIZONTAL TAB]*	41	29	)	73	49	**I**	105	69	**i**
10	A	*[LINE FEED]*	42	2A	*	74	4A	**J**	106	6A	**j**
11	B	*[VERTICAL TAB]*	43	2B	+	75	4B	**K**	107	6B	**k**
12	C	*[FORM FEED]*	44	2C	,	76	4C	**L**	108	6C	**l**
13	D	*[CARRIAGE RETURN]*	45	2D	-	77	4D	**M**	109	6D	**m**
14	E	*[SHIFT OUT]*	46	2E	.	78	4E	**N**	110	6E	**n**
15	F	*[SHIFT IN]*	47	2F	/	79	4F	**O**	111	6F	**o**
16	10	*[DATA LINK ESCAPE]*	48	30	**0**	80	50	**P**	112	70	**p**
17	11	*[DEVICE CONTROL 1]*	49	31	**1**	81	51	**Q**	113	71	**q**
18	12	*[DEVICE CONTROL 2]*	50	32	**2**	82	52	**R**	114	72	**r**
19	13	*[DEVICE CONTROL 3]*	51	33	**3**	83	53	**S**	115	73	**s**
20	14	*[DEVICE CONTROL 4]*	52	34	**4**	84	54	**T**	116	74	**t**
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	**5**	85	55	**U**	117	75	**u**
22	16	*[SYNCHRONOUS IDLE]*	54	36	**6**	86	56	**V**	118	76	**v**
23	17	*[ENG OF TRANS. BLOCK]*	55	37	**7**	87	57	**W**	119	77	**w**
24	18	*[CANCEL]*	56	38	**8**	88	58	**X**	120	78	**x**
25	19	*[END OF MEDIUM]*	57	39	**9**	89	59	**Y**	121	79	**y**
26	1A	*[SUBSTITUTE]*	58	3A	:	90	5A	**Z**	122	7A	**z**
27	1B	*[ESCAPE]*	59	3B	;	91	5B	[	123	7B	{
28	1C	*[FILE SEPARATOR]*	60	3C	<	92	5C	\	124	7C	|
29	1D	*[GROUP SEPARATOR]*	61	3D	=	93	5D	]	125	7D	}
30	1E	*[RECORD SEPARATOR]*	62	3E	>	94	5E	^	126	7E	~
31	1F	*[UNIT SEPARATOR]*	63	3F	?	95	5F	_	127	7F	*[DEL]*

UTF-8, _Unicode Transformation Format_



Cadeias de Carateres: Exemplo 2

Exemplo *to_upper*

In [2]:
```
def to_upper(s):
    i = 0
    while i < len(s):
        if 'a' <= s[i] <= 'z':
            s = s[:i] + chr(ord(s[i]) - ord('a') + ord('A')) + s[i+
1:]
        i = i + 1
    return s

def to_upper2(s):
    ns = ''
    for c in s:
        if 'a' <= c <= 'z':
            ns = ns + chr(ord(c) + ord('A') - ord('a'))
        else:
            ns = ns + c
    return ns

print(to_upper('aBceF.'))
print(to_upper2('aBceF.'))
chr(ord('c') - (ord('a') - ord('A')))
```

```
ABCEF.
ABCEF.
```

Out[2]:
```
'C'
```

Cadeias de Caracteres: Mais Operações

- Como as strings correspondem a sequências de códigos numéricos (Unicode), as seguintes operações são possíveis:

```
>>> 'a' < 'z'
True
>>> 'a' < 'Z'
False
>>> 'a' > 'Z'
True

>>> 'Fundamentos' > 'Programacao'
False
>>> 'fundamentos' > 'Programacao'
True
>>> 'fundamentos' > 'fundao'
False
>>> 'fundamentos' < 'fundao'
True
>>>
```

In [ ]:

Cadeias de Caracteres: Formatação

- Como formatar *strings* com `''.format()` (novo estilo)
- Mais informação neste link (https://pyformat.info)

In [16]:
```
print('Inteiros: {} e {:d}'.format(1, 2))
print('Floats: {} e {:f}'.format(1.456, 2.3007))
print('Strings: {} e {:s}'.format('um','dois'))
print('Primeiro é {} e Segundo é {}'.format(1, 2))
print('Segundo é {1} e Primeiro é {0}'.format(1, 2))
print('Primeiro é {first} e Segundo é {second}'.format(first=1, sec
ond='dois'))
print('Primeiro é {first} e Segundo é {second}'.format(second='dois
', first=1))
print('{:<20}'.format('FP'))
print('{:>20}'.format('FP'))
print('{:^20}'.format('FP'))
from math import pi
print('{:.2f}'.format(pi))
print('{:.6f}'.format(pi))
print('{:0>20.6f}'.format(pi))
print('{:0>20.7f}'.format(pi))
```

```
Inteiros: 1 e 2
Floats: 1.456 e 2.300700
Strings: um e dois
Primeiro é 1 e Segundo é 2
Segundo é 2 e Primeiro é 1
Primeiro é 1 e Segundo é dois
Primeiro é 1 e Segundo é dois
FP
                  FP
          FP
3.14
3.141593
0000000000003.141593
000000000003.1415927
```

