
ΠΕΡΙΕΧΟΜΕΝΑ:



Κώστας Μ.
Σμαραγδένιος Χορηγός Μαθήματος

Γρηγόρης Μ.
Σμαραγδένιος Χορηγός Μαθήματος

- **Συμβολοσειρά (string)** είναι μία **immutable ακολουθία (sequence) από χαρακτήρες**.
  - Αποθηκεύεται εσωτερικά σαν ένας πίνακας από bytes που κωδικοποιούν Unicode χαρακτήρες.
  - Όμως προσοχή, δεν υπάρχει τύπος δεδομένων "χαρακτήρας", το κάθε στοιχείο είναι μία συμβολοσειρά με ένα χαρακτήρα.
  - Δεν τροποποιείται (immutable).

- **Τρόποι Ορισμού Συμβολοσειράς**
  - Ορίζουμε μία συμβολοσειρά χρησιμοποιώντας κάποιο όνομα μεταβλητής και ενθέτοντας τους χαρακτήρες σε μονά ή διπλά εισαγωγικά.
    - Επιτρέπονται μονά εισαγωγικά στη συμβολοσειρά αν αυτή ορίζεται από διπλά εισαγωγικά και το αντίθετο.
  - Συμβολοσειρά πολλών γραμμών: Την ενθέτουμε μέσα σε τριπλά εισαγωγικά (μονά ή διπλά)

**Παράδειγμα 1: string.definition.py**

Στο παράδειγμα βλέπουμε τον ορισμό δύο λεξικών:

```
empty_str = ""
string = 'Hello World!'
string2 = "I said 'Hello'!"
mult_string = """...
Hello there!
 I am a multi-line string!
"""
print(string2)
print(mult_string)
```

- **Συμπεριφορά:**
  - Κοινή συμπεριφορά με όλες τις ακολουθίες (προσοχή στο indexing – όχι για τροποποίηση)
  - (βλ. και Μάθημα 6 – διαφ.3 – "Οι Πλειάδες ως Ακολουθίες"

**Παράδειγμα 2: string.as.sequence.py**

```
string = "Sample String"

print((string + " ") * 3)              # mult
print(string[1])                       # indexing
print(string[1:4] + string[-4:-1])  # slicing
print(len(string))                     # length
print(max("sample"))                   # max
print(min("String"))                   # min
print(string.index("am"))          # searching
print(string.count("S"))           # counting
# str[2] = "c"  not working..
new_str = string[:2] + "c" + string[3:]
print(new_str)
```

## Χαρακτήρες διαφυγής (Escape Characters)

Οι ακόλουθοι χαρακτήρες είναι ειδικοί και έχουν την εξής λειτουργικότητα:

| Χαρακτήρας | Εξήγηση |
|---|---|
| \" | Διπλό εισαγωγικό |
| \' | Μονό εισαγωγικό |
| \\ | Backslash |
| \n | Αλλαγή γραμμής |
| \r | Carriage Return |
| \t | Tab |
| \b | Backspace |
| \ooo | Οκταδική τιμή χαρακτήρα |
| \xhh | Δεκαεξαδική τιμή χαρακτήρα |

Το μήκος του tab είναι by default 4, και μπορούμε να το τροποποιήσουμε με τη μέθοδο expandtabs:

1. string.**expandtabs(n):** Ορίζει το μήκος του tab σε n

## Παράδειγμα 3: string.escape.characters.py

```
print("quotes: \", \'")
print("slashes: / \\")
print("tabs:\n\t|\t|a\t|aa\t|aaa\t|")
print("change line: \n newline \r carriage return")
print("Backspace: bb\baa\bc\b\baa\b")
```

## Παράδειγμα 4: string.tab.length.py

```
string = "tabs:\n\t|\t|a\t|aa\t|aaa\t|"
print(string)
new_string = string.expandtabs(5)
print(new_string)
```

## Άσκηση 1:

Ορίστε μία λίστα 3x3 με τυχαίους ακέραιους στο διάστημα 0...999
Έπειτα τυπώστε τον πίνακα στην εξής μορφή (προσέξτε τη χρήση των tabs)

```
+----+----+----+
|1   |99  |331 |
+----+----+----+
|312 |5   |12  |
+----+----+----+
|44  |991 |1   |
+----+----+----+
```

**Μετατροπές**

Η **str**() μετατρέπει οποιονδήποτε άλλο τύπο δεδομένων σε συμβολοσειρά.

