Codici Python del Primo Quadrimestre

Calcolatrice Con Python

L'Addizione

```
numero1 = int(input("Inserisci il primo numero: "))
numero2 = int(input("Inserisci il secondo numero: "))
somma = numero1 + numero2
print("La somma è:", somma)

Inserisci il primo numero: 2
Inserisci il secondo numero: 3
La somma è: 5
```

La Sottrazione

```
sottrazione = numero1 - numero2
print("La sottrazioneè:", sottrazione)

La sottrazioneè: -1
```

La Moltiplicazione

```
numero1 = int(input("Inserisci il pirmo numero: "))
numero2 = int(input("Inserisci il secondo numero: "))
moltiplicazione = numero1 * numero2
print("La moltiplicazione è:", moltiplicazione)

Inserisci il pirmo numero: 2
Inserisci il secondo numero: 3
La moltiplicazione è: 6
```

Divisione

```
numero1 = int(input("Inserisci il pirmo numero: "))
numero2 = int(input("Inserisci il secondo numero: "))
divisione = numero1 / numero2
print("La divisione è:", divisione)

Inserisci il pirmo numero: 3
Inserisci il secondo numero: 2
La divisione è: 1.5
```

Calcolare la Somma

```
n = int(input("Inserisci un numero intero positivo: "))
somma = 0

for numero in range(1, n+1):
    #somma = somma + numero
    somma += numero
print("La somma dei primi", n, "numeri interi è:", somma)
```

```
Inserisci un numero intero positivo:  6

La somma dei primi 6 numeri interi è: 21
```

Il Loop e le Ripetizioni

```
for numero in range(1,20): #il range parte da 0 ed arriva a 11
    print(numero)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

Calcolatrice con le Condizioni e le Decisioni

```
operazione = input("Inserisci l'operazione (+, -, *, /): ")

numero1 = float(input("Inserisci il primo numero: "))
numero2 = float(input("Inserisci il secondo numero: "))

if operazione == "+":
```

```
    risultato = numero1 + numero2
elif operazione == "-":
    risultato = numero1 - numero2
elif operazione == "*":
    risultato = numero1 * numero2
elif operazione == "/":
    risultato = numero1 / numero2
else:
    risultato = "Operazione non valida"

print("Il risultato è:", risultato)
```

```
Inserisci l'operazione (+, -, *, /):  +
Inserisci il primo numero:  8
Inserisci il secondo numero:  4

Il risultato è: 12.0
```

Contare fino al Numero

```
n = int(input("Inserisci un numero intero positivo:"))

for numero in range(1,n+1):
    print(numero)
```

```
Inserisci un numero intero positivo: 21

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
```

Calcolare il Fattoriale

```
n = int(input("Inserire un numero intero positivo: "))
fattoriale = 1

for numero in range(1, n+1):
    fattoriale *= numero
print("Il fattoriale di", n, "è:", fattoriale)
```

```
Inserire un numero intero positivo:  10

Il fattoriale di 10 è: 3628800
```

Calcolare la media di una Lista di Numeri

```
numeri = []

n = int(input("Quanti numeri vuoi inserire= "))

for i in range(n):
    numero = float(input("Inserisci un numero: "))
    numeri.append(numero)

media = sum(numeri) / len(numeri)

print("La media dei numeri inseriti è:", media, numeri)
```

```
Quanti numeri vuoi inserire=  6
Inserisci un numero:  10
Inserisci un numero:  12
Inserisci un numero:  8
Inserisci un numero:  90
Inserisci un numero:  54
Inserisci un numero:  4

La media dei numeri inseriti è: 29.666666666666668 [10.0, 12.0, 8.0,
90.0, 54.0, 4.0]
```

Indovina il numero

```
import random

numero_da_indovinare =random.randint(1, 100)
tentativi = 0

#chiede un numero
while True:
```

```
    tentativo = int(input("Indovina il numero (1-100): "))
    tentativi += 1

    if tentativo == numero_da_indovinare:
        print("Bravo! Hai indovinato il numero", numero_da_indovinare,
"in", tentativi, "tentativi")
        break
    elif tentativo < numero_da_indovinare:
        print("Il numero è più grande")
    else:
        print("Il numero è più piccolo")
```

```
Indovina il numero (1-100):  50

Il numero è più piccolo

Indovina il numero (1-100):  25

Il numero è più grande

Indovina il numero (1-100):  40

Il numero è più piccolo

Indovina il numero (1-100):  7

Il numero è più grande

Indovina il numero (1-100):  37

Il numero è più grande

Indovina il numero (1-100):  38

Il numero è più grande

Indovina il numero (1-100):  39

Bravo! Hai indovinato il numero 39 in 7 tentativi
```

La Morra Cinese

```
import random

mosse = ["carta", "forbice", "sasso"]

computer_mossa = random.choice(mosse)

print("Benvenuti al gioco del Morra Cinese!")
scelta_giocatore = input("Scegli la tua mossa (Carta, forbici, sasso):
")
```

```
if scelta_giocatore not in mosse:
    print("Mossa non permessa")
else:
    print("Il computer ha scelto:", computer_mossa)
    if scelta_giocatore == computer_mossa:
        print("Pareggio!")
    elif (scelta_giocatore == "carta" and computer_mossa == "sasso")
or \
         (scelta_giocatore == "forbici" and computer_mossa == "carta")
or \
         (scelta_giocatore == "sasso" and computer_mossa ==
"forbici"):
        print("Hai Vinto!")
    else:
        print("Hai Perso!")
```

```
Benvenuti al gioco del Morra Cinese!

Scegli la tua mossa (Carta, forbici, sasso):  carta

Il computer ha scelto: sasso
Hai Vinto!
```

Calcolo del fattoriale

```
n = int(input("Inserisci un numero intero: "))

fattoriale = 1

if n<0:
    print("Numero Negativo")
elif n ==0:
    print("Il numero di zero è un 1 per definizione")
else:
    for numero in range(1, n+1):
        fattoriale*=numero
print(f"Il fattoriale di {n} è {fattoriale}")
```

```
Inserisci un numero intero: 02
Il fattoriale di 2 è 2
```

somma numeri n

```
N = int(input("Inserisci un numero intero positivo N: "))

somma = 0
```

```
for numero in range(2, 2 * N + 1, 2):
    somma += numero

print(f"la somma dei primi {N} numeri pari è {somma}")
```

```
Inserisci un numero intero positivo N:  40

la somma dei primi 40 numeri pari è 1640
```

Lista numeri pari

```
N = int(input("Inserisci un numero intero positivo N: "))
lista = []

for numero in range(2, 2 * N + 1,2):
    lista.append(numero)

print(lista)
```

```
Inserisci un numero intero positivo N:  26

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36,
38, 40, 42, 44, 46, 48, 50, 52]
```

SIMULAZIONE POPOLAZIONE

```
popolazione = int(input("inserisci popolazione iniziale: "))
anni = int(input("Inserisci il numero di anni da simulare: "))

tasso_natalità = float(input("Inserisci tasso natalità: "))
tasso_mortalità = float(input("Inserisci tasso mortalità: "))


for anno in range(anni):
    nascite = (popolazione * tasso_natalità) / 100
    morti = (popolazione * tasso_mortalità) / 100
    popolazione += (nascite - morti)

    print(f"Anno {anni}, Popolazione {int(popolazione)}")

print("Simulazione completata")
```

```
inserisci popolazione iniziale: 20
Inserisci il numero di anni da simulare: 1
Inserisci tasso natalità: 2
Inserisci tasso mortalità: 2
```

```
Anno 1, Popolazione 20
Simulazione completata
```

calcolo interessi

```
def calcola_interessi(importo_iniziale, tasso_interesse,
periodi_investimento):
    importo_finale = importo_iniziale * (1 + tasso_interesse / 100) **
periodi_investimento
    return importo_finale

print("Benvetuo nel Calolatore d'interessi!")

importo = float(input("Inserisci l'importo iniziale: "))
tasso = float(input("Inserisci il tasso d'interesse annuale (%): "))
periodo = int(input("Inserisci il periodo d'investimento (anni): "))

importo_finale = calcola_interessi(importo, tasso, periodo)

print(f"l'importo finale dopo {periodo} anni è di {importo_finale:.2f}
euro.")
```

```
Benvetuo nel Calolatore d'interessi!

Inserisci l'importo iniziale:  50
Inserisci il tasso d'interesse annuale (%):  15
Inserisci il periodo d'investimento (anni):  5

l'importo finale dopo 5 anni è di 100.57 euro.
```

calcolatore di anagrammi

```
from itertools import permutations
k=0

def trova_anagrammi(parola):
    anagrammi = ["".join(p) for p in permutations(parola)]
    return anagrammi

print("Benvenuto nel risolutore di anagrammi!")

parola_input = input("Inserisci una parola: ").strip().lower()

if len(parola_input) < 2:
    print("Inserisci una parola con almeno 2 caratteri: ")
else:
    anagrammi = trova_anagrammi(parola_input)
```

```
    for anagramma in anagrammi:
        if anagramma != parola_input:
            k+=1
            print(anagramma)
        print(f"gli anagrammi di '{parola_input}' sono: '{k}'")
```

```
Benvenuto nel risolutore di anagrammi!

Inserisci una parola:  casa

gli anagrammi di 'casa' sono: '0'
caas
gli anagrammi di 'casa' sono: '1'
csaa
gli anagrammi di 'casa' sono: '2'
csaa
gli anagrammi di 'casa' sono: '3'
caas
gli anagrammi di 'casa' sono: '4'
gli anagrammi di 'casa' sono: '4'
acsa
gli anagrammi di 'casa' sono: '5'
acas
gli anagrammi di 'casa' sono: '6'
asca
gli anagrammi di 'casa' sono: '7'
asac
gli anagrammi di 'casa' sono: '8'
aacs
gli anagrammi di 'casa' sono: '9'
aasc
gli anagrammi di 'casa' sono: '10'
scaa
gli anagrammi di 'casa' sono: '11'
scaa
gli anagrammi di 'casa' sono: '12'
saca
gli anagrammi di 'casa' sono: '13'
saac
gli anagrammi di 'casa' sono: '14'
saca
gli anagrammi di 'casa' sono: '15'
saac
gli anagrammi di 'casa' sono: '16'
acas
gli anagrammi di 'casa' sono: '17'
acsa
gli anagrammi di 'casa' sono: '18'
```

```
aacs
gli anagrammi di 'casa' sono: '19'
aasc
gli anagrammi di 'casa' sono: '20'
asca
gli anagrammi di 'casa' sono: '21'
asac
gli anagrammi di 'casa' sono: '22'
```

dizionario

```
tassi_di_cambio= {
    "dollari": 1.0,
    "euro": 0.85,
    "yen": 110.41,
}

importo = float(input("Inserisci l'importo da convertire: "))
valuta_di_partenza = input("Inserisci la valuta di partenza:
").lower()
valuta_destinazione = input("Inserisci la valuta di destinazione:
").lower()

if valuta_di_partenza in tassi_di_cambio and valuta_destinazione in
tassi_di_cambio:
    tasso_di_cambio = tassi_di_cambio[valuta_destinazione] /
tassi_di_cambio[valuta_di_partenza]
    importo_convertito = importo * tasso_di_cambio

    print(f"{importo} {valuta_di_partenza} sono equivalenti a
{importo_convertito:.2f} {valuta_destinazione}")
else:
    print("Valute non supportate. Assicurati di inserire valute
valide.")

Inserisci l'importo da convertire:  35
Inserisci la valuta di partenza:  dollari
Inserisci la valuta di destinazione:  euro

35.0 dollari sono equivalenti a 29.75 euro
```

Utilizzo della libreria "Datetime"

```
from datetime import datetime
import pytz

print("Benvenuto nell'orologio mondiale")
```

```python
citta_fusi_orari = {
    "New York": "America/New_York",
    "Londra": "Europe/London",
    "Tokyo": "Asia/Tokyo",
    "Sydney": "Australia/Sydney",
    "Rio de Janeiro": "America/Sao_Paulo",
}

while True:
    print("\n\nCittà disponibili:")
    for citta in citta_fusi_orari.keys():
        print(citta)
    scelta_citta = input("Inserisci il nome della città per
visualizzare l'ora (o 'esci' per uscire): ").strip()
    if scelta_citta.lower() == 'esci':
        print("ok")
        break

    if scelta_citta in citta_fusi_orari.keys():
        fuso_orario = pytz.timezone(citta_fusi_orari[scelta_citta])
        ora_corrente = datetime.now(fuso_orario)
        print(f"\nL'ora corrente a {scelta_citta} è:
{ora_corrente.strftime('%H:%M:%S')}")
    else:
        print("Città non valida. Riprova")
```

```
Benvenuto nell'orologio mondiale


Città disponibili:
New York
Londra
Tokyo
Sydney
Rio de Janeiro

Inserisci il nome della città per visualizzare l'ora (o 'esci' per
uscire):  Tokyo


L'ora corrente a Tokyo è: 18:05:13


Città disponibili:
New York
Londra
Tokyo
Sydney
Rio de Janeiro
```

```
Inserisci il nome della città per visualizzare l'ora (o 'esci' per
uscire):  Londra


L'ora corrente a Londra è: 09:05:18


Città disponibili:
New York
Londra
Tokyo
Sydney
Rio de Janeiro

Inserisci il nome della città per visualizzare l'ora (o 'esci' per
uscire):  New York


L'ora corrente a New York è: 04:05:26


Città disponibili:
New York
Londra
Tokyo
Sydney
Rio de Janeiro

Inserisci il nome della città per visualizzare l'ora (o 'esci' per
uscire):  esci

ok
```

Dizionari e main

```python
def paolo():
    print("Mi chiamo Paolo")

if __name__ == "__main__" :
    paolo()
```

```
Mi chiamo Paolo
```

Calcolatore BMI

```python
def calcola_bmi(peso, altezza):
    return peso / (altezza ** 2)
```

```
def valuta_bmi(bmi):
    if bmi < 10.5:
        return "Sottopeso"
    elif 10.5 <= bmi < 24.9:
        return "Normopeso"
    elif 25 <= bmi < 29.9:
        return "Sovrappeso"
    else:
        return "Obeso"

def main():
    print("Benvenuto nella calcolatri bmi!")
    peso = float(input("Inserisci i tuo peso in chilogrammi: "))
    altezza = float(input("Inserisci la tua altezza in metri: "))

    bmi = calcola_bmi(peso, altezza)
    valutazione = valuta_bmi(bmi)

    print(f"il tuo BMI è {bmi:.2f}, sei classificato come
'{valutazione}'.")

if __name__ == "__main__":
    main()
```

```
Benvenuto nella calcolatri bmi!

