UNIVERSITATEA TEHNICĂ A MOLDOVEI

Programare Interactivă

Tema: Introducere în Python

lect. univ. Victoria ALEXEI
Departament Informatică și Ingineria Sistemelor
Facultatea Calculatoare, Informatică și Microelectronică

UNIVERSITATEA TEHNICĂ A MOLDOVEI

În acest curs vom învăța:

- Structurile iterative (while, for)
- Bibliotecile predefinite în Python
- Crearea șirurilor de caractere
- Operații asupra șirurilor
- Accesul la caracterele individuale a șirului
- Slice-ing în șirurile de caractere
- Formatarea stringurilor
- Metodele specifice de prelucrare a stringurilor

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Structurile iterative - Ciclurile

- **Structurile iterative,** cu pași repetați, care au variabile de iterație care se modifică la fiecare iterare.

Adesea, aceste **variabile de iterare** trec printr-o secvență de numere ce satisfac condiției.

Program:

```
n = 5

while n > 0:
    print(n)
    n = n - 1
print('Atat!')
print(n)
```

```
5
4
3
2
1
Atat!
0
>>>
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Cicluri nedefinite

```
>>> a=0
>>> while(a<7):
        a=a+1
        print(a)
```

```
1
2
3
4
5
6
7
```

- Ciclurile **while** sunt numite "**bucle nedefinite**", pentru că se execută atâta timp până când ***condiția logică*** **devine falsă**

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Cicluri nedefinite

```
>>> a=0
>>> while(a<7):
        a=a+1
        print(a)
```

```
1
2
3
4
5
6
7
```

- Ciclurile **while** sunt numite "**bucle nedefinite**", pentru că se execută atâta timp până când ***condiția logică*** **devine falsă**

```
>>> a=0
>>> while(a<7):
        a=a+1
        print(a, end=" ")
```

```
1 2 3 4 5 6 7
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

WHILE

```
>>> def inmultire(baza):
        rezultat=[]
        n=1
        while n < 11:
                b = n * baza
                rezultat.append(b)
                n = n+1
        return rezultat
```

```
>>> ti9=inmultire(9)
>>> ti9
[9, 18, 27, 36, 45, 54, 63, 72, 81, 90]
>>>
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

WHILE

```
>>> def inmultire(baza):
        rezultat=[]
        n=1
        while n < 11:
                b = n * baza
                rezultat= rezultat + [b]
                n = n+1
        return rezultat
```

```
>>> print(inmultire(8)[3])
32
```

```
>>> ti7= inmultire(7)
>>> ti7
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
>>>
```

```
>>> print(inmultire(8))
[8, 16, 24, 32, 40, 48, 56, 64, 72, 80]
>>>
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Cicluri definite

- Deseori avem o listă de elemente, linii într-un fișier, sau într-o listă, adică **un set finit de lucruri**
- Putem să scriem un script pentru a rula ciclul o dată pentru fiecare dintre elementele dintr-un set folosind **for ...in**
- Aceste cicluri sunt numite “**definite**", deoarece ele execută un număr exact de ori
- Noi spunem că **"bucle bine definite iterează membrii unui set"**

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Cicluri definite

- Ciclurile definite **for** au **variabile de iterație explicite** care se schimbă de fiecare dată prin ciclu. Aceste **variabile de iterație** se deplasează prin secvență sau set.
- Variabila de iterație "iterează", **prin secvență (setul ordonat)**
- Blocul de intrucțiuni este executat o singură dată pentru fiecare valoare **în** secvență
- **Variabila de iterație** se mută prin toate valorile din secvență

UNIVERSITATEA TEHNICĂ A MOLDOVEI

FOR

```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('ATAT!')
```

```
5
4
3
2
1
ATAT!
>>>
```

```
for letter in "hello world":
    print(letter*2, end='')
```

```
hheellllloo  wwoorrlldd
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

For cu stringuri

```
friends = ['Ion', 'Maria', 'Elena']
for friend in friends :
    print('La multi ani:',  friend)
print('Atat!')
```

```
La multi ani: Ion
La multi ani: Maria
La multi ani: Elena
Atat!
>>>
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Exemplu:

```
count = 0
sum = 0
print ('Inainte', count, sum)
for value in [9, 41, 12, 3, 74, 15]:
    count = count + 1
    sum = sum + value
    print (count, sum, value)
print('Dupa', count, sum, sum / count)
```

```
Inainte 0 0
1 9 9
2 50 41
3 62 12
4 65 3
5 139 74
6 154 15
Dupa 6 154 25.666666666666668
>>>
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Exemplu: căutare utilizând o variabilă booleană

```
found = False
print('Inainte', found)
for value in [9, 41, 12, 3, 74, 15]:
    if value == 3:
        found = True
    print(found, value)
print('Dupa', found)
```