Cadeias de Caracteres: Exemplo 3

Exemplo de formatação, horário

In [ ]:
```
slot = 0
for h in range (0,24):
    for m in range(0,60,30):
        print("Time slot {:03d} --> {:02d}:{:02d}".format(slot, h,
m))
        slot += 1
```

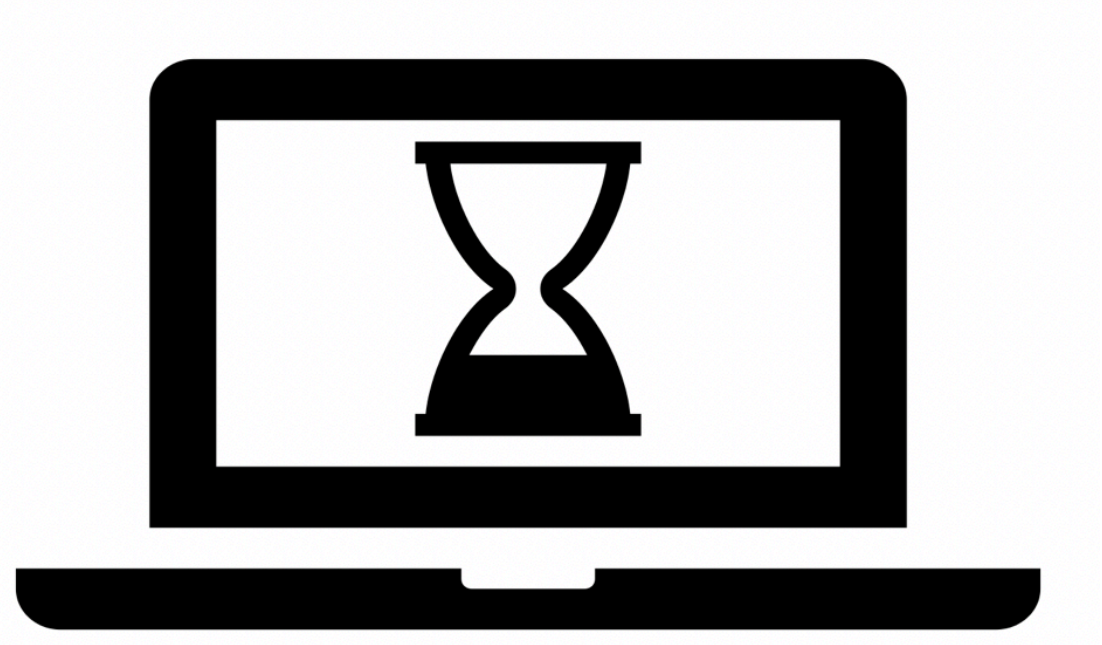
Cadeias de Caracteres: f-strings (opcional)

- Um novo e melhorado método de formatar *strings* no Python (>= 3.6)
- Mais informação neste link (https://realpython.com/python-f-strings/)

```
>>> first_name = "Alberto"
>>> last_name = "Abad"
>>> age = 25

>>> print("Olá, {} {}. Tens {} anos.".format(first_name, last_name,
age))
Olá, Alberto Abad. Tens 25 anos.

>>> print(f"Olá, {first_name} {last_name}. Tens {age} anos.")
Olá, Alberto Abad. Tens 25 anos.
```

In [ ]:
```

```

A treinar!!!!