Inserisci i tuo peso in chilogrammi:  45
Inserisci la tua altezza in metri:  2

il tuo BMI è 11.25, sei classificato come 'Normopeso'.
```

Calcolatore Del Cibo

```
cibo_calorie = {
    "pizza": 285,
    "hamburger": 250,
    "insalata": 100,
    "pollo arrosto": 335,
    "yogurt": 150
}

def calorie_consumate(cibo, quantita):
    if cibo not in cibo_calorie.keys():
        print("Cibo non presente")
    elif cibo in cibo_calorie:
        calorie_per_100g = cibo_calorie[cibo]
        calorie_totali = (calorie_per_100g / 100) * quantita
        return calorie_totali
```

```
    else:
        return 0

def main():
    cibo_consumato = []

    while True:
        print("menù:")
        print("\n 1. aggiungi cibo consmato")
        print("\n 2. calcola calorie totali")
        print("\n 3 esci")

        scelta = input("scegli un opzione: ")

        if scelta == "1":
            print("\n")
            for key, value in cibo_calorie.items():
                print(key,value)

            cibo = input("inserisci il cibo consumto: ").lower()
            quantita = float(input("Inserisci la quantita in grammi:
"))
            cibo_consumato.append((cibo,quantita))
        elif scelta == "2":
            calorie_totali = sum(calorie_consumate(c,q) for c, q in
cibo_consumato)
            print(f"\ncalore totali consumate: {calorie_totali}
calorie")
        elif scelta == "3":
            print("ok")
            break
        else:
            print("\nscelta non valida. riprova")
if __name__ == "__main__":
    main()
```

```
menù:

 1. aggiungi cibo consmato

 2. calcola calorie totali

 3 esci

scegli un opzione:  3

ok
```

Generatori di Personggi

```
import random

speci = ["Elfo", "Umano" "Nano", "Orco", "Gnomo"]
classi = ["Guerriero", "Mago", "Ranger", "Ladro", "Chierico"]
armi = ["Spada", "Arco", "Bacchetta magica", "Ascia", "Daga"]
abilità = ["Furtività", "Magia dell'acqua", "Camuffamento",
"Estrazione mineraria", "Incantesimi di guarigione"]

specie = random.choice(speci)
classe = random.choice(classi)
arma = random.choice(armi)
abilità_scelte = random.sample(abilità, random.randint(1,3))

print(f"Personaggio Fantasy Generato: ")
print(f"Specie: {specie}")
print(f"Classe: {classe}")
print(f"Arma: {arma}")
print(f"Abilità: {', '.join(abilità_scelte)}")
```

```
Personaggio Fantasy Generato:
Specie: UmanoNano
Classe: Guerriero
Arma: Bacchetta magica
Abilità: Magia dell'acqua, Estrazione mineraria
```

```
import random

speci = ["Elfo", "Umano" "Nano", "Orco", "Gnomo"]
classi = ["Guerriero", "Mago", "Ranger", "Ladro", "Chierico"]
armi = ["Spada", "Arco", "Bacchetta magica", "Ascia", "Daga"]
abilità = ["Furtività", "Magia dell'acqua", "Camuffamento",
"Estrazione mineraria", "Incantesimi di guarigione"]

def crea_personaggio():
    return {
        "Specie": random.choice(speci),
        "Classe": random.choice(classi),
        "Arma": random.choice(armi),
        "Abilità": random.sample(abilità, random.randint(1,3))
    }

def main():
    personaggio_generato = crea_personaggio()

    print("Personaggi Fantasy Generato:")
    for chiave, valore in personaggio_generato.items():
        if chiave == "Abilità":
            valore = ', '.join(valore)
```

```
        print(f"{chiave}: {valore}")

if __name__ == "__main__":
    main()
```

```
Personaggi Fantasy Generato:
Specie: Elfo
Classe: Mago
Arma: Spada
Abilità: Magia dell'acqua
```

frase del giorno

```
import random
citazioni = [
    "Giorno Di sera, qualcosa non c'era",
    "Non c'è male senza male",
    "La via piu'veloce è buttarsi",
    "Quando la vita ti da i limoni, limona",
]

def genera_citazione():
    return random.choice(citazioni)

def main():
    print("Benvevuto nel generatore di citazioni")
    input("Premi invio per ottenere una citazione causale...")

    citazione = genera_citazione()
    print(f"citazione del giorno: {citazione}")

if __name__ == "__main__":
    main()
```

```
Benvevuto nel generatore di citazioni

Premi invio per ottenere una citazione causale...

citazione del giorno: Quando la vita ti da i limoni, limona
```

GENERATORE FRASI DA INFLUENCER!!!

```
import random
frammenti = [
    "yo bro",
    "Come butta brosky",
    "babba",
```

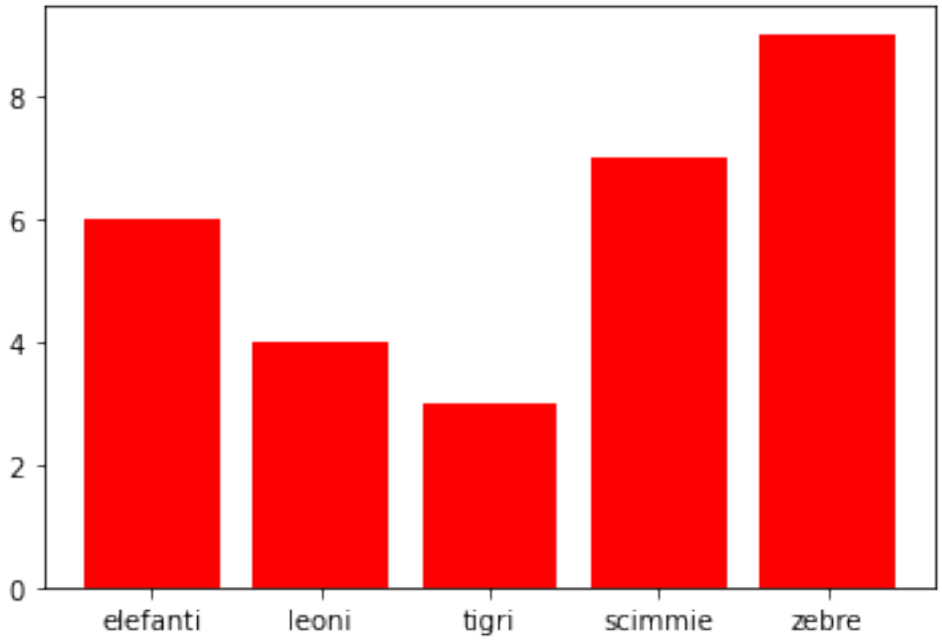
```
    "SGREVE!!!",
    "Come butta?",
    "Fra, ma quanto sei cool?",
    "Troppo cool!",
    "Ma sei real?",
    "Non tutto il gold luccica, bro"
]
def crea_citazione():
    num_frammenti = random.randint(5,7) #scegli un numero casuale di frammenti da utilizzare
    citazione_rimescolata = random.sample(frammenti, num_frammenti)
    nuova_citazione = " ".join(citazione_rimescolata)
    return nuova_citazione

nuova_citazione = crea_citazione()
print("Nuova citazione generata:")
print(nuova_citazione)
```

```
Nuova citazione generata:
Non tutto il gold luccica, bro Come butta brosky Troppo cool! yo bro Fra, ma quanto sei cool?
```

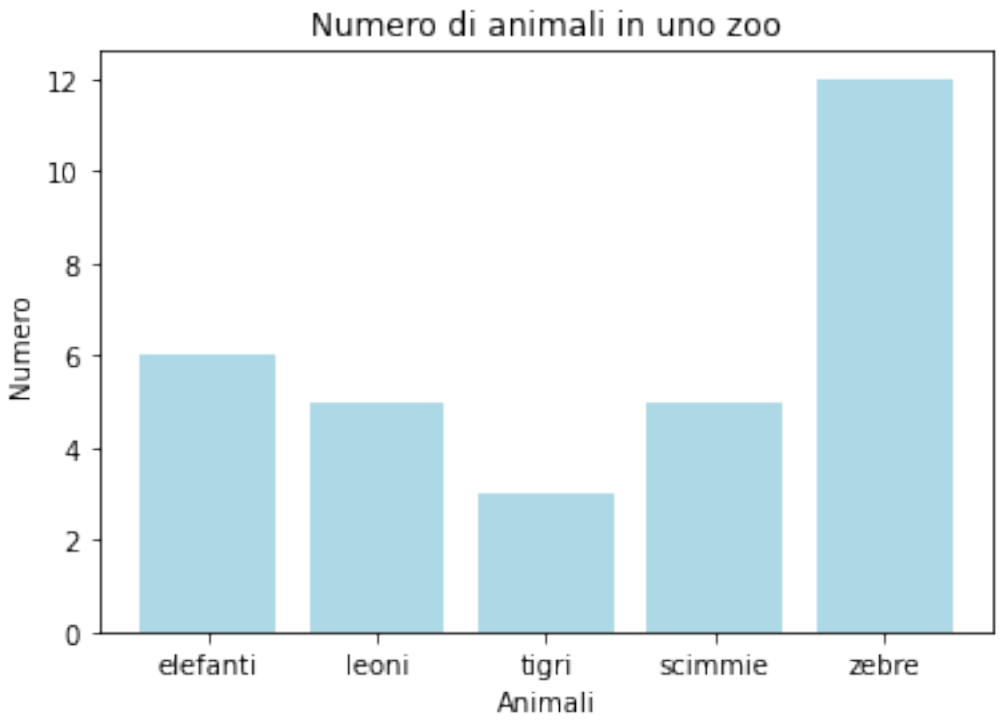
SECONDA PARTE: I GRAFICI

grafico

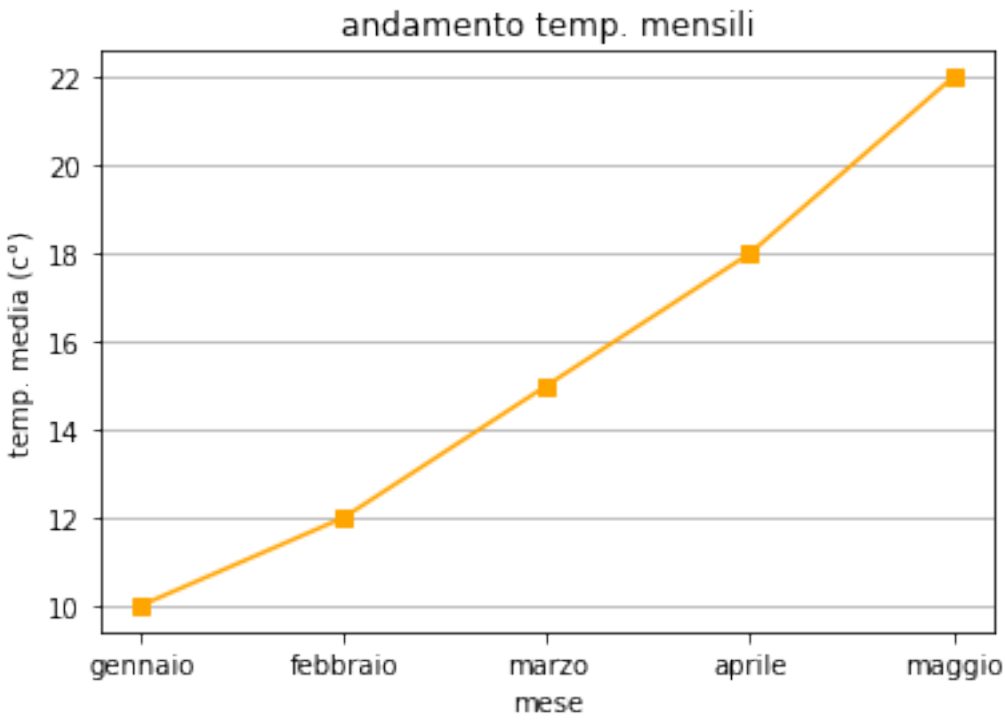
```
import matplotlib.pyplot as plt
animali = ["elefanti", "leoni", "tigri", "scimmie", "zebre"]
numero_animali = [6,4,3,7,9]
plt.bar(animali, numero_animali, color="red")
plt.show()
```



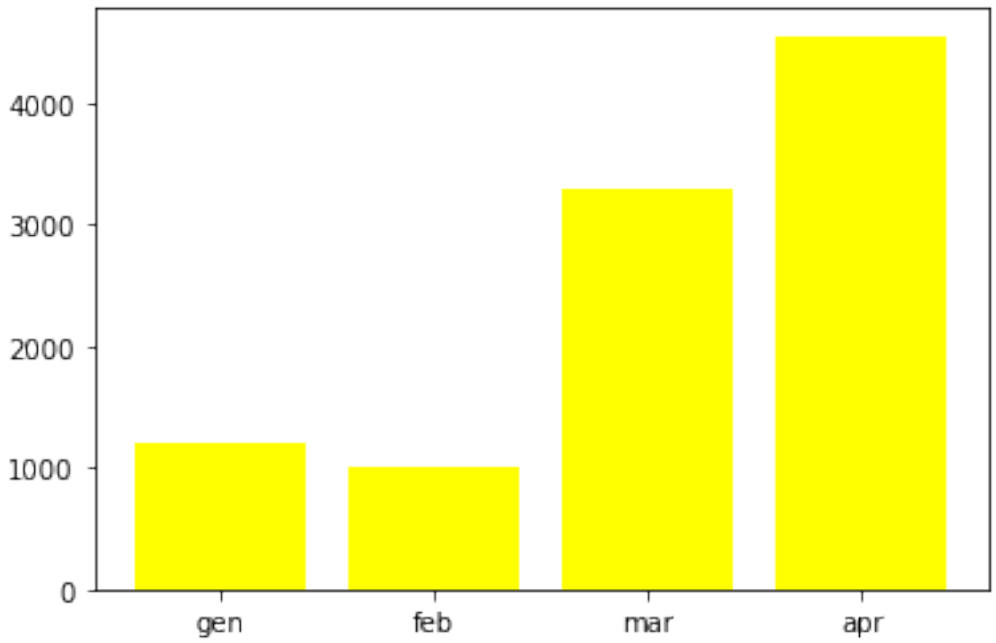

