

# Șiruri de caractere

# Șiruri de caractere

## Clasa `str`

- Constante (literali) delimitate de:
  - `' ... '`
  - `" ... "`
  - `' ' ' ... ' ' '` sau `""" ... """` – se pot întinde pe mai multe linii

# Șiruri de caractere

```
s = 'acesta este un sir'
```

```
t = "acesta este un alt sir"
```

```
versuri = """A fost odata ca in povesti  
A fost ca niciodata"""
```

```
print(s)
```

```
print(t)
```

```
print(versuri)
```



# Șiruri de caractere

## Clasa `str`

- ▶ Pot fi create și folosind `str()` = constructor

```
x = str(3.1415)
```

- ▶ Nu există `char`, `s[0]` este tot de tip `str`

# Șiruri de caractere

- ▶ Imutabil – nu putem modifica valoarea după creare

```
s[0] = 'A' => TypeError: 'str' object does not support item assignment
```

- ▶ Apelul metodelor care efectuează modificări:

```
s_nou = s.metoda()
```

# Șiruri de caractere

```
s = "acest sir"
id_initial_s = id(s)
s.replace("acest", "Alt")
print(s) #nemodificat
s = s.replace("acest", "Alt")
print(s) #modificat
print( id(s), id_initial_s) #de fapt s-a
creat obiect nou, cu alt id
```

# Șiruri de caractere

- Secvențe escape – pentru a insera caractere speciale:

`\n` newline

`\t` tab

`\\` [https://docs.python.org/3/reference/lexical\\_analysis.html#string-and-bytes-literals](https://docs.python.org/3/reference/lexical_analysis.html#string-and-bytes-literals)

`\'`

`\"`

```
s = 'Programarea\talgoritmilor'
print(s)
```

# Șiruri de caractere

```
s = "That's it"
```

```
print(s)
```





# Şiruri de caractere

```
s = "That's it"
```

```
print(s)
```

**s = 'That's it' ?!?!?!?!?**

# Șiruri de caractere

```
s = "That's it"
```

```
print(s)
```

```
s = 'That\'s it'
```

```
print(s)
```



# Șiruri de caractere

```
s = "That's it"
```

```
print(s)
```

```
s = 'That\'s it'
```

```
print(s)
```

```
s = """I say "That's it" again"""
```

```
print(s)
```



# Șiruri de caractere

## Caractere Unicode

- ▶ Prefixam cu `\u` (cod hexa 16) sau `\U` (cod hexazecimal 32 biti) sau `\N{numele lor}`

```
s = "Aceasta este o pisic\u0103 \N{Cat}"  
print(s)
```

# Șiruri de caractere

## Metode uzuale

### Cele comune pentru subsecvențe:

- Parcurgere, Accesarea elementelor, feliere (slice)
- Operatori
  - de concatenare +, \*n
  - in, not in
  - < , <=, >=, >, ==, is
- Funcții uzuale: len, min, max,
- ord, chr

# Șiruri de caractere

```
s = "un sir"
```

```
print(s, "inversat = ", s[::-1])
```

```
print("caracterul", s[0], "are codul", ord(s[0]))
```

```
print("urmatoarea litera dupa", s[0], "este",  
chr(ord(s[0]) + 1))
```

```
s2 = "alt sir"
```

```
print(s < s2)
```

# Șiruri de caractere – Metode specifice

## Căutare

Amintim

`s.index(x[,i[,j]])` => prima apariție a lui `x` în `s` (începând cu indicele `i`, până la indicele `j` exclusiv, dacă sunt specificați, ca la feliere)

**ValueError** dacă nu există

`s.count(x[, i[, j]])` = numărul de apariții

`s.rindex(x[,i[,j]])` => ultima apariție

# Șiruri de caractere – Metode specifice

## Căutare

`s.find(x[,i[,j]])` => ca și `s.index` , dar  
returnează -1 dacă nu găsește

`s.rfind`





# Șiruri de caractere – Metode specifice

```
s = 'Programarea'
print(s.find("a"))
print(s.find("z")) #nu da eroare, ca s.index
print(s.rfind('a'))
print(s.index("z"))
```

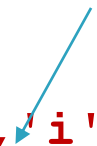
# Șiruri de caractere – Metode specifice