```
Inainte False
False 9
False 41
False 12
True 3
True 74
True 15
Dupa True
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Instrucțiunea **break**

```
while True:
    line = input('> ')
    if line == 'atat':
        break
    print(line)
print('Atat!')
```

```
> unu
unu
> doi
doi
> trei
trei
> patru
patru
> cinci
cinci
> atat
Atat!
```

Instrucțiunea **break** termină ciclul curent și trece la instrucțiunea următoare ce urmează imediat după buclă

Este ca un **test în buclă** care se poate plasa oriunde în corpul ciclului

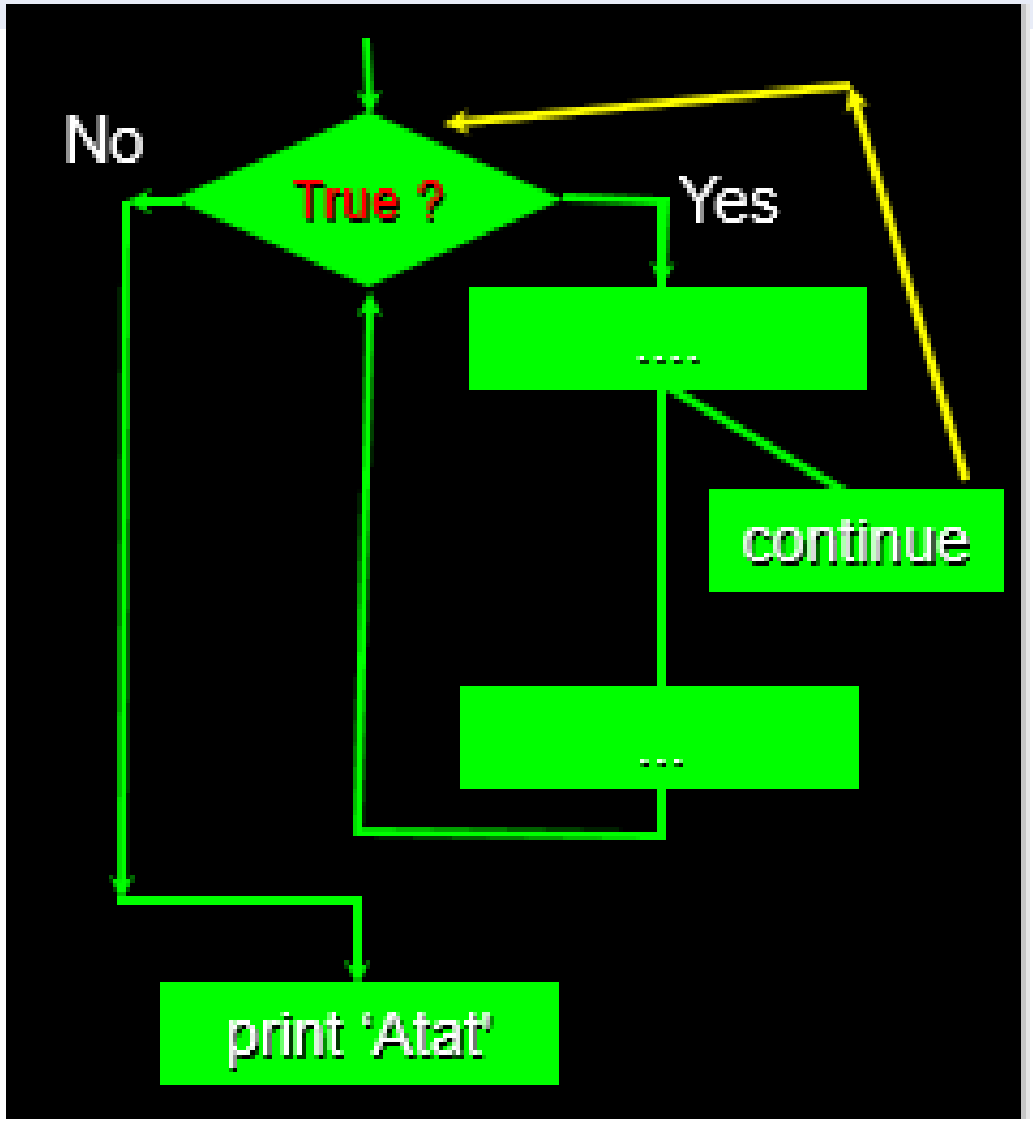
UNIVERSITATEA TEHNICĂ A MOLDOVEI

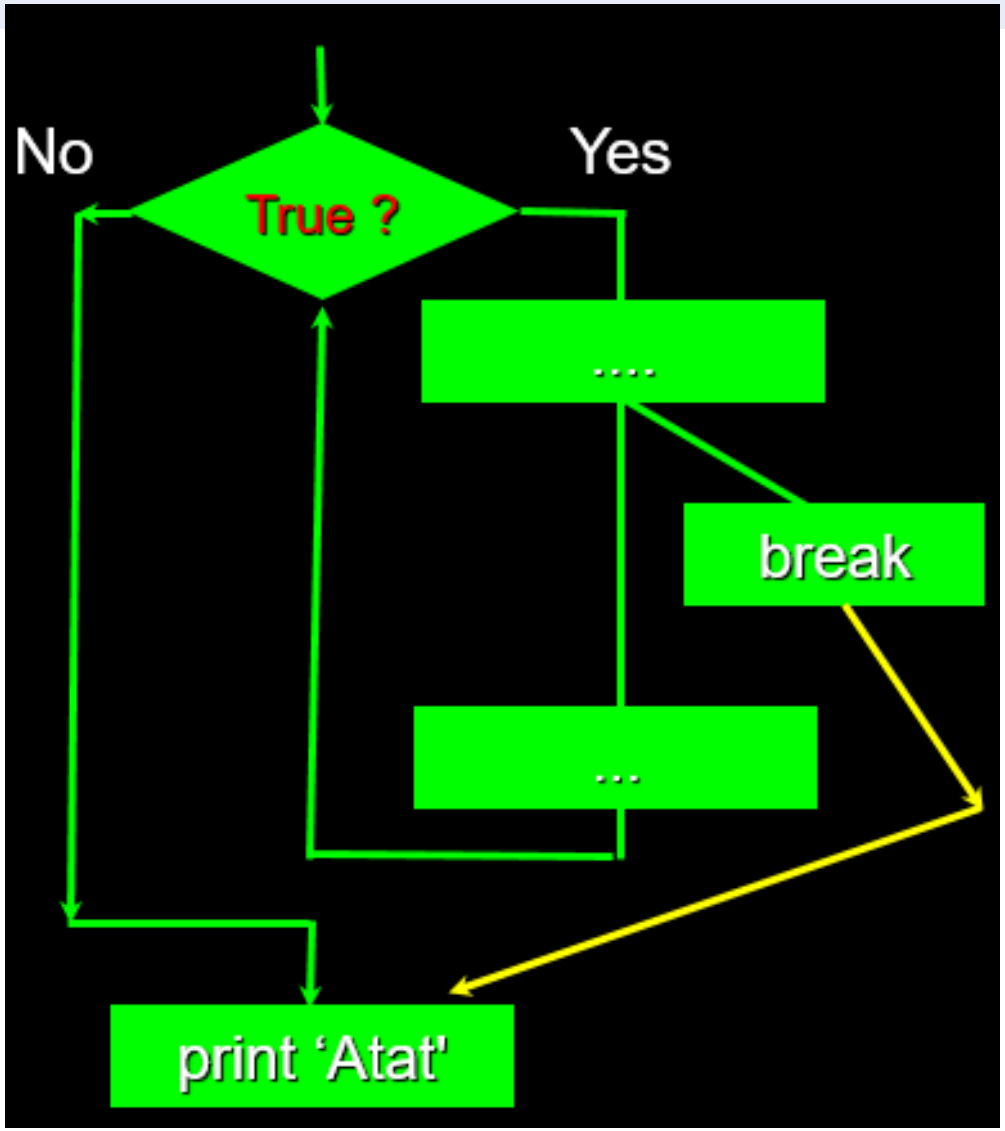
Instrucțiunea **continue**

- Instrucțiunea **continue** termină iterația curentă și sare la începutul ciclului și pornește următoarea iterație

```
while True:
    line = input('> ')
    if line[0] == '#':
        continue
    if line == 'atat':
        break
    print(line)
print('Atat!')
```

```
> afisam
afisam
> # nu afisam
> afisam aceasta
afisam aceasta
> atat
Atat!
>>>
```


No
True ?
Yes
....
continue
...
print 'Atat'


No
True ?
Yes
....
break
...
print 'Atat'

UNIVERSITATEA TEHNICĂ A MOLDOVEI

While, For ... else

- Instrucțiunea **else** se utilizează în ciclurile while și for, verifică dacă s-a efectuat ieșirea din ciclu cu instrucțiunea break sau în mod natural. Blocul de instrucțiuni în interiorul la else se execută doar în cazul dacă ieșirea din ciclu s-a efectuat fără ajutorul instrucțiunii break.

```
for letter in "hello world":
    if letter == 'a':
        break
else:
        print('Litera "a" nu este in acest sir')
```

```
Litera "a" nu este in acest sir
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operatorii ”is” și ”is not”

```
smallest = None
print('Inainte')
for value in [3, 41, 12, 9, 74, 15]:
    if smallest is None:
        smallest = value
    elif value < smallest:
        smallest = value
    print (smallest, value)
print('După', smallest)
```

