EPICODE

Programmazione in Python
Materiale Extra
Giorno 2

EPICODE

Extra - Funzioni per i costrutti `for`

- Funzione **`range()`** per eseguire un ciclo uno specifico numero di volte
- Sintassi: **`range(`***`start`*`, `*`stop`*`, `*`[step]`***`)`**

```
for numero in range(10):
    print(numero)
```

- Cosa produce **`range()`**?

```
range(15)  # ?
list(range(15))
list(range(3, 15))
list(range(2, 15, 2))
list(range(5, 0, -1))
```

- Funzione **`enumerate()`** per contare gli elementi di un iterabile durante un ciclo
- Sintassi: **`enumerate(`***iterabile*`)`**

```
libri = ["enciclopedia", "atlante", "dizionario"]

for libro in libri:
    print(libro)

for contatore, libro in enumerate(libri):
    print(contatore, libro)
```

- Funzione **`zip()`** per iterare su più iterabili contemporaneamente
- Sintassi: **`zip(`***iterabile1*`,` *iterabile2*`,` *[iterabile3]*`,` *[…]*`)`

```
alunni = ["Lucio", "Silvio", "Michela", "Natalia"]
corsi = ["ingegneria", "medicina",
     "cinema", "giurisprudenza"]
anni = [19, 22, 25, 21]
for alunno, corso, eta in zip(alunni, corsi, anni):
    print(alunno, "frequenta", corso,
          "e ha", eta, "anni.\n")
```

EPICODE

Extra – `break` / `continue`

- Può capitare di poter stabilire il momento in cui è necessario terminare un ciclo solo mentre il flusso di esecuzione si trova nel bel mezzo del corpo
- In questi casi possiamo usare l'istruzione **`break`** per interrompere il ciclo e uscire da esso

- Esempio, supponiamo di voler ricevere dei numeri dall'utente, finché non scriverà un numero maggiore di 100
- Useremo l'istruzione **`break`** in un costrutto **`if`**

```
numeri = []  # inizializzazione
while True:  # ciclo infinito! Dovremo contare sul 'break' per uscire
    numero = int(input("Inserisci un numero"))
    if numero > 100:
        break
    else:
        numeri.append(numero)
print("Hai inserito i numeri:", numeri)
```

- La condizione del ciclo è `True`, che è sempre vera per definizione, quindi il ciclo è destinato a continuare, a meno che non incontri l'istruzione `break`
- Se viene eseguita, l'istruzione `break` interrompe il ciclo, altrimenti il programma ripete quello che l'utente ha scritto e ritorna da capo
- Questo modo di scrivere i cicli `while` è frequente, perché vi permette di controllare la condizione ovunque all'interno del ciclo (e non solo all'inizio) e di esprimere la condizione di stop in modo affermativo ("fermati quando succede questo") invece che negativo ("continua fino a quando non succede questo")

- Esempio, supponiamo di voler ricevere dei valori dall'utente, finché non scrive la parola "finito"
- Useremo l'istruzione **`break`** in un costrutto **`if`**

```
amici = []  # inizializzazione
while True:  # ciclo infinito! Dovremo contare sul 'break' per uscire
     dato = input("Inserisci il nome di un tuo amico!
                Per terminare, scrivi 'finito'")
     if dato == "finito":
          break
     else:
          amici.append(dato)
if len(amici) == 0:
     print("Oh no! Non hai amici!")
else:
     print("I tuoi amici sono:", amici)
```

- Può capitare di voler saltare una determinata iterazione, ma senza interrompere il ciclo, controllando quindi il flusso di esecuzione per procedere con l'iterazione successiva
- Per svolgere questa operazione si usa la parola chiave `continue`

- Esempio, visualizziamo i numeri da 1 a 20, ma solo quelli dispari
- Useremo l'istruzione **`continue`** in un costrutto **`if`**

```
numero = 0  # inizializzazione
while numero < 20:
    numero += 1
    if numero % 2 == 0:
        continue
    else:
        print(num)
```

- Domande per lo studente: perché stavolta l'incremento (in questo caso della variabile **`numero`**) è all'inizio? Che succede se lo mettiamo in fondo al corpo?

- Come per il costrutto **`while`**, anche con il **`for`** è possibile utilizzare i comandi **`break`** e **`continue`**
- L'uso più comune del **`break`** in un ciclo **`for`** è quando si deve effettuare una ricerca in un iterabile: in tal caso, una volta trovato ciò che si cercava, è inutile continuare a ciclare sui dati rimanenti
- Esempio pratico: abbiamo una lista di proventi in una lista, in corrispondenza posizionale con delle date di un'altra lista; c'è stato un mese di Agosto in cui i proventi hanno superato i 400 €?

```
proventi = [100, 200, 300, 400, 100, 500, 600, 400, 300, 100, 100]
date = ["Luglio 2021", "Agosto 2023", "Maggio 2020", "Agosto 2021",
    "Marzo 2023", "Agosto 2022", "Gennaio 2021", "Aprile 2023",
"Agosto 2020", "Marzo 2022", "Gennaio 2022"]
for cifra, data in zip(proventi, date):
    if cifra > 400 and "Agosto" in data:
        print("Ad", data, "i proventi sono stati", cifra, "€")
        break
```

EPICODE

GRAZIE

Epicode