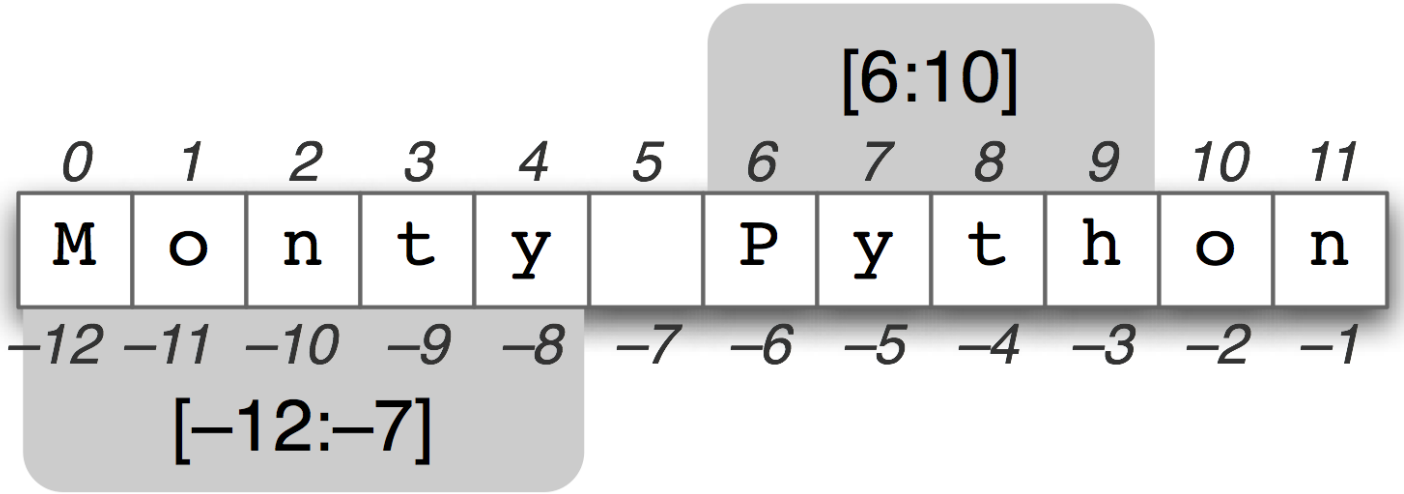


stringhe

Python

- necessaria ***convenzione*** per codifica numerica (***binaria***) dei caratteri
- codifica ***ASCII*** (***A***merican ***S***tandard ***C***ode for ***I***nformation ***I***nterchange)
  - inizialmente 7 bit $\Rightarrow 2^7 = 128$ caratteri
- caratteri ***alfanumerici***: *lettere maiuscole, minuscole, numeri, spazio*
- simboli e ***punteggiatura***: @, #, …
- caratteri di ***controllo*** (***non tutti visualizzabili***): TAB, LF, CR, BELL ecc.

Byte	Cod.	Char
00000000	0	Null
00000001	1	Start of heading
00000010	2	Start of text
00000011	3	End of text
00000100	4	End of transmit
00000101	5	Enquiry
00000110	6	Acknowledge
00000111	7	Audible bell
00001000	8	Backspace
00001001	9	Horizontal tab
00001010	10	Line feed
00001011	11	Vertical tab
00001100	12	Form Feed
00001101	13	Carriage return
00001110	14	Shift out
00001111	15	Shift in
00010000	16	Data link escape
00010001	17	Device control 1
00010010	18	Device control 2
00010011	19	Device control 3
00010100	20	Device control 4
00010101	21	Neg. acknowledge
00010110	22	Synchronous idle
00010111	23	End trans. block
00011000	24	Cancel
00011001	25	End of medium
00011010	26	Substitution
00011011	27	Escape
00011100	28	File separator
00011101	29	Group separator
00011110	30	Record Separator
00011111	31	Unit separator

Byte	Cod.	Char
00100000	32	Spc
00100001	33	!
00100010	34	"
00100011	35	#
00100100	36	$
00100101	37	%
00100110	38	&
00100111	39	'
00101000	40	(
00101001	41	)
00101010	42	*
00101011	43	+
00101100	44	,
00101101	45	-
00101110	46	.
00101111	47	/
00110000	48	0
00110001	49	1
00110010	50	2
00110011	51	3
00110100	52	4
00110101	53	5
00110110	54	6
00110111	55	7
00111000	56	8
00111001	57	9
00111010	58	:
00111011	59	;
00111100	60	<
00111101	61	=
00111110	62	>
00111111	63	?