```
Inainte
3 3
3 41
3 12
3 9
3 74
3 15
După 3
```

Python are un operaror „**is**" care poate fi utilizat în expresii logice

El implică ”**este același lucru cu**„

Este similar, dar mai puternic decît **==**

„**is not**", de asemenea, este un operator logic

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Bibliotecile python

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'cla
ss', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from
', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pas
s', 'raise', 'return', 'try', 'while', 'with', 'yield']
>>>
```

```
>>> dir(keyword)
['__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '
__name__', '__package__', '__spec__', 'iskeyword', 'kwlist', 'main']
>>>
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Bibliotecile python

```
>>> help(iskeyword)
Traceback (most recent call last):
  File "<pyshell#25>", line 1, in <module>
    help(iskeyword)
NameError: name 'iskeyword' is not defined
>>> help(keyword.iskeyword)
Help on built-in function __contains__:

__contains__(...) method of builtins.frozenset instance
    x.__contains__(y) <==> y in x.

>>>
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Bibliotecile Python

```
>>> import math
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh'
, 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh',
'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fm
od', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'is
inf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan'
, 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'ta
u', 'trunc']
>>>
```

```
>>> help(math.exp)
Help on built-in function exp in module math:

exp(x, /)
    Return e raised to the power of x.
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Bibliotecile python

```
>>> from math import*
>>> sqrt(121)
11.0
>>> pi
3.141592653589793
>>> round(pi,2)
3.14
>>> pow(2,3)
8.0
>>> int(pow(2,3))
8
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Definiție. Crearea șirurilor de caractere

Un string reprezintă orice tip de text inclus între apostrofuri, ghilimele sau ghilimele /apostrofuri triple.

```
ex1 = "string"
ex2 = 'string'
ex3 = """string"""
```

Exemple de șiruri de caractere

```
ex4 = '''Acesta este
un string pe mai multe randuri'''
ex5 ="""acesta tot e un string
pe mai multe randuri"""
```

Docstring-urile sunt similare cu comentariile, dar sunt o versiune îmbunătățită, mai logică și utilă

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Crearea șirurilor de caractere

NB!
Încheiem reprezentarea șirului cu acelaşi tip de ghilimele cu care am început

Exemplu

```
ex6 = "string'
```

Rezultat

```
In [34]: runfile('C:/Users/Admin/Desktop/Sir.py', wdir='C:/Users/Admin/
Desktop')
  File "C:\Users\Admin\Desktop\Sir.py", line 14
    ex6 = "string'
                  ^
SyntaxError: EOL while scanning string literal
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Crearea șirurilor de caractere

La crearea șirului cu ghilimele, noi putem include în șir - caracterul apostrof, atunci Python va trata caracterul respectiv ca parte a șirului

Exemplu

```
ex7 ="acesta string contine apostrof ' in interior"
ex8 = 'string cu apostrof \' in interior'
```

Rezultat

```
acesta string contine apostrof ' in interior
string cu apostrof ' in interior
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Crearea șirurilor de caractere

Semnul backslash sau bara inversă ascunde semnificația specială a unui apostrof și ghilimele duble.

Exemplu

```
ex9 = 'Utilizam un citat "In ghilimele duble" intr-un sir'
ex10 = "Utilizam un citat \"In ghilimele duble\" intr-un sir"
```

Rezultat

```
Utilizam un citat "In ghilimele duble" intr-un sir
Utilizam un citat "In ghilimele duble" intr-un sir
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operații asupra șirurilor

Operaţiile aritmetice de adunare şi înmulţire sunt posibile asupra șirurilor de caractere

Operatorul + este utilizat pentru concatenarea / unirea a două sau mai multe șiruri. Returnează șirul rezultat concatenat.

Exemplu

```
s1 = 'Salutare'
s2 = 'tuturor!'

s = s1 + s2
print(s)
```

Concatinarea șirurilor

Rezultat

```
Salutaretuturor!
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operații asupra șirurilor

Operaţiile aritmetice de adunare şi înmulţire sunt posibile asupra șirurilor de caractere

Operatorul * este utilizat pentru a repeta un șir de un anumit număr de ori. Returnează șirul rezultat repetat.

NB! pentru ***adunare***, ambele elemente trebuie să fie string, iar pentru ***înmulţire*** este necesar un string şi un număr întreg.