Cadeias de Caracteres: Exemplo 4

Exemplo verifica ISBN

- O *International Standard Book Number* (ISBN) é um sistema de identificação de livros e softwares que utiliza números únicos para classificá-los por título, autor, país, editora e edição:
  - **ISBN-10** (antes de 2007): O último dígito (x_9) é de controlo e varia de 0 a 10 (o símbolo 'X' é usado em vez de 10) e deve ser tal que:

    $$(1 * x_0 + 2 * x_1 + 3 * x_2 + 4 * x_3 + 5 * x_4 + 6 * x_5 + 7 * x_6 + 8 * x_7 + 9 * x_8 + 10 * x_9)$$

  - **ISBN-13** (desde 2007): O último dígito (x_{12}) é de controlo e varia de 0 a 9 e deve ser tal que:

    $$(x_0 + 3 * x_1 + x_2 + 3 * x_3 + x_4 + 3 * x_5 + x_6 + 3 * x_7 + x_8 + 3 * x_9 + x_{10} + 3 * x_{11} + x$$

Cadeias de Caracteres: Exemplo 4

Exemplo verifica ISBN-10

In [ ]:
```
def verifica_isbn10(isbn):
    def codigo_control(isbn): # ATENÇÃO - Função interna!!
        if isbn[-1] == 'X':
            return 10
        else:
            return int(isbn[-1])
        ## advanced (ternary operator)
        ## return 10 if (isbn[-1] == 'X') else int(isbn[-1])

    soma = 0
    for i in range(len(isbn) - 1):
        soma += (i + 1) * int(isbn[i])
    soma += 10 * codigo_control(isbn)

    return soma % 11 == 0

verifica_isbn10('054792822X') # https://isbnsearch.org/isbn/054792822X
```

Cadeias de Caracteres: Exemplo 4

Exemplo verifica ISBN-13

In [2]:
```
def verifica_isbn13(isbn):
    soma = 0
    for i in range(len(isbn)):
        soma += ((2*i +1)%4)*int(isbn[i])


    return soma%10 == 0

verifica_isbn13('9780547928227') # https://isbnsearch.org/isbn/9789898481474
```

Out[2]: True

Cadeias de Caracteres: Exemplo 4

Exemplo verifica ISBN, valida argumentos

In [ ]:
```
def allValidChars(isbn):
    if len(isbn) == 10:
        for i in range(len(isbn) -1):
            if not '0'<= isbn[i] <= '9':
                return False
        if not ('0' <= isbn[-1] <= '9' or  isbn[-1] == 'X'):
            return False
    elif len(isbn) == 13:
        for i in range(len(isbn)):
            if not '0'<= isbn[i] <= '9':
                return False
    else:
        return False
    return True

#NOTA: Quando aprendamos funcionais, podemos fazer esta função numa linha de codigo
```

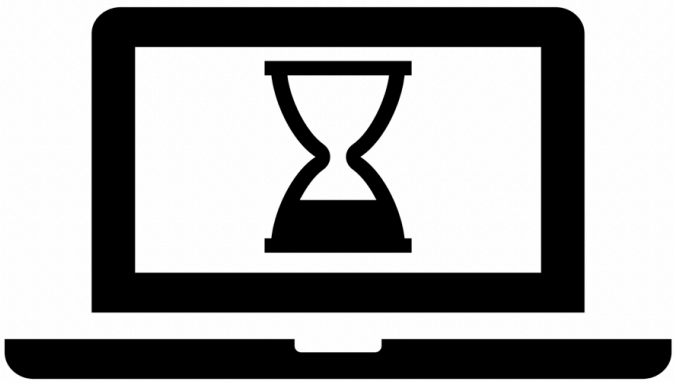
Cadeias de Caracteres: Exemplo 4

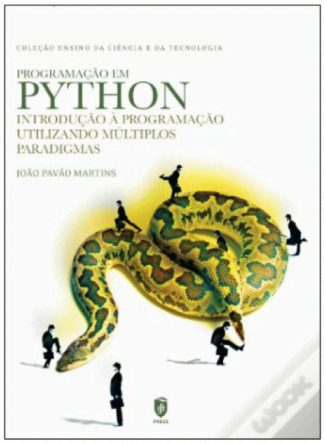
Exemplo verifica ISBN

In [ ]:
```
def verifica_isbn(isbn):
    if not (type(isbn) == str and allValidChars(isbn)):
        raise ValueError("Invalid ISBN string")
    elif len(isbn) == 10:
        return verifica_isbn10(isbn)
    else:
        return verifica_isbn13(isbn)

verifica_isbn('054792822X') # https://isbnsearch.org/isbn/054792822X
verifica_isbn('9789898481474') # https://isbnsearch.org/isbn/9789898481474
```

Tuplos e ciclos contados - Tarefas próxima semana

- Trabalhar matéria apresentada esta semana:
  - Fazer todos os programas dos slides
  - Olhar exercío *Cifra do César* no livro
- Ler capítulo 5 do livro da UC
- Na próxima aula laboratorial L04:
  - Ficha sobre elementos básicos + funções
  - Tópico da aula: tuplos e ciclos contados
- Ler o enunciado do Projeto 1!!





In [ ]:
```
```