Corso ITS:

PROGETTISTA E SVILUPPATORE SOFTWARE:

FULL STACK DEVELOPER E CLOUD SPECIALIST

Modulo: **Programmazione in Python**

Docente: *Andrea Ribuoli*

Mercoledì **2 Aprile** 2025

09:00 - 14:00

---

elaborazione con controllo eccezione per fine input

In [ ]:
```
ALIQUOTA_1  = 0.23
ALIQUOTA_2  = 0.35
ALIQUOTA_3  = 0.43
SCAGLIONE_1 = 28000
SCAGLIONE_2 = 50000
MSG = "Inserisci il reddito: "
# MSG = ""

while True:
    try :
        reddito = int(input(MSG))
    except EOFError :
        break
    imposta = 0
    if reddito <= SCAGLIONE_1:
        imposta = reddito * ALIQUOTA_1
    else:
        if reddito > SCAGLIONE_1 and reddito <= SCAGLIONE_2:
            imposta = ( SCAGLIONE_1 * ALIQUOTA_1 +
                        (reddito - SCAGLIONE_1) * ALIQUOTA_2)
        else:
            imposta = ( SCAGLIONE_1 * ALIQUOTA_1 +
                         ( SCAGLIONE_2 - SCAGLIONE_1 ) * ALIQUOTA_2 +
```

```
                        ( reddito - SCAGLIONE_2 ) * ALIQUOTA_3)
    print(f"L'imposta da pagare per un reddito di {'%9.2f' % reddito} € è {'
```

In [3]:
```
!python imposta.py < imposta_input.txt
```

```
L'imposta da pagare per un reddito di  10000.00 € è   2300.00 €
L'imposta da pagare per un reddito di  28000.00 € è   6440.00 €
L'imposta da pagare per un reddito di  38000.00 € è   9940.00 €
L'imposta da pagare per un reddito di  50000.00 € è  14140.00 €
L'imposta da pagare per un reddito di  60000.00 € è  18440.00 €
```

esempio con valore sentinella

In [11]:
```
# media.py

# Acquisire tramite input() una sequenza di valori numerici
# fino a che l'utente non scriva 'exit'.
# In tal caso stampare la media dei valori letti.

somma = 0.0
numeri_letti = 0
valore_stringa = input("Passami il prossimo valore oppure 'exit': ")
while valore_stringa != "exit":
    try :
        addendo = float(valore_stringa)
    except ValueError as errore :
        print(errore)
        valore_stringa = input("Passami il prossimo valore oppure 'exit': ")
        continue
    numeri_letti += 1
    somma += addendo
    valore_stringa = input("Passami il prossimo valore oppure 'exit': ")
print("Numeri letti sono : ", numeri_letti)
print("La somma dei numeri è : ", somma)
if numeri_letti == 0 :
    print("La media ha senso quando esiste almeno un dato.")
else :
    print("La media dei numeri letti è: ", somma / numeri_letti)
```

```
Passami il prossimo valore oppure 'exit':  10e2
Passami il prossimo valore oppure 'exit':  exit
Numeri letti sono :  1
La somma dei numeri è :  1000.0
La media dei numeri letti è:  1000.0
```

In [19]:
```
# metodi isalpha, isdigit and isalnum

valore_stringa = input("Passami un numero: ")
valore_stringa.isalnum()
```

```
Passami un numero:  123w
```

Out[19]:
```
True
```

check_data.py

Acquisire l'input da utente controllando se il contenuto della stringa può essere interpretato come una data.

Restituirlo (`print`) in un formato standard.

esempi

```
3/3/25   => 03-03-2025
7-5-24   => 07-05-2024
29/3     => 29-03-2025

2 2 25   => 02-02-2025
5 Maggio => 05-05-2025
```

In [20]:
```
valore_stringa = input("Passami una data da verificare: ")
```

```
Passami una data da verificare:  3/3/25
```

In [21]:
```
len(valore_stringa)
```

Out[21]: 6

In [22]:
```
indice = 0
while indice < len(valore_stringa):
    print(valore_stringa[indice])
    indice += 1
```

```
3
/
3
/
2
5
```

