Python di base: scripts

Python è ora un linguaggio di programmazione molto diffuso e performante perché è adatto allo sviluppo di applicazioni da molto semplici a complesse.

Questo compendio richiama gli argomenti di base di Python mediante esempi di script che sono presentati e proposti per essere eseguiti in ambiente interattivo: shell di python, Iupyter. Spyder, shell di thonny ...

Sommario

C1 – Creare ed eseguire il primo script python:

Puoi scrivere ed eseguire un semplice script Python dal terminale senza creare alcun file Python. Se lo script è di grandi dimensioni, richiede la scrittura e salva lo script in qualsiasi file Python utilizzando qualsiasi editor. Puoi utilizzare qualsiasi editor di testo o editor di codice come sublime, Visual Studio Code o qualsiasi software IDE sviluppato solo per Python come PyCharm o Spyder per scrivere lo script. L'estensione del file Python è .py. La versione 3.8 di Python e l'IDE spyder3 di Python vengono utilizzati in questo articolo per scrivere lo script Python. Devi installare Spyder IDE nel tuo sistema per usarlo.

Se desideri eseguire qualsiasi script dal terminale, esegui il comando "python" o "python3" per aprire Python in modalità interazione. Il seguente script Python stamperà il testo "Hello World" come output.

>>> print("Hello World")

```
fahmida@fahmida-VirtualBox:~$ python3
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>>
```

Ora salva lo script in un file denominato c1.py. Devi eseguire il seguente comando dal terminale per eseguire c1.py.
$ python3 c1.py

```
fahmida@fahmida-VirtualBox:~/python$ python3 c1.py
Hello World
fahmida@fahmida-VirtualBox:~/python$
```

Se desideri eseguire il file dal Jupyter, devi inviare il comando ↑ (shift) + (invio)

Se desideri eseguire il file dall'IDE Thonny, devi fare clic sul pulsante Esegui 

Sommario

C2 – Unire due stringhe:

Esistono molti modi per unire valori di stringa in Python. Il modo più semplice per combinare due valori stringa in Python è utilizzare l'operatore "+". Prova il seguente script per conoscere il modo di unire due stringhe. **Qui, due valori stringa vengono assegnati in due variabili e un'altra variabile viene utilizzata per memorizzare i valori uniti che verranno stampati successivamente.**

There are many ways to join string values in python. The most simple way to combine two string values in python is to use the '+' operator. Create any python with the following script to know the way to join two strings. Here, two string values are assigned in two variables, and another variable is used to store the joined values that are printed later.

c2_0.py # unisci due stringhe

```
string1 = "Linux"
string2 = "Hint"
joined_string = string1 + string2
print(joined_string)
```

c2_1.py # assegnazione multipla di variabili

```
COGNOME, Nome, CF = 'paperino', 'paolino', 'pprpln34b18e682k'
print(COGNOME.upper(),Nome.capitalize(),CF.upper())
scheda = [COGNOME.upper(),Nome.capitalize(),CF.upper()]

scheda
```

>>> ['PAPERINO', 'Paolino', 'PPRPLN34B18E682K']

c2_2.py # input multiplo di variabili

```
COGNOME,Nome,CF = input('COGNOME,Nome,Codice fiscale').split()
print(COGNOME.upper(),Nome.capitalize(),CF.upper())
scheda = [COGNOME.upper(),Nome.capitalize(),CF.upper()]

scheda
```

>>> ['PAPERINO', 'Paolino', 'PPRPLN34B18E682K']

c2_3.py # input multiplo in una lista

```
skill = input('cognome nome,cf').split()

# output di lista

skill
```

>>> ['PAPERINO', 'Paolino', 'PPRPLN34B18E682K']

Sommario

C3 – Formattazione di numeri reali in stringa:

Il numero in virgola mobile è richiesto nella programmazione per generare numeri frazionari e talvolta richiede la formattazione a scopo di programmazione. Esistono molti modi in Python per formattare il numero in virgola mobile.
La formattazione e l'interpolazione delle stringhe vengono utilizzate nello script seguente per

formattare un numero a virgola mobile.
Il metodo format() con larghezza del formato viene utilizzato nella formattazione del numero, vengono impostate 5 cifre prima del punto decimale e 2 cifre dopo il punto decimale.

