RAGAZZE DIGITALI ER

Digital Girl Power

Cosa faremo oggi

Costruiremo un programma che sarà in grado di convertire un normale testo in un codice segreto e viceversa.

Caesar Cipher

Vuoi criptare o decriptare un messaggio?
criptare
Scrivi il tuo messaggio:
Questo è il corso Ragazze digitali, idee per un futuro smart
Inserisci la chiave (1-52)
13
Ecco il testo criptato:
dHrFGB zèz zvy zpBEFB zentnMMr zqvtvGnyv,z zvqrr zCrE zHA zsHGHEB zFznEG

Outline

Crittografia 1/3

Vediamo di imparare qualche nozione elementare di crittografia che ci può essere utile per scrivere il nostro programma

Per iniziare

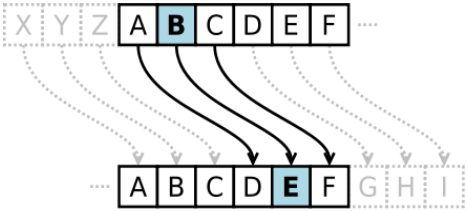
- **cipher** = è il sistema, l'insieme delle regole secondo le quali *criptiamo* un messaggio
- **plaintext** = testo che vogliamo nascondere e mantenere segreto
- **ciphertext** = testo trasformato
- un *plaintext* viene **criptato** in un *ciphertext*
- un *ciphertext* viene **decriptato** in un *plaintext*
- **chiave** = valore segreto con il quale si decripta un messaggio criptato usando un determinato **cipher**

Crittografia 2/3

Esistono tantissimi cipher, sistemi per crittografare un messaggio, ciascuno con la propria tipologia di chiavi. Per costruire il nostro programma ci interesseremo al **Caeser cipher**, ovvero del *cifrario di Cesare*... Si, proprio quel Cesare che pensate! È parecchio vecchio come cipher, ma tutt'ora ancora perfettamente funzionante.

Crittografia 3/3

Con questo Caeser cipher, i messaggi vengono criptati rimpiazzando ciascuna lettera con una lettera *"shiftata"*, ovvero spostata.
Per esempio, se *shiftiamo* di 3 lettere, avremo che la lettera **B** diventerà una **E**, e così via. Per decriptare i messaggi, verranno shiftate indietro le lettere e quindi una **E** diventerà una **B**.



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

La **chiave** del Caeser cipher è il numero di lettere shiftate.

Outline

Metodo `find()`

Come funziona

- È un metodo che restituisce la posizione in cui si trova la stringa passata al metodo rispetto alla stringa su cui è invocato il metodo.
- Proviamo a vedere con degli esempi come funziona

```
>'Hello world'.find('H')
0
>'Hello world'.find('o')
4
>'Hello world'.find('ell')
1
>'Hello world'.find('xyz')
-1
```

- Gli indici partono da 0 e non da 1!
- Di 'o' ce ne sono due, ma viene restituito l'indice della prima occorrenza trovata
- Se si ricerca una stringa, viene restituito l'indice dell'inizio della stringa
- Se si cerca una stringa non presente, viene restituito -1

Outline

Funzione `len()`

Come funziona sulle stringhe?

- Restituisce il numero di caratteri presenti nella stringa passata in input
- Proviamo a vedere con degli esempi come funziona

```
1 SYMBOLS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
2 MAX_KEY_SIZE = len(SYMBOLS) # restituisce 26
```

- In questo esempio, la costante `SYMBOLS` contiene una stringa rappresentante tutte le lettere dell'alfabeto.
- La costante `MAX_KEY_SIZE`, invece, contiene il risultato della chiamata alla funzione `len()` invocata con parametro la stringa di cui vogliamo calcolare la lunghezza

Outline

Funzione `bool()`

Come funziona

- È una funzione simile a `str()` e `int()` visti in precedenza; restituisce `True` o `False` in base a cosa viene passato in input.
  Ogni data types ha un valore che viene considerato *falso* mentre tutti gli altri sono considerati come *veri*.
- Copiamo una riga alla volta premendo poi invio nella console e vediamo cosa accade:

```
1 >>> bool(0)
2 >>> bool(0.0)
3 >>> bool('')
4 >>> bool([])
5 >>> bool(1)
6 >>> bool('Hello')
7 >>> bool([1, 2, 3, 4, 5])
```

Outline

Caeser cipher 1/5

Ora tocca a voi!

- Definisci innanzitutto due costanti, una contenente tutte le lettere dell'alfabeto (minuscole e maiuscole) e un'altra che contenga il numero delle lettere definite in precedenza
- Costruisci una funzione che chieda all'utente se vuole criptare o decriptare un messaggio e che restituisca la modalità scelta dall'utente; altrimenti, se l'utente inserisce un carattere o una stringa non inerente alla scelta, viene mostrato un messaggio di spiegazione su cosa bisogna inserire e viene riproposta la domanda iniziale.
- **Suggerimento**: per controllare cosa inserisce l'utente, può essere di aiuto convertire l'input in caratteri *lowercase* tramite il metodo delle stringhe `lower()`

Caeser cipher 2/5

Ora tocca a voi!

- Costruisci una funzione che chieda all'utente di inserire il messaggio che vuole criptare/decriptare e lo restituisca come valore di ritorno della funzione.
- Costruisci una funzione che chieda all'utente di inserire la chiave di cifratura, che sarà un numero compreso tra 1 e il numero delle lettere definite all'inizio. Controllare che il valore inserito sia all'interno di questo range. Nel caso non lo fosse, deve essere richiesto all'utente di inserire la chiave; se è dentro al range, viene restituito dalla funzione.