In [ ]:
```
ANNO_DEFAULT = 2025
giorno = ""
mese = ""
anno = ""
fase = "G"   #   "M",  "A"
valore_stringa = input("Passami una data da verificare: ")
indice = 0
while indice < len(valore_stringa):
    current = valore_stringa[indice]
    if not current.isdigit():
        if fase == "G":
            fase = "M"
            indice += 1
            continue
        if fase == "M":
            fase = "A"
```

```
            indice += 1
            continue
        if fase == "A":
            break
    else:
        if fase == "G":
            giorno += current
        if fase == "M":
            mese += current
        if fase == "A":
            anno += current
    indice += 1
giorno = int(giorno)
mese = int(mese)
if anno == "":
    anno = ANNO_DEFAULT
else :
    anno = int(anno)
if anno < 50:
    anno += 2000
if anno < 100:
    anno += 1900
print(f"{giorno:02d}-{mese:02d}-{anno:04d}")
```

In [45]:
```
nomeRegione = "Marche"
for letter in nomeRegione :
   print(letter)
```

```
M
a
r
c
h
e
```

In [46]:
```
nomeRegione = "Marche"
i = 0
while i < len(nomeRegione) :
   letter = nomeRegione[i]
   print(letter)
   i = i + 1
```

```
M
a
r
c
h
e
```

In [47]:
```
b"Ciao"
```

Out[47]:
```
b'Ciao'
```

In [49]:
```
type("€".encode())
```

Out[49]:
```
bytes
```

In [51]:
```
euro = "€".encode()
```

In [52]:
```
for byte in euro :
    print(byte)
```

```
226
130
172
```

In [53]:
```
for byte in euro :
    print(chr(byte))
```

```
â

¬
```

In [62]:
```
for y in range(1, 4):
    for x in range(1, 4):
        print("x =", x, "y =", y, "prodotto", x * y)
    print("---")
```

```
x = 1 y = 1 prodotto 1
x = 2 y = 1 prodotto 2
x = 3 y = 1 prodotto 3
---
x = 1 y = 2 prodotto 2
x = 2 y = 2 prodotto 4
x = 3 y = 2 prodotto 6
---
x = 1 y = 3 prodotto 3
x = 2 y = 3 prodotto 6
x = 3 y = 3 prodotto 9
---
```

In [1]:
```
for y in range(1, 5):
    for x in range(1, 5):
        print("x =", x, "y =", y, "prodotto", x * y)
    print("---")
```

```
x = 1 y = 1 prodotto 1
x = 2 y = 1 prodotto 2
x = 3 y = 1 prodotto 3
x = 4 y = 1 prodotto 4
---
x = 1 y = 2 prodotto 2
x = 2 y = 2 prodotto 4
x = 3 y = 2 prodotto 6
x = 4 y = 2 prodotto 8
---
x = 1 y = 3 prodotto 3
x = 2 y = 3 prodotto 6
x = 3 y = 3 prodotto 9
x = 4 y = 3 prodotto 12
---
x = 1 y = 4 prodotto 4
x = 2 y = 4 prodotto 8
x = 3 y = 4 prodotto 12
x = 4 y = 4 prodotto 16
---
```

In [64]:
```
regione = "Trentino Alto Adige"
regione.find(" ")
```

Out[64]: 8

In [66]:
```
regione = "Trentino Alto Adige"
regione.find(" ", 9)
```

Out[66]: 13

In [68]:
```
regione = "Trentino Alto Adige"
indice = regione.find(" ")
print(indice)
indice += 1
indice = regione.find(" ", indice)
print(indice)
```

```
8
13
```

In [69]:
```
regione = "Trentino Alto Adige"
indice = regione.find(" ")
print(indice)
indice += 1
indice = regione.find(" ", indice)
print(indice)
indice += 1
indice = regione.find(" ", indice)
print(indice)
```

```
8
13
-1
```

In [70]:
```
regione = "Trentino Alto Adige"
indice = 0
```

```
indice = regione.find(" ", indice)
print(indice)
indice += 1
indice = regione.find(" ", indice)
print(indice)
indice += 1
indice = regione.find(" ", indice)
print(indice)
```

```
8
13
-1
```

In [76]:
```
regione = input("Stringa in cui cercare: ")
carattere_da_cercare = input("Indicami il carattere da cercare: ")
indice = 0
indice = regione.find(carattere_da_cercare, indice)
while indice != -1:
    print(indice)
    indice += 1
    indice = regione.find(carattere_da_cercare, indice)
```

```
Stringa in cui cercare:  Trentino Alto Adige Emilia Romagna
Indicami il carattere da cercare:  o
7
12
28
```

In [77]:
```
def volumeCubo(lunghezzaLato) :
    volume = lunghezzaLato ** 3
    return volume
```