Byte	Cod.	Char
01000000	64	@
01000001	65	A
01000010	66	B
01000011	67	C
01000100	68	D
01000101	69	E
01000110	70	F
01000111	71	G
01001000	72	H
01001001	73	I
01001010	74	J
01001011	75	K
01001100	76	L
01001101	77	M
01001110	78	N
01001111	79	O
01010000	80	P
01010001	81	Q
01010010	82	R
01010011	83	S
01010100	84	T
01010101	85	U
01010110	86	V
01010111	87	W
01011000	88	X
01011001	89	Y
01011010	90	Z
01011011	91	[
01011100	92	\
01011101	93	]
01011110	94	^
01011111	95	_

Byte	Cod.	Char
01100000	96	`
01100001	97	a
01100010	98	b
01100011	99	c
01100100	100	d
01100101	101	e
01100110	102	f
01100111	103	g
01101000	104	h
01101001	105	i
01101010	106	j
01101011	107	k
01101100	108	l
01101101	109	m
01101110	110	n
01101111	111	o
01110000	112	p
01110001	113	q
01110010	114	r
01110011	115	s
01110100	116	t
01110101	117	u
01110110	118	v
01110111	119	w
01111000	120	x
01111001	121	y
01111010	122	z
01111011	123	{
01111100	124	|
01111101	125	}
01111110	126	~
01111111	127	Del

- ***unicode*** associa un preciso ***code-point*** (***32 bit***) a ciascun simbolo
  - possibile rappresentare miliardi di simboli
  - primi 256 code-point = Latin1

- attualmente più di 30 sistemi di scrittura
  - rappresentazione di geroglifici e caratteri cuneiformi
  - emoticon ed emoji ☺: ideogrammi per espressioni facciali, oggetti comuni, posti, eventi meteo e animali
  - proposta per Klingon (da Star Trek) … *rifiutata* ☹

Python - stringhe

- le ***stringhe*** sono sequenze immutabili di caratteri
- i caratteri sono rappresentati medianti codici ***unicode***
- una costante stringa (tipo str) è rappresentata tra ***singoli*** o ***doppi apici***
  - `'hello'` oppure `"hello"`
- in Python ***non esiste il tipo carattere***
  - un singolo carattere è rappresentato da una stringa di lunghezza 1
- le ***parentesi quadre*** permettono di selezionare un carattere di una stringa
  - `nome = "Itis"`
  - `nome[1]` rappresenta il secondo carattere

operare con le stringhe

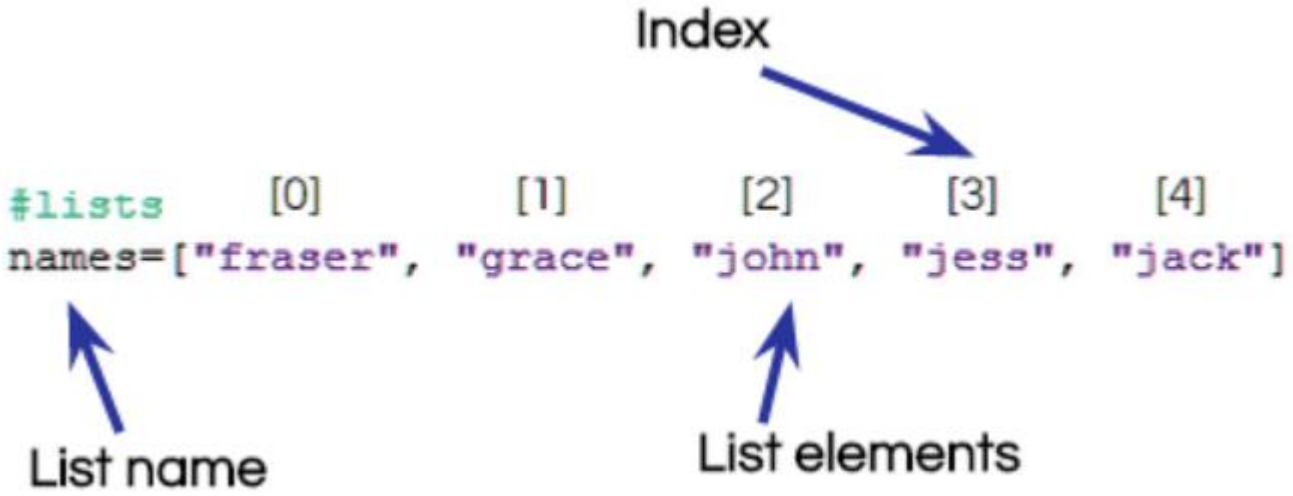
A. Ferrari

```
s = 'hello'
c = s[1]          # selezione di un carattere 'e'
# s[0] = 'H'      # errore le stringhe sono immutabili
l = len(s)        # lunghezza della stringa
for i in range (0,len(s)):# ciclo con accesso mediante indice
    print(s[i],end="|")
print()

for c in s:       # ciclo sugli elementi (caratteri) della stringa
    print(c,end="|")
print()
print(s[1:4])     # selezione di sottostringa (ell)
s = '    Hello  Word!    '
s = s.strip()     # eliminazione di spazi in testa e in coda
print(s)
print(s.upper()) # conversione in maiuscolo
print(s.lower()) # conversione in minuscolo
print(s.replace('o', 'xyz'))
nome = input('inserisci il tuo nome: ')
print(nome+' says: '+s)
```