## Căutare

- ▶ `s.startswith(prefix [, [start [, [stop]]])`
- ▶ `s.endswith`

```
if s.endswith('nt'):  
    print('Fazan!')  
  
if s.startswith(('a', 'e', 'i', 'o', 'u')):  
    print('incepe cu o vocala')  
  
if s.startswith(tuple("aeiouAEIOU")):  
    print('incepe cu o vocala')
```

tuplu



# Șiruri de caractere – Metode specifice

## Căutare+înlocuire

`s.replace(old, new[, count])` => returnează o copie a lui s în care toate aparițiile subsecvenței `old` sunt înlocuite cu `new`;

- dacă este dat și parametrul opțional `count`, atunci sunt înlocuite doar primele `count` apariții ale lui `old` ( !!! nu se modifică `s`, este imutabil)

# Șiruri de caractere – Metode specifice

```
t = s = 'Programarea algoritmilor'  
s.replace('a', 'aaa')  
print(s)
```

# Șiruri de caractere – Metode specifice

```
t = s = 'Programarea algoritmilor'
s.replace('a', 'aaa')
print(s)
s = s.replace('a', 'aaa')
print(s)
print(t)
```

# Șiruri de caractere – Metode specifice

```
t = s = 'Programarea algoritmilor'
s.replace('a', 'aaa')
print(s)

s = s.replace('a', 'aaa')
print(s)

print(t)

s = s.replace('o', '', 1)
print(s)
```

# Șiruri de caractere – Metode specifice

## Căutare+înlocuire

`s.translate` – pentru a face mai multe înlocuiri  
simultan – v. seminar și laborator

# Șiruri de caractere – Metode specifice

## Transformare la nivel de caracter

- `s.lower()` – **returnează** șirul scris cu minuscule (!nu îl modifică)
- `s.upper()`
- `s.capitalize()`
- `s.title()`
- `etc`



# Șiruri de caractere – Metode specifice

```
s = ' Programarea Algoritmilor'
s.lower()
print(s) #nu s-a modificat
s = s.lower()
print(s)
s = s.capitalize ()
print(s) #' Programarea algoritmilor'
```

# Șiruri de caractere – Metode specifice

## Transformare la nivel de caracter

- `s.strip([chars])` elimină caracterele din *chars* de la începutul și sfârșitul șirului;  
implicit `chars=None` elimina caracterele albe
- `s.rstrip()` / `s.lstrip()`

# Șiruri de caractere – Metode specifice

```
s = '    Programarea algoritmilor!!**    '  
s = s.strip()  
print(s)  
print(len(s))  
print(s.rstrip("!*"))
```

# Șiruri de caractere – Metode specifice

## Transformare la nivel de caracter

- `s.center()` / `s.ljust(width, fillchar=' ')` / `rjust`

# Șiruri de caractere – Metode specifice

```
print('Programarea algoritmilor'.center(40))  
print('Programarea algoritmilor'.center(40,"*"))  
print('Programarea algoritmilor'.rjust(30,">"))
```

# Șiruri de caractere – Metode specifice

## Testare/clasificare la nivel de caracter

- `s.islower()` – returnează **True** dacă toate literele din șir sunt minuscule, **False** altfel
- `s.isupper()`
- `s.isdigit()`
- ...

# Șiruri de caractere – Metode specifice

## Parsare, divizare si unificare cu separatori

```
s.split(sep = None, maxsplit =-1)
```

- ▶ Împarte `s` în cuvinte folosind `sep` ca separator (delimiter) și returnează o lista acestor cuvinte.
- ▶ Dacă parametrul opțional `maxsplit` este specificat, sunt făcute cel mult `maxsplit` împărțiri (se obține o listă de maxim `maxsplit+1` cuvinte).