Floating point number is required in programming for generating fractional numbers, and sometimes it requires formatting the floating-point number for programming purposes. There are many ways to exist in python to format the floating-point number. String formatting and string interpolation are used in the following script to format a floating-point number. ***format()*** *method with format width is used in string formatting, and '%" symbol with the format with width is used in string interpolation. According to the formatting width, 5 digits are set before the decimal point, and 2 digits are set after the decimal point.*

c3_1.py # formattazione di numeri reali

```
# inizializzazione variabile
float1 = 563.78453

# output variabile
float1

>>> 563.78453

# definizione del formato di stringa
print("{:5.3f}".format(float1))

>>> 563.785
```

provare a modificare il numero di cifre decimali
cosa succede se metto 4 cifre decimali?
e se ne metto 6?

C4 – Elevare un numero a potenza:

Esistono molti modi in Python per calcolare x^n in Python. Nello script seguente vengono mostrati tre modi per calcolare x^n in Python. Per calcolare x^n vengono utilizzati l'operatore doppio "*", il metodo pow() e il metodo math.pow().
I valori di x e n vengono inizializzati con valori numerici. I metodi double '*' e pow() vengono utilizzati per calcolare la potenza dei valori interi. math.pow() può calcolare la potenza dei numeri frazionari; inoltre, questo viene mostrato nell'ultima parte dello script.

Many ways exist in python to calculate the $\boldsymbol{x^n}$ *in Python. In the following script, three ways are shown to calculate the* $\boldsymbol{xn}$ *in Python. Double '**'** operator,* ***pow()*** *method, and* ***math.pow()*** *method are used for calculating the xn. The values of* ***x*** *and* ***n*** *are initialized with numeric values. Double '**'** and* ***pow()*** *methods are used for calculating the power of integer values.* ***math.pow()*** *can calculate the power of fractional numbers; also, that is shown in the last part of the script.*

c4_0.py # elevazione a potenza

```
# inizializzazione variabili
x = 4
n = 3

# metodo valido solo per esponenti interi
power = x ** n

# output formattato
print("%d to the power %d is %d" % (x, n, power))

# FUNZIONA MA E' ASSOLUTAMENTE DEPRECABILE, DA METTERE ALL'INIZIO DELLO SCRIPOT

import math
# chiamata alla funzione di libreria
power = pow(x, n)
# output formattato
print("%d elevato a %d vale %d" % (x, n, power))

# potenza con esponente frazionario
power = math.pow(2, 0.5)
print("%d elevato a  %d vale %5.4f" % (x, n, power))
```

C5 – Lavorare con variabili logiche:

I diversi usi dei tipi booleani sono mostrati nello script seguente. Il primo output stamperà il valore di val1 che contiene il valore booleano, true. Tutti i numeri positivi sono negativi restituiscono vero come valore booleano e solo zero restituisce falso come valore booleano. Pertanto, il secondo e il terzo output verranno stampati come veri per i numeri positivi e negativi. Il quarto output stamperà false per 0, e il quinto output stamperà false perché l'operatore di confronto restituisce false.

The different uses of Boolean types are shown in the following script. The first output will print the value of val1 that contains the Boolean value, ***true.*** *All positive are negative numbers return* ***true*** *as Boolean value and only zero returns* ***false*** *as a Boolean value. So, the second and third outputs will print* ***true*** *for positive and negative numbers. The fourth output will print false for 0, and the fifth output will print* ***false*** *because the comparison operator returns* ***false****.*

c5_0.py # Valori logici (Booleani)

```
# inizializzazione variabili
val1 = True
```

```
print(val1)

# da numero intero a booleano
number = 10
print(bool(number))

# da numero intero a booleano
number = 0
print(bool(number))

number = 0
print(bool(number))

# risultato logico di una comparazione
val1 = 6
val2 = 3
print(val1 == val2)
```

Sommario

C6 – Uso del controllo condizionale if:

Il seguente script mostra l'uso di un'istruzione condizionale in Python. La dichiarazione dell'istruzione if-else in Python è leggermente diversa rispetto ad altri linguaggi. Non sono necessarie parentesi graffe per definire il blocco if-else in Python come altri linguaggi, ma il blocco di indentazione deve essere utilizzato correttamente altrimenti lo script mostrerà un errore. Qui, nello script viene utilizzata un'istruzione if-else molto semplice che controllerà che il valore della variabile numerica sia maggiore o uguale a 70 o meno. I due punti (:) vengono utilizzati dopo il blocco "if" e "else" per definire l'inizio del blocco.