Από συμβολοσειρά σε άλλο τύπο δεδομένων, ενδιαφέρον έχουν:

- Η μετατροπή σε λίστα (ή tuple) που χωρίζει τη συμβολοσειρά στους διαδοχικούς χαρακτήρες.
- Η μετατροπή σε σύνολο εμφανίζει κάθε χαρακτήρα ακριβώς μία φορά.
- Η μετατροπή σε Boolean. Κάθε συμβολοσειρά είναι True εκτός από την κενή που είναι False.

Η μέθοδος join μπορεί επίσης να χρησιμοποιηθεί για τη μετατροπή ενός τύπου δεδομένων που επιδέχεται επανάληψη σε συμβολοσειρα:

- string.**join**(iterable): Ενώνει όλα τα στοιχεία του iterable (π.χ. λίστα ή tuple) ενθέτωντας μεταξύ τους τη συμβολοσειρά string.
  - Ειδικά για λεξικά, κάνει συνένωση των κλειδιών.
  - Τα στοιχεία του iterable πρέπει να είναι συμβολοσειρές.

**Παράδειγμα 5: string.convert.py**

```
print("int: " + str(5))
print("float: " + str(5.1))
print("boolean: " + str(True))
print("list: " + str([1,2]))
print("tuple: " + str((1,2)))
print("set: " + str({1,2}))
print("dict: " + str({1:2,3:4}))

print(list("Find what you love and let it kill you."))

print(" ".join(["a","b","c"]))
print(", ".join({"a","b","c"}))
print(" # ".join({"a":1, "k":10}))
```

**Άσκηση 2:**

*"I guess the only time most people think about injustice is when it happens to them."* [exercise02.initial.py]

Βρείτε το πλήθος των εμφανίσεων κάθε γράμματος στο παραπάνω ρητό του Charles Bukowski (του οποίου ειναι όλα τα ρητά στο σημερινό μάθημα) και εκτυπώστε τους σε αλφαριθμητική σειρα

Οι πρώτες γραμμές της εκτύπωσης θα πρέπει να είναι:

: 15
I: 1
a:2

## Μορφοποίηση Συμβολοσειράς

Με τον όρο μορφοποίηση (format) συμβολοσειράς εννοούμε την ενσωμάτωση σε μια συμβολοσειρά των τιμών μεταβλητών

- Π.χ. όπως έχουμε δει με το + (π.χ. **string = "result: " + str(x)** )
- και τους νέους τρόπους που θα δούμε τώρα

## F-Strings

Νέα προσθήκη από την έκδοση 3.6 και έπειτα. Γράφουμε f, και μέσα στα εισαγωγικά μπορούμε μέσα σε άγκιστρα να βάλουμε μια υπολογιζόμενη έκφραση:

- Έτσι το προηγούμενο π.χ. γράφεται: **string = f"result: {x}"**

Επιπλέον υποστηρίζουν:

- F-strings πολλών γραμμών: Υποχρεωτικά σε παρενθέσεις και ο χαρακτήρας αλλαγής γραμμής να μπαίνει στο τέλος κάθε γραμμής
- Για πραγματικούς: {f} ή {f:a} ή {f:.b} ή {**f:a.b**}
  - f: Ένας πραγματικός αριθμός
  - a: Το πλήθος θέσεων που θα δεσμεύσει ο αριθμός (αν είναι λιγότερες, συμπληρώνονται με κενά)
  - b: Το πλήθος των ψηφίων που θα χρησιμοποιηθούν
- Για ακέραιους: {i} ή {i:x} ή {i:o} ή {i:e}
  - i: Ένας ακέραιος
  - x: Αποτύπωση ως δεκαεξαδικός
  - o: Αποτύπωση ως οκταδικός
  - e: Αποτύπωση σε επιστημονική μορφή

