Costrutti di controllo

Flusso di controllo - Branching

```
if <condition>:
    <expression>
    <expression>
    ...
```

```
if <condition>:
    <expression>
    <expression>
    ...
else:
    <expression>
    <expression>
    ...
```

```
if <condition>:
    <expression>
    <expression>
    ...
elif <condition>:
    <expression>
    <expression>
    ...
else:
    <expression>
    <expression>
    ...
```

- `<condition>` può essere `True` o `False`
- Si valuta l'espressione nel blocco se `<condition>` è `True`

Indentazione

- **L'indentazione in Python conta!**
- Viene utilizzata per definire un blocco di istruzioni

```
x = float(input("Enter a number for x: "))
y = float(input("Enter a number for y: "))
if x == y:
    print("x and y are equal")
    if y != 0:
        print("therefore, x / y is", x/y)
elif x < y:
    print("x is smaller")
else:
    print("y is smaller")
print("thanks!")
```

Operatore = vs Operatore ==

```
x = float(input("Enter a number for x: "))
y = float(input("Enter a number for y: "))
if x == y:
    print("x and y are equal")
    if y != 0:
        print("therefore, x / y is", x/y)
elif x < y:
    print("x is smaller")
else:
    print("y is smaller")
print("thanks!")
```

What if x = y here?

get a SyntaxError

Flusso di controllo: ciclo while

```
while <condition>:
    <expression>
    <expression>
    ...
```

1. `<condition>` viene valutata come un booleano
2. se `<condition>` è `True`, allora esegui tutte le istruzioni nel blocco while
3. controlla `<condition>`
4. ripeti finché `<condition>` è `False`

Flusso di controllo: ciclo for

```
for <variable> in range(<some_num>):
    <expression>
    <expression>
    ...
```

- Ogni volta che siamo nel ciclo, `<variable>` assume un valore nella lista ottenuta con range `range(<some_num>)`
  - Il tipo `range` rappresenta una **sequenza immutabile** di numeri (https://docs.python.org/3/library/stdtypes.html#range)
- La prima iterazione, `<variable>` assume l'estremo inferiore dell'insieme specificato in range
- Le successive volte, `<variable>` assume il valore precedente sommato di 1 (il default step è 1)
- E così via…

Flusso di controllo: ciclo for

range(start,stop,step)

- E’ possibile utilizzare range specificando l’estremo inferiore, superiore, e lo step di incremento
- Di default di start = 0 e step = 1
- l ciclo esegue fino a che il valore diventa `stop - 1`
- E.g.:

```
mysum = 0
for i in range(7, 10):
    mysum += i
print(mysum)
```

```
mysum = 0
for i in range(5, 11, 2):
    mysum += i
print(mysum)
```

Istruzione break

- Esce immediatamente qualunque sia il tipo e lo stato corrente del ciclo utilizzato
- Salta la valutazione delle espressioni rimanenti nel blocco di codice
- Esce soltanto nel contesto del ciclo più interno!

```
while <condition_1>:
    while <condition_2>:
        <expression_a>
        break
        <expression_b>
    <expression_c>
```

Istruzione break: esempio

```
mysum = 0
for i in range(5, 11, 2):
    mysum += i
    if mysum == 5:
        break
    mysum += 1
print(mysum)
```

- Cosa succede a questo programma?

Stringhe e Cicli

- Gli snippet di codice seguenti fanno la stessa cosa
- Quello più in basso è più “pythonic”

```
s = "abcdefgh"

for index in range(len(s)):
    if s[index] == 'i' or s[index] == 'u':
        print("There is an i or u")
```

```
for char in s:
    if char == 'i' or char == 'u':
        print("There is an i or u")
```

Stringhe e Cicli:
Esempio ROBOT CHEERLEADERS

```
an_letters = "aefhilmnorsxAEFHILMNORSX"

word = input("I will cheer for you! Enter a word: ")
times = int(input("Enthusiasm level (1-10): "))

i = 0
while i < len(word):
    char = word[i]
    if char in an_letters:
        print("Give me an " + char + "! " + char)
    else:
        print("Give me a  " + char + "! " + char)
    i += 1
print("What does that spell?")
for i in range(times):
    print(word, "!!!")
```

```
for char in word:
```

Stringhe e Cicli: Esempio 2

```
s1 = "mit u rock"
s2 = "i rule mit"
if len(s1) == len(s2):
    for char1 in s1:
        for char2 in s2:
            if char1 == char2:
                print("common letter")
                break
```