*The following script shows the use of a conditional statement in python. The declaration of **the if-else** statement in python is a little bit different than other languages. No curly brackets are required to define the if-else block in python like other languages, but the indentation block must be used properly other the script will show an error. Here, a very simple **if-else** statement is used in the script that will check the value of the number variable is more than or equal to 70or not. A **colon(:)** is used after the **'if'** and **'else'** block to define the starting of the block.*

c6_1.py # IF costrutto

```
# Assegna il valore alla variabile
number = 70
# Verifica se il numero è maggiore di settanta o no
if (numero >= 70):
    print('Il numero è maggiore di settanta')
else:
    print('Il numero è INFERIORE a settanta')
```

c6_2.py # IF costrutto con assegnazioni multiple

```
# Assegna il valore alle variabil x e y
x, y = 30,40
# Verifica se la somma è maggiore di settanta o no
if (x+y >= 70):
    print('La somma è maggiore o ugualedi settanta')
else:
    print('La somma è INFERIORE a settanta')
```

c6_3.py # IF costrutto, tema

```
# Chiede all'utente due numeri e valuta se il loro prodotto è maggiore o minore
di 100
```

Sommario

C7 – Uso degli operatori logici AND e OR:

The following script shows the uses of **AND** and **OR** operators in the conditional statement. **AND** operator returns **true** when the two conditions return **true,** and **OR** operator returns **true** when any condition of two conditions returns **true**. Two floating-point numbers will be taken as MCQ and theory marks. Both AND and OR operators are used in the '**if'** statement. According to the condition, if the MCQ marks are more than equal to 40 and theory marks is more than or equal to 30 then the '**if'** statement will return **true** or if the total of MCQ and theory is more than or equal to 70 then the '**if'** statement will also return **true**.

c7.py

```
# Take MCQ marks
mcq_marks = float(input("Enter the MCQ marks: "))
# Take theory marks
theory_marks = float(input("Enter the Theory marks: "))

# Check the passing condition using AND and OR operator
if (mcq_marks >= 40 and theory_marks >= 30) or (mcq_marks + theory_marks) >= 70:
   print("\nYou have passed")
else:
   print("\nYou have failed")
```

According to the following output, **if** statement returns **false** for the input values 30 and 35, and returns **true** for the input values 40 and 45.

Sommario

C8 – Istruzione switch case:

Python non supporta un'istruzione switch-case come altri linguaggi di programmazione standard, ma questo tipo di istruzione può essere implementato in Python utilizzando una funzione personalizzata.

La funzione Employee_details() viene creata nello script seguente per funzionare come l'istruzione switch-case. La funzione contiene un parametro e un dizionario denominato switcher. Il valore del parametro della funzione viene controllato con ciascun indice del dizionario. Se viene trovata una corrispondenza, la funzione restituirà il valore corrispondente dell'indice; in caso contrario, verrà restituito il valore del secondo parametro del metodo switcher.get().

*Python does not support **a switch-case** statement like other standard programming languages, but this type of statement can be implemented in python by using a custom function. **employee_details()** function is created in the following script to work like the switch-case statement. The function contains one parameter and a dictionary named **switcher.** The value of the function parameter is checked with each index of the dictionary. If any match found, then the corresponding value of the index will be returned from the function; otherwise, the second parameter value of the **switcher.get()** method will be returned.*

c8_1.py # Selettore per implementere switch case

```
# c8_1.py # Switcher per implementare il costrutto switch case
def employee_details(ID):

'''Il nome verrà restituito se viene trovata la chiave, verrà
restituito 'sconosciuto' se non viene trovata alcuna
corrispondenza'''
    switcher = {"1004": "MD. Mehrab","1009": "Mita Rahman","1010":
"Sakib Al Hasan",}
    return switcher.get(ID, "sconosciuto")
```

```
# Chiede la matricola utente
ID = input("Inserire la matricola: ")
# Stampa il nome corrispondente
print('matricola -',ID,': ',employee_details(ID))


in uscita:
>>> Inserire la matricola:  1000
>>> matricola - 1000 :  sconosciuto
```

Sommario

C9 – Uso del ciclo while:

(M2_07 - Ciclo while, istruzioni break e continue)

