Șiruri de caractere (strings)

Secventa de caractere

Un string este reprezentat de o secventa de caractere. Acestea pot fi accesate folosind operatorul [ ].

Exemple:

```
>>> fruct = 'banana'
>>> litera = fruct[1]
```

A doua instrucțiune extrage caracterul de pe poziția 1 din variabila fruct și o atribuie variabilei litera.

```
>>> print(litera)
a
```

În Python indicii încep de la valoarea 0, astfel încât fruct[0] va avea valoarea ‘b’.

Lungimea unui sir de caractere

Funcția care returnează lungimea unui string este len.

```
>>> fruct = 'banana'
>>> len(fruct)
6
```

Pentru a obține ultima literă a unui cuvânt, este tentant să folosim următoarele instrucțiuni:

```
>>> lungime= len(fruct)
>>> ultim = fruct[lungime]
IndexError: string index out of range
```

Motivul erorii îl repreintă faptul că nu există un indice cu valoarea 6 pentru literele din cuvântul ‘banana’. Varianta corectă este:

```
>>> ultim = fruct[lungime-1]
>>> print(ultim)
a
```

Parcurgerea unui șir de caractere cu o buclă

Parcurgerea unui șir de caractere se referă la procesarea caracter cu caracter a șirului. Exemplu de parcurgere cu o buclă while:

```
index = 0
```

```
while index < len(fruct):
        litera = fruct[index]
        print(litera)
        index = index + 1
```

Această buclă parcurge șirul de caractere literă cu literă și afișează litera pe câte o linie.

Exercițiu:

Scrieți o buclă while care începe cu ultimul caracter dintr-un string și îl parcurge în sens invers, afișând câte o literă pe câte o linie.

Felii de șiruri de caractere (Slices)

Un segment dintr-un șir de caractere se numește felie (slice). Selectarea unui slice se face astfel:

```
>>> s = 'Monty Python'
>>> print(s[0:5])
Monty
>>> print(s[6:12])
Python
```

Operatorul [ ] returnează partea din șirul de caractere care începe cu al n-lea caracter, până la al m-lea caracter, incluzându-l pe primul, dar excluzându-l pe ultimul.

Dacă omitem primul indice înainte de :, felia începe la începutul șirului de caractere. Dacă îl omitem pe al doilea, felia se va termina la capătul șirului de caractere:

```
>>> fruct = 'banana'
>>> fruct[:3]
'ban'
>>> fruct[3:]
'ana'
```

Dacă primul indice este mai mare sau egal cu al doilea, rezultatul va fi un șir vid:

```
>>> fruct = 'banana'
>>> fruct[3:3]
''
```

Atenție! Este tentant de folosit operatorul de atribuire pentru a modifica conținutul unui șir de caractere, însă acest lucru va returna o eroare, întrucât șirurile de caractere sunt inamovibile:

```
>>> greeting = 'Hello, world!'
>>> greeting[0] = 'J'
TypeError: 'str' object does not support item assignment
```

Numărarea caracterelor printr-o buclă

```
word = 'banana'
```

```
count = 0
for letter in word:
        if letter == 'a':
                count = count + 1
print(count)
```

Exercițiu: Încapsulați secvența **de cod de mai sus într-o funcție numită**count și generalizați-o astfel încât să accepte șirul de caractere și litera de numărat ca și argumente.

Operatorul in

Operatorul in este un operator Boolean care ia valoarea True dacă primul șir de caractere este un subșir din al doilea:

```
>>> 'a' in 'banana'
True
>>> 'seed' in 'banana'
False
```

Compararea șirurilor

Șirurile de caractere pot fi comparate:

```
if word < 'banana':
        print('Cuvantul,' + word + ', este inainte de banana.')
elif word > 'banana':
        print('Cuvantul,' + word + ', este dupa banana.')
else:
        print('Egalitate')
```

Pentru a obține rezultate inteligibile, este recomandat să fie transformate șirurile la un format standard, cum ar fi trecerea numai la litere mici, înainte de a face comparația.

Metodele șirurilor de caractere

```
>>> cuvant = 'Hello'
>>> type(cuvant)
<class 'str'>
>>> dir(cuvant)
['capitalize', 'casefold', 'center', 'count', 'encode',
'endswith' , 'expandtabs', 'find', 'format', 'format_map',
'index' , 'isalnum', 'isalpha', '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']
```

```
>>>  help(str.capitalize)
Help on method_descriptor:
capitalize(...)
S.capitalize() -> str
Return a capitalized version of S, i.e. make the first character
have upper case and the rest lower case.
>>>
```

Exemple de utilizare a metodelor:

```
>>> word = 'banana'
>>> index = word.find('a')
>>> print(index)
1
```

```
>>> word.find('na', 3)
4
```

```
>>> line = 'Have a nice day'
>>> line.startswith('Have')
True
>>> line.startswith('h')
False
```