# Șiruri de caractere – Metode specifice

## Parsare, divizare si unificare cu separatori

```
s.split(sep = None, maxsplit =-1)
```

- ▶ Dacă nu este specificat sep – caracterele albe consecutive sunt considerate un separator
- ▶ între delimitatori consecutivi => cuvinte vide
- ▶ NU se pot specifica mai mulți delimitatori (seminare.split())



# Șiruri de caractere – Metode specifice

```
s = "acesta     este un sir"
```

```
print(s.split())
```

```
['acesta', 'este', 'un', 'sir']
```

# Șiruri de caractere – Metode specifice

```
s = "acesta  este un sir"
```

```
print(s.split())
```

```
['acesta', 'este', 'un', 'sir']
```

```
print(s.split(" "))
```

```
['acesta', '', '', 'este', 'un', 'sir']
```

# Șiruri de caractere – Metode specifice

```
s = "acesta     este un sir"
```

```
print(s.split())
```

```
#['acesta', 'este', 'un', 'sir']
```

```
print(s.split(" "))
```

```
['acesta', '', '', 'este', 'un', 'sir']
```

```
print(s.split(maxsplit=1))
```

```
#['acesta', 'este un sir']
```



# Șiruri de caractere – Metode specifice

```
s = input("numere pe o linie\n")  
ls = s.split()  
print(ls)  
s = 0  
for x in ls:  
    s = s + int(x)  
print(s)
```

# Șiruri de caractere – Metode specifice

## Parsare, divizare si unificare cu separatori

`s.join(iterable)`

- ▶ Returnează șirul obținut prin concatenarea șirurilor din parametrul `iterable`, folosind șirul `s` ca separator între șirurile concatenate

# Șiruri de caractere – Metode specifice

```
ls = ["2", "3", "5"]
```

```
s = "*".join(ls)
```

```
print(s)
```



# Șiruri de caractere – Metode specifice

```
ls = ["2","3","5"]
```

```
s = "*".join(ls)
```

```
print(s)
```

```
ls = ["ab"]*4
```

```
print(ls, id(ls[0]),id(ls[1]))
```

```
s = "".join(ls)
```

```
print(s)
```



# Șiruri de caractere – Metode specifice

## Formatare

`template.format(<positional_arguments>,<keyword_arguments>)`

- ▶ `template` – Șir conține secvențe speciale cuprinse între `{}` care vor fi înlocuite cu parametri ai metodei `format` (numite **campuri de formatare**), de tipul

`{ [<camp>] [!<fct_conversie>] [:<format_spec>] }`



# Șiruri de caractere – Metode specifice

## Formatare

`template.format(<positional_arguments>,<keyword_arguments>)`

- ▶ `template` – Șir conține secvențe speciale cuprinse între `{}` care vor fi înlocuite cu parametri ai metodei `format` (numite **campuri de formatare**), de tipul

`{ [<camp>] [!<fct_conversie>] [:<format_spec>] }`

- ▶ Parametri
  - **poziționali** – se identifica prin poziție
  - **cu nume (numiți)** – se pot identifica și prin nume

# Șiruri de caractere – Metode specifice

## Formatare

`template.format(<positional_arguments>,<keyword_arguments>)`

- ▶ `template` – Șir conține secvențe speciale cuprinse între `{}` care vor fi înlocuite cu parametri ai metodei `format` (numite **campuri de formatare**), de tipul

`{ [<camp>] [!<fct_conversie>] [:<format_spec>] }`

- ▶ Parametri
  - **poziționali** – se identifica prin poziție
  - **cu nume (numiți)** – se pot identifica și prin nume
- ▶ Returnează sirul `template` formatat, înlocuind câmpurile de formatare cu valorile parametrilor (formatate conform cu specificațiilor din câmpuri)

# Șiruri de caractere – Metode specifice

**Variante de a specifica ce parametru pozițional se folosește într-un câmp de formatare:**

- Specificăm numărul parametrului pozițional pe care îl folosim (numerotare de la 0)
- Nu specificăm nimic => parametrii poziționali sunt considerați în ordine (numerotare automata)
- Nu se pot combina cele două abordări

# Șiruri de caractere – Metode specifice

```
s = "Nota la {} = {}".format("PA",10)  
print(s)
```

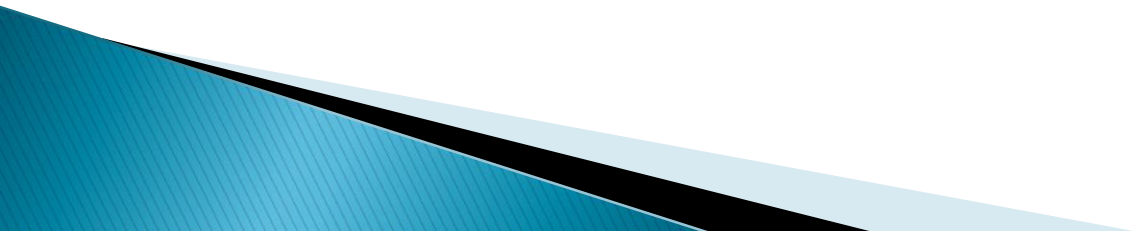
# Șiruri de caractere – Metode specifice

```
s = "Nota la {} = {}".format("PA",10)
```

```
print(s)
```

```
x=3;y=4
```

```
print("x={},y={}".format(x,y))
```