Exemplu

```
s1 = 'Salutare'

si = s1 * 3

print(si)
```

Repetarea șirurilor

Rezultat

```
SalutareSalutareSalutare
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operații asupra șirurilor

Două **șiruri pot fi egale**, dacă au aceeași lungime și aceleași caractere și sunt în aceeași ordine.

Operatorii de comparare egal și diferit: “==“ , “!=“

Operatorul **==** returnează **True** dacă există o potrivire exactă, altfel **False** va fi returnat.
În schimb, operatorul **! =** returnează **True** dacă nu există potrivire.

Exemplu

```
print("ziua" == 'ziua')

print("Ziua" == 'ziua')

print("Ziua" != 'ziua')

print("iuza" != 'ziua')

print("iuza" == 'ziua')
```

Rezultat

```
True
False
True
True
False
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operații asupra șirurilor

Pentru o comparație cu privire la **o ordine lexicografică** se vor utiliza operatorii de comparație **<,>, <= și > =**. Comparația în sine se face caracter cu caracter. Ordinea depinde de ordinea caracterelor din alfabet.

Ordinea este diferențiată și de majuscule și minuscule.
Ca exemplu pentru alfabetul latin, „Autobuz” vine înainte de „autobuz”.

Exemplu

```
print("aloha" > 'ziua')
print("Aloha" < 'ziua')
```

Rezultat

```
False
True
```

```
print(ord("A"))
print(ord("a"))

print(ord("z"))
print(ord("Z"))
```

```
65
97
122
90
```

Funcția **ord()** este utilizată pentru a găsi valoarea întreagă a unui caracter.

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operații asupra șirurilor

Operatorii **in, not in**

in: Acesta verifică dacă un șir este prezent în alt șir sau nu.

Returnează True dacă întregul șir este găsit altfel returnează False.

not in: Funcționează exact opusul la ceea ce face operatorul „in”.

Returnează True dacă șirul nu este găsit în șirul specificat, altfel returnează False.

Exemplu

```
st = 'Scriem un mesaj'
print('un' in st)

print('scr' in st)

print('s' in st)

print('un' not in st)
```

Rezultat

```
True
False
True
False
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operații asupra șirurilor

Conversia număr - șir, șir - număr

Dacă avem nevoie să transformăm tipul numeric în șir de caractere atunci conversia explicită spre acest tip se efectuează prin funcția **str()**.

Exemplu

```
n = 23
s3 = str(n)
print(s3)
print(type(s3))

f = -5.6
s4 = str(f)
print(s4,type(s4))

print(int(s3) + float(s4))
```

Rezultat

```
23
<class 'str'>
-5.6 <class 'str'>
17.4
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operații asupra șirurilor

Lungimea șirului funcția **len()**

Exemplu

```
st = 'Scriem un mesaj'
print(len(st))
```

Rezultat

```
15
```

min(s), max(s) - returnează litera minimă/maximă din string (**NB!**, toate literele mari vor fi considerate "mai mici" decât orice literă mică).

```
print(max('numar'), min('numar'))
```

```
u a
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Accesul la elementele șirurilor, indexare

Un șir de caractere nu este altceva decât un tablou unidimensional, astfel încât putem folosi indecșii pentru a accesa caracterele acestuia. La fel ca tablourile, indecșii încep de la **0** până la **lungimea-1**.

Putem obține orice caracter dintr-un șir de caractere folosind indexul specificat în **paranteze pătrate**

den_string[index]

```
st = 'Scriem un mesaj'
```

S	c	r	i	e	m		u	n		m	e	s	a	j
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Accesul la elementele șirurilor, indexare

```
st = 'Scriem un mesaj'
```

S	c	r	i	e	m		u	n		m	e	s	a	j
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Exemplu

```
print(st[2])
print(st[5])
print(st[14])
print(st[-1])
print(st[-7])
print(st[15])
```

Rezultat

```
r
m
j
j
n
```

```
Traceback (most recent call last):

  File "C:\Users\Admin\Desktop\Sir.py", line 111, in <module>
    print(st[15])

IndexError: string index out of range
```

NB! Nu trebuie să accesați elemente care sunt în afara lungimii șirului pentru a evita **IndexError**.

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Slice-ing în șirurile de caractere

Slicing – felierea stringurilor

Putem accesa o secvență continuă de caractere, numită substring, printr-un proces numit **feliere (slicing)**

den_string[start : stop : [step]]

Tipărește de la indexul **start**(inclus) până la indexul **stop**(exclus), cu pasul **step**(opțional)

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Slice-ing în șirurile de caractere

Slicing – felierea stringurilor **den_string[start : stop : [step]]**

S	c	r	i	e	m		u	n		m	e	s	a	j
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Exemplu

```
print(st[0:8])
print(st[:8])
print(st[9:])
print(st[2:11])
print(st[:])
print(st[14:9:-1])
print(st[::-1])
print(st[:10:2])
print(st[::-2])
```

Rezultat

```
Scriem u
Scriem u
 mesaj
riem un m
Scriem un mesaj
jasem
jasem nu meircS
Sre n
jsmn erS
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Accesul la elementele șirurilor

Stringurile sunt secvențe imutabile

Exemplu

```
st = 'Scriem un mesaj'
st[0]="A"
```

Rezultat

```
Traceback (most recent call last):

  File "C:\Users\Admin\Desktop\Sir.py", line 116, in <module>
    st[0]="A"

TypeError: 'str' object does not support item assignment
```

Crearea unui string nou cu modificările dorite

```
st1 = 'A'+ st[1:]
print(st1)
```

```
Acriem un mesaj
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Formatarea șirurilor

O parte importantă în lucrul cu stringurile o reprezintă formatarea acestora.

Prin formatare vom intelege posibilitatea de a crea un string în mod dinamic folosind valori păstrate în variabile.

Prima şi cea mai simplă metodă de formatare o reprezintă concatenarea stringurilor implicate folosind operaţia de adunare

Exemplu

```
nume = "Maria"
anul = 2

print("Ma numesc " + nume + ' si sunt studenta in anul ' + str(anul) + ' la universitate')
```

Rezultat

```
Ma numesc Maria si sunt studenta in anul 2 la universitate
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Formatarea șirurilor

O altă metodă de formatare o reprezintă folosirea caracterului %

Exemplu

```
nume = "Maria"
anul = 2
print('Ma numesc %s si sunt studenta in anul %d la universitate' %(nume, anul))
```

`%s` indică faptul că pe acea pozitie se asteaptă un element de tip string, iar `%d` indică prezenţa unui element de tip întreg. Sunt mai multe tipuri de astfel de simboluri specifice diferitelor tipuri de date

Rezultat

```
Ma numesc Maria si sunt studenta in anul 2 la universitate
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Formatarea șirurilor

Altă metodă de formatare a stringurilor este utilizarea metodei **format().**

Acoladele pot conţine informaţii despre parametrii. Astfel, dacă punem un număr în interiorul acoladelor, atunci Python va asocia acel număr cu poziţia parametrilor din metoda format

NB! numărătoarea începe de la 0 şi se poate continua până la numărul parametrilor - 1.

Exemplu

```
print('Ma numesc {} si sunt studenta in anul {} la universitate'.format(nume, anul))

print('Ma numesc {1} si sunt studenta in anul {0} la universitate'.format(anul, nume))
```

Rezultat

```
Ma numesc Maria si sunt studenta in anul 2 la universitate
Ma numesc Maria si sunt studenta in anul 2 la universitate
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Formatarea șirurilor

Această metodă ne permite refolosirea parametrilor prin repetarea unui index (numărul parametrului)

Exemplu

```
print('2**10 = {} si {} * {} = {}'.format(2**10, 4, 5, 4*5))

print('2**10 = {0} si {1} * {1} = {2}'.format(2**10, 4, 4*4))
```

Rezultat

```
2**10 = 1024 si 4 * 5 = 20
2**10 = 1024 si 4 * 4 = 16
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Formatarea șirurilor

Exemplu

```
print('2**10 = {} si {1} * {1} = {}'.format(2**10, 4, 4*4))
```

Rezultat

```
Traceback (most recent call last):

  File "C:\Users\Admin\Desktop\Sir.py", line 138, in <module>
    print('2**10 = {} si {1} * {1} = {}'.format(2**10, 4, 4*4))

ValueError: cannot switch from automatic field numbering to manual field
specification
```

NB! Nu putem folosi în acolade și completarea automată și manuală

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Formatarea șirurilor

Metoda format() are o mulțime de opțiuni. Mai jos vedem un exemplu cu opțiuni de poziționare în șir.

Exemplu

```
print('2**10 = {:<10} si {:>10} * {:<6} = {:>10}'.format(2**10, 4, 5, 4*5))
print('2**10 = |{:<10}| si |{:>10}| * |{:<6}| = |{:>10}|'.format(2**10, 4, 5, 4*5))
```

Rezultat

```
2**10 = 1024       si          4 * 5      =         20
2**10 = |1024      | si |          4| * |5     | = |        20|
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Formatarea șirurilor

Ultima metodă de formatare a fost adăugată în Python 3.6 şi presupune adăugarea caracterului **f** înaintea şirului nostru de caractere, iar între acolade, vom trece numele variabilelor pe care vrem să le includem în formatare. Numele variabilelor din acolade trebuie să coincidă cu cel din declarare.

Exemplu

```
nume = "Maria"
anul = 2
print(f'Ma numesc {nume} si sunt studenta in anul {anul} la universitate')

a = 1
b = 2
c = a + b
# a, b și c sunt înlocuite cu valorile lor
print (F"Suma de {a} + {b} este egală cu {c}")
```

Rezultat

```
Ma numesc Maria si sunt studenta in anul 2 la universitate
Suma de 1 + 2 este egală cu 3
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

- Python are un număr de funcții string, care se află în biblioteca string
- Aceste funcții sunt predefinite - noi doar le invocăm prin alipirea funcției la variabila șir
- Aceste funcții nu modifică șirul inițial, în schimb, ele returnează un nou șir de caractere care a fost modificat

```
s='Cartile sunt prietenii nostri'
print(dir(s))
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

```
s='Cartile sunt prietenii nostri'
print(dir(s))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
'__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__',
'__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize',
'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find',
'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal',
'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace',
'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans',
'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit',
'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title',
'translate', 'upper', 'zfill']
```

https://docs.python.org/3/library/string.html

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

Metodele upper() - returnează șirul original în majuscule,
lower() - returnează șirul original în litere mici,
capitalize() - returnează șirul original unde primul cuvânt începe cu literă mare
title() - returnează șirul original unde toate cuvintele încep cu o literă mare

```
s='Cartile sunt prietenii nostri'
```

Exemplu

```
print(s.upper())
print(s.lower())
print(s.capitalize())
print(s.title())
```

Rezultat

```
CARTILE SUNT PRIETENII NOSTRI
cartile sunt prietenii nostri
Cartile sunt prietenii nostri
Cartile Sunt Prietenii Nostri
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

Metodele strip(), rstrip(), lstrip()

Funcția **strip()** elimină spațiile de la începutul și sfârșitul șirului.
Funcția **lstrip()** elimină spațiile de la începutul șirului (l -left)
Funcția **rstrip()** elimină spațiile de la sfârșitul șirului (r -right)

Exemplu

```
s1='   Cartile sunt prietenii nostri  '
print(s1)
print(s1.strip())
s1='   Cartile sunt prietenii nostri  '
print(s1.lstrip())
```

Rezultat

```
   Cartile sunt prietenii nostri
Cartile sunt prietenii nostri
Cartile sunt prietenii nostri
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

Metodele strip(), rstrip(), lstrip()

Exemplu

```
s1='=====Cartile sunt prietenii nostri+++'
print(s1.lstrip('='))
print(s1.rstrip('+'))
print(s1.strip('=+'))
```

Rezultat

```
Cartile sunt prietenii nostri+++
=====Cartile sunt prietenii nostri
Cartile sunt prietenii nostri
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

Metodele find(), rfind()

find() găsește un subșir în șir și returnează poziția unde începe subșirul
Dacă subșirul nu poate fi găsit, atunci funcția find() returnează -1

Exemplu

```
s1='Cartile sunt prietenii nostri  '
print(s1)
print(s1.find('sunt'))
print(s1.find('Sunt'))
```

Rezultat

```
8
-1
```

```
print(s1.find('prie',3,len(s1)))
```

```
13
```

3 este indexul unde dorim să începem căutarea
len(s1) este indexul unde dorim să oprim căutarea

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

Metodele find(), rfind()

rfind() găsește un subșir în șir și returnează poziția unde începe subșirul

Dacă subșirul nu poate fi găsit, atunci funcția find() returnează -1

Exemplu

```
s2='Sunt prieteni cei acei oameni ce sunt aproape la bine si la greu'
print(s2.find('sunt'))
print(s2.rfind('sunt'))
```

Rezultat

```
33
33
```

```
s2='Sunt prieteni cei acei oameni ce sunt aproape la bine si la greu'
print(s2.lower().find('sunt'))
print(s2.rfind('sunt'))
```

```
0
33
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

Metodele index(), rindex()

index() returnează poziția primei apariții a șirului căutat

Dacă șirul nu poate fi găsit, atunci funcția returnează excepție

rindex() returnează poziția ultimei apariții a șirului căutat

Exemplu

```
s2='Sunt prieteni cei acei oameni ce sunt aproape la bine si la greu'
print(s2.index('acei'))
print(s2.index('unt', 5))
print(s2.index('Acei'))
```

Rezultat

```
18
34
Traceback (most recent call last):

  File "C:\Users\Admin\Desktop\Sir.py", line 174, in <module>
    print(s2.index('Acei'))

ValueError: substring not found
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

Metodele count(), replace()

count() numără aparițiile unui subșir într-un șir

replace(old_subs, new_subs) înlocuiește un subșir cu un altul

Exemplu

```
s2='Sunt prieteni cei acei oameni ce sunt aproape la bine si la greu'
print(s2.count('e'))
print(s2.count('ei'))

print(s2.replace('ei','asi'))
print(s2.replace('e','X'))
```

Rezultat

```
9
2
Sunt prieteni casi acasi oameni ce sunt aproape la bine si la greu
Sunt priXtXni cXi acXi oamXni cX sunt aproapX la binX si la grXu
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

Metodele split(), splitlines()

s.split() împarte șirul **s** într-o listă de subșiruri

Separarea se face în funcție de parametrul pe care îl primește funcția split().

