Appunti Python - Coltellino Svizzero

Struttura Algoritmo

- Inizializzazioni
- Input
- Cicli
- Stampe

IDE

- Ambiente di esecuzione integrato

Input e Output

- Per leggere l'input da tastiera, usa la funzione `input()`:

```
nome = input("Inserisci il tuo nome: ")
```

- Per stampare l'output a schermo, usa la funzione `print()`:

```
print("Ciao,", nome)
```

Variabili e Tipi di Dati

- Le variabili in Python sono dichiarate assegnando loro un valore:

```
x = 10
y = "Hello"
```

- I principali tipi di dati in Python sono:
  - Numeri interi (`int`): `x = 5`
  - Numeri a virgola mobile (`float`): `y = 3.14`
  - Stringhe (`str`): `nome = "Mario"`
  - Booleani (`bool`): `is_true = True`

Operatori

- Operatori aritmetici: `+`, `-`, `*`, `/`, `%` (modulo), `**` (potenza)
- Operatori di confronto: `==`, `≠`, `<`, `>`, `≤`, `≥`
- Operatori logici: `and`, `or`, `not`

Condizioni

- If/Elif/Else:

```
if condizione:
    # codice se la condizione è vera
elif altra_condizione:
    # codice se l'altra condizione è vera
else:
    # codice se nessuna delle condizioni precedenti è vera
```

```
Osservazioni:
- Servono a dire percorsi logici diversi

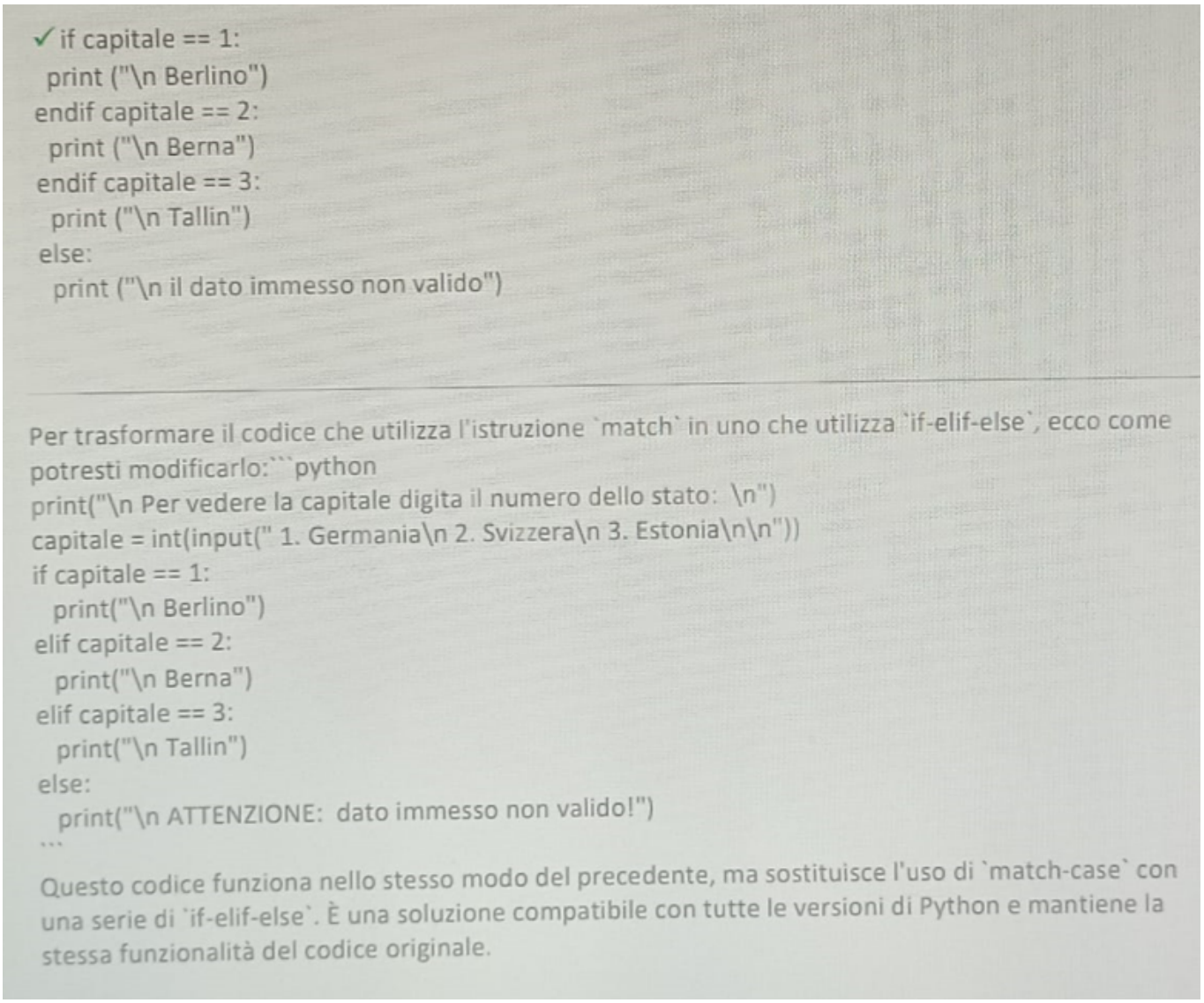
- Match = Modo diverso di scrivere gli if
```python
match parameter: # parameter = variabile che controlli
    case prima_variabile:
        # code for pattern 1
    case seconda_variabile:
        # code for pattern 2

