LINGUAGGIO PYTHON

ELEMENTI DI GESTIONE DEGLI ERRORI
STRUTTURE DI CONTROLLO

SISTEMI INFORMATIVI E DBMS

CORSO DI LAUREA MAGISTRALE IN
MANAGEMENT E MONITORAGGIO DEL TURISMO SOSTENIBILE

PROF. ANDREA PINNA

AA 2020/2021

GESTIONE ERRORI

Abbiamo visto che se l'interprete non riesce ad eseguire una istruzione viene generato un errore. Gli errori interrompono l'esecuzione del programma.

Il programmatore può prevedere dove il programma potrebbe generare un errore, stabilire cosa fare in caso di errore ed evitare l'interruzione dell'esecuzione.

Questa pratica si chiama gestione degli errori o gestione delle eccezioni

GESTIONE ERRORI

Costrutto base.
Il modo più semplice per gestire l'errore è usare il costrutto try – except.
Sintassi:

```
try:
   #istruzioni che potrebbero dare errore
except:
   #istruzioni da eseguire in caso di errore
```

GESTIONE ERRORI

Esempio: provo a stampare la variabile a senza che sia definita.

```
try:
   print(a)
except:
   print("qualcosa non ha funzionato")
```

ESERCIZIO DI RIEPILOGO

Scrivere un programma che chiede all'utente di inserire un numero.

Il programma elabora l'input e stampa a video il numero elevato 3 ed il numero diviso per due.

Se il dato inserito non è convertibile in un numero gestire l'eccezione mandando a schermo la stringa: "Impossibile valutare l'input".

STRUTTURE DI CONTROLLO

Un programma scritto in un linguaggio imperativo come Python viene eseguito in ordine, partendo dalla prima istruzione fino ad arrivare all'ultima.

Quest'ordine può essere modificato utilizzando le strutture di controllo.

Le strutture di controllo sono `if`, `for` e `while`

COSTRUTTO IF

l'istruzione if si usa quando il programmatore vuole far si che una certa sequenza di istruzioni venga eseguita solo a certe condizioni. La sintassi è questa:

```
if espressione:
    istruzione1
    istruzione2
```

Si può leggere come: "se il valore dell'espressione tra la parola `if` e i due punti è `True` o diverso da zero, allora esegui le istruzioni del blocco . Altrimenti saltale."

COSTRUTTO IF

Esempio:

```
if espressione:
    istruzione1
    istruzione2
istruzione3
```

Le `istruzione1` e `istruzione2` sono spostate a destra di un tab rispetto a `if`. Esse costituiscono un blocco di codice. `L'istruzione3` non fa parte del blocco e non viene saltata se l'`if` ha argomento `False`.

ESEMPIO COSTRUTTO IF

Vogliamo che l'utente possa inserire un numero e che il programma lo mostri a video. Se però il dato supera il valore 2 voglio che venga resettato a zero e che il programma mi avverta.

```
a = float(input("inserisci il valore di a: "))
if a>2:
   a = 0
   print("Ho portato a zero il valore")
print(a)
```

COSTRUTTO IF-ELSE

Il costrutto if-else si usa per definire un'alternativa obbligatoria. Sintassi:

```
if espressione:
   istruzione1
   istruzione2
else:
   istruzione3
```

Else = "altrimenti"

ESEMPIO IF-ELSE

Se nella variabile a c'è un numero positivo allora stampa a video "positivo". Altrimenti "negativo". Se il numero è maggiore o uguale a 10 stampa anche la stringa "Valore elevato"

```
if a>=0:
   print("positivo")
   if a>=10:
      print("valore elevato")
else:
   print("negativo")
```

CICLI

I linguaggi imperativi prevedono i cicli.

Un ciclo è un costrutto che indica all'interprete di eseguire il blocco di istruzioni contenuto nel ciclo e di ritornare alla prima istruzione del blocco, fino a che sussiste una certa condizione.

CICLI

In molti problemi è necessario ripetere lo stesso blocco di istruzioni fino a che non si ottiene un certo risultato.
Ad esempio: consideriamo le seguenti istruzioni:

```
a = 1
a += 1 #ora a vale 2
```

Vogliamo far si che `a` raggiunga il valore 10 continuando a sommare 1 al suo valore.
Si tratterebbe di ripetere l'istruzione `a += 1` fino a che il valore di `a` non sia uguale a 10.

CICLO WHILE

Il ciclo while esegue un blocco di istruzioni fino a che il valore di una certa espressione di controllo è True (o non 0).

Sintassi:

```
while espressione:
    istruzione1
    istruzione2
```

CICLO WHILE

```
while espressione:
    istruzione1
    istruzione2
istruzione3
```

Quando il programma arriva all'istruzione `while` l'interprete valuta il valore dell'espressione. Se è `True` allora esegue il blocco di istruzioni tabulate che parte dal simbolo : (come con l'if).
Ma una volta concluso il blocco di istruzioni l'interprete torna indietro all'istruzione `while` e valuta nuovamente l'espressione.
Se questa ora è `False` allora l'interprete salta il blocco di istruzioni e prosegue dalla `istruzione3`

ESEMPIO CICLO WHILE

Abbiamo una variabile a con valore 10. Vogliamo che finché il valore di a è maggiore o uguale a zero il programma stampi il valore di a e sottragga 1 al valore di a.

```
a = 10
while a>=0:
   print(a)
   a-=1 #a = a-1
print("operazione conclusa")
```

CICLO FOR

Il ciclo `for` esegue un blocco di istruzioni per un numero determinato di volte ed aggiorna in automatico il valore di una variabile.

Sinstassi:

```
for variabile in sequenza:
    istruzione1
    istruzione2
```

Si può leggere in questo modo:
Fino a che ci sono elementi nella sequenza e assegna il valore dell'elemento alla variabile ed esegui il blocco di istruzioni.

ESEMPIO I CICLO FOR

Esempio: in una lista (tipo di dato che vedremo più avanti)

```
for numero in [1,4,9]:
    print(numero)
```

```
1
4
9
```

Il `for` prende uno ad uno i valori della tupla e li assegna alla variabile `numero`

ESEMPIO 2 CICLO FOR

Esempio: in una stringa

```
for lettera in "ciao":
    print(lettera)
```

```
c
i
a
o
```

Il `for` prende uno ad uno i caratteri della stringa e li assegna alla variabile `lettera`

ESEMPIO 3 CICLO FOR

Esempio: in una sequenza range

```
for x in range(1,6):
    print(x)
```

```
1
2
3
4
5
```

RANGE

Sintassi:

```
range(stop)
range(start,stop)
range(start,stop,step)
```

sostituire a `start, stop e step` un valore numerico

Il `range` è un'istruzione che restituisce un valore diverso ad ogni chiamata. Il primo valore restituito è uguale al primo argomento (`start`). Ad ogni iterazione il valore cresce fino al valore del secondo argomento (`stop`) sempre escluso, con incrementi unitari. Si può inserire un terzo argomento chiamato `step` per cambiare l'incremento. Se uso un solo argomento, il valore di partenza è uguale a zero.

ESEMPIO DI RIEPILOGO

Vogliamo un programma che chiede all'utente di inserire un valore maggiore di 1. Il programma continua a chiedere di inserire il numero fino a che questo non sia maggiore di 1.
Successivamente, prende il valore inserito e stampa a video la sua tabellina. Se l'utente inserisce 4, l'output a video deve essere:

```
Tabellina del 4.
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
…
4 x 10 = 40
```