L'uso di un ciclo while in Python è mostrato nell'esempio seguente. I due punti(:) vengono utilizzati per definire il blocco iniziale del ciclo e tutte le istruzioni del ciclo devono essere definite utilizzando il rientro corretto; in caso contrario, verrà visualizzato un errore di rientro. Nello script seguente, il valore del contatore viene inizializzato su 1 utilizzato nel ciclo. Il ciclo ripeterà 5 volte e stamperà i valori del contatore in ogni iterazione. Il valore del contatore viene incrementato di 1 in ogni iterazione per raggiungere la condizione di terminazione del ciclo.

c9_1.py # contatore

```
# Initialize counter
counter = 1
# Iterate the loop 5 times
while counter < 6:
     # Print the counter value
     print("The current counter value: %d" % counter)
     # Increment the counter
     counter = counter + 1
```

c9_2.py # contatore

```
# Inizializza il contatore
counter = 1
# Iterate the loop 5 times
while counter < 6:
    # Stampa il valore del contatore
    print("Il valore corrente del contatore: %d" % counter)
    # Incrementa il contatore
    counter = counter + 1
```

c9_3.py # menu

```
# c9_3 Menu

while True:
    print("\nOpzioni:")
    print("1. Crea base dati")
    print("2. Elenco soci")
    print("3. aggiungi socio")
    print("4. Esci\n")
    choice = input("Scegli l'azione")

    if choice == '1':
        print("\nCrea base dati")
    elif choice == '2':
        print("\nElenco soci")
```

```
elif choice == '3':
    print("\nAggiungi socio")
elif choice == '4':
    break
else:
    print("\nOpzione inesistente. Seleziona da 1 .. a 4.")
```

Sommario

C10 – Uso del ciclo for:

(M2_08 - Il ciclo for e la funzione range)

Il ciclo for viene utilizzato per molti scopi in Python. Il blocco iniziale di questo ciclo deve essere definito dai due punti (:) e le istruzioni del ciclo vengono definite utilizzando il tabulatore.

Nello script seguente viene definito un elenco di nomi di giorni feriali e viene utilizzato un ciclo for per stampare ogni elemento della lista. Il metodo len() viene utilizzato per contare gli elementi totali della lista e definire il limite della funzione range().

c10_1.py # uso del ciclo for

```
'''
     c10_1.py uso del ciclo for
     - inizializza la lista dei giorni della settimana
     - calcola il numero degli elementi della lista per
dimensionare il ciclo for
     - stampa i giorni della settimana
'''

# Inizializza la lista
giorni = ["Domenica", "Lunedi", "Martedi","Mercoledi", "Giovedi",
"Venerdi", "Sabato"]

n_giorni = len(giorni)
print("I ",n_giorni," giorni della settimana sono:\n")

giorni         # visualizza la lista


# implementa il ciclo mediante l'operatore for
n_giorni = len(giorni)
print("I ",n_giorni," giorni della settimana sono:\n")

for j in range(len(giorni)):
    print(j,' - ',giorni[j])
```

Sommario

C11 – Richiamare uno script python da un altro:

A volte è necessario utilizzare lo script di un file Python da un altro file Python. Può essere fatto facilmente utilizzando la parola chiave import, come importare qualsiasi altro modulo. Nell'esempio il file Vacations.py contiene due variabili inizializzate da valori stringa. Questo file viene importato nel file c11.py con il nome alias "v". Qui viene definito un elenco di nomi di mesi. La variabile flag viene utilizzata qui per stampare il valore della variabile Vacation1 per una volta per i mesi "Giugno" e "Luglio". Il valore della variabile Vacation2 verrà stampato per il mese "Dicembre". Gli altri nomi dei nove mesi verranno stampati quando verrà eseguita la parte else dell'istruzione if-elseif-else.

*Sometimes it is required to use the script of a python file from another python file. It can be done easily, like importing any module by using **the import** keyword. Here, **vacations.py** file contains two variables initialized by string values. This file is imported in **c11.py** file with the alias name '**v**'. A list of month names is defined here. The **flag** variable is used here to print the value of **vacation1** variable for one time for the months '**June**' and '**July**'. The value of the **vacation2** variable will print for the month '**December**'. The other nine-month names will be printed when the else part of the **if-elseif-else** statement will be executed.*

```
vacations.py
# Initialize values
vacation1 = "Summer Vacation"
vacation2 = "Winter Vacation"
```

```
c11_1.py # Import another python script
import vacations as v

# Initialize the month list
months = ["January", "February", "March", "April", "May",
"June","July", "August", "September", "October", "November",
"December"]


# Initial flag variable to print summer vacation one time
flag = 0

# Iterate the list using for loop
for month in months:
    if month == "June" or month == "July":
        if flag == 0:
            print("Now", v.vacation1)
            flag = 1
    elif month == "December":
        print("Now", v.vacation2)
    else:
        print("The current month is", month)
```

Sommario

C12 – Uso di comandi di sistema:

Gli script che seguono mostrano l'uso degli argomenti della riga di comando di sistema in Python. Esistono molti moduli in Python per impiegare gli argomenti della riga di comando.