    case tuttelevariabili
    case _:
        # default code block = nessuna delle opzioni coperte (già coperte)
```

```
if voto1 == 1:
    gatto += 1
elif voto1 == 2:
        cane += 1
elif voto1 == 3:
```
```

```
        criceto += 1

match voto1:
        case 1:
                gatto += 1
        case 2:
                cane += 1
        case 3:
                criceto += 1
        case _:
                nonvalido += 1
```



✓ if capitale == 1:
print ("\n Berlino")
endif capitale == 2:
print ("\n Berna")
endif capitale == 3:
print ("\n Tallin")
else:
print ("\n il dato immesso non valido")

Per trasformare il codice che utilizza l'istruzione `match` in uno che utilizza `if-elif-else`, ecco come potresti modificarlo:```python
print("\n Per vedere la capitale digita il numero dello stato: \n")
capitale = int(input(" 1. Germania\n 2. Svizzera\n 3. Estonia\n\n"))
if capitale == 1:
print("\n Berlino")
elif capitale == 2:
print("\n Berna")
elif capitale == 3:
print("\n Tallin")
else:
print("\n ATTENZIONE: dato immesso non valido!")
```
Questo codice funziona nello stesso modo del precedente, ma sostituisce l'uso di `match-case` con una serie di `if-elif-else`. È una soluzione compatibile con tutte le versioni di Python e mantiene la stessa funzionalità del codice originale.

# Cicli

- Cicli `for`: (da usare quando so che devo contare fino ad un certo/stabilito numero di iterazioni)
```

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

Se ho più cicli:

- primo for: scorre le righe
- secondo for: scorre le colonne

```
for i:1 to 3 #righe
        for j:1 to 4 # colonne

[1 2 3 4
4 5 6 5
7 8 9 10]

# Indici = i/j (ma puoi chiamarli come vuoi)

# Se separo i for
for i:1 to 3 #righe

end

1 2 3

for j:1 to 4 # colonne

end

1 2 3 4
```

- Cicli `while` (si usa se mi serve una condizione - se prevedi che il ciclo possa essere interrotto da una condizione):

```
i = 0
while i < 10:
    print(i)
    i += 1
```

Formattazione di Stringhe

- Per formattare le stringhe, puoi usare gli f-string (a partire da Python 3.6):