```
import matplotlib.pyplot as plt
animali = ["elefanti", "leoni", "tigri", "scimmie", "zebre"]
numero_animali = [6,5,3,5,12]
plt.bar(animali, numero_animali, color="lightblue")
plt.title("Numero di animali in uno zoo")
plt.xlabel("Animali")
plt.ylabel("Numero")

plt.show()
```



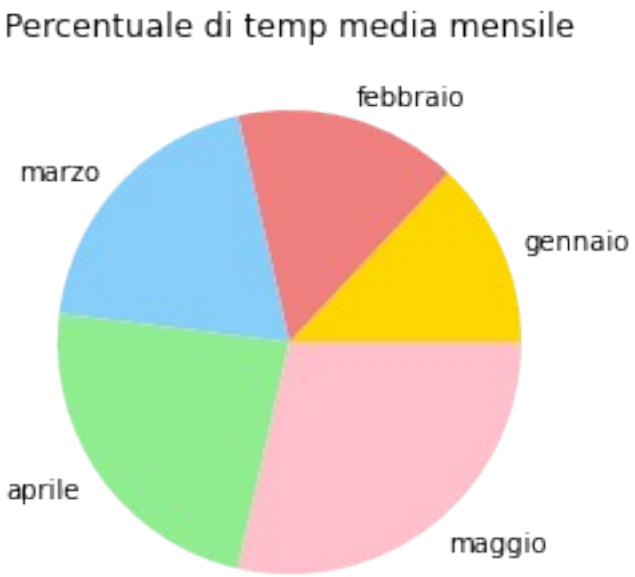

```
import matplotlib.pyplot as plt
mese = ["gennaio", "febbraio", "marzo", "aprile", "maggio"]
temperatura_media = [10,12,15,18,22]
plt.plot(mese, temperatura_media, marker="s", linestyle="-",
color="orange")
plt.title("andamento temp. mensili")
plt.xlabel("mese")
plt.ylabel("temp. media (c°)")
plt.grid(True, axis="y")
plt.show()
```




```
vendite_mensili={
    "gen":1200,
    "feb":1000,
    "mar":3300,
    "apr":4555
}
plt.bar(vendite_mensili.keys(), vendite_mensili.values(),
color="yellow")
plt.show()
```

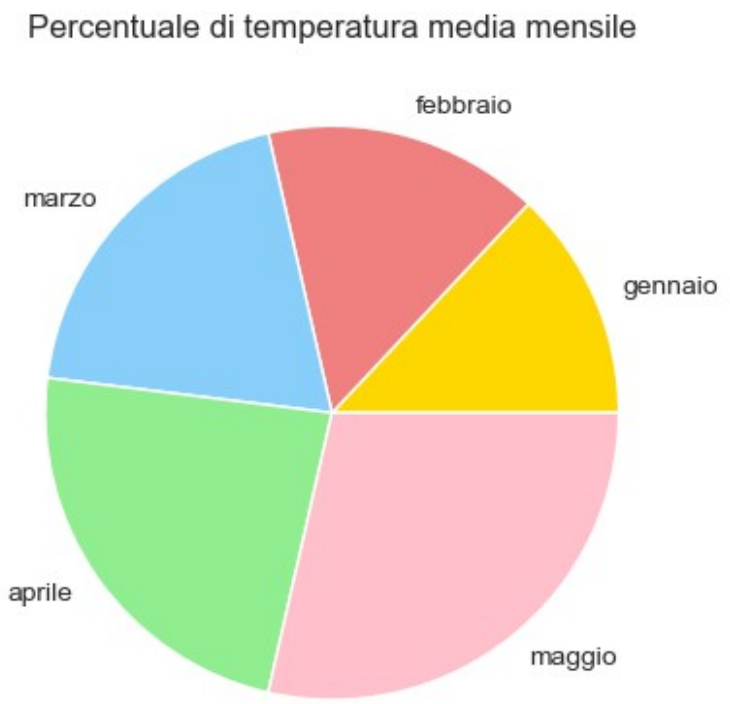



```
colori= ["gold", "lightcoral", "lightskyblue", "lightgreen", "pink"]
mese = ["gennaio", "febbraio", "marzo", "aprile", "maggio"]
temperatura_media = [10, 12, 15, 18, 22]
plt.pie(temperatura_media, labels=mese, colors=colori)
plt.title("Percentuale di temp media mensile")
plt.show()
```

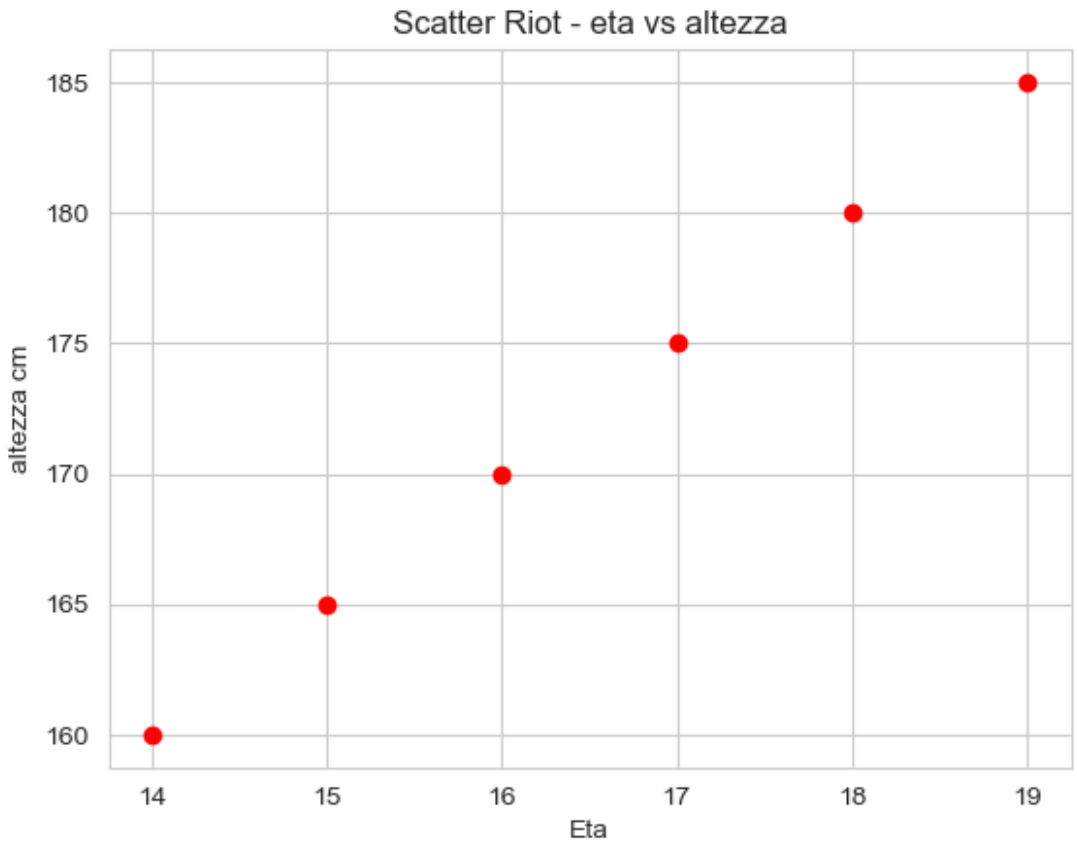



```
import matplotlib.pyplot as plt
```

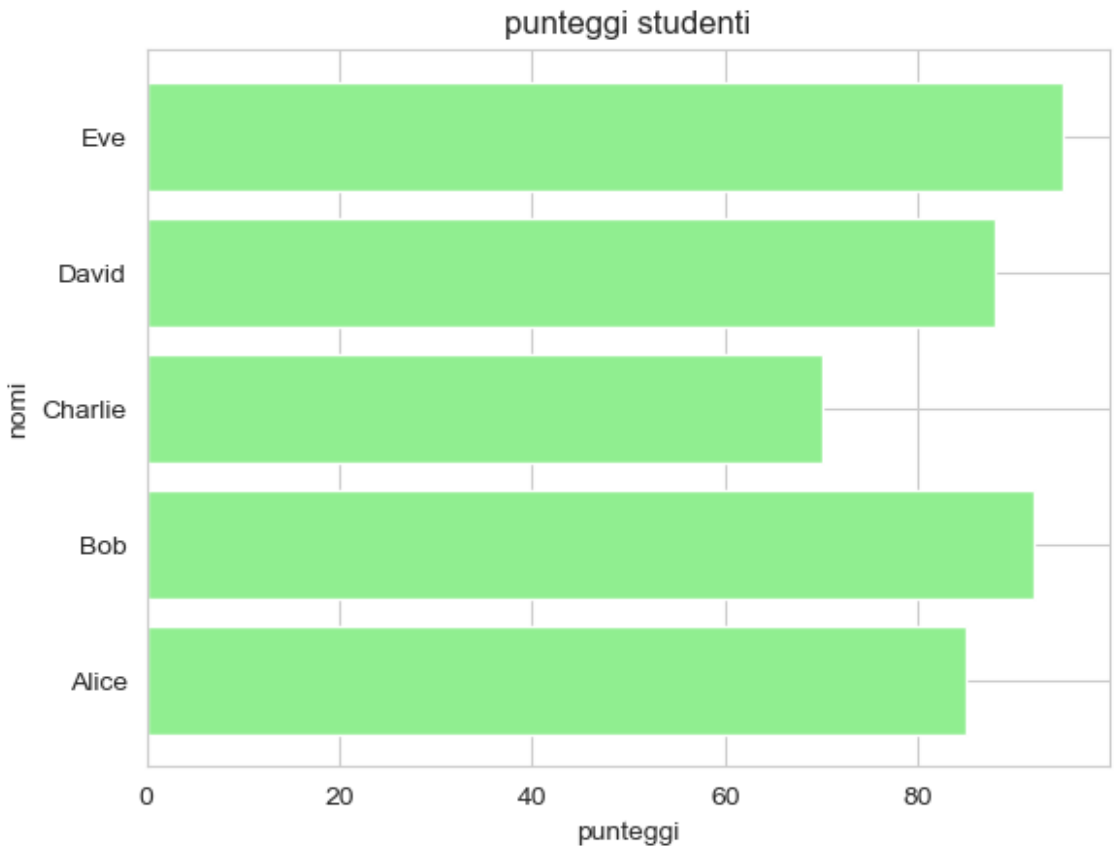
```
colori = ["gold", "lightcoral", "lightskyblue", "lightgreen", "pink"]
mese = {
    "gennaio": 10,
    "febbraio": 12,
    "marzo": 15,
    "aprile": 18,
    "maggio": 22
}

plt.pie(mese.values(), labels=mese.keys(), colors=colori)
plt.title('Percentuale di temperatura media mensile')
plt.show()
```




```
import matplotlib.pyplot as plt
eta = [14,15,16,17,18,19]
altezza = [160,165,170,175,180,185]
plt.scatter(eta,altezza,color="red",marker="o")
plt.title("Scatter Riot - eta vs altezza")
plt.xlabel("Eta")
plt.ylabel("altezza cm")
plt.grid(True)
plt.show()
```




```
import matplotlib.pyplot as plt
nomi_studenti = ['Alice', 'Bob', 'Charlie', 'David', 'Eve']
punteggi = [85, 92, 70, 88, 95]

plt.barh(nomi_studenti, punteggi, color='lightgreen')
plt.title('punteggi studenti')
plt.xlabel("punteggi")
plt.ylabel("nomi")
plt.show()
```




```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Genera dati di esempio
data = {
    'Feature1': [1, 2, np.nan, 4, 5],
    'Feature2': [np.nan, 2, 3, 4, np.nan],
    'Feature3': [1, np.nan, 3, 4, 5]
}

# Crea un DataFrame
df = pd.DataFrame(data)

# Calcola la matrice di missing values
missing_matrix = df.isnull()
missing_matrix
```

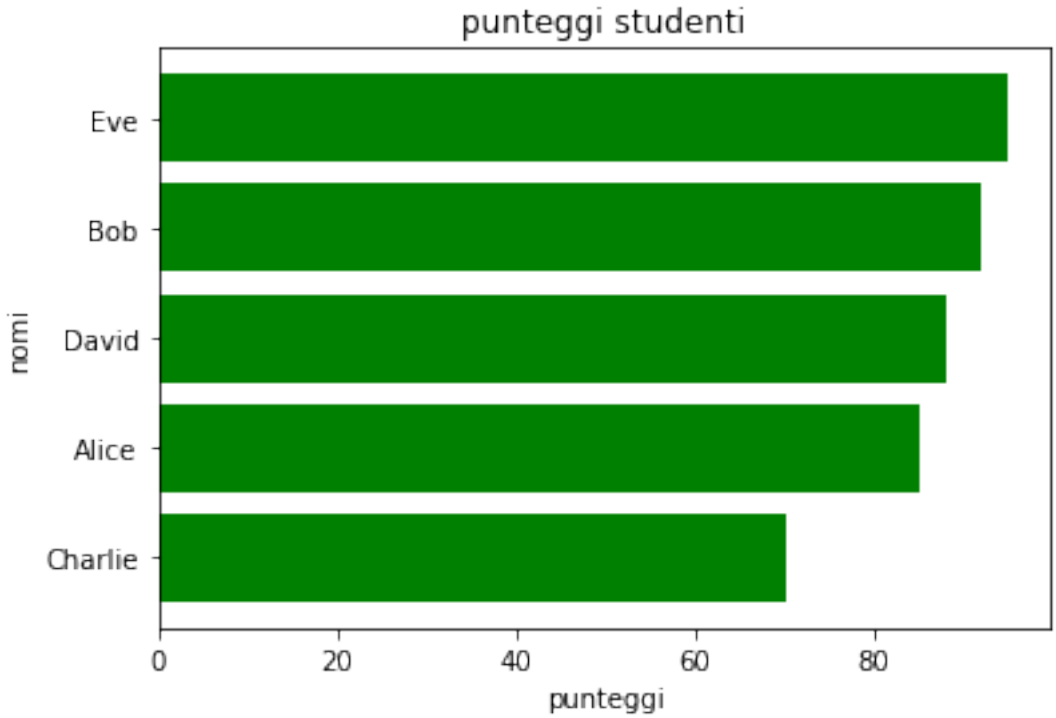
```
   Feature1  Feature2  Feature3
0     False      True     False
1     False     False      True
2      True     False     False
```