# Șiruri de caractere – Metode specifice

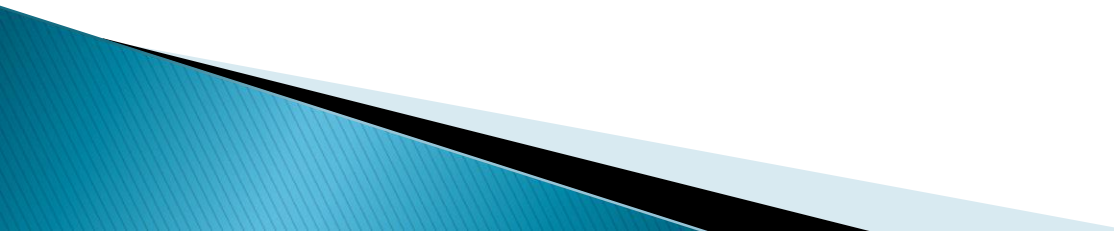
```
s = "Nota la {} = {}".format("PA",10)
```

```
print(s)
```

```
x=3;y=4
```

```
print("x={},y={}".format(x,y))
```

```
print("x={0},y={1}".format(x,y))
```



# Șiruri de caractere – Metode specifice

```
s = "Nota la {} = {}".format("PA",10)
```

```
print(s)
```

```
x=3;y=4
```

```
print("x={},y={}".format(x,y))
```

```
print("x={0},y={1}".format(x,y))
```

```
print("x={1},y={0},x+y={1}+{0}={2}".format(y,x,x+y))
```

# Șiruri de caractere – Metode specifice

```
s = "Nota la {} = {}".format("PA",10)
```

```
print(s)
```

```
x=3;y=4
```

```
print("x={},y={}".format(x,y))
```

```
print("x={0},y={1}".format(x,y))
```

```
print("x={1},y={0},x+y={1}+{0}={2}".format(y,x,x+y))
```

```
print("x={0},y={1}, s={2}".format(x,y))
```

```
#eroare IndexError: Replacement index 2 out of range
```

```
#for positional args tuple
```





# Șiruri de caractere – Metode specifice

Pentru a indica ce parametru numit (cu nume) se folosește într-un câmp de formatare putem folosi direct numele parametrului

```
x=3
```

```
y=4
```

```
print("x={p1},y={p2},s={suma}".format(suma=x+y,p1=x,p2=y))
```

# Șiruri de caractere – Metode specifice

Se pot combina parametri poziționali cu cei numiți, dar cei numiți se dau la final

```
x=3;y=4
```

```
print("{p1}+{p2}={suma}, {p1}*{p2}={}"
```

```
format(x*y,suma=x+y,p1=x,p2=y))
```

# Șiruri de caractere – Metode specifice

Pentru a include acolada în șirul template fără a fi interpretat ca delimitator de câmp se dublează:

```
x = 3; y = 4; z = 5  
s = "Multimea {{{}}, {}, {}}".format(x, y, z)  
print(s)
```

# Șiruri de caractere – Metode specifice

```
z = 1 + 3j
```

```
print('z = {0.real}+ {0.imag}i'.format(z))
```

```
ls = [10,20,30]
```

```
print("primul si al doilea element din lista:  
{0[0]} si {0[1]}".format(ls))
```

# Șiruri de caractere – Metode specifice

## Specificarea formatului de afisare

Lungimea ocupată la afișare, precizie, aliniere, baza în care se afișează, formatul (notație cu exponent etc)

# Șiruri de caractere – Metode specifice

{[<camp>][!<fct\_conversie>][:<format\_spec>]}

<fct\_conversie>

- ▶ !s – convertește cu str()
- ▶ !r – convertește cu repr()
- ▶ !a – convertește cu ascii()