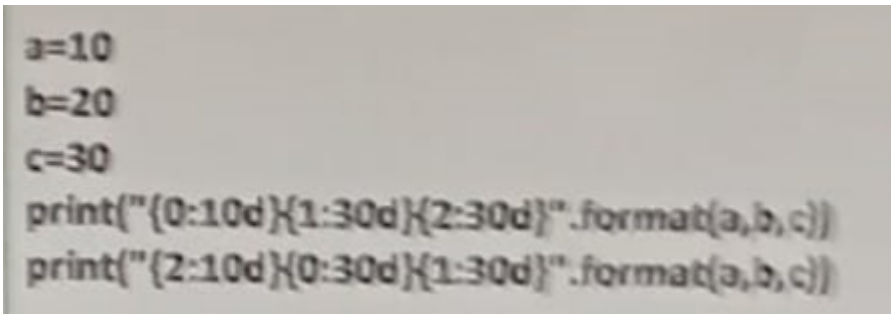
```
nome = "Mario"
età = 25
print(f"Il mio nome è {nome} e ho {età} anni.")
```

- In alternativa, puoi usare il metodo `format()` per rimpiazzare le variabili nelle graffe:

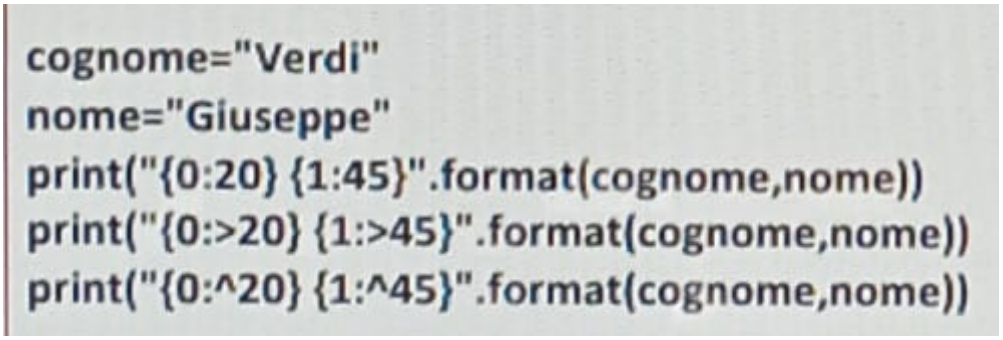
```
print("Il mio nome è {} e ho {} anni.".format(nome, età))
```

- `>` = destra
- `^` = centro
- `<` = sinistra

Esempi di codice:



Il metodo format viene utilizzato per inserire i valori delle variabili a, b, c in una stringa, formattandoli secondo le specifiche indicate tra le parentesi graffe, come larghezza e allineamento del campo.




```
cognome = "Verdi"
nome = "Giuseppe"
print("{0:20}{1:45}".format(cognome, nome))
```

```
print("{0:>20}{1:>45}".format(cognome, nome))
print("{0:^20}{1:^45}".format(cognome, nome))
```

In sintesi:

- `{0:20}` e `{1:45}` stampano i valori in campi di larghezza fissa, allineati a sinistra.
- `>` allinea i valori a destra all'interno dei campi di larghezza specificata.
- `^` allinea i valori al centro all'interno dei campi di larghezza specificata.

Formattazione avanzata delle stringhe

Formattazione di Stringhe in Python

Metodi di Formattazione

F-String (Python 3.6+)

- Utilizzare la lettera `f` o `F` prima delle virgolette per creare una f-string.
- Inserire le variabili o le espressioni tra parentesi graffe `{}` all'interno della stringa.

```
nome = "Mario"
età = 25
print(f"Il mio nome è {nome} e ho {età} anni.")
```

Metodo format()

- Utilizzare il metodo `format()` per sostituire i segnaposto `{}` con i valori delle variabili.

```
nome = "Mario"
età = 25
print("Il mio nome è {} e ho {} anni.".format(nome, età))
```

- È possibile specificare l'indice delle variabili all'interno delle parentesi graffe `{}`.

```
nome = "Mario"
età = 25
print("Il mio nome è {0} e ho {1} anni.".format(nome, età))
```