## Παράδειγμα 6: fstrings.py

```
print(f"With a result: {1+4}")
x = 3
print(f"For debugging: {x=}")
mult_line = (
  f"multiline: {x} value\n"
  f"multiline: {x*x} square"
)
print(mult_line)
print(f"A float with 2 decimals: {1/3:.2}|")
print(f"A float with width 6: {1/4:6}|")
print(f"A float with width 6 and 2 decimals: {1/3:6.2}|")
print(f"An integer(hexadecimal): {155:x}")
print(f"An integer(octal): {155:o}")
print(f"An integer(scientific): {155:e}")
```

## Άσκηση 3:

Τροποποιήστε το πρόγραμμα του προηγούμενου μαθήματος (Άσκηση 9 – Πέτρα Ψαλίδι Χαρτι), έτσι ώστε να χρησιμοποιεί f-strings.

- **Η συμβολοσειρά έχει (πάρα) πολλές μεθόδους:**
  - Η επεξεργασία κειμένου είναι από τις πιο συχνές προγραμματιστικές υποχρεώσεις
  - Προσοχή ότι όλες οι μέθοδοι δεν επηρεάζουν τη συμβολοσειρά, αλλά επιστρέφουν καινούργια συμβολοσειρά (immutable)

**Πακέτο 1: Μέθοδοι για Κεφαλαία – Μικρά**

1. string.**capitalize()**
   - Μετατρέπει μόνο τον πρώτο χαρακτήρα σε κεφαλαίο και τους υπόλοιπους σε μικρούς.
2. string.**lower()**
   - Μετατρέπει όλους τους χαρακτήρες σε μικρούς.
3. string.**casefold()**
   - Μετατρέπει όλους τους χαρακτήρες σε μικρούς (πιο ισχυρή, για «περίεργες» γλώσσες)
4. string.**upper()**
   - Μετατρέπει όλους τους χαρακτήρες σε κεφαλαίους.
5. string.**title()**
   - Μετατρέπει τον πρώτο χαρακτήρα κάθε λέξης σε κεφαλαίο.
6. string.**swapcase()**
   - Μετατρέπει κάθε μικρό σε κεφαλαίο και κάθε κεφαλαίο σε μικρό.

**Παράδειγμα 7: string.upper.lower.py**

```
string = "Burning in Water, Drowning in Flame"

print("capitalize: " + string.capitalize())
print("lower: " + string.lower())
print("casefold: " + string.casefold())
print("upper: " + string.upper())
print("title: " + string.title())
print("swapcase: " + string.swapcase())
```

**Άσκηση 4:**

*"How the hell could a person enjoy being awakened at 6:30AM, by an alarm clock, leap out of bed, dress, force-feed, shit, piss, brush teeth and hair, and fight traffic to get to a place where essentially you made lots of money for somebody else and were asked to be grateful for the opportunity to do so?"* (exercise04.initial.py)
Πραγματοποιήστε την ίδια μελέτη με την άσκηση 2, αλλά να προηγηθεί η μετατροπή σε μικρούς χαρακτήρες και να αφαιρεθεί από το αποτέλεσμα οτιδήποτε δεν είναι γράμμα.

**Πακέτο 2: Έλεγχος χαρακτήρων**

Οι παρακάτω μέθοδοι ελέγχουν αν όλοι χαρακτήρες έχουν μία κοινή ιδιότητα. Επιστρέφουν T/F