Caeser cipher 3/5

Ora tocca a voi!

- Costruisci una funzione che effettivamente cripta o decripta il messaggio in base a cosa è stato scelto dall'utente e alla chiave scelta.
- Creiamo una stringa vuota che sarà il risultato da restituire.
- Per ogni simbolo (carattere) del nostro messaggio, se il carattere è presente nella nostra "lista" caratteri (ovvero vuol dire che appartiene all'alfabeto), dobbiamo trovare l'indice del nuovo carattere da sostituire, in base alla chiave scelta.
- Una volta trovato il nuovo indice, inseriamo il carattere sostitutivo nella stringa del risultato.
- Se il carattere non è presente nel nostro alfabeto, copiamo semplicemente il vecchio carattere senza sostituirlo.

Caeser cipher 4/5

Ora tocca a voi!

- In questa ultima funzione per prima cosa bisogna verificare se è stata scelta la modalità decriptazione: con essa, infatti, è necessario rendere negativa la chiave, così che nella fase di sostituzione del carattere esso venga sostituito con il corrispettivo simbolo.
- Inseriamo quindi questo pezzetto di codice:

```
1 if mode[0] == 'd':
2     key = -key
3 translated = '' # la stringa del risultato, inizialmente vuota
```

Caeser cipher 5/5

Ora tocca a voi!

- Come ultima cosa richiamate le funzioni, salvate i valori restituiti e stampate la stringa criptata o decriptata!
- Ed ecco, il nostro cifrario è completato!

Soluzione

Outline

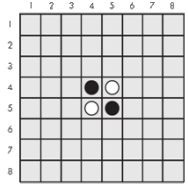
Othello game 1/6

Conosciamo già tutti gli strumenti per poter creare l'**Othello game**, o Reversegam game. A questo link troverete il codice sorgente nel quale mancano delle parti, che inserirete seguendo le indicazioni delle prossime slide.

Othello game 2/6 - Come funziona

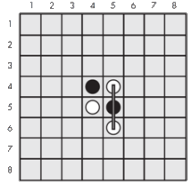
Il gioco si svolge su una griglia 8x8 e due giocatori con due pedine diverse (nel nostro caso: $\boldsymbol{x}$ e $\boldsymbol{o}$)

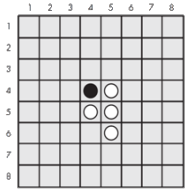
Si parte da una situazione iniziale di questo tipo:



L'obiettivo del gioco è conquistare più pedine del proprio colore. Come fare?

Inserendo una pedina del proprio colore in modo da circondare la pedina avversaria, la si conquisterà.





Conquistata la pedina, la situazione successiva sarà quella appena mostrata. Si possono conquistare pedine anche in diagonale!

Othello game 3/6

Ora tocca a te!

- #**1** Crea due variabili globali, una per l'altezza e una per la larghezza, che servono a definire la grandezza della griglia.
- #**2** All'interno della funzione `isValidMove()` inserire un controllo per cui, se la pedina (`tile`) è *'X'*, allora l'altra pedina (che potete chiamare, ad esempio `othertile`) sarà *'O'*. Altrimenti, il contrario.
- #**3** Crea una funzione chiamandola `getBoardCopy(board)`, che, presa in ingresso una griglia (`board`), ne crei una copia e la restituisca.

Othello game 4/6

Ora tocca a te!

- #**4** Creare un meccanismo per cui si scorre tutta la griglia e si incrementano le variabili `xscore` e `oscore` in base al numero di pedine presenti.
- #**5** Completare la funzione `whoGoesFirst()` in modo che, dato un numero casuale (*0* o *1*), se questo è 0, il turno sarà del computer e quindi verrà restituita una stringa *'computer'*, altrimenti si restituisca la stringa *'player'*
- #**6** Creare un meccanismo, all'interno della funzione `getPlayerMove()` tale per cui se la mossa fatta (contenuta nella variabile `move`) è *'quit'* o `hints`, venga restituita la mossa stessa.

Othello game 5/6

Ora tocca a te!

- #**7** Creare un meccanismo, all'interno della funzione `getPlayerMove()` tale per cui se la mossa fatta contiene 2 numeri e sia il primo che il secondo numero sono contenuti all'interno del vettore `DIGITS1TO8`, allora il primo numero venga inserito in una variabile chiamata `x` e il secondo in a chiamata `y`. Da qui, se la mossa **non** è valida si stampi un errore e su continui il ciclo, altrimenti si esca.
  Se la mossa non è composta da due numeri o composta da numeri al di fuori del range, si stampino dei messaggi per aiutare l'utente a inserire numeri validi.
- #**8** Completare la funzione `printScore` in modo che, vengano recuperati i punteggi della partita dalla funzione che si occupa di calcolarli, e li stampi con un messaggio.

Othello game 6/6

Ora tocca a te!

- #**9** Creare una nuova griglia (*board*) nella quale inserire la configurazione iniziale delle pedine (come nella prima figura mostrata alla slide **22**)
- #**10** Inserire un meccanismo per cui, se lo score del computer è maggiore di quello dell'utente, viene stampato un messaggio di perdita stampando il punteggio.
  Se, invece, l'utente ha ottenuto un punteggio superiore del computer, viene stampato un messaggio di vittoria.
  Se, altrimenti, si è ottenuto un punteggio pari, si stampi un altri messaggio inerente.

Soluzione

Materiale rilasciato con licenza

Creative Commons - Attributions, Share-alike 4.0