- il confronto tra stringhe segue l'ordine lessicografico
- operatori di confronto: **<, <=, >, >=, ==, !=**

```
>>> 'first' < 'second'
True
>>> 'Second' <  'first'
True
>>> chr(90)
'Z'
>>> ord('Z')
90
```



liste

Python

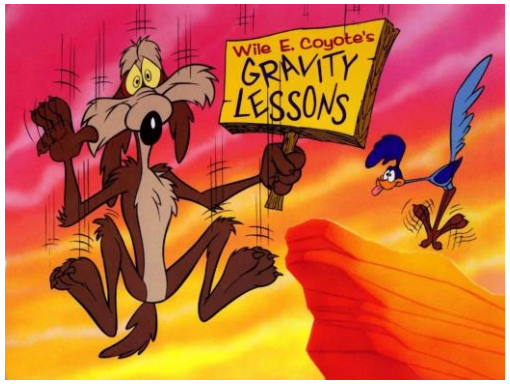
- ***sequenza*** di elementi (*di solito dello stesso tipo*)
- l'intera lista può essere assegnata ad una variabile, così diamo un nome alla lista
- i singoli ***elementi*** sono ***numerati*** per ***posizione***
- gli ***indici*** partono da ***0***

```
spesa = ["pasta", "pane", "vino"]

valori = [13, 24, 18, 15]

mesi = ["Gen", "Feb", "Mar",
        "Apr", "Mag", "Giu",
        "Lug", "Ago", "Set",
        "Ott", "Nov", "Dic"]
```

- attenzione ad usare ***indici validi!***
  - lunghezza attuale di una lista x: **`len(x)`**
  - elementi numerati da **`0`** a **`len(x)-1`**
  - indici negativi contano dalla fine




```
n = len(mesi)                  # 12
mesi[3]                        # "Apr"
mesi[-2]                       # "Nov", analogo a n - 2
spesa = ["pasta", "pane", "vino"]

spesa[1] = "ketchup"           # modifica elemento
```

appartenenza, inserimento, rimozione


```
spesa = ["pasta", "pane", "vino"]

"vino" in spesa              # True, spesa contiene "vino"
spesa.append("olio")         # aggiunge un elemento alla fine
spesa.pop() # toglie e restituisce l'ultimo elemento

spesa.insert(1, "mele")      # gli altri elementi shift
tolto = spesa.pop(1)     # restituisce e toglie el. a index

spesa.remove("pane")         # elimina elemento per valore
```

slice: porzioni di lista


```
primav = mesi[2:5]       # ["Mar", "Apr", "Mag"]
primi = mesi[:3]         # ["Gen", "Feb", "Mar"]
ultimi = mesi[-2:]        # ["Nov", "Dic"]
anno = mesi[:]     # copia in un'altra lista

spesa1 = ["pasta", "pane", "vino"]
spesa2 = ["olio", "sale"]
comprare = spesa1 + spesa2     # concatenazione
copie = [1, 2] * 3     # ripetizione: # [1, 2, 1, 2, 1, 2]
valori = [0] * 12
```

```
a = [3, 4, 5]
b = a[:]        # b = [3, 4, 5] -- a new list!
b == a          # True, they contain the same values
b is a          # False, they are two objects in memory
                # (try and modify one of them...)
c = a
c is a          # True, same object in memory
                # (try and modify one of them...)
```