- È possibile utilizzare le parole chiave per assegnare i valori ai segnaposto.

```
nome = "Mario"
età = 25
print("Il mio nome è {nome} e ho {età} anni.".format(nome=nome, età=età))
```

Allineamento e Riempimento

- Utilizzare i due punti `:` seguiti da un carattere di allineamento per specificare l'allineamento del testo.
  - `<`: allineamento a sinistra
  - `>`: allineamento a destra
  - `^`: allineamento al centro

```
print("{:<10}".format("Ciao"))    # Output: "Ciao      "
print("{:>10}".format("Ciao"))    # Output: "      Ciao"
print("{:^10}".format("Ciao"))    # Output: "   Ciao   "
```

- Specificare un carattere di riempimento prima del carattere di allineamento per riempire gli spazi vuoti.

```
print("{:*<10}".format("Ciao"))   # Output: "Ciao******"
print("{:0>10}".format(42))       # Output: "0000000042"
```

Formattazione di Numeri

- Utilizzare il formato `{:.n}` per specificare il numero di cifre decimali per i numeri in virgola mobile, dove `n` è il numero di cifre decimali desiderato.

```
print("{:.2f}".format(3.14159))   # Output: "3.14"
print("{:.4f}".format(3.14159))   # Output: "3.1416"
```

- Combinare l'allineamento, il riempimento e la larghezza del campo con i valori numerici.

```
print("{:0<10.2f}".format(3.14159))   # Output: "3.14000000"
print("{:*>10.3f}".format(3.14159))   # Output: "***3.142"
```

```
print("{:^10.4f}".format(3.14159))    # Output: " 3.1416 "
```

- Utilizzare il carattere `%` per formattare le percentuali.

```
print("{:.2%}".format(0.7514))   # Output: "75.14%"
print("{:.0%}".format(0.7514))   # Output: "75%"
```

- Utilizzare il carattere `,` per separare le migliaia nei numeri.

```
print("{:,}".format(1234567890))          # Output: "1,234,567,890"
print("{:,.2f}".format(1234567890.987))   # Output: "1,234,567,890.99"
```

- Utilizzare le lettere `e` o `E` per formattare i numeri in notazione scientifica.

```
print("{:.2e}".format(1234567890))   # Output: "1.23e+09"
print("{:.2E}".format(0.0000000987))   # Output: "9.87E-08"
```

Caratteri Speciali

- Utilizzare la sequenza di escape `\` per inserire caratteri speciali nelle stringhe.
  - `\n`: nuova riga
  - `\t`: tabulazione
  - `\\`: backslash
  - `\"`: virgolette doppie
  - `\'`: virgoletta singola

```
print("Prima riga\nSeconda riga")
print("Tabulazione\tQui")
print("Backslash: \\")
print("Virgolette doppie: \"")
print("Virgoletta singola: \'")
```

- Utilizzare i codici Unicode per rappresentare caratteri speciali.

```
print("\u00A9")   # Output: ©
print("\u2122")   # Output: ™
print("\u03C0")   # Output: π
```

Algoritmi utili

Ricerca massimo e posizione massimo

```
# Inizializzazione variabili

MAX = 0

num = int(input("Dammi il numero di numeri: ")) #input
pos = 0 # posizione

for i in range(1, N+1): # il ciclo scorre tutti gli elementi
        if num > MAX:
                MAX = num  # salvo il nuovo massimo
                pos = i    # salviamo l'indice come posizione
        end
end
```

Ricerca minimo e posizione minimo

```
# Inizializzazione variabili

MIN = 0

num = int(input("Dammi il numero di numeri: ")) #input
pos = 0 # posizione

for i in range(1, N+1): # il ciclo scorre tutti gli elementi
        if num < MIN:
                MIN = num  # salvo il nuovo minimo
                pos = i    # salviamo l'indice come posizione
        end
end
```

Calcolo della media

```
somma = 0
n = int(input("Inserisci il numero di elementi: "))
```

```python
for i in range(n):
    num = float(input(f"Inserisci il numero {i+1}: "))
    somma += num

media = somma / n
print(f"La media è: {media}")
```