Python e costrutti

UNIVERSITÀ DI PISA

Costrutti

UNIVERSITÀ DI PISA

Un *costrutto* modifica la sequenza normale di istruzioni, permettendoci di...

- Eseguire alcune istruzioni piuttosto che altre
- Ripeto un insieme di istruzioni

Scelta (`if`)

Università di Pisa

```
# temperatura letta da un sensore
temperature = ...

# se temperature > 150, allora il tardigrado e' in stato di stasi
status = "stasis"
# altrimenti, no
status = "normal"
```

Vogliamo assegnare solo un valore a `status`, ma questo dipende da un valore che deriva da un sensore, e non posso sapere a prescindere.

Scelta (`if`)

Università di Pisa

Per effettuare una scelta su che istruzioni eseguire, sfruttiamo i valori di verità (`bool`, con valori `True, False`) e il costrutto `if` (*se*).

- Se (`if`) una condizione è vera, allora esegui queste istruzioni...
- Altrimenti (`else`), esegui queste altre

```
# temperatura letta da un sensore
temperature = ...

if temperature > 150:
        # se temperature > 150, allora il tardigrado e' in stato di stasi
        status = "stasis"
else:
        # altrimenti, no
        status = "normal"
```

Reminder su condizioni booleane

Oltre a `True, False`, possiamo esprimere condizioni booleane con diversi operatori.

- Uguaglianza (`==`) e disuguaglianza (`!=`)
- Relazioni numeriche (`>, <, >=, <=`)
- Negazione di altre espressioni booleane (`not`)

Scelta su condizioni multiple

UNIVERSITÀ DI PISA

Spesso non è sufficiente una sola condizione per definire che istruzioni eseguire, e.g., se

- La temperatura è sopra i 150
- La concentrazione di ossigeno è bassa
- La densità dell'atmosfera è bassa

Allora il tardigrado entra in stato di stasi. **Tutte** queste condizioni devono essere vere!

Scelta su condizioni multiple

Università di Pisa

Date due espressioni di verità (`bool`), l'operatore `and` le valuta a `True` se sono entrambe vere. `False` altrimenti. Tutte le espressioni sono condizioni necessarie, ma nessuna da sola è sufficiente.

```
if temperature > 150 and oxygen < 0.001 and concentration < 0.0000001:
        status = "stasis"
else:
        status = "normal"
```

A	B	and
True	True	True
False	False	False
True	False	False
False	True	False

Scelta su condizioni multiple

UNIVERSITÀ DI PISA

Date due espressioni di verità (`bool`), l'operatore `or` le valuta a `True` se almeno una è vera, e `False` altrimenti. Tutte le espressioni sono sufficienti. Un tardigrado si riproduce se

- È di sesso femminile, e capace di partenogenesi, oppure
- Si è accoppiata con un tardigrado di sesso maschile

Basta che una qualsiasi delle due condizioni sia vera perché l'intera condizione sia vera.

Scelta su condizioni multiple

Università di Pisa

Date due espressioni di verità (`bool`), l'operatore `or` le valuta a `True` se almeno una è vera. `False` altrimenti. Tutte le espressioni sono condizioni sufficienti.

```
supports_parthogenesis = ...
is_fertilized = ...

if supports_parthogenesis or is_fertilized:
        reproduces = True
else:
        reproduces = False
```

A	B	or
True	True	True
False	False	False
True	False	True
False	True	True

Scelta su condizioni multiple

Università di Pisa

Possiamo combinare `and` e `or` (`and` ha precedenza).

```
supports_parthogenesis = ...
is_fertilized = ...
sex = ...

if sex == "F" and supports_parthogenesis or is_fertilized:
        reproduces = True
else:
        reproduces = False
```

Quiz time

Esprimi queste condizioni booleane.

1. Una cellula sintetizza una proteina sfruttando l'elicasi, che separa il DNA, offrendo due filamenti che permettono la trascrizione dell'RNA. Considerando questi come due processi separati, che espressione booleana useremmo per decidere se l'intero processo di trascrizione ha successo?
2. Un organismo sopravvive in un ambiente entro un dato intervallo di temperature. Come esprimeremmo questo in una condizione booleana?
3. Un organismo sopravvive in un ambiente entro un dato intervallo luminoso e di pressione. In alternativa, è in grado di entrare in uno stato di stasi in cui sopravvive a pressioni molto più alte.

Iterazione (`for` e `while`)


```
# riceviamo del DNA da degli esperimenti
dna = ...

# cerca
marker = "CGGGTACT"
first_occurrence = dna.index(marker)
second_occurrence = dna[first_occurrence:].index(marker)
...
```

Vogliamo trovare tutte le posizioni di un certo marker, ma non sappiamo quante volte lo troveremo.

Iterazione (`for` e `while`)

UNIVERSITÀ DI PISA

L'iterazione ci permette di ripetere delle istruzioni su un insieme. Abbiamo due casi: sappiamo quante volte *iterare* (costrutto `for`), oppure no (costrutto `while`).

```
for element in collection:
  ...
```

- `element` è una variabile in cui viene assegnato l'elemento corrente di `collection`
- `collection` può essere una qualsiasi collezione
- Le istruzioni (corpo) sono indentate

Iterazione (`for` e `while`)

UNIVERSITÀ DI PISA

L'iterazione ci permette di ripetere delle istruzioni su un insieme. Abbiamo due casi: sappiamo quante volte *iterare* (costrutto `for`), oppure no (costrutto `while`).

```
conto_timine = 0
for base in "ACCCGTGAC":
  if base == "T":
    conto_timine += 1
```

Iterazione (`for` e `while`)

Università di Pisa

L'iterazione ci permette di ripetere delle istruzioni su un insieme. Abbiamo due casi: sappiamo quante volte *iterare* (costrutto `for`), oppure no (costrutto `while`).

```
conto_timine = 0
for i in range(10):
  print(i)
```

Iterazione (`for` e `while`)

UNIVERSITÀ DI PISA

L'iterazione ci permette di ripetere delle istruzioni su un insieme. Abbiamo due casi: sappiamo quante volte *iterare* (costrutto `for`), oppure no (costrutto `while`).

```
while condition:
  ...
```

- `condition` è una condizione booleana, come nell'`if`
- Le istruzioni (corpo) sono indentate

Iterazione (`for` e `while`)

UNIVERSITÀ DI PISA

L'iterazione ci permette di ripetere delle istruzioni su un insieme. Abbiamo due casi: sappiamo quante volte *iterare* (costrutto `for`), oppure no (costrutto `while`).

```
dna = ...
current_index = 0

while current_index != -1:
  current_index = dna[current_index:].index(marker)
```