- ***stringa***: sequenza ***immutabile*** di caratteri
- ***join*** e ***split***: da lista a stringa e viceversa

```
txt = "Monty Python's Flying Circus"
txt[0]    # 'M'
txt[1]    # 'o'
txt[-1]   # 's'
txt[6:12] # "Python"
txt[-6:]  # "Circus"

days = ["tue", "thu", "sat"]
txt = "|".join(days)  # "tue|thu|sat"

days = "mon|wed|fri".split("|")  # ["mon", "wed", "fri"]
```

- il ciclo ***for*** permette di iterare su qualsiasi tipo di sequenza
  - ***lista, stringa, tupla, range…***
- *nell'esempio* ad **ogni iterazione**, alla variabile ***el*** è assegnato un elemento della lista ***corsi***

```
classi = ['3Ainf', '4Ainf', '5Ainf']

for el in classi:
    print(el)
```

```
def limita_valori(lis, limite):
    '''
    fissa un limite massimo ai valori della lista lis
    '''
    for i in range(len(lis)):
        if lis[i] > limite:
            lis[i] = limite

def stampa_valori(lis):
    for i, val in enumerate(lis):
        print('indice', i, 'valore', val)

dati = [5, 4, 2]
limita_valori(dati, 4)
print(dati)
stampa_valori(dati)
```

The enumerate() method adds counter to an iterable and returns it (the enumerate object)

python
introduzione e primi programmi

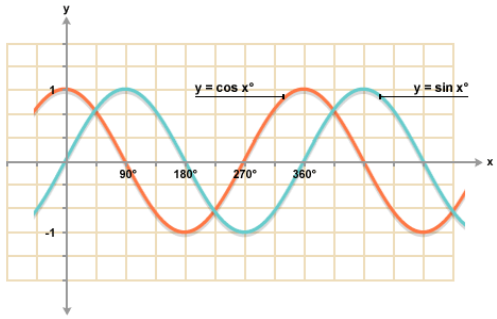
esercizi

esercizi stringhe - 1

- ***es01*** il programma riceve in input una stringa e calcola il numero di occorrenze del carattere 'a' presenti in essa
- ***es02*** il programma riceve in input 4 stringhe e si calcola la percentuale totale delle vocali minuscole sulle 4 stringhe
- ***es03*** il programma riceve in input una stringa e visualizza il numero di caratteri alfabetici presenti in essa
  - il programma deve fare uso delle funzioni che ricevono come parametro un stringa di un carattere:
    - isMinuscola(c: str) -> bool
    - isMaiuscola(c: str) -> bool

- ***es04*** il programma riceve in input una stringa che rappresenta una data nella forma GG/MM/AA e la trasforma e visualizza nel formato AA/MM/GG
  - se l'input non è in forma corretta il programma visualizza un messaggio di errore
  - per verificare se la stringa è in forma corretta è obbligatorio definire e utilizzare la funzione
  - isData(d: str) -> bool che restituisce True se d è una stringa di esattamente 8 caratteri nel formato AA/MM/GG, False altrimenti

- ***es05*** valori precalcolati
  - riempire una lista con i valori di sin(x)
    - 360 elementi, indice x tra 0 e 359
  - poi, ciclicamente...
  - chiedere un angolo all'utente
  - visualizzare il corrispondente valore precalcolato del seno
  - *nota*
    - math.sin opera su radianti
    - calcolare math.sin(x * math.pi / 180), anzichè math.sin(x)



- ***es06*** fattori primi
  - funzione che trova tutti i fattori primi di un numero n
    - parametro: n
    - risultato: lista, contenente i fattori primi di n
  - algoritmo: scorrere tutti i valori d'interesse, e cercare i divisori
    - x è divisore di n sse n % x == 0
    - non considerare i fattori non primi
  - provare la funzione con valori inseriti dall'utente

quando si trova un divisore x, dividere ripetutamente n per x, finché resta divisibile

valutare l'uso di un ciclo while, anzichè for