# Șiruri de caractere – Metode specifice

```
s = "Programarea\talgoritmilor Ă"  
print('{!s}'.format(s))  
print('{!r}'.format(s))  
print('{!a}'.format(s))
```

# Șiruri de caractere – Metode specifice

`<format_spec>` – poate include (printre altele)

`[0] [<dimensiune>] [.<precizie>] [<tip>]`

**`<tip>`**

- **d**: întreg zecimal
- **b,o x,X**: întreg în baza 2,8 respectiv 16 cu litere mici/mari
- **s** șir de caractere
- **f**: număr real în virgulă mobilă (afișat implicit 6 cifre)  
... etc



# Șiruri de caractere – Metode specifice

```
z = 1.1 + 2.2
```

```
print('{}'.format(z))
```

```
print('{:f}'.format(z))
```

```
print('{:.2f}'.format(z))
```

# Șiruri de caractere – Metode specifice

```
z = 12
```

```
print('{}'.format(z))
```

```
print('{:8}'.format(z))
```

```
print('{:8b}'.format(z))
```

```
print('{:08b}'.format(z))
```



# Șiruri de caractere – Metode specifice

## Formatare – Interpolarea șirurilor: f-stringuri

– începând cu Python 3.6

- șir <template> precedat de f sau F
- în câmpurile de formatare putem folosi direct nume de **variabile**, chiar și expresii

# Șiruri de caractere – Metode specifice

```
nume = 'Ionescu'
```

```
prenume = 'Ion'
```

```
varsta = 20
```

```
s = f"{nume} {prenume}: {varsta} ani"
```

```
print(s)
```



```
s = "{} {}: {} ani".format(nume, prenume, varsta)
```

```
print(s)
```

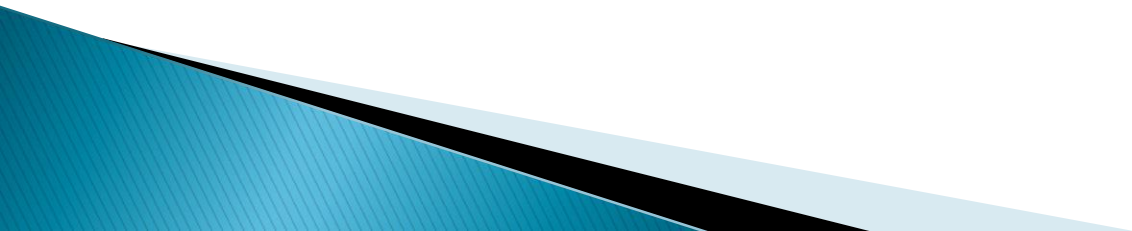
# Șiruri de caractere – Metode specifice

```
x=3;y=4
```

```
print(f" {x} ")
```

```
print(f" {x}+{y}={x+y} , {x}*{y}={x*y} ")
```

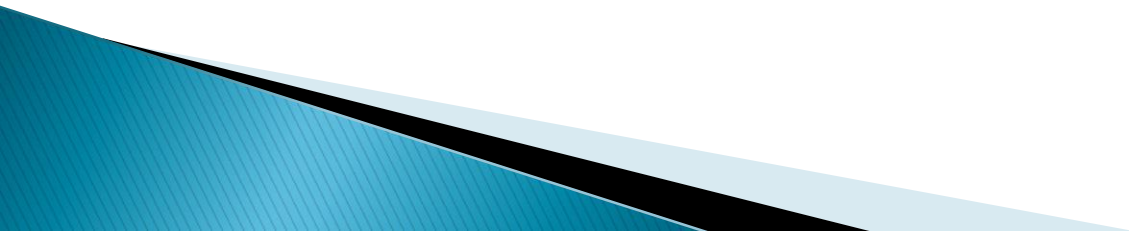
```
print(f' {x:08b} ')
```



# Șiruri de caractere – Metode specifice

Formatare

<https://realpython.com/python-formatted-output/>



# Șiruri de caractere – Metode specifice

## Formatare cu operatorul %

- similar cu limbajul C (funcția printf)
- ▶ old- style – nu se mai recomandă folosirea ei

`<template> % (<values>)`

# Șiruri de caractere – Metode specifice

```
s = "Nota la %s = %d" % ("PA",10)
```

```
print(s)
```

```
x=3.1415
```

```
print("%d %.2f" % (x,x))
```