1. string.**isalnum():** Είναι αλφαριθμητικά (αριθμοί ή γράμματα)
2. string.**isalpha():** Είναι γράμματα
3. string.**isdigit():** Είναι ψηφίο (αριθμός)
   - Μικρές διαφορές με αυτήν σε ειδικές μορφές αριθμών (π.χ. λατινικοί αριθμοί, ιδεογράμματα κ.α.): **isdecimal()** και **isnumeric()**
4. string.**isspace():** Όλοι οι χαρακτήρες είναι κενά
5. string.**isprintable():** Οι χαρακτήρες είναι εκτυπώσιμοι
   - Εκτυπώσιμοι χαρακτήρες είναι τα γράμματα, το κενό, οι αριθμοί, τα σημεία στίξης. Δεν είναι εκτυπώσιμος π.χ. το \n ή το \t
6. string.**isidentifier():** Η συμβολοσειρά είναι identifier
   - identifier είναι έγκυρο όνομα μεταβλητής (ξεκινά με γράμμα, περιέχει μόνο αριθμούς, γράμματα και το underscore)
7. string.**islower():** Όλοι οι χαρακτήρες είναι μικροί
8. string.**isupper():** Όλοι οι χαρακτήρες είναι κεφαλαίοι
9. string.**istitle():** Έχει τη μορφή τίτλου

**Παρατήρηση:**

Κρατάμε το διαχωρισμό των χαρακτήρων σε: αριθμούς, γράμματα, σημεία στίξης, το κενό και τους ειδικούς χαρακτήρες.

**Παράδειγμα 8: string.character.checking.py**

```
print("alnum    : 123abc  : " + str("123abc".isalnum()))
print("alnum    : 123abc# : " + str("123abc#".isalnum()))
print("alpha    : abcDδΔ  : " + str("abcDδΔ".isalpha()))
print("alpha    : δαεςι2  : " + str("δαεςι2".isalpha()))
print("digit    : 123456  : " + str("123456".isdigit()))
print("digit    : 123a34  : " + str("123a34".isdigit()))
print("space    :         : " + str("    ".isspace()))
print("printable : aA12 #  : " + str("aA12 #".isprintable()))
print("printable : \\na    : " + str("\na   ".isprintable()))
print("identifier: is_var  : " + str("is_var".isidentifier()))
print("identifier: 12ar    : " + str("12ar".isidentifier()))
print("lower    : abca34  : " + str("abca34".islower()))
print("upper    : ABFa34  : " + str("ABFa34".isupper()))
print("title    : Tik Tok : " + str("Tik Tok".istitle()))
```

**Άσκηση 5:**

Παρατηρήστε ότι με την εντολή
x = int(input("Give an integer")
αν ο χρήστης πληκτρολογήσει κάτι που δεν είναι ακέραιος, προκαλείται σφάλμα.
Σκεφθείτε πως θα κάνετε έλεγχο της εισόδου, ώστε να εμφανίζεται μήνυμα στο χρήστη να ξαναεισάγει τον αριθμό, εφόσον δεν εισάγει ακέραιο αριθμό.

**Πακέτο 3: Διόρθωση Κενών**

Οι παρακάτω μέθοδοι διορθώνουν την ύπαρξη κενών (trimming) σε μια συμβολοσειρά.

1. string.**strip():** Αποκόπτει τα κενά στην αρχή και στο τέλος
2. string.**lstrip():** Αποκόπτει τα κενά στην αρχή
3. string.**rstrip():** Αποκόπτει τα κενά στο τέλος

**Πακέτο 4: Στοίχιση**

Οι παρακάτω μέθοδοι στοιχίζουν τη συμβολοσειρά αριστερά, δεξιά ή στο κέντρο ορίζοντας ένα μήκος το οποίο συμπληρώνεται με κενά.

1. string.**ljust(n):** Στοιχίζει αριστερά τη συμβολοσειρά συμπληρώνοντας με κενά μέχρι το μήκος n
2. string.**rjust(n):** Ομοίως για δεξιά στοίχιση
3. string.**center(n):** Ομοίως γα κεντράρισμα.

και οι 3 μέθοδοι μπορούν να δεχτούν 2ο όρισμα με ένα χαρακτήρα που γεμίζει το κενό (padding) που δεν είναι το κενό.

4. string.**zfill(n):** Γεμίζει με 0 από αριστερά τη συμβολοσειρά μέχρι η συμβολοσειρά να γίνει μήκους n

**Παράδειγμα 9: string.trimming.justifying.py**

```
string = "   Factotum "
print("lstrip : |" + string.lstrip() + "|")
print("rstrip : |" + string.rstrip() + "|")
print("strip  : |" + string.strip() + "|")
print("ljust  : |" + string.strip().ljust(30) + "|")
print("rjust  : |" + string.strip().rjust(30) + "|")
print("center : |" + string.strip().center(30,"-") + "|")

string = "abcd"
print("zfill(2) : " + string.zfill(2))
print("zfill(10): " + string.zfill(10))
```

**Άσκηση 6:**

Κατασκευάστε πρόγραμμα που θα διαβάζει από την είσοδο το όνομα και το επώνυμο του χρήστη.

Να γίνεται ένας τυπικός έλεγχος των συμβολοσειρών:

- Να γίνεται trimming
- Να διαπιστώνεται ότι αποτελείται μόνο από γράμματα
- Να διορθώνονται τα μικρά-κεφαλαία σε περίπτωση λάθος εισόδου.

Έπειτα να τυπώνεται σε ένα πλαίσιο πλάτους 30 χαρακτήρων, κεντραρισμένα, ένας χαιρετισμός προς το χρήστη, π.χ.:

```
+------------------------------+
|    Hello Dimitris Psounis!   |
+------------------------------+
```

**Πακέτο 5: Αναζήτηση και Αντικατάσταση**

Οι παρακάτω μέθοδοι αναζητούν μιας συμβολοσειρά μέσα σε μια άλλη.

1. string.**startswith(search,[start],[end]):** Επιστρέφει T/F αν η συμβολοσειρά ξεκινά με τη search (optional: στο μέρος από start έως end).
2. string.**endswith(search,[start],[end]):** Επιστρέφει T/F αν η συμβολοσειρά τελειώνει με τη search (optional: στο μέρος από start έως end).
3. string.**find(search,[start],[end]):** Επιστρέφει τη θέση (index) που βρήκε τη search (από το [start] στο [end]). -1 αν δεν το βρει.
   [Η string.**index** έχει την ίδια συμπεριφορά, αλλά αν δεν το βρει, επιστρέφει μία εξαίρεση (exception)]
4. string.**rfind(search,[start],[end]):** Επιστρέφει τη θέση (index) που βρήκε την τελευταία εμφάνιση της search (από το [start] στο [end]). -1 αν δεν το βρει.
   [Η string.**rindex** έχει την ίδια συμπεριφορά, αλλά αν δεν το βρει, επιστρέφει μία εξαίρεση (exception)]
5. string.**replace(old_value,new_value,[count]):** Αντικαθιστά τις [count] πρώτες εμφανίσεις της old_value με την new_value

**Παράδειγμα 10: string.searching.py**

```
string = "Some people never go crazy. What truly horrible
lives they must lead."
print("startswith: " + str(string.startswith("Some")))
print("startswith with bounds: " +
str(string.startswith("people", 5)))
print("startswith with bounds: " +
str(string.startswith("people", 5,8)))
print("endswith: " + str(string.endswith("people",0,11)))
print("find: " + str(string.find("crazy")))
print("position(s) of \"o\": ", end="")
pos = -1
lpos = string.rfind("o")
while pos != lpos:
    pos = string.find("o", pos+1)
    print(pos, end=" ")

print("\nreplace: " + string.replace("o","0",3))
```

**Άσκηση 7:**

*"I don't hate them...I just feel better when they're not around." (exercise07.initial.py)*

Κατασκευάστε πρόγραμμα που θα λαμβάνει επαναληπτικά είσοδο από το χρήστη μία λέξη, θα την εντοπίζει στο ρητό και θα τυπώνει το ρητό με τη λέξη σε κεφαλαία, εωσότου ο χρήστης γράψει "quit"

**Πακέτο 6: Χωρισμός Συμβολοσειράς σε Μέρη**

Οι παρακάτω μέθοδοι χωρίζουν μια συμβολοσειρά σε μέρη:

1. string.**split([separator],[maxsplit]):** Επιστρέφει μία λίστα με τις υπο-συμβολοσειρες(substrings) της string χωρισμένες με βάση το κενό (ή τον separator). Επιστρέφει το πολύ maxsplit χωρισμούς.
2. string.**rsplit([separator],[maxsplit]):** Όμοια με τη split, αλλά ο χωρισμός ξεκινά από τα δεξιά.
3. string.**partition(middle):** Χωρίζει τη συμβολοσειρά σε 3 μέρη και επιστρέφει μία λίστα με αυτά τα μέρη: Αριστερά της middle, η middle και δεξιά της middle. Αν υπάρχουν περισσότερες από μία εμφανίσεις, ο χωρισμός γίνεται με βάση την πρώτη της εμφάνιση.
4. string.**rpartition(middle):** Όμοια με τη partition, αλλά ο χωρισμός γίνεται με βάση την τελευταία εμφάνιση της middle.
5. string.**splitlines([inclusion]):** Χωρίζει τη συμβολοσειρά σε μέρη με βάση τους χαρακτήρες αλλαγής γραμμής. Αν το [inclusion] τεθεί True, ο χαρακτήρας αλλαγής γραμμής ενσωματώνεται στις συμβολοσειρές.

**Παράδειγμα 11: string.partitioning.py**

```
string = "You have to die a few times before you can really live."
print("split(by blanks): " + str(string.split()))
print("split(by a's): " + str(string.split("a")))
print("split(by blanks): " + str(string.split(" ", 2)))
print("rsplit(by blanks): " + str(string.rsplit(" ", 2)))
print("partition(by 'few'): " + str(string.partition("few")))
print("rpartition(by 'r'): " + str(string.rpartition("r")))

mult_line = """Some lose all mind and become soul, insane.
some lose all soul and become mind, intellectual.
some lose both and become accepted"""
print("splitlines: " + str(mult_line.splitlines()))
```

**Άσκηση 8:**

Στο (exercise08.initial.py) θα βρείτε ένα ποίημα του Bukowski. Κατασκευάστε πρόγραμμα που θα λαμβάνει από το χρήστη μία λέξη από την είσοδο. Έπειτα θα αναζητά σε ποιες γραμμές του ποιήματος εμφανίζεται η λέξη και θα τυπώνει τον αριθμό κάθε γραμμής και το περιεχόμενό της, θέτοντας τη λέξη σε κεφαλαία γράμματα. Π.χ.:

```
Give the word: too
Line 6: but I'm TOO tough for him,
```

Εκτός από τα F-strings υπάρχουν και άλλοι τρόποι για να **μορφοποιήσουμε μια συμβολοσειρά**. Πλέον θεωρούνται παρωχημένοι, αλλά είναι καλό να έχουμε μια ιδέα γι' αυτούς:

ΠΟΛΥ ΠΑΛΙΟΣ ΤΡΟΠΟΣ: **Μορφοποίηση α λά C** (string.format.c.py)

```
print("int value: %d %d" % (1, 5+1))
print("int width(right align): %10d " % 1)
print("int width(left align): %-10d " % 1)
print("float: %f" % (1/3))
print("float: decimals: %.4f %.2f" % (1/3, 1/8))
print("string: %s" % "Hello there! ")
```

ΠΑΛΙΟΣ ΤΡΟΠΟΣ: **Η μέθοδος format()** (string.format.py)

```
print("int value: {} {}".format(1, 5+1))
print("int width(right align): {:>10d} ".format(1))
print("int width(left align): {:<10d} ".format(1))
print("float: {}".format(1/3))
print("float: decimals: {:.4f} {:.2f}".format(1/3, 1/8))
print("string: {}".format("Hello there! "))
```

**Σημείωση:**

Η format έχει πολλές ακόμη χρήσεις για μορφοποιημένη έξοδο, που ξεφεύγουν από τα όρια της παρουσίασης.