Il modulo "sys" viene importato qui per analizzare gli argomenti della riga di comando. il metodo len() viene utilizzato per contare gli argomenti totali, incluso il nome del file di script. Successivamente, verranno stampati i valori degli argomenti.

c12.py

```
# Importa il modulo sys
import sys

# Numero totale degli argomenti inviati
print('Argomenti inviati:', len(sys.argv))

print("I valori degli argomenti inviati:\n")
# Stampa recursiva degli argomenti inviati
for i in sys.argv:
    print(i)
```

If the script is executed without any command-line arguments, then the following output will appear that is showing the script filename.

The command-line argument values can be set in spyder editor by opening the **Run configuration per file** dialog box by clicking on the **Run** menu. Set the values with space by clicking the Command line options of General settings part of the dialog box.

If the script is executed after setting the values shown above, then the following output will appear.

The command-line argument values can be passed in the python script easily from the terminal. The following output will appear if the script is executed from the terminal.

If you want to know more about command-line arguments in python, then you can check the tutorial, How to parse arguments on command-line in Python.

Sommario

C13 - Use of regex:

L'espressione regolare o regex viene utilizzata in Python per abbinare o cercare e sostituire qualsiasi porzione particolare di una stringa in base al modello particolare. Il modulo 're' viene utilizzato in Python per utilizzare un'espressione regolare. Il seguente script mostra l'uso di regex in Python. Il modello utilizzato nello script corrisponderà alle stringhe in cui il primo carattere della stringa è una lettera maiuscola. Verrà preso un valore stringa e corrisponderà al modello utilizzando il metodo match(). Se il metodo restituisce true, verrà stampato un messaggio di successo, altrimenti verrà stampato un messaggio di istruzioni.

Regular expression or regex is used in python to match or search and replace any particular portion of a string based on the particular pattern. ***'re'*** *module is used in python to use a regular expression. The following script shows the use of regex in python. The pattern used in the script will match those string where the first character of the string is a capital letter. A string value will be taken and match the pattern using* ***match()*** *method. If the method returns true, then a success message will print otherwise an instructional message will print.*

c13.py

```
# Import re module
import re

# Take any string data
string = input("Enter a string value: ")
# Define the searching pattern
pattern = '^[A-Z]'

# match the pattern with input value
found = re.match(pattern, string)

# Print message based on the return value
if found:
    print("The input value is started with the capital letter")
else:
    print("You have to type string start with the capital letter")
```

The script is executed two times in the following output. **match()** function returns false for the first execution and returns true for the second execution.

Sommario

C14 - Use of getpass:

getpass is a useful module of python that is used to take password input from the user. The following script shows the use of the getpass module. getpass() method is used here to take the input as password and '**if'** statement is used here to compare the input value with the defined password. "**you are authenticated"** message will print if the password matches otherwise it will print "**You are not authenticated**" message.

c14.py

```
# import getpass module
import getpass

# Take password from the user
passwd = getpass.getpass('Password:')

# Check the password
if passwd == "fahmida":
    print("You are authenticated")
else:
    print("You are not authenticated")
```

If the script runs from the spyder editor, then the input value will be shown because the editor console doesn't support password mode. So, the following output shows the input password in the following output.

If the script executes from the terminal, then input value will not be shown like other Linux password. The script is executed two times from the terminal with an invalid and valid password that is shown in the following output.

Sommario

C15 – Data (misura del tempo):

Il valore della data può essere formattato in Python in vari modi. Lo script seguente utilizza il modulo datetime per impostare il valore della data corrente e personalizzato. Il metodo today() viene utilizzato qui per leggere la data e l'ora correnti del sistema. Successivamente, il valore formattato della data viene stampato utilizzando nomi di proprietà diversi dell'oggetto data. Il modo in cui un valore di data personalizzato può essere assegnato e stampato viene mostrato nella parte successiva dello script.

c15.py

```
from datetime import date

# Read the current date
```

```
current_date = date.today()

# Print the formatted date
print("Today is :%d-%d-%d" %
      (current_date.day, current_date.month, current_date.year))

# Set the custom date
custom_date = date(2020, 12, 16)
print("The date is:", custom_date)
```

The following output will appear after executing the script.