Pentru că metoda startswith este case sensitive, poate fi utilizată și metoda lower pentru a obține rezultatul corect:

```
>>> line.lower()
'have a nice day'
>>> line.lower().startswith('h')
True
```

Parsarea șirurilor de caractere

Deseori este nevoie să verificăm apariția unui subșir intr-un șir. De exemplu avem următoarea linie:

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

Ne dorim să extragem domeniul adresei de email din această linie. Vom folosi următoarea secvență

```
>>> data = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
>>> atpos = data.find('@')
>>> print(atpos)
21
>>> sppos = data.find(' ',atpos)
>>> print(sppos)
31
>>> host = data[atpos+1:sppos]
>>> print(host)
uct.ac.za
>>>
```

Documentația pentru metodele pentru lucrul cu șiruri de caractere este disponibilă la adresa:

https://docs.python.org/library/stdtypes.html#string-methods.

Operatorul de formatare

Operatorul de formatare % ne permite să construim șiruri de caractere prin înlocuirea unor părți ale lor cu date din variabile.

Exemple:

%d – formatarea unor numere întregi

camile= 42

print('Am vazut %d camile.' % camile)

Am vazut 42 camile.

Dacă avem mai mult de un operator de formatare în string, al doilea argument trebuie să fie o tuplă, iar fiecare formatare va fi asociată cu un element din tuplă, în ordine:

print('In %d ani am vazut %g %s.' % (3, 0.1, 'camile')*)*

Mai multe despre operatorul de formatare puteți găsi la adresa:

https://docs.python.org/library/stdtypes.html#printf-style-string-formatting.

Exerciții

1. Plecând de la următoarea instrucțiune folosiți metoda find și operațiunile de slice pentru a extrage partea de text de după ":" și folosiți apoi funcția float pentru a converti valoarea extrasă într-un număr real pe care să îl afișați.

   text = 'X-DSPAM-Confidence:**0.8475**'

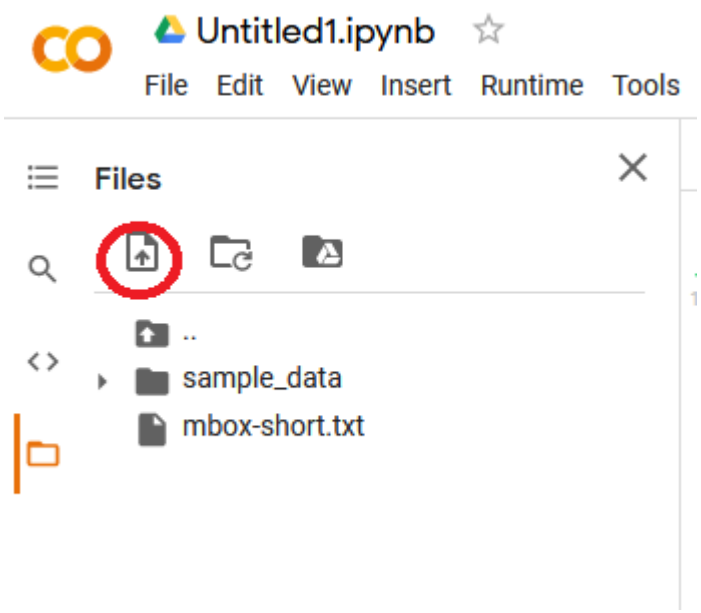
Lucrul cu fișiere

Deschiderea fișierelor

Pentru a citi sau a scrie fișiere, prima operațiune estea aceea de a deschide fișierul (open).

Descărcați fișierele mbox.txt și mbox-short.txt în folderul în care salvați programele Python.

In utlizarea Google Colab , fisierul va fi incarcat in notebook prin utilizarea explorer.



Alternativ se poate utiliza biblioteca “files”, ceea ce va avea acelasi rezultat.

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

In continuare se vor utiliza aceleasi instructiuni ca si in variantele instalate local.

```
>>> f= open('mbox.txt')
>>> print(f)
<_io.TextIOWrapper name='mbox.txt' mode='r' encoding='cp1252'>
```

Sistemul afișează informații despre fișierul deschis folosind funcția print, dacă deschiderea a fost cu succes.

Fișierele text și liniile din fișier

Un fișier text este o secvență de linii de text, așa cum șirul de caractere este o secvență de caractere.

Parcurgerea unui fișier se poate face folosind un ciclu for. De exemplu, pentru numărarea liniilor dintr-un fișier folosim următorul cod:

```
f = open('mbox.txt')
count = 0
for line in f:
        count = count + 1
print('Nr linii:', count)
```

Pentru fișierele foarte mari, Python nu încarcă în memorie întreg fișierul la deschidere, iar ciclul for, parcurgând linie cu linie fișierul, reușește să il parcurgă într-un mod eficient, astfel încât să nu supraîncarce memoria internă.

