Libreria standard: `math`, `os`, `os.path` e `re`

Rachele Sprugnoli

rachele.sprugnoli@unipr.it

UNIVERSITÀ
DI PARMA

Libreria

- Raccolta di codice scritto da altri e pronto per essere utilizzato
  - LIBRERIA STANDARD: inserita di default negli ambienti di sviluppo, fornita con ogni installazione di Python

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

    (abbiamo già visto `csv`)
  - LIBRERIE ESTERNE: sviluppate per gestire problemi specifici, es. gestione file csv, analisi linguistica, creazione grafici
- È organizzata in MODULI che contengono FUNZIONI
- Devono essere esplicitamente importate nel programma prima di essere utilizzate

Importare moduli

- Chiamare il modulo tramite il comando → `import nome_libreria`

```
import math
```

- Importare solo specifiche classi, variabili e funzioni del modulo → `from nome_modulo import nome_oggetto`

```
from math import sqrt, exp
```

- Richiamare tutte le funzioni, le classi e le variabili incluse nel modulo tramite l'asterisco → `from nome_libreria import *`

```
from math import *
```

Libreria standard: os()

- Contiene funzioni per interagire con il sistema operativo, ad esempio per navigare nelle cartelle
- Percorso (*path*): individua la collocazione di un file o di una cartella
  - Percorso assoluto: `/home/testi/text.txt`
  - Percorso relativo: `text.txt`

  https://docs.python.org/3/library/os.html
  - `os.getcwd()` → restituisce la cartella di lavoro
  - `os.chdir(nome_cartella)` → cambia cartella di lavoro
  - `os.listdir(path)` → restituisce una lista di file e cartelle
  - `os.rename(sorgente,destinazione)` → rinomina file sorgente in destinazione

Libreria standard: os.path()

- Contiene funzioni per manipolare i nomi dei percorsi (*pathname*)

  https://docs.python.org/3/library/os.path.html

  - `os.path.join(percorso,nome)` → restituisce una stringa ottenuta anteponendo il percorso al nome
  - `os.path.exists(nome)` → restituisce un valore booleano (True/False) che indica se esiste o meno la cartella o il file di nome `nome`

Libreria standard: math()

- Da usare per eseguire operazioni matematiche; la maggioranza delle sue funzioni restituisce valori di tipo float
  https://docs.python.org/3/library/math.html
  - `math.sqrt(x)` → radice quadrata di x
  - `math.pow(x, y)` → x elevamento a potenza di y
  - `math.ceil(x)` → arrotondamento per eccesso di x a un intero
  - `math.floor(x)` → arrotondamento per difetto di x a un intero
  - `math.sin(x)` → seno di x
  - `math.cos(x)` → coseno di x
  - `math.tan(x)` → tangente di x

Libreria standard: re()

- Modulo per compiere vari tipi di operazioni con le espressioni regolari (re è l'acronimo di *regular expression*)

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

- RE = linguaggio formale per definire stringhe di testo
  - Tutorial interattivo: https://www.w3schools.com/python/python_regex.asp
  - Piattaforma di test online: https://regexr.com/

Libreria standard: re()

- METACARATTERI (caratteri con un significato speciale)

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"planet$"
*	Zero or more occurrences	"he.*o"
+	One or more occurrences	"he.+o"
?	Zero or one occurrences	"he.?o"
{}	Exactly the specified number of occurrences	"he.{2}o"
|	Either or	"falls|stays"
()	Capture and group	

Libreria standard: re()

- SEQUENZE SPECIALI (lettere preceduti dal carattere di *escape*)

N.B.

- \t → tabulazione
- \n → a capo

Character	Description	Example
\A	Returns a match if the specified characters are at the beginning of the string	"\AThe"
\b	Returns a match where the specified characters are at the beginning or at the end of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\bain" r"ain\b"
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\Bain" r"ain\B"
\d	Returns a match where the string contains digits (numbers from 0-9)	"\d"
\D	Returns a match where the string DOES NOT contain digits	"\D"
\s	Returns a match where the string contains a white space character	"\s"
\S	Returns a match where the string DOES NOT contain a white space character	"\S"
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	"\w"
\W	Returns a match where the string DOES NOT contain any word characters	"\W"
\Z	Returns a match if the specified characters are at the end of the string	"Spain\Z"

Libreria standard: re()

- Uso delle parentesi quadre per indicare gruppi di caratteri
  - [a-z] tutte le lettere minuscole
  - [A-Z] tutte le lettere maiuscole