C16 - Add and remove the item from a list:

L'oggetto list viene utilizzato in Python per risolvere vari tipi di problemi. Python ha molte funzioni integrate per lavorare con l'oggetto list. Il modo in cui un nuovo elemento può essere inserito e rimosso da un elenco viene mostrato nell'esempio seguente. Nello script viene dichiarato un elenco di quattro elementi. Il metodo Insert() viene utilizzato per inserire un nuovo elemento nella seconda posizione dell'elenco. Il metodo rimuovi() viene utilizzato per cercare e rimuovere un particolare elemento dall'elenco. L'elenco viene stampato dopo l'inserimento e la cancellazione.

list object is used in python for solving various types of problems. Python has many built-in functions to work with the list object. How a new item can be inserted and removed from a list is shown in the following example. A list of four items is declared in the script. ***Insert()*** *method is used to insert a new item in the second position of the list.* ***remove()*** *method is used to search and remove the particular item from the list. The list is printed after the insertion and deletion.*

c16.py # Declare a fruit list

```
fruits = ["Mango", "Orange", "Guava", "Banana"]

# Insert an item in the 2nd position
fruits.insert(1, "Grape")

# Displaying list after inserting
print("The fruit list after insert:")
print(fruits)

# Remove an item
fruits.remove("Guava")

# Print the list after delete
print("The fruit list after delete:")
print(fruits)
```

If you want to know more details about the insertion and deletion of the python script, then you can check the tutorial, “How to add and remove items from a list in Python”.

C17 - List comprehension:

List comprehension is used in python to create a new list based on any string or tuple or another list. The same task can be done using for loop and lambda function. The following script shows two different uses of list comprehension. A string value is converted into a list of characters using list comprehension. Next, a tuple is converted into a list in the same way.

c17.py

```
# Create a list of characters using list comprehension
char_list = [char for char in "linuxhint"]
print(char_list)

# Define a tuple of websites
websites = ("google.com", "yahoo.com", "ask.com", "bing.com")

# Create a list from tuple using list comprehension
site_list = [site for site in websites]
print(site_list)
```

Sommario

C18 - Slice data:

Il metodo slice() è usato in python per tagliare una porzione specifica di una stringa. **Questo metodo ha tre parametri**. Questi parametri sono **start, stop e step**. **Stop è il parametro obbligatorio, mentre gli altri due sono facoltativi.** Lo script seguente mostra gli usi del metodo slice() con uno, due e tre parametri. Quando un parametro è usato nel metodo slice(), allora userà il parametro obbligatorio, stop. Quando i due parametri sono usati nel metodo slice(), allora vengono usati i parametri start e stop. Quando tutti e tre i parametri sono usati, allora vengono usati i parametri start, stop e step.

slice() *method is used in python to cut the particular portion of a string. This method has three parameters. These parameters are* ***start***, ***stop,*** *and* ***step***. *The* ***stop*** *is the mandatory parameter, and the other two parameters are optional. The following script shows the uses of the* ***slice()*** *method with one, two, and three parameters. When one parameter is used in the* ***slice()*** *method, then it will use the mandatory parameter,* ***stop***. *When the two parameters are used in the* ***slice()*** *method, then* ***start*** *and* ***stop*** *parameters are used. When all three parameters are used, then* ***start***, ***stop***, *and* ***step*** *parameters are used.*

c18.py

```
# Assign string value
text = "Learn Python Programming"
```

```
# Slice using one parameter
sliceObj = slice(5)
print(text[sliceObj])

# Slice using two parameter
sliceObj = slice(6, 12)
print(text[sliceObj])

# Slice using three parameter
sliceObj = slice(6, 25, 5)
print(text[sliceObj])
```

The following output will appear after running the script. In the first **slice()** method, 5 is used as the argument value. It sliced the five characters of **text** variables that are printed as output. In the second **slice()** method, 6 and 12 are used as arguments. The slicing is started from position 6 and stopped after 12 characters. In the third **slice()** method, 6, 25, and 5 are used as arguments. The slicing is started from position 6, and the slicing stopped after 25 characters by omitting 5 characters in each step.