Pentru a încărca întreg fișierul în memorie, se poate folosi metoda read().

```
>>> f2 = open('mbox-short.txt')
>>> inp = f2.read()
>>> print(len(inp))
94626
>>> print(inp[:20])
From stephen.marquar
```

Căutarea în fișiere

În practică, atunci când căutăm date într-un fișier, sunt ignorate de obicei cele mai multe informații, cu excepția celor care îndeplinesc o condiție.

De exemplu, dacă dorim să căutăm în fișier numai liniile care încep cu prefixul “From:”, putem folosi metoda startswith pentru a identifica acelel linii:

```
f = open('mbox-short.txt')
count = 0
for line in f:
        if line.startswith('From:'):
                print(line)
```

Rezultat:

From: stephen.marquard@uct.ac.za

From: louis@media.berkeley.edu

From: zqian@umich.edu

From: rjlowe@iupui.edu
...

In afisare apare o spatiere suplimentara intre linii datorita faptului ca in fisier fiecare linie contine un caracter special \n care face sa se treaca la un rand nou la apelul functiei print. Pentru a evita acest efect, putem utiliza metoda rstrip, care elimina spatiile de la dreapta unui string:

```
f = open('mbox-short.txt')
for line in f:
        line = line.rstrip()
        if line.startswith('From:'):
                print(line)
```

Rezultat:

From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: cwen@iupui.edu
...

Alegerea numelui fișierului de către utilizator

Nedorind sa editam codul de câte ori procesăm un fișier diferit, este mai util să întrebăm utilizatorul în legătură cu numele fișierului ce urmează a fi procesat.

Numele fișierului va fi citit folosind funcția input:

```
fname = input('Introduceți numele fisierului: ')
f = open(fname)
count = 0
for line in f:
        if line.startswith('Subject:'):
                count = count + 1
print('Au fost gasite', count, 'linii de Subiect in', fname)
```

Folosirea try, except și open

Pentru a evita situatiile in care utilizatorul greseste introducerea numelui fisierului, este recomandata folosirea structurii try/except pentru a semnala eroarea.

```
fname = input('Nume fisier: ')
try:
        f = open(fname)
except:
```

```
        print('Fisierul nu a fost gasit:', fname)
        exit()
count = 0
for line in f:
        if line.startswith('Subject:'):
                count = count + 1
print('Au fost gasite, count, 'linii de Subiect in', fname)
```

Funcția exit termina executia programului.

Scrierea în fișier

Pentru a putea scrie într-un fișier, acesta trebuie deschis folosind “w” ca și parametru:

```
>>> fout = open('output.txt', 'w')
>>> print(fout)
<_io.TextIOWrapper name='output.txt' mode='w' encoding='cp1252'>
```

Dacă fișierul există deja, deschiderea sa în modul scriere șterge datele vechi și începe de la început! Dacă fișierul nu există, este creat unul nou.

Metoda write aferentă fisierului scrie date în fișier și returnează numărul de caractere scrise.

```
fout = open('output.txt', 'w')

line1 = "Nu credeam sa-nvat a muri vreodata\n"

fout.write(line1)

line2 = 'Pururi tanar, infasurat in manta-mi\n'

fout.write(line2)

fout.close()
```

Pentru fisierele deschise pentru scriere, este recomandabil sa fie inchise explicit folosind metoda close().

Fisierele sunt salvate in drive si pot fi descarcate local manual sau folosind instructiunea

```
files.download('example.txt')
```

Exerciţii

1. Scrieti un program care citeste dintr-un fisier si afiseaza continutul linie cu linie folosind majuscule. Executia va arata astfel:

```
python shout.py
Enter a file name: mbox-short.txt
FROM STEPHEN.MARQUARD@UCT.AC.ZA SAT JAN 5 09:14:16 2008
RETURN-PATH: <POSTMASTER@COLLAB.SAKAIPROJECT.ORG>
RECEIVED: FROM MURDER (MAIL.UMICH.EDU [141.211.14.90])
BY FRANKENSTEIN.MAIL.UMICH.EDU (CYRUS V2.3.8) WITH LMTPA;
        SAT, 05 JAN 2008 09:14:16 -0500
```

2. Scrieti un program care citeste dintr-un fisier si cauta linii de forma:
   X-DSPAM-Confidence: 0.8475

Cand intalniti o linie ce incepe cu "**X-DSPAM-Confidence:"** prelucrati acea linie si extrageti valoarea reală. Numărați aceste linii și calculați suma valorilor pentru spam confidence. Când ați ajuns la finalul fișierului afișați media spam confidence.

Exemplu de output:

```
Enter the file name: mbox.txt
Average spam confidence: 0.894128046745

Enter the file name: mbox-short.txt
Average spam confidence: 0.750718518519
```

Testați codul folosind cele două fișiere de mai sus.