Implicit separatorul este caracterul spațiu

Exemplu

```
s2='Sunt prieteni cei acei oameni ce sunt aproape la bine si la greu'
print(s2.split())
```

Rezultat

```
['Sunt', 'prieteni', 'cei', 'acei', 'oameni', 'ce', 'sunt', 'aproape', 'la',
'bine', 'si', 'la', 'greu']
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

Metodele split()

Separator poate fi orice caracter

Exemplu

```
s3 = '23-76-6-4-9-0-45'
print(s3.split('-'))
print('Omul iubeste cu ochii, natura, frumosul!'.split(','))
```

Rezultat

```
['23', '76', '6', '4', '9', '0', '45']
['Omul iubeste cu ochii', ' natura', ' frumosul!']
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

Metodele split(), splitlines()

splitlines() împarte șirul la semnul ‘\n’ (rând nou) și returnează o listă

Exemplu

```
ex6 ='!avem un sir \npe mai multe \nranduri!'
print(ex6.splitlines())
```

Rezultat

```
['!avem un sir ', 'pe mai multe ', 'randuri!']
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

Metode de verificare

- **isalpha()** returnează True dacă șirul constă doar din litere, altfel - False
- **isalnum()** returnează True dacă șirul este format din cifre și litere, altfel – False
- **isdigit()** returnează True dacă șirul este format din cifre, altfel – False
- **islower()** returnează True dacă șirul este format din caractere minuscule, în caz contrar - False
- **isupper()** returnează True dacă șirul este format din caractere mari, altfel - False
- **isspace()** returnează True dacă șirul constă din caractere care nu sunt afișate (spații, etc.), altfel – False
- **istitle ()** returnează True dacă toate cuvintele încep cu o literă mare în șir

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metodele specifice de prelucrare a stringurilor

Metode de verificare

Exemple

```
print('     '.isspace())
print('litere'.isalpha())
print('litere678'.isalnum())
print('236598'.isdigit())
print('Verificam Titlul'.istitle())
print('TOATE CU MAJUSCULE'.isupper())
print('toate mici'.islower())
```

Rezultatele

```
True
True
True
True
True
True
True
True
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Parcurgerea stringurilor

Folosind **while,** o **variabilă de iterație**, și funcția **len**, putem construi o buclă să parcurgem fiecare literă din șirul de caractere în mod individual

```
s5 = 'banana'
index = 0
while index < len(s5):
    letter = s5[index]
    print(index, letter)
    index = index + 1
```

```
0 b
1 a
2 n
3 a
4 n
5 a
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Parcurgerea stringurilor

Ciclul **for** este mult mai *elegant*

Variabila de iterație este complet luată de către **for**

```
s4 = "cursuri"
for elem in s4:
    print(elem)
```

```
c
u
r
s
u
r
i
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Exemple de probleme cu stringuri

```python
def string_reverse(str1):
    '''
    Returnează stringul inversat.

    Parameters:
        str1 (str): Stringul ce va fi inversat.

    Returns:
        reverse(str1): Stringul ce este inversat.
    '''

    reverse_str1 = ''
    i = len(str1)
    while i > 0:
        reverse_str1 += str1[i - 1]
        i = i- 1
    return reverse_str1


print(string_reverse('BazeleProgramariiCalculatoarelor'))
```

```
roleraotaluclaCiiramargorPelezaB
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Exemple de probleme cu stringuri

```
# Elimina caracterele ce se repeta
sir= input('Introdu un sir:> ')
r=[]
for c in sir:
    if not(c in r):
        r.append(c)
print(r)
myStr = ''.join(r)
print(myStr)
```

```
Introdu un sir:> caractere
['c', 'a', 'r', 't', 'e']
carte
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Exemple de probleme cu stringuri

```
#crearea sirurilor noi
prefix = "ODTPSLM"
sufixul = "ac"
for litera in prefix:
    print(litera + sufixul)
```

```
Oac
Dac
Tac
Pac
Sac
Lac
Mac
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Propuneri pentru lucrul individual

1. Revizuiți conținutul cursului

2. Fiind dat un șir format din cuvinte separate prin spații, să se numere câte cuvinte conține șirul.

3. Să se scrie un program, care solicită utilizatorului o propoziție și scrie întreaga propoziție cu majuscule și fără spații albe.

4. Se dă un șir de caractere. Șirul de caractere de împărțit în fragmente a câte trei simboluri consecutive. În fiecare fragment simbolul de mijloc se înlocuiește cu un caracter aleatoriu care nu coincide cu oricare dintre caracterele acestui fragment.

5. Se dau două șiruri. Să se afișeze șirul de dimensiune mai mare de atâtea ori, de câte numărul de elemente sunt diferite.

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Mulțumesc de atenție!

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

UNIVERSITATEA TEHNICĂ
A MOLDOVEI