Set	Description
[arn]	Returns a match where one of the specified characters (`a` , `r` , or `n`) is present
[a-n]	Returns a match for any lower case character, alphabetically between `a` and `n`
[^arn]	Returns a match for any character EXCEPT `a` , `r` , and `n`
[0123]	Returns a match where any of the specified digits (`0` , `1` , `2` , or `3`) are present
[0-9]	Returns a match for any digit between `0` and `9`
[0-5][0-9]	Returns a match for any two-digit numbers from `00` and `59`
[a-zA-Z]	Returns a match for any character alphabetically between `a` and `z` , lower case OR upper case
[+]	In sets, `+` , `*` , `.` , `|` , `()` , `$` , `{}` has no special meaning, so `[+]` means: return a match for any `+` character in the string

Libreria standard: re() - funzioni

- `re.search(pattern, string, flag)` → cerca la stringa `pattern` all'interno della stringa `string` e restituisce la prima occorrenza trovata e lo `span` con gli indici (da zero) della posizione iniziale e finale della corrispondenza nella stringa

  N.B. Un **flag** è un'opzione aggiuntiva, ad es.:

  - `re.IGNORECASE`: esegue la ricerca senza considerare maiuscole e minuscole
  - `re.MULTILINE`: esegue la ricerca considerando inizio e fine delle singole righe di una stringa posta su più righe

Libreria standard: re() - funzioni

- ESEMPIO:

```
import re
pattern = "signor.?"
string = """\"Signor Marchese! se volete diventare mio
genero, non sta che a voi\".
Il marchese, con mille reverenze, gradì l'alto onore
fattogli dal Re, e il giorno dopo sposò la Principessa.
Il gatto diventò gran signore, e se seguitò a dar la
caccia ai topi, lo fece unicamente per passatempo."""
re.search(pattern, string)
```

→ signore

Libreria standard: re() - funzioni

- ESEMPIO:

```
import re
pattern = "signor.?"
string = """\"Signor Marchese! se volete diventare mio
genero, non sta che a voi\".
Il marchese, con mille reverenze, gradì l'alto onore
fattogli dal Re, e il giorno dopo sposò la Principessa.
Il gatto diventò gran signore, e se seguitò a dar la
caccia ai topi, lo fece unicamente per passatempo."""
re.search(pattern, string, flags = re.IGNORECASE)
```

→ Signor

Libreria standard: re() - funzioni

- `re.findall(pattern, string, flag)` → cerca la stringa `pattern` all'interno della stringa `string` e restituisce una lista di stringhe → trova TUTTE le occorrenze
- `re.sub(pattern, repl, string, count, flag)` → sostituisce le occorrenze della stringa `pattern` all'interno della stringa `string` con la stringa `repl`; l'argomento `count` indica il numero massimo di occorrenze del `pattern` da sostituire → 0 = TUTTE

Libreria standard: re() - funzioni

- `re.split(pattern, string, maxsplit, flags)` → restituisce una lista in cui la stringa `string` è stata divisa a ogni corrispondenza del `pattern`
- `re.compile(pattern,flag)` → compila un pattern salvandolo per essere utilizzato in seguito

```
import re
string = "Elenco autori dal 1500 al 1600:"
pattern = re.compile(r"\d{4}")
result = pattern.findall(string)
print(result)
```

Un po’ di pratica

- Lezione3.ipynb

  https://colab.research.google.com/drive/1IGgcV2QDp2S320uYCtHFDJ06TCT7a2x7?usp=sharing

Esercizio 1

- Sostituisci la parola “mille” con 1.000 nel testo “Dammi mille baci, poi cento poi altri mille, poi ancora cento poi altri mille, poi cento ancora.”

Esercizio 2

- Sostituisci solo la prima occorrenza della parola “mille” con 1.000 nel testo “Dammi mille baci, poi cento poi altri mille, poi ancora cento poi altri mille, poi cento ancora.”

Esercizio 3

- Sostituisci “mille” con “1.000” e “cento” con “100” nel testo “Dammi mille baci, poi cento poi altri mille, poi ancora cento poi altri mille, poi cento ancora.”

Esercizio 4

- Filtrare dalla lista seguente sole le parole che iniziano con una maiuscola: Giove, Marte, gratia, uscio, varco, Amor
  - Come fare a filtrare le parole che NON iniziano con una maiuscola?

Esercizio 5

- Trovare tutte le occorrenze di “io” come parola singola nel testo:
  Et io, da che comincia la bella alba
  a scuoter l' ombra intorno de la terra
  svegliando gli animali in ogni selva,
  non ò mai triegua di sospir' col sole;
  poi quand' io veggio fiammeggiar le stelle
  vo lagrimando, et disïando il giorno.

Esercizio 6

- Scrivere uno script che riconosca in un testo le date in formato DD/MM/AAAA, poi stampare solo l'anno

Soluzioni esercizi

- https://colab.research.google.com/drive/1s7YXYhxc6lCwU_d1TJd-5Eih0K0DLwfA?usp=sharing