**Άσκηση 9:**

Ορίστε έναν ακέραιο, έναν πραγματικό και μία συμβολοσειρά. Έπειτα, τυπώστε και τα τρία σε μία γραμμή με 4 τρόπους:

- Με string concatenation (+)
- Με f-strings
- Με μορφοποίηση α λά C
- Με τη μέθοδο format

**Τελεστές** που είναι υπερφορτωμένοι ώστε να δουλεύουν με συμβολοσειρές (string.operators.py):

```
string = "Hello "
concatenation = string + string
print(f"Concatenation(+): {concatenation}")
multiplication = string * 5
print(f"Multiplication(*): {multiplication}")
string += "World!"
print(f"Increment: {string}")
print(f"Char in string: 'W' in {string}: {'W' in string}")
print(f"Char not in string: 'W' not in {string}: {'W' not in string}")
comparison = "abc" < "abd" # also: >, <=, >=
print(f"Comparison: {comparison}")
equality = "aa" == "AA".lower() # also not equal: !=
print(f"Equality: {equality}")
```

Παρατήρηση: Ο τελεστής += είναι μια συντομογραφία της έκφρασης: string = string + "World!"

- **Ένας χαρακτήρας αναπαρίσταται εσωτερικά (στη μνήμη) από έναν αριθμό.**
- Έπειτα ο αριθμός "μεταφράζεται" σε χαρακτήρα και π.χ. εκτυπώνεται.
- Μπορούμε να δούμε τον αριθμό που αντιστοιχεί στον χαρακτήρα, χρησιμοποιώντας τη συνάρτηση ord. Πχ η εκτύπωση:

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

  - θα εκτυπώσει το 97, που είναι ο ακέραιος που αντιστοιχεί στο a.
- Αντίστοιχα η συνάρτηση chr, επιστρέφει το χαρακτήρα που αντιστοιχεί σε έναν αριθμό. Π.χ. η εκτύπωση:

```
print(chr(100))
```

  - θα εκτυπώσει το d, που είναι ο χαρακτήρας που αντιστοιχεί στο 100.
- Από τους πρώτους πίνακες αντιστοίχισης αριθμών – χαρακτήρων ήταν ο ASCII ο οποίος αποθήκευε 128 χαρακτήρες
- Δείτε τους πρώτους 128 χαρακτήρες στην κωδικοποίηση της Python (python.encoding.128.py)

```
for i in range(128):
    if i%10==0:
        print(f"\n{i}-{i+10}: ", end="")
    print(chr(i), end="")
```

- Σήμερα, το πρότυπο **Unicode**, είναι ένας τεράστιος πίνακας αντιστοίχισης που περιέχει >130.000 χαρακτήρες (και επεκτείνεται συνεχώς με νέα σύμβολα) και περιλαμβάνει όλα τα συστήματα γραφής που χρησιμοποιούνται σήμερα.
- Ένας χαρακτήρας του Unicode αποθηκεύεται στη μνήμη με βάση κάποια **κωδικοποίηση**.
  - π.χ. η **UTF-8** μετατρέπει τον αριθμό του χαρακτήρα σε bytes και **ένας χαρακτήρας μπορεί να πιάνει 1,2,3, ή 4** bytes (και είναι ο τρόπος που χρησιμοποιεί η Python)
  - Υπάρχουν και άλλες κωδικοποίησεις όπως η UTF-16 και η UTF-32 (η οποία χρησιμοποιεί σταθερό μήκος).
- Και η μέθοδος που μπορεί να μετατρέψει ένα string σε μια κωδικοποίηση είναι η encode (βλ. python.encoding.py):

```
print("a".encode("ascii"))
print("a".encode("utf-8"))
print("α".encode("utf-8"))
print("麦".encode("utf-8"))
print("a".encode("utf-32"))
print("α".encode("utf-32"))
print("麦".encode("utf-32"))
```

**Άσκηση 10:**
Εκτυπώστε τους πρώτους 10.000 χαρακτήρες σε ομάδες (γραμμές) των 50 χαρακτήρων