```
3     False     False     False
4     False      True     False
```

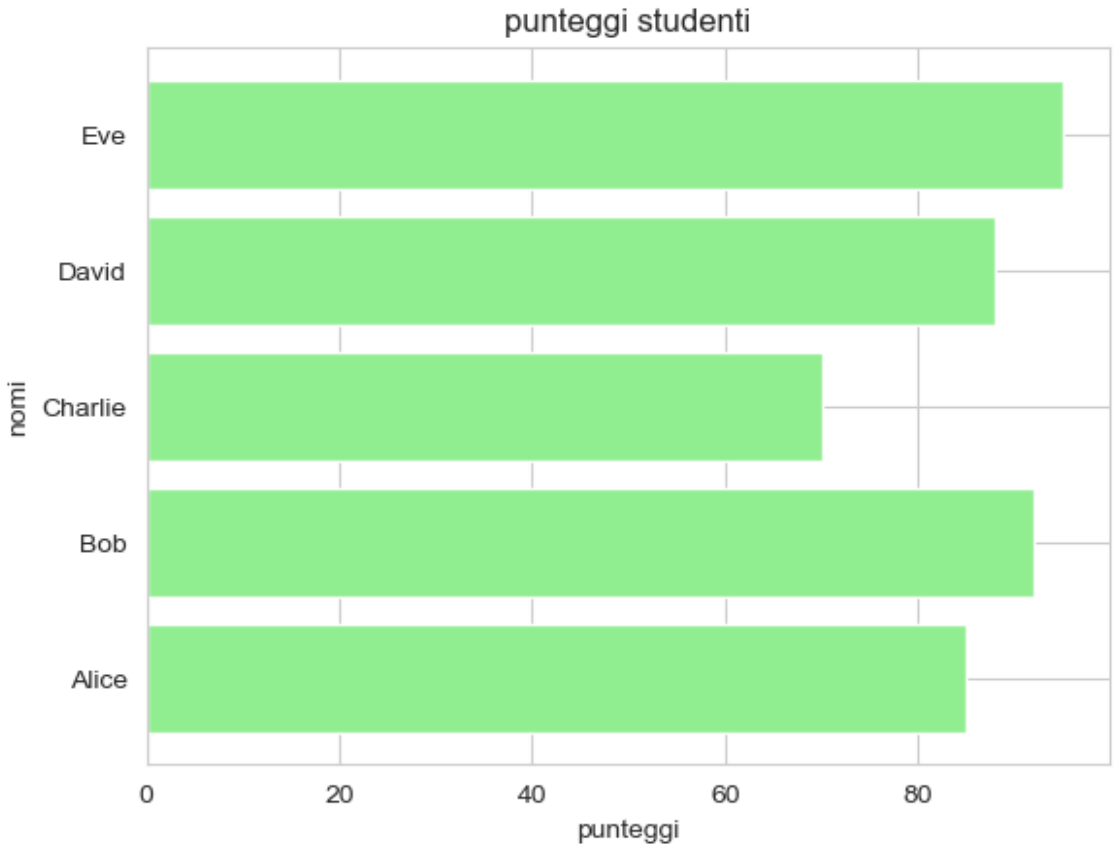
```
import matplotlib.pyplot as plt
import pandas as pd

nomi_studenti = ['Alice', 'Bob', 'Charlie', 'David', 'Eve']
punteggi = [85, 92, 70, 88, 95]

data = {'nome dello studente': nomi_studenti, 'punteggio': punteggi}
df = pd.DataFrame(data)
df.sort_values(by='punteggio', inplace=True)
df
```

```
  nome dello studente  punteggio
2             Charlie         70
0               Alice         85
3               David         88
1                 Bob         92
4                 Eve         95
```

```
plt.barh(df['nome dello studente'], df['punteggio'], color='green')
plt.title('punteggi studenti')
plt.xlabel("punteggi")
plt.ylabel("nomi")
plt.show()
```



```
import matplotlib.pyplot as plt
import pandas as pd

nomi_studenti = ['Alice', 'Bob', 'Charlie', 'David', 'Eve']
punteggi = [85, 92, 70, 88, 95]

data = {'nome dello studente': nomi_studenti, 'punteggio': punteggi}
df = pd.DataFrame(data)
df.sort_values(by='punteggio', inplace=False)

plt.barh(df['nome dello studente'], df['punteggio'],
color='lightgreen')
plt.title('punteggi studenti')
plt.xlabel("punteggi")
plt.ylabel("nomi")
plt.show()
```




```
import pandas as pd
import numpy as np

np.random.seed(41)

df = pd.DataFrame()
```

```
n_rows = 10000
df['CatCol1'] = np.random.choice(['A', 'B', 'C'], size=n_rows)
df['CatCol2'] = np.random.choice(['X', 'Y'], size=n_rows)
df['NumCol1'] = np.random.randn(n_rows)
df['NumCol2'] = np.random.randint(1, 100, size=n_rows)
df['NumCol3'] = np.random.uniform(0, 1, size=n_rows)

total_missing_values = int(0.03 * n_rows * len(df.columns))

for column in df.columns:
    num_missing_values = np.random.randint(0, total_missing_values +
1)
    missing_indices = np.random.choice(n_rows,
size=num_missing_values, replace=False)
    df.loc[missing_indices, column] = np.nan
df
```

```
     CatCol1 CatCol2   NumCol1  NumCol2   NumCol3
0          A     NaN  0.440877     49.0  0.246007
1          A       Y  1.945879     28.0  0.936825
2          C       X  0.988834     42.0  0.751516
3          A       Y -0.181978     73.0  0.950696
4          B       X  2.080615     74.0  0.903045
...      ...     ...       ...      ...       ...
9995       C       Y  1.352114     61.0  0.728445
9996       C       Y  1.143642     67.0  0.605930
9997       A       X -0.665794     54.0  0.071041
9998       C       Y  0.004278      NaN       NaN
9999       A       X  0.622473     95.0  0.751384

[10000 rows x 5 columns]
```

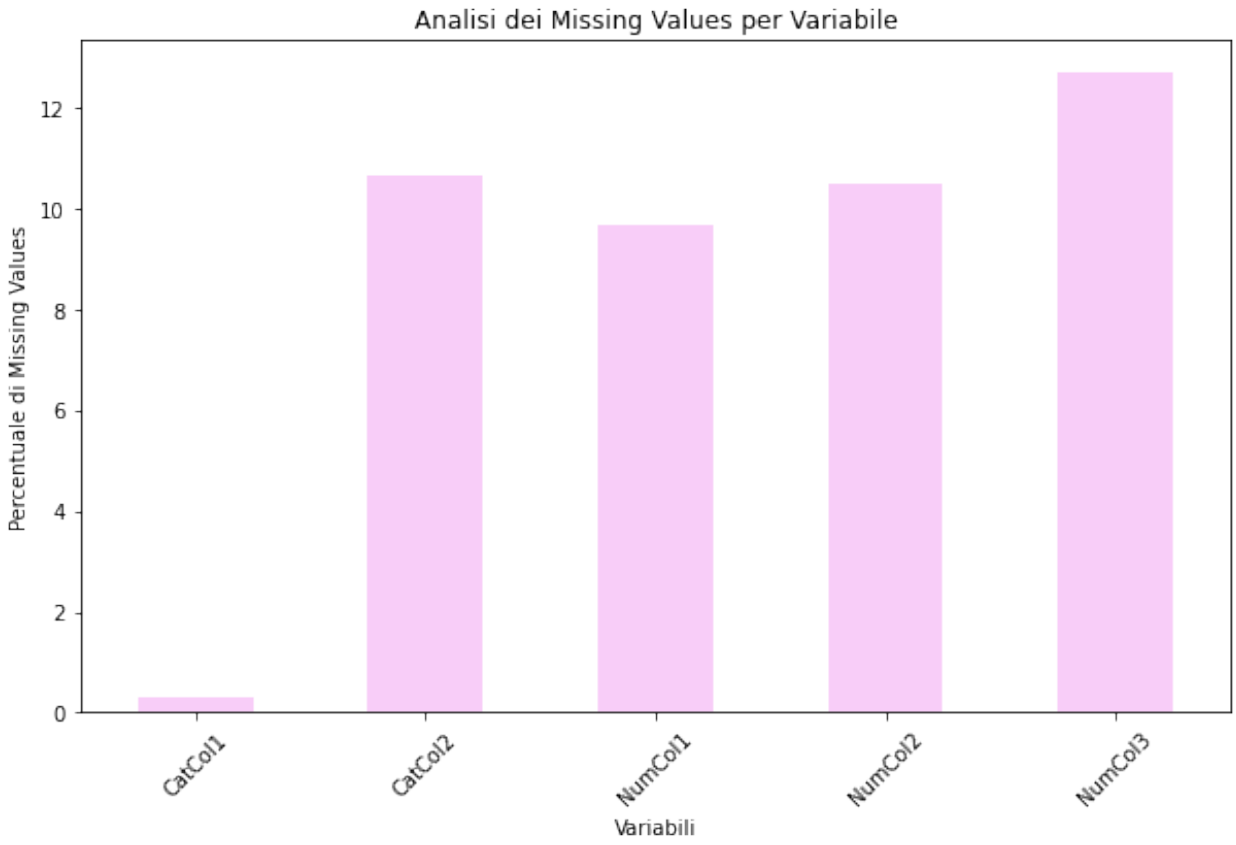
```
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
righe_con_dati_mancanti
```

```
     CatCol1 CatCol2   NumCol1  NumCol2   NumCol3
0          A     NaN  0.440877     49.0  0.246007
5          B     NaN       NaN     71.0  0.752397
6          B       X  0.080686     31.0       NaN
8          B       Y -1.291483      NaN  0.868791
12         C       Y -1.193705      8.0       NaN
...      ...     ...       ...      ...       ...
9986       C       X -0.909994      NaN  0.767918
9988       A       Y       NaN     35.0  0.149513
9989       A     NaN -0.148047      NaN  0.326089
9992       A       Y -0.048300     58.0       NaN
9998       C       Y  0.004278      NaN       NaN

[3648 rows x 5 columns]
```

```
missing_percent = (df.isnull().sum() / len(df)) * 100
missing_percent
```

```
CatCol1     0.29
CatCol2    10.63
NumCol1     9.67
NumCol2    10.48
NumCol3    12.69
dtype: float64
```

```
plt.figure(figsize=(10,6))
missing_percent.plot(kind='bar', color='violet', alpha=0.4)
plt.xlabel('Variabili')
plt.ylabel('Percentuale di Missing Values')
plt.title('Analisi dei Missing Values per Variabile')
plt.xticks(rotation=45)
plt.show()
```

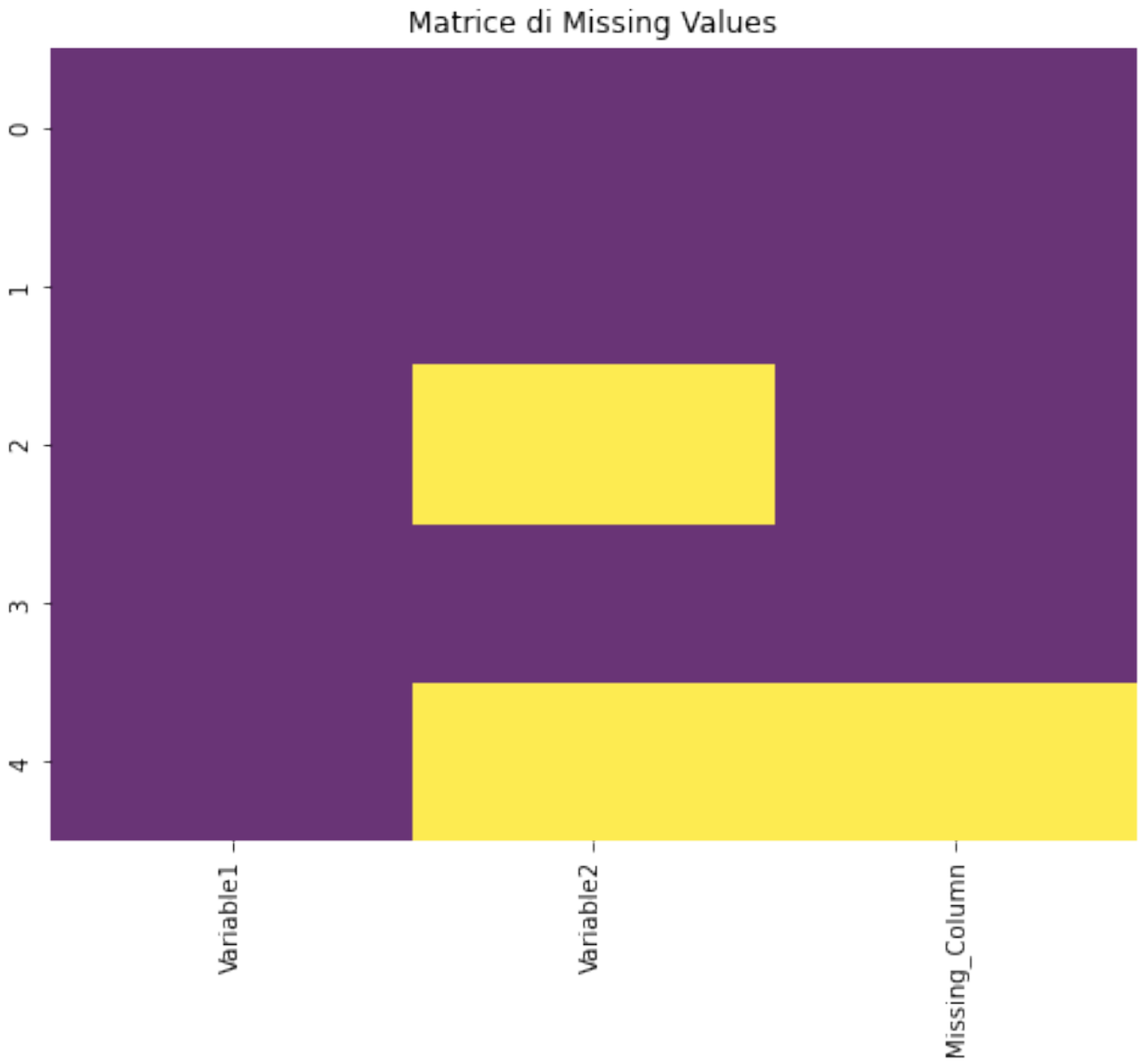



```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

```
data = {
    'Variable1': [1, 2, 3, 4, 5],
    'Variable2': [1, 2, np.nan, 4, np.nan],
    'Missing_Column': ['A', 'B', 'A', 'C', np.nan]
}

df = pd.DataFrame(data)
df1=pd.DataFrame()
df
```

	Variable1	Variable2	Missing_Column
0	1	1.0	A
1	2	2.0	B
2	3	NaN	A
3	4	4.0	C
4	5	NaN	NaN

```
missing_matrix = df.isnull()

plt.figure(figsize=(8,6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=False, alpha=0.8)
plt.title('Matrice di Missing Values')
plt.xticks(rotation=90)

plt.show()
```




```
import pandas as pd

dataset = [
    {"età": 25, "punteggio": 90, "ammesso": 1},
    {"età": None, "punteggio": 85, "ammesso": 0},
    {"età": 28, "punteggio": None, "ammesso": 1},
    {"età": None, "punteggio": 75, "ammesso": 1},
    {"età": 23, "punteggio": None, "ammesso": None},
    {"età": 23, "punteggio": 77, "ammesso": None},
]
df = pd.DataFrame(dataset)
df
```

```
    età  punteggio  ammesso
0  25.0       90.0      1.0
```

```
1   NaN       85.0      0.0
2  28.0        NaN      1.0
3   NaN       75.0      1.0
4  23.0        NaN      NaN
5  23.0       77.0      NaN
```

```
df['punteggio']
```

```
0    90.0
1    85.0
2     NaN
3    75.0
4     NaN
5    77.0
Name: punteggio, dtype: float64
```

```
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
righe_con_dati_mancanti
```

```
    età  punteggio  ammesso
1   NaN       85.0      0.0
2  28.0        NaN      1.0
3   NaN       75.0      1.0
4  23.0        NaN      NaN
5  23.0       77.0      NaN
```

```
print('righe con dati mancanti:')
print(righe_con_dati_mancanti)
print('Totale dati mancanti: ',totale_dati_mancanti)
```

```
righe con dati mancanti:
    età  punteggio  ammesso
1   NaN       85.0      0.0
2  28.0        NaN      1.0
3   NaN       75.0      1.0
4  23.0        NaN      NaN
5  23.0       77.0      NaN
Totale dati mancanti:  5
```

```
import pandas as pd

dataset = [
    {"nome": "Anna", "età": 25, "punteggio": 90, "email":
"anna@email.com"},
    {"nome": "Dario", "età": 22, "punteggio": None, "email": None},
    {"nome": "Carolina", "età": 28, "punteggio": 75, "email":
"carolina@email.com"},
]

df = pd.DataFrame(dataset)
df
```

```
       nome  età  punteggio               email
0      Anna   25       90.0      anna@email.com
1     Dario   22        NaN                None
2  Carolina   28       75.0  carolina@email.com
```

```
df1=df.dropna(inplace=False)

df1
```

```
       nome  età  punteggio               email
0      Anna   25       90.0      anna@email.com
2  Carolina   28       75.0  carolina@email.com
```

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

data = {
    'Variable1': [1, 2, 3, 4, 5],
    'Variable2': [1, 2, np.nan, 4, np.nan],
    'Missing_Column': ['A', 'B', 'A', 'C', np.nan]
}

df = pd.DataFrame(data)
df1=pd.DataFrame()
df
```

```
   Variable1  Variable2 Missing_Column
0          1        1.0              A
1          2        2.0              B
2          3        NaN              A
3          4        4.0              C
4          5        NaN            NaN
```

```
numeric_cols = df.select_dtypes(include=['number'])
numeric_cols
```

```
   Variable1  Variable2
0          1        1.0
1          2        2.0
2          3        NaN
3          4        4.0
4          5        NaN
```

```
df1[numeric_cols.columns] =
df[numeric_cols.columns].fillna(df[numeric_cols.columns].mean())
df1
```

```
   Variable1  Variable2
0          1   1.000000
1          2   2.000000
2          3   2.333333
3          4   4.000000
4          5   2.333333
```

```
df1[numeric_cols.columns] =
df[numeric_cols.columns].fillna(df[numeric_cols.columns].mean().iloc[1
])
df1
```

```
   Variable1  Variable2
0          1   1.000000
1          2   2.000000
2          3   2.333333
3          4   4.000000
4          5   2.333333
```

```
print(f"il primo coi valori mancati \n{df} \ne")
```

```
il primo coi valori mancati
   Variable1  Variable2 Missing_Column
0          1        1.0              A
1          2        2.0              B
2          3        NaN              A
3          4        4.0              C
4          5        NaN            NaN
e
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

data = {
    'Feature1': [1, 2, np.nan, 4, 5],
    'Feature2': [np.nan, 2, 3, 4, np.nan],
    'Feature3': [1, np.nan, 3, 4, 5]
}
df = pd.DataFrame(data)
df
```

```
   Feature1  Feature2  Feature3
0       1.0       NaN       1.0
1       2.0       2.0       NaN
2       NaN       3.0       3.0
3       4.0       4.0       4.0
4       5.0       NaN       5.0
```

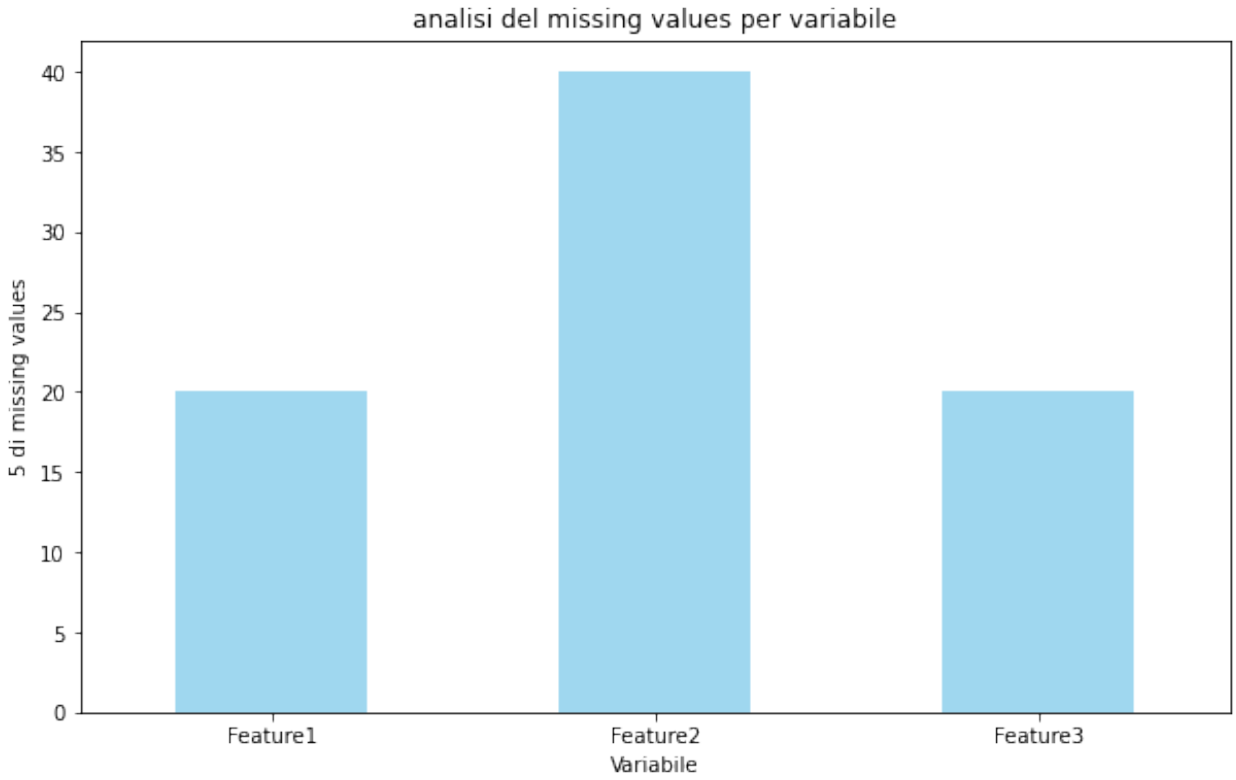
```
df.isnull()
```

```
   Feature1  Feature2  Feature3
0     False      True     False
1     False     False      True
2      True     False     False
3     False     False     False
4     False      True     False
```

```
missing_percent = df.isnull().sum() / len(df) * 100
missing_percent
```

```
Feature1    20.0
Feature2    40.0
Feature3    20.0
dtype: float64
```

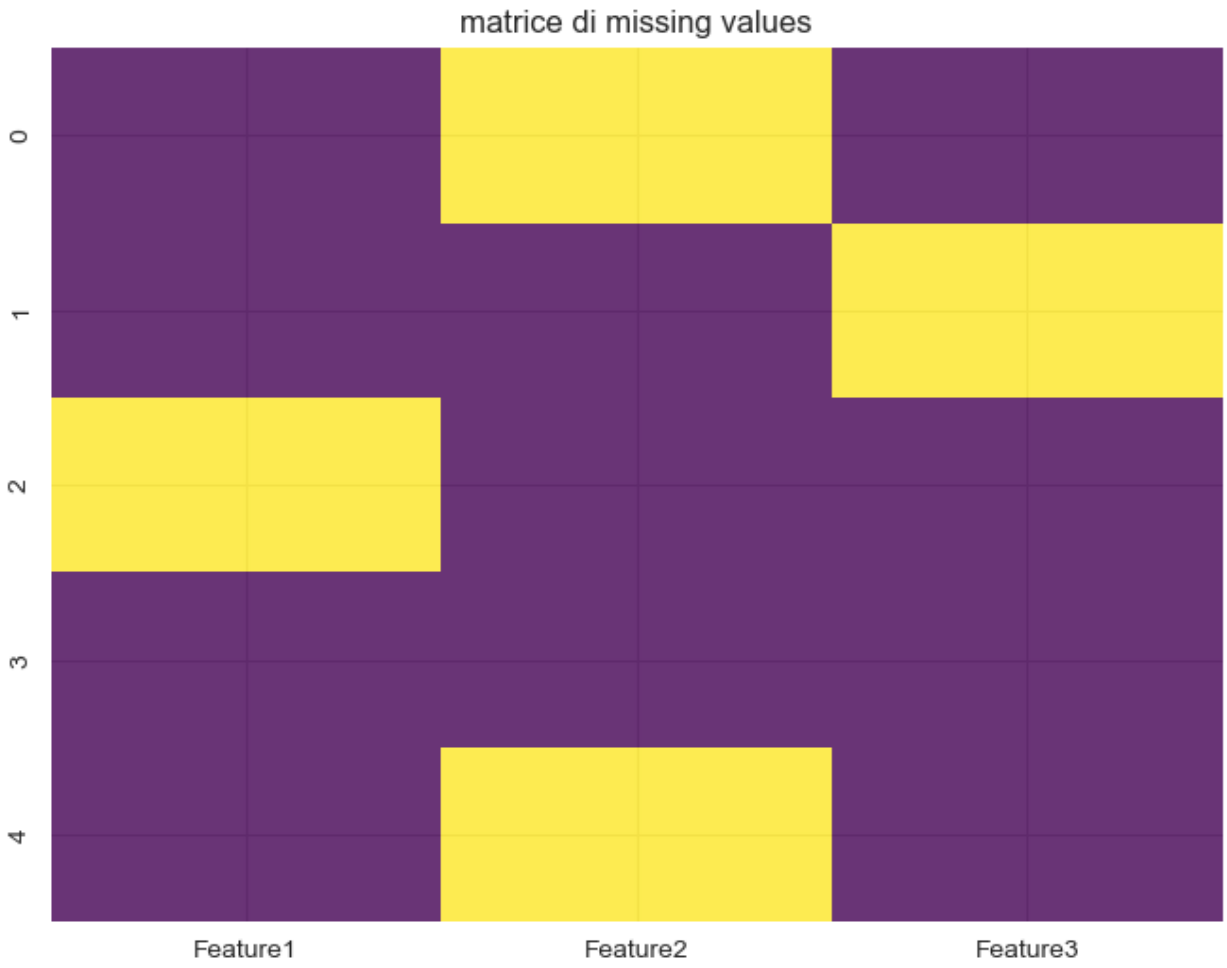
```
missing_percent= (df.isnull().sum()) / len(df) * 100

plt.figure(figsize=(10,6))
missing_percent.plot(kind='bar', color='skyblue', alpha=0.8)
plt.xlabel('Variabile')
plt.ylabel('5 di missing values')
plt.title('analisi del missing values per variabile')
plt.xticks(rotation=0)
plt.show()
```



```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

data = {
    'Feature1': [1, 2, np.nan, 4, 5],
    'Feature2': [np.nan, 2, 3, 4, np.nan],
    'Feature3': [1, np.nan, 3, 4, 5]
}

df = pd.DataFrame(data)

missing_matrix = df.isnull()
missing_matrix
```

	Feature1	Feature2	Feature3
0	False	True	False
1	False	False	True
2	True	False	False
3	False	False	False
4	False	True	False

```
plt.figure(figsize=(8,6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=False,alpha=0.8)
plt.title('matrice di missing values')
plt.show()
```




```
plt.figure(figsize=(8,6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=True,alpha=0.8)
plt.title('matrice di missing values')
plt.show()
```

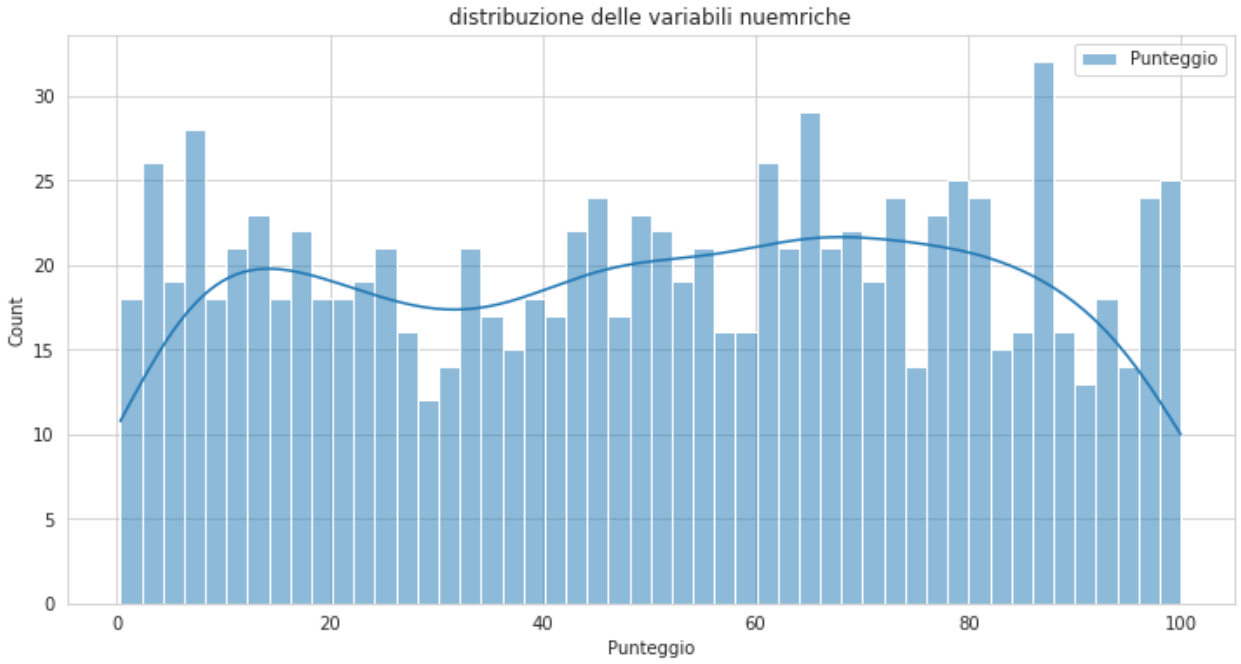

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

np.random.seed(42)
data = {
    'Età': np.random.randint(18, 70, size=1000),
    'Genere': np.random.choice(['Maschio', 'Femmina'], size=1000),
    'Punteggio': np.random.uniform(0, 100, size=1000),
    'Reddito': np.random.normal(50000, 15000, size=1000)
}

df = pd.DataFrame(data)

print(df.head())
```

```
   Età   Genere  Punteggio       Reddito
0   56  Maschio  85.120691  52915.764524
1   69  Maschio  49.514653  44702.505608
2   46  Maschio  48.058658  55077.257652
3   32  Femmina  59.240778  45568.978848
4   60  Maschio  82.468097  52526.914644
```

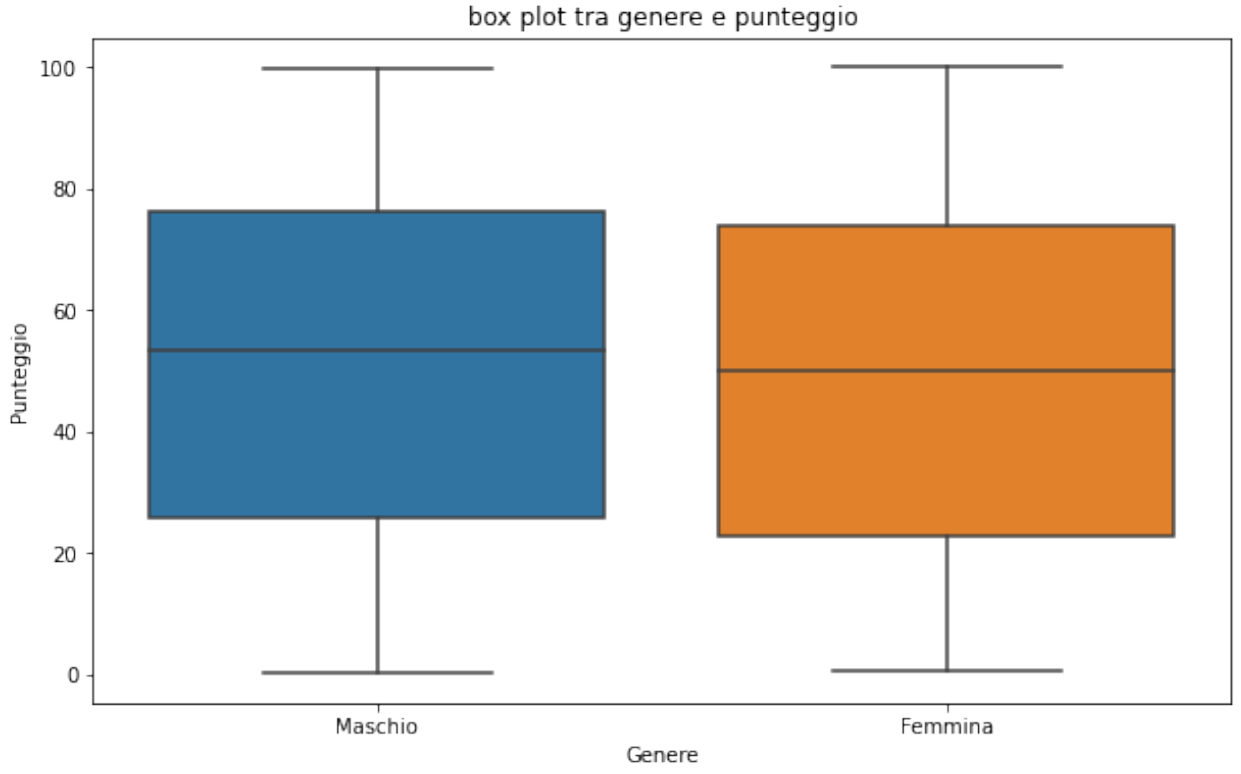
```
print(df.info())

print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Età        1000 non-null   int64  
 1   Genere     1000 non-null   object 
 2   Punteggio  1000 non-null   float64
 3   Reddito    1000 non-null   float64
dtypes: float64(2), int64(1), object(1)
memory usage: 31.4+ KB
None
              Età    Punteggio       Reddito
count  1000.00000  1000.000000   1000.000000
mean     43.81900    50.471078  50241.607607
std      14.99103    29.014970  14573.000585
min      18.00000     0.321826   4707.317663
25%      31.00000    24.690382  40538.177863
50%      44.00000    51.789520  50099.165858
75%      56.00000    75.549365  60089.683773
max      69.00000    99.941373  97066.228005
```

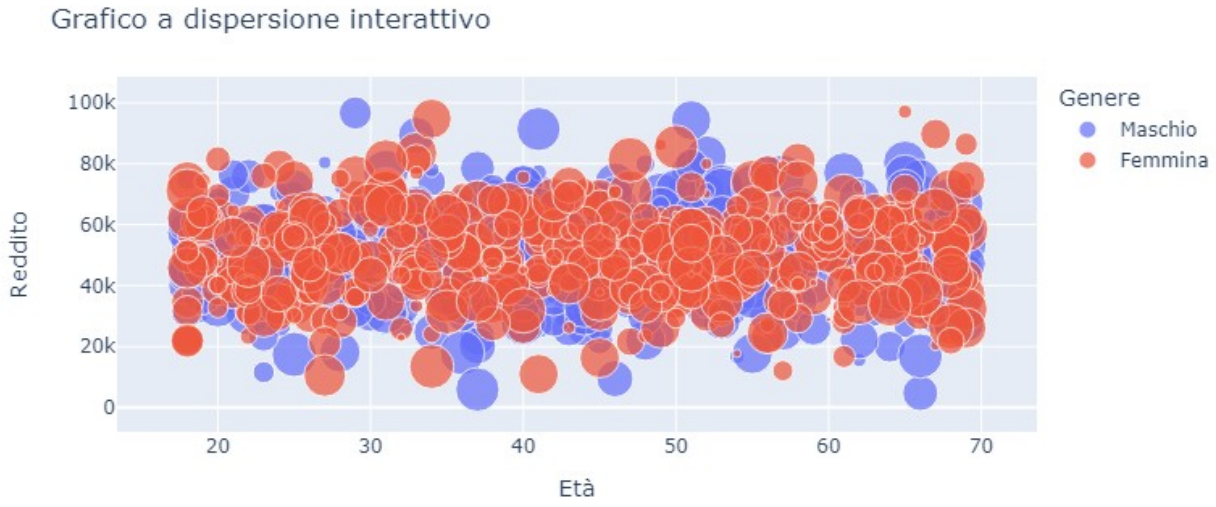
```
plt.figure(figsize=(12,6))
sns.set_style('whitegrid')
sns.histplot(df['Punteggio'], kde=True, bins=50, label='Punteggio')
plt.legend()
plt.title('distribuzione delle variabili nuemriche')
plt.show()
```




```
plt.figure(figsize=(10,6))
sns.boxplot(x='Genere',y='Punteggio', data=df)
plt.title('box plot tra genere e punteggio')
plt.show()
```



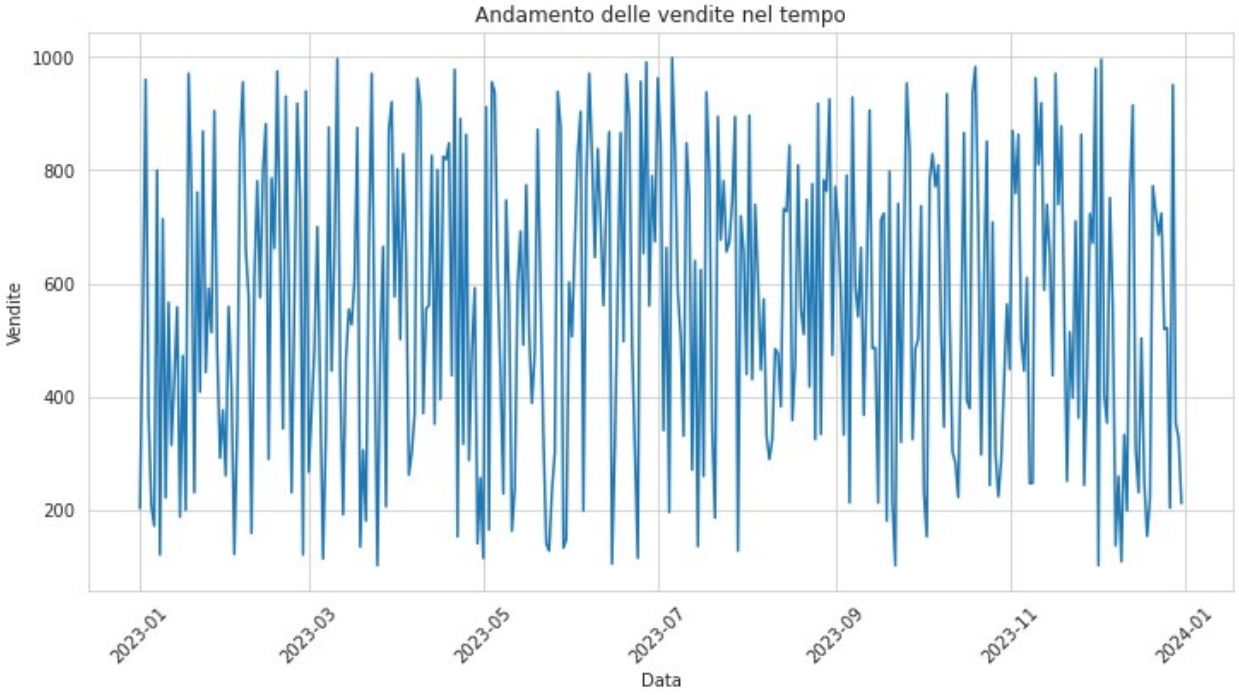
```
import plotly.express as px
fig = px.scatter(df, x='Età', y='Reddito', color='Genere',
size='Punteggio')
fig.update_layout(title='Grafico a dispersione interattivo')
fig.show()
```

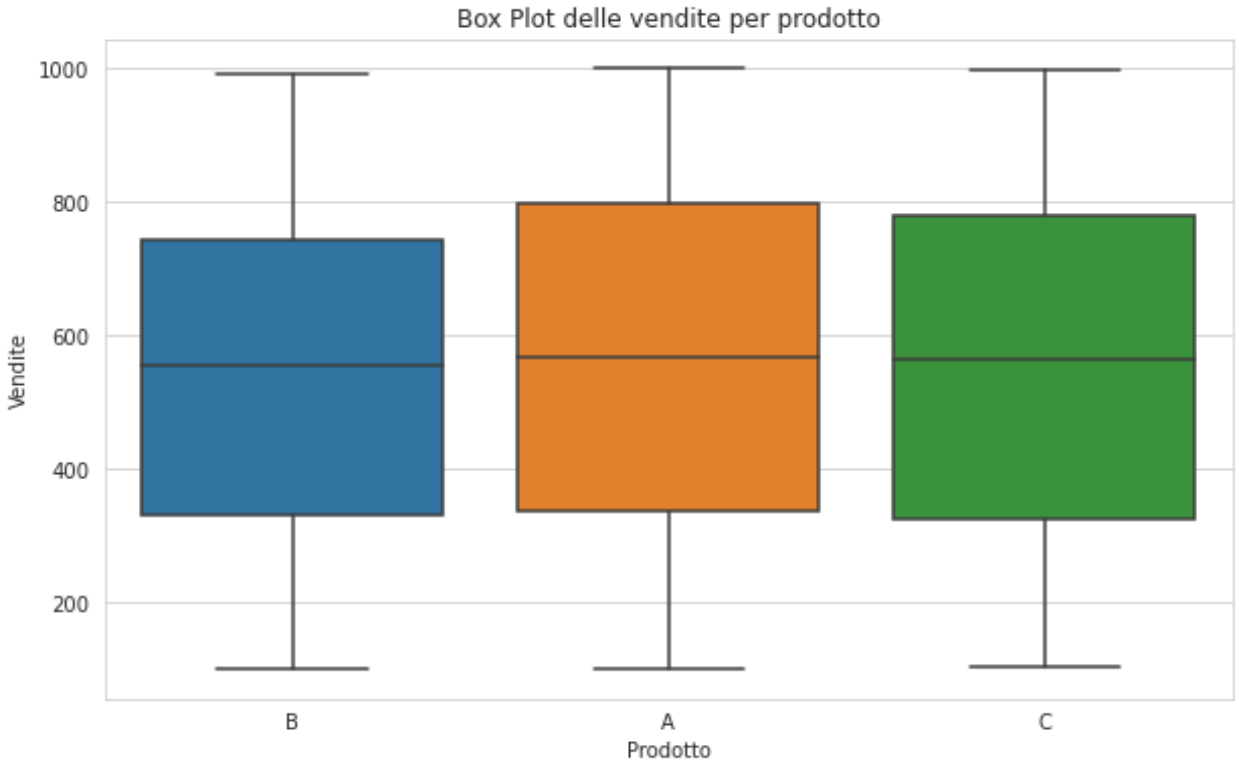



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

np.random.seed(42)
data = {
    'Data': pd.date_range(start='2023-01-01', end='2023-12-31',
freq='D'), #data casuale
    'Vendite': np.random.randint(100, 1000, size=365), #100 numeri
casuali tra1 100 e 1000
    'Prodotto': np.random.choice(['A', 'B', 'C'], size=365) #
}

df = pd.DataFrame(data)

print(df.head())
```

```
        Data  Vendite Prodotto
0 2023-01-01      202        B
1 2023-01-02      535        A
2 2023-01-03      960        C
3 2023-01-04      370        A
4 2023-01-05      206        A
```

```
plt.figure(figsize=(12, 6))
sns.lineplot(x='Data', y='Vendite', data=df)
plt.title('Andamento delle vendite nel tempo')
plt.xlabel('Data')
plt.ylabel('Vendite')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(10, 6))
sns.boxplot(x='Prodotto', y='Vendite', data=df)
plt.title('Box Plot delle vendite per prodotto')
plt.xlabel('Prodotto')
plt.ylabel('Vendite')
plt.show()
```



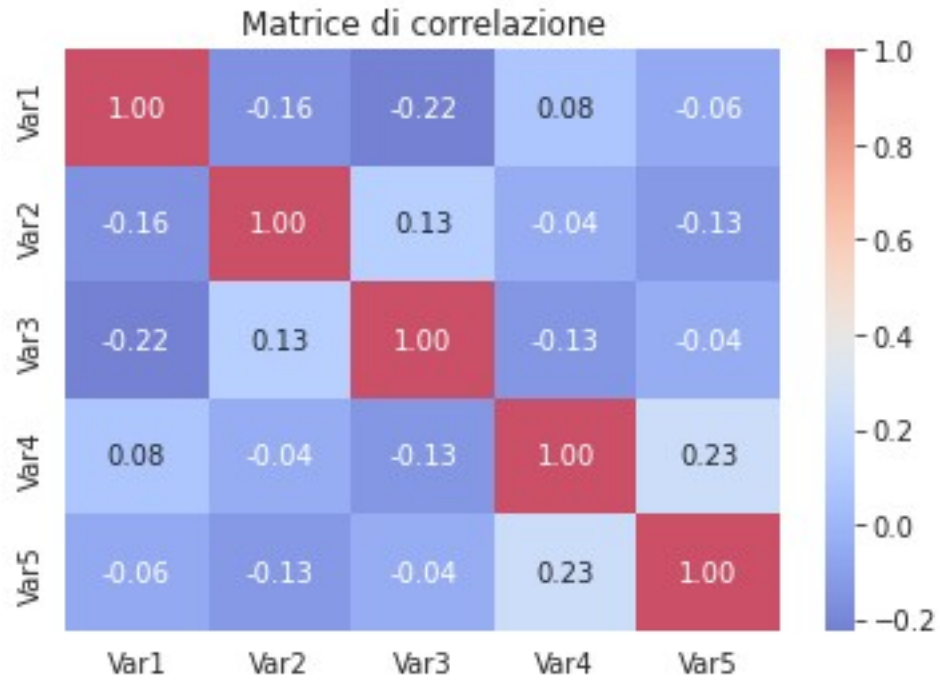



```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

# Genera dati di esempio
data = {
    'Numeric_Var': [1, 2, 3, 4, np.nan, 6],
    'Categorical_Var': ['A', 'B', 'A', 'B', 'A', 'B']
}

# Crea un DataFrame
df = pd.DataFrame(data)
print(df)
```

```
   Numeric_Var Categorical_Var
0          1.0               A
1          2.0               B
2          3.0               A
3          4.0               B
4          NaN               A
5          6.0               B
```

```python
import seaborn as sns
import matplotlib.pyplot as plt
```

```
np.random.seed(42)
data = pd.DataFrame(np.random.rand(100, 5), columns=['Var1', 'Var2',
'Var3', 'Var4', 'Var5'])

data['Categoria1'] = np.random.choice(['A', 'B', 'C'], size=100)
data['Categoria2'] = np.random.choice(['X', 'Y'], size=100)

correlation_matrix = data.corr()


sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f", alpha=0.7)

plt.title('Matrice di correlazione')
plt.show()
```




```
import pandas as pd
import numpy as np

np.random.seed(41)

df = pd.DataFrame()

n_rows = 10000
df['CatCol1'] = np.random.choice(['A', 'B', 'C'], size=n_rows)
df['CatCol2'] = np.random.choice(['X', 'Y'], size=n_rows)
df['NumCol1'] = np.random.randn(n_rows)
df['NumCol2'] = np.random.randint(1, 100, size=n_rows)
```

```
df['NumCol3'] = np.random.uniform(0, 1, size=n_rows)

total_missing_values = int(0.03 * n_rows * len(df.columns))

for column in df.columns:
    num_missing_values = np.random.randint(0, total_missing_values +
1)
    missing_indices = np.random.choice(n_rows,
size=num_missing_values, replace=False)
    df.loc[missing_indices, column] = np.nan
df
```

```
     CatCol1 CatCol2   NumCol1  NumCol2   NumCol3
0          A     NaN  0.440877     49.0  0.246007
1          A       Y  1.945879     28.0  0.936825
2          C       X  0.988834     42.0  0.751516
3          A       Y -0.181978     73.0  0.950696
4          B       X  2.080615     74.0  0.903045
...      ...     ...       ...      ...       ...
9995       C       Y  1.352114     61.0  0.728445
9996       C       Y  1.143642     67.0  0.605930
9997       A       X -0.665794     54.0  0.071041
9998       C       Y  0.004278      NaN       NaN
9999       A       X  0.622473     95.0  0.751384

[10000 rows x 5 columns]
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

percorso_file_csv = "pokemons.csv"

df = pd.read_csv(percorso_file_csv)

print(df.head())
```

```
   id        name      rank    generation evolves_from  type1   type2
hp  \
0   1   bulbasaur  ordinary  generation-i      nothing  grass  poison
45
1   2     ivysaur  ordinary  generation-i    bulbasaur  grass  poison
60
2   3    venusaur  ordinary  generation-i      ivysaur  grass  poison
80
3   4  charmander  ordinary  generation-i      nothing   fire    None
39
4   5  charmeleon  ordinary  generation-i   charmander   fire    None
58

   atk  def  spatk  spdef  speed  total  height  weight  \
```

```
0   49   49     65     65     45    318       7      69
1   62   63     80     80     60    405      10     130
2   82   83    100    100     80    525      20    1000
3   52   43     60     50     65    309       6      85
4   64   58     80     65     80    405      11     190

               abilities
desc
0  overgrow chlorophyll   A strange seed was planted on its back at
birt...
1  overgrow chlorophyll   When the bulb on its back grows large, it
appe...
2  overgrow chlorophyll   The plant blooms when it is absorbing solar
en...
3     blaze solar-power   Obviously prefers hot places. When it rains,
s...
4     blaze solar-power   When it swings its burning tail, it elevates
t...
```

```
import pandas as pd

percorso_file_excel = "serieA.xlsx"

df = pd.read_excel(percorso_file_excel, sheet_name='09-10')

df
```

```
    position                    team  Pt  Played  Won  Net  lose  Goals
made  \
0          1             Inter Inter  82      38   24   10     4
75
1          2               Roma Roma  80      38   24    8     6
68
2          3             Milan Milan  70      38   20   10     8
60
3          4    Sampdoria Sampdoria  67      38   19   10     9
49
4          5         Palermo Palermo  65      38   18   11     9
59
5          6           Napoli Napoli  59      38   15   14     9
50
6          7      Juventus Juventus  55      38   16    7    15
55
7          8             Parma Parma  52      38   14   10    14
46
8          9             Genoa Genoa  51      38   14    9    15
57
9         10               Bari Bari  50      38   13   11    14
49
10        11  Fiorentina Fiorentina  47      38   13    8    17
```

```
48
11        12            Lazio Lazio  46      38   11   13    14
39
12        13        Catania Catania  45      38   10   15    13
44
13        14          Chievo Chievo  44      38   12    8    18
37
14        15        Udinese Udinese  44      38   11   11    16
54
15        16      Cagliari Cagliari  44      38   11   11    16
56
16        17        Bologna Bologna  42      38   10   12    16
42
17        18      Atalanta Atalanta  35      38    9    8    21
37
18        19            Siena Siena  31      38    7   10    21
40
19        20        Livorno Livorno  29      38    7    8    23
27

    Goals suffered  Difference goals
0               34                41
1               41                27
2               39                21
3               41                 8
4               47                12
5               43                 7
6               56                -1
7               51                -5
8               61                -4
9               49                 0
10              47                 1
11              43                -4
12              45                -1
13              42                -5
14              59                -5
15              58                -2
16              55               -13
17              53               -16
18              67               -27
19              61               -34
```

```
import os
import pandas as pd

percorso_cartella = "serieAnuovo"

lista_dataframes = []

for nome_file in os.listdir(percorso_cartella):
```

```
    if nome_file.endswith(".csv"):
        percorso_file_csv = os.path.join(percorso_cartella, nome_file)
        df = pd.read_csv(percorso_file_csv)
    lista_dataframes.append(df)

df
```

```
    Div      Date    HomeTeam    AwayTeam  FTHG  FTAG FTR  HTHG  HTAG
HTR  \
0    I1  25/08/01     Bologna    Atalanta     1     0   H     1     0
H
1    I1  26/08/01     Brescia       Milan     2     2   D     2     0
H
2    I1  26/08/01  Fiorentina      Chievo     0     2   A     0     1
A
3    I1  26/08/01       Inter     Perugia     4     1   H     2     0
H
4    I1  26/08/01    Juventus     Venezia     4     0   H     3     0
H
..   ..       ...         ...         ...   ...   ...  ..   ...   ...
..
301  I1  05/05/02       Parma     Venezia     2     1   H     1     1
D
302  I1  05/05/02     Perugia  Fiorentina     2     0   H     2     0
H
303  I1  05/05/02    Piacenza      Verona     3     0   H     1     0
H
304  I1  05/05/02      Torino        Roma     0     1   A     0     0
D
305  I1  05/05/02     Udinese    Juventus     0     2   A     0     2
A

     ...  LBA   SBH    SBD    SBA   SYH   SYD    SYA    WHH   WHD
WHA
0    ...  NaN   NaN    NaN    NaN  1.91  3.00   3.75   1.90  3.00
3.75
1    ...  NaN   NaN    NaN    NaN  3.20  3.00   2.10   3.30  2.87
2.10
2    ...  NaN   NaN    NaN    NaN  1.40  3.75   7.00   1.44  3.75
6.00
3    ...  NaN   NaN    NaN    NaN  1.29  4.33   8.50   1.25  4.50
10.00
4    ...  NaN   NaN    NaN    NaN  1.20  5.00  11.00   1.20  5.00
11.00
..   ...  ...   ...    ...    ...   ...   ...    ...    ...   ...    .
..
301  ...  NaN   NaN    NaN    NaN   NaN   NaN    NaN   1.33  4.00
8.00
302  ...  NaN   NaN    NaN    NaN   NaN   NaN    NaN   1.30  4.33
8.00
```

```
303  ...  NaN   2.0  2.375  4.500   NaN   NaN    NaN   2.00  2.37
5.00
304  ...  NaN   8.0  4.000  1.300   NaN   NaN    NaN  10.00  4.50
1.25
305  ...  NaN  10.0  4.500  1.182   NaN   NaN    NaN    NaN   NaN
NaN

[306 rows x 28 columns]
```

```
lista_dataframes[0]
```

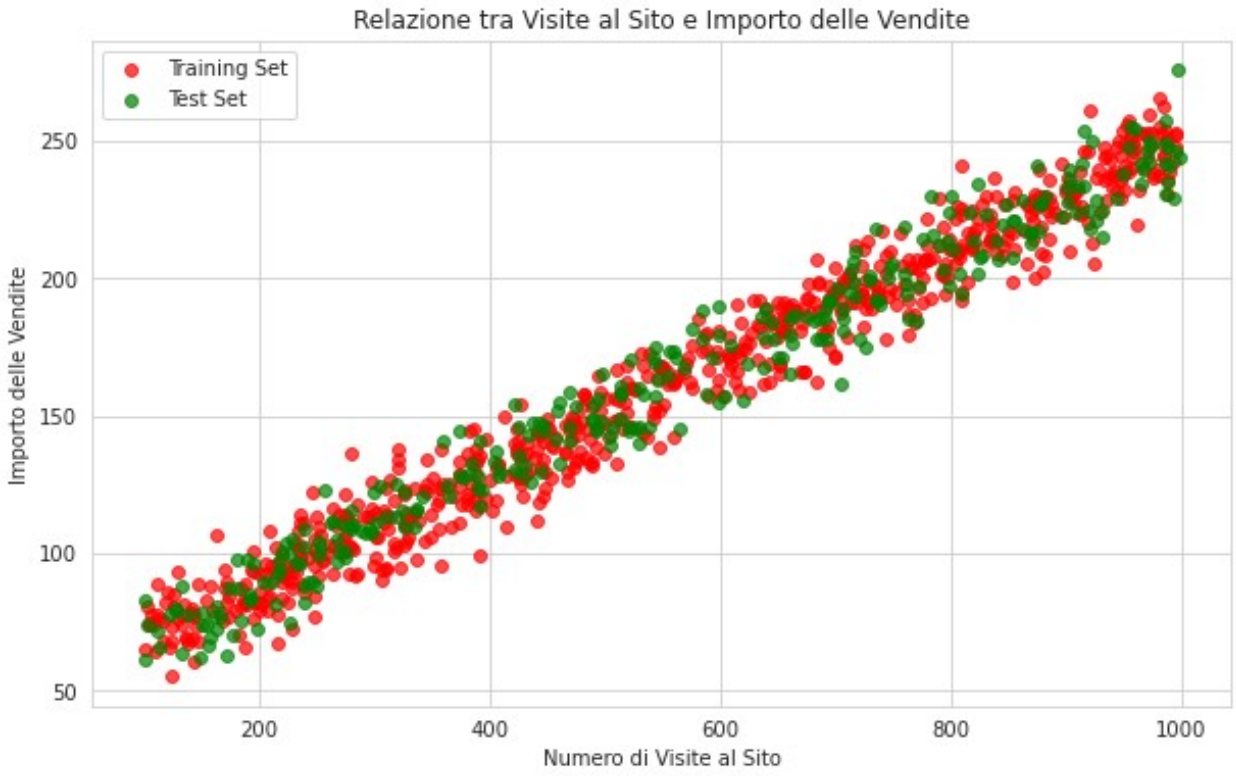
```
    Div      Date   HomeTeam   AwayTeam  FTHG  FTAG FTR
0    I1  29/08/93   Atalanta   Cagliari     5     2   H
1    I1  29/08/93      Genoa       Roma     2     0   H
2    I1  29/08/93      Inter   Reggiana     2     1   H
3    I1  29/08/93   Juventus  Cremonese     1     0   H
4    I1  29/08/93      Lazio     Foggia     0     0   D
..   ..       ...        ...        ...   ...   ...  ..
301  I1  01/05/94      Lecce   Cagliari     0     1   A
302  I1  01/05/94      Milan   Reggiana     0     1   A
303  I1  01/05/94      Parma   Piacenza     0     0   D
304  I1  01/05/94       Roma     Torino     2     0   H
305  I1  01/05/94  Sampdoria      Lazio     3     4   A

[306 rows x 7 columns]
```

```
import numpy as np
from sklearn.model_selection import train_test_split

np.random.seed(0)
altezze = np.random.normal(160, 10, 100)
pesi = 0.5 * altezze + np.random.normal(0, 5, 100)

X_train, X_test, y_train, y_test = train_test_split(altezze, pesi,
test_size=0.3, random_state=42)

print("dimensioni del training set (altezze e pesi):", X_train.shape,
y_train.shape)
print("dimensioni del test set (altezze e pesi:)", X_test.shape,
y_test.shape)
```

```
dimensioni del training set (altezze e pesi): (70,) (70,)
dimensioni del test set (altezze e pesi:) (30,) (30,)
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

np.random.seed(0)
visite_al_sito = np.random.randint(100, 1000, 1000)
importo_vendite = 50 + 0.2 * visite_al_sito + np.random.normal(0, 10,
```

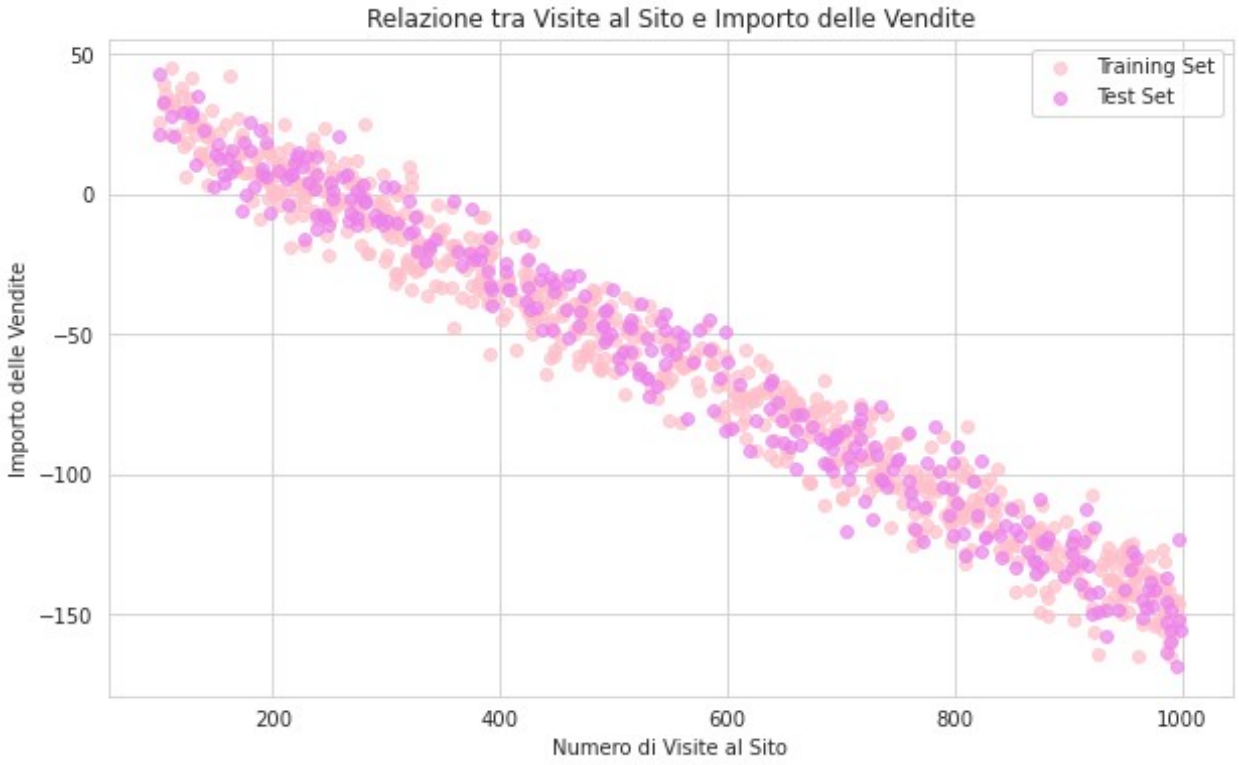
```
1000)

X_train, X_test, y_train, y_test = train_test_split(visite_al_sito,
importo_vendite, test_size=0.3, random_state=42)

plt.figure(figsize=(10, 6))
plt.scatter(X_train, y_train, label='Training Set', color='red',
alpha=0.7)
plt.scatter(X_test, y_test, label='Test Set', color='green',
alpha=0.7)
plt.xlabel('Numero di Visite al Sito')
plt.ylabel('Importo delle Vendite')
plt.title('Relazione tra Visite al Sito e Importo delle Vendite')
plt.legend()
plt.grid(True)
plt.show()

print("Dimensioni del Training Set (visite al sito e importo delle
vendite):", X_train.shape, y_train.shape)
print("Dimensioni del Test Set (visite al sito e importo delle
vendite):", X_test.shape, y_test.shape)
```




```
Dimensioni del Training Set (visite al sito e importo delle vendite):
(700,) (700,)
```

```
Dimensioni del Test Set (visite al sito e importo delle vendite):
(300,) (300,)
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

np.random.seed(0)
visite_al_sito = np.random.randint(100, 1000, 1000)
importo_vendite = 50 - 0.2 * visite_al_sito + np.random.normal(0, 10,
1000)

X_train, X_test, y_train, y_test = train_test_split(visite_al_sito,
importo_vendite, test_size=0.3, random_state=42)

plt.figure(figsize=(10, 6))
plt.scatter(X_train, y_train, label='Training Set', color='pink',
alpha=0.7)
plt.scatter(X_test, y_test, label='Test Set', color='violet',
alpha=0.7)
plt.xlabel('Numero di Visite al Sito')
plt.ylabel('Importo delle Vendite')
plt.title('Relazione tra Visite al Sito e Importo delle Vendite')
plt.legend()
plt.grid(True)
plt.show()

print("Dimensioni del Training Set (visite al sito e importo delle
vendite):", X_train.shape, y_train.shape)
print("Dimensioni del Test Set (visite al sito e importo delle
vendite):", X_test.shape, y_test.shape)
```




```
Dimensioni del Training Set (visite al sito e importo delle vendite):
(700,) (700,)
Dimensioni del Test Set (visite al sito e importo delle vendite):
(300,) (300,)
```

```
from sklearn.model_selection import train_test_split
import numpy as np

X = np.random.rand(100, 2)
y = np.random.choice(['A', 'B'], size=100)

proporzione_classe_A = sum(y == 'A') / len(y)
proporzione_classe_B = 1 - proporzione_classe_A

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

proporzione_classe_A_train = sum(y_train == 'A') / len(y_train)
proporzione_classe_B_train = 1 - proporzione_classe_A_train

proporzione_classe_A_test = sum(y_test == 'A') / len(y_test)
proporzione_classe_B_test = 1 - proporzione_classe_A_test



print("proporzione classe A nel dataset completo:",
proporzione_classe_A)
```

```
print("proporzione classe B nel dataset completo:",
proporzione_classe_B)
print("proporzione classe A nel training set:",
proporzione_classe_A_train)
print("proporzione classe B nel training set:",
proporzione_classe_B_train)
print("proporzione classe A nel test set:", proporzione_classe_A_test)
print("proporzione classe B nel test set:", proporzione_classe_B_test)
```

```
proporzione classe A nel dataset completo: 0.53
proporzione classe B nel dataset completo: 0.47
proporzione classe A nel training set: 0.44285714285714284
proporzione classe B nel training set: 0.5571428571428572
proporzione classe A nel test set: 0.7333333333333333
proporzione classe B nel test set: 0.2666666666666667
```

```
labels = ['Classe A', 'Classe B']

colors = ['red', 'blue']

plt.pie([proporzione_classe_A, proporzione_classe_B], labels=labels,
colors=colors, autopct='%1.1f%%')
plt.title('Proporzione delle classi nel training set')
plt.show()
```


```
labels = ['Classe A', 'Classe B']

colors = ['red', 'blue']

plt.pie([proporzione_classe_A_test, proporzione_classe_B_test],
```

```
labels=labels, colors=colors, autopct='%1.1f%%')
plt.title('Proporzione delle classi nel training set')
plt.show()
```


```
import random
import numpy as np

dataset=[]
for i in range(1000):
    dataset.append(random.randint(1, 100))

campione_casuale = random.sample(dataset, 300)

media_campione = np.mean(campione_casuale)
deviazione_standard_campione = np.std(campione_casuale)

media_dataset = np.mean(dataset)
deviazione_standard_dataset = np.std(dataset)

print(f"Media del campione casuale: {media_campione: .2f}")
print(f"Deviazione standard del campione casuale:
{deviazione_standard_campione: .2f}")
print(f"Media del dataset completo: {media_dataset: .2f}")
print(f"Deviazione standard del dataset completo:
{deviazione_standard_dataset: .2f}")
```

```
Media del campione casuale:  50.76
Deviazione standard del campione casuale:  29.91
```

```
Media del dataset completo:  50.45
Deviazione standard del dataset completo:  28.68
```

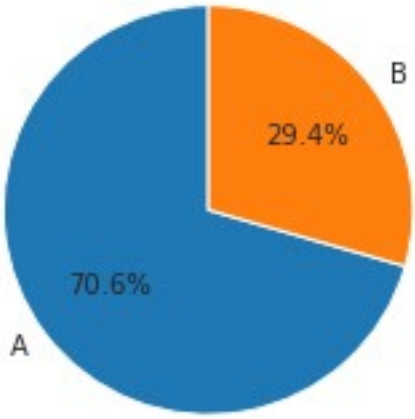
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

np.random.seed(42)
num_elementi = 1000
percentuale_A = 0.7

colonna = np.random.choice(['A', 'B'], size=num_elementi,
p=[percentuale_A, 1 - percentuale_A])


df = pd.DataFrame({'ColonnaAB': colonna})
df
```

```
    ColonnaAB
0           A
1           B
2           B
3           A
4           A
..        ...
995         A
996         B
997         A
998         B
999         A

[1000 rows x 1 columns]
```

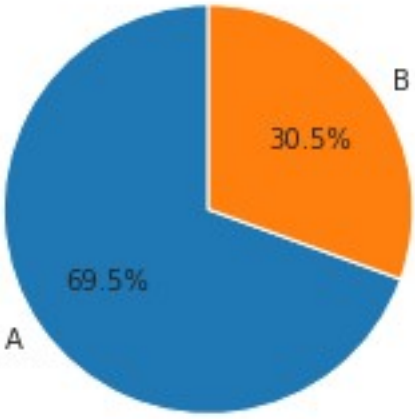
```
percentuali_subset1 =
subset1['ColonnaAB'].value_counts(normalize=True)
percentuali_subset2 =
subset2['ColonnaAB'].value_counts(normalize=True)
percentuali_subset3 =
subset3['ColonnaAB'].value_counts(normalize=True)

fig, axs = plt.subplots(3, 1, figsize=(6, 12))

axs[0].pie(percentuali_subset1, labels=percentuali_subset1.index,
autopct='%1.1f%%', startangle=90)
axs[0].set_title('Subset 1')

axs[1].pie(percentuali_subset2, labels=percentuali_subset2.index,
autopct='%1.1f%%', startangle=90)
axs[1].set_title('Subset 2')
```

```
axs[2].pie(percentuali_subset3, labels=percentuali_subset3.index,
autopct='%1.1f%%', startangle=90)
axs[2].set_title('Subset 3')

plt.show()
```

Subset 1



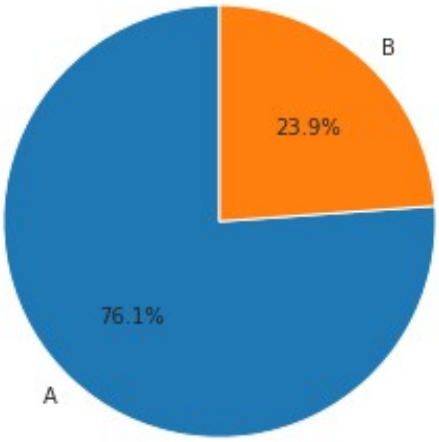
Subset 2



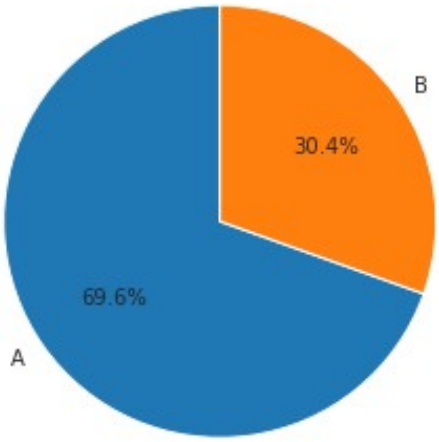
Subset 3

```
train_subset1, test_subset1 = train_test_split(subset1, test_size=0.2,
random_state=42)
train_subset2, test_subset2 = train_test_split(subset2, test_size=0.2,
random_state=42)
train_subset3, test_subset3 = train_test_split(subset3, test_size=0.2,
random_state=42)

fig, axs = plt.subplots(3, 2, figsize=(10, 12))

def draw_pie(ax, data, title):
    ax.pie(data, labels=data.index, autopct='%1.1f%%', startangle=90)
    ax.set_title(title)

draw_pie(axs[0, 0],
train_subset1['ColonnaAB'].value_counts(normalize=True), 'Train Subset
1')
draw_pie(axs[0, 1],
test_subset1['ColonnaAB'].value_counts(normalize=True), 'Test Subset
1')

draw_pie(axs[1, 0],
train_subset2['ColonnaAB'].value_counts(normalize=True), 'Train Subset
2')
draw_pie(axs[1, 1],
test_subset2['ColonnaAB'].value_counts(normalize=True), 'Test Subset
2')

draw_pie(axs[2, 0],
train_subset3['ColonnaAB'].value_counts(normalize=True), 'Train Subset
3')
draw_pie(axs[2, 1],
test_subset3['ColonnaAB'].value_counts(normalize=True), 'Test Subset
3')

plt.tight_layout()

plt.show()
```

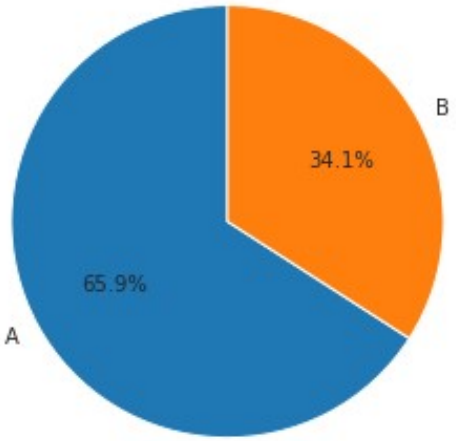
Train Subset 1



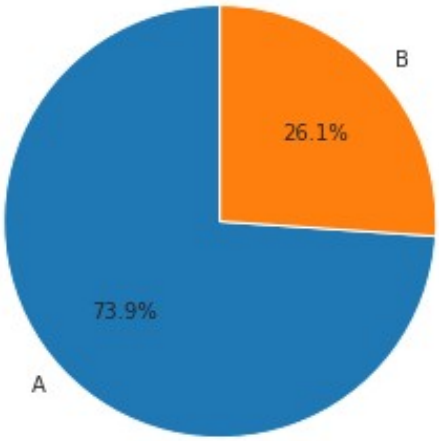
Test Subset 1



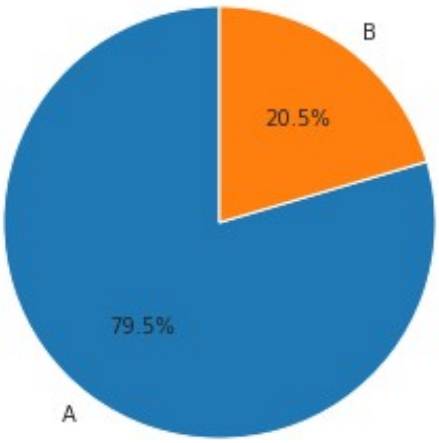
Train Subset 2



Test Subset 2



Train Subset 3



Test Subset 3