In [78]:
```
volumeCubo(3.5)
```

Out[78]:
```
42.875
```

In [79]:
```
def trovaOccorrenze(stringa, char) :
    indice = 0
    indice = stringa.find(char, indice)
    while indice != -1:
        print(indice)
        indice += 1
        indice = stringa.find(char, indice)
```

In [80]:
```
regione = input("Stringa in cui cercare: ")
carattere_da_cercare = input("Indicami il carattere da cercare: ")
trovaOccorrenze(regione, carattere_da_cercare)
```

```
Stringa in cui cercare:  Trentino Alto Adige Emilia Romagna
Indicami il carattere da cercare:  o
7
12
28
```

In [82]:
```
trovaOccorrenze("Trentino Alto Adige Emilia Romagna", "A")
```

```
9
14
```

In [84]:
```
trovaOccorrenze("Trentino Alto Adige Emilia Romagna")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [84], in <cell line: 1>()
----> 1 trovaOccorrenze("Trentino Alto Adige Emilia Romagna")

TypeError: trovaOccorrenze() missing 1 required positional argument: 'char'
```

In [83]:
```

```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [83], in <cell line: 1>()
----> 1 prova()

NameError: name 'prova' is not defined
```

In [85]:
```
def trovaOccorrenze(stringa, char = " ") :
    indice = 0
    indice = stringa.find(char, indice)
    while indice != -1:
        print(indice)
        indice += 1
        indice = stringa.find(char, indice)
```

In [86]:
```
trovaOccorrenze("Trentino Alto Adige Emilia Romagna")
```

```
8
13
19
26
```

In [87]:
```
trovaOccorrenze("Trentino Alto Adige Emilia Romagna", " ")
```

```
8
13
19
26
```

In [88]:
```
trovaOccorrenze("Trentino Alto Adige Emilia Romagna", "A")
```

```
9
14
```

In [90]:
```
!python myfunc.py
```

```
9
14
```

In [97]:
```
for letter in range(127462, 127488) :
    print(chr(letter), end = " ")
```

```
🇦 🇧 🇨 🇩 🇪 🇫 🇬 🇭 🇮 🇯 🇰 🇱 🇲 🇳 🇴 🇵 🇶 🇷 🇸 🇹 🇺 🇻 🇼 🇽 🇾 🇿
```

In [98]:
```
print(chr(127470))
```

In [100…
```
print(chr(127481))
```

In [101…
```
print(chr(127470) + chr(127481))
```

In [103…
```
italia = chr(127470) + chr(127481)
len(italia)
```

Out[103… 2

In [105…
```
def flag(sigla) :
    DIFFERENZA = 127397
    flag  = ""
    for lettera in sigla :
        flag += chr(ord(lettera) + DIFFERENZA)
    return flag
```

In [106…
```
flag("IT")
```

Out[106… '🇮🇹'

In [107…
```
flag("FR")
```

Out[107… '🇫🇷'

In [109…
```
print(flag("IT"), 4, "-", flag("DE"), 3)
```

🇮🇹 4 - 🇩🇪 3

saldo.py

(Es. P5.22 pag 334)

Scrivete un funzione che calcoli il saldo di un conto bancario dopo che siano trascorsi un dato numero di anni, a partire da un dato saldo iniziale e con un dato tasso di interesse annuo, accreditando gli interessi annualmente.

In [116…
```
def saldo(saldo_iniziale, tasso, numero_di_anni = None) :
    if numero_di_anni == None :
        saldo_corrente = saldo_iniziale
        for i in range(20) :
            print(saldo_corrente)
            saldo_corrente *= ( 1 + tasso )
        pass
    else :
        return saldo_iniziale * ( 1 + tasso ) ** numero_di_anni
```

In [117…
```
saldo(10000, 0.04)
```

```
10000
10400.0
10816.0
11248.640000000001
11698.585600000002
12166.529024000003
12653.190184960004
13159.317792358404
13685.690504052742
14233.118124214852
14802.442849183446
15394.540563150784
16010.322185676816
16650.73507310389
17316.764476028045
18009.435055069167
18729.812457271935
19479.004955562814
20258.16515378533
21068.491759936744
```

In [115…
```
saldo(10000, 0.04, 2)
```

Out[115… 10816.000000000002

10000 0.04 10000 * 0.04 = 400 10000 + 10000 * 0.04 = 10000 * (1 + 0.04)

(10000 * (1 + 0.04)) * (1 + 0.04) 10000 * ((1 + 0.04) * (1 + 0.04))

saldo_finale = saldo_iniziale * (1 * tasso) ** numero_di _anni