Add and search data in the dictionary:

dictionary object is used in python to store multiple data like the associative array of other programming languages. The following script shows how a new item can be inserted, and any item can be searched in the dictionary. A dictionary of customer information is declared in the script where the index contains the customer ID, and the value contains the customer name. Next, one new customer information is inserted at the end of the dictionary. A customer ID is taken as input to search in the dictionary. **'for'** loop and **'if'** condition is used to iterate the indexes of the dictionary and search the input value in the dictionary.

c19.py

```
# Define a dictionary
customers = {'06753': 'Mehzabin Afroze', '02457': 'Md. Ali',
        '02834': 'Mosarof Ahmed', '05623': 'Mila Hasan', '07895': 'Yaqub Ali'}

# Append a new data
customers['05634'] = 'Mehboba Ferdous'

print("The customer names are:")
# Print the values of the dictionary
for customer in customers:
   print(customers[customer])

# Take customer ID as input to search
```

```
name = input("Enter customer ID:")

# Search the ID in the dictionary
for customer in customers:
    if customer == name:
        print(customers[customer])
        break
```

The following output will appear after executing the script and taking '**02457'** as ID value.

If you want to know more about the other useful methods of the dictionary, then you can check the tutorial, "10 most useful Python Dictionary Methods".

Add and search data in set:

The following script shows the ways to add and search data in a python set. A set of integer data is declared in the script. **add()** method is used to insert new data in the set. Next, an integer value will be taken as input to search the value in the set by using **for** loop and **if** condition.

c20.py

```
# Define the number set
numbers = {23, 90, 56, 78, 12, 34, 67}

# Add a new data
numbers.add(50)
# Print the set values
print(numbers)

message = "Number is not found"

# Take a number value for search
search_number = int(input("Enter a number:"))
# Search the number in the set
for val in numbers:
    if val == search_number:
        message = "Number is found"
        break

print(message)
```

The script is executed two times with the integer value 89 and 67. 89 does not exist in the set, and "**Number is not found**" is printed. 67 exists in the set, and "**Number is found**" is printed.

If you want to know about the **union** operation in the set, then you can check the tutorial, "How to use union on python set".

Sommario

Count items in the list:

count() method is used in python to count how many times a particular string appears in other string. It can take three arguments. The first argument is mandatory, and it searches the particular string in the whole part of another string. The other two arguments of this method are used to limit the search by defining the searching positions. In the following script, the **count()** method is used with one argument that will search and count the word '**Python**' in the **string** variable.

c21.py

```
# Define the string
string = 'Python Bash Java Python PHP PERL'
# Define the search string
search = 'Python'
# Store the count value
count = string.count(search)
# Print the formatted output
print("%s appears %d times" % (search, count))
```

The following output will appear after executing the script.

If you want to know more details about the count() method, then you can check the tutorial, How to use count() method in python.

Sommario

C22 – Defininizione e chiamata di funzione:

How user-defined function can be declared and called in python is shown in the following script. Here, two functions are declared. **addition()** function contains two arguments to calculate the sum of two numbers and print the value. **area()** function contains one argument to calculate the area of a circle and return the result to the caller by using **the return** statement.

c22.py

```
# Define addition function
def addition(number1, number2):
    result = number1 + number2
```

```
    print("Addition result:", result)

# Define area function with return statement
def area(radius):
    result = 3.14 * radius * radius
    return result

# Call addition function
addition(400, 300)
# Call area function
print("Area of the circle is", area(4))
```

The following output will appear after executing the script.

If you want to know details about the return values from a python function, then you can check the tutorial, Return Multiple Values from A Python Function.

Use of throw and catch exception:

try and **catch** block are used to throw and catch the exception. The following script shows the use of a **try-catch** block in python. In the **try** block, a number value will be taken as input and checked the number is even or odd. If any non-numeric value is provided as input, then a **ValueError** will be generated, and an exception will be thrown to the **catch** block to print the error message.

c23.py

```
# Try block
try:
    # Take a number
    number = int(input("Enter a number: "))
    if number % 2 == 0:
        print("Number is even")
    else:
        print("Number is odd")

# Exception block
except (ValueError):
    # Print error message
    print("Enter a numeric value")
```

The following output will appear after executing the script.

If you want to know more details about the exception handling in python, then you can check the tutorial, “Exception Handling in Python”.

Sommario

C24 - Read and Write File:

Il seguente script mostra come leggere e scrivere in un file in Python. Il nome del file è definito nella variabile filename. Il file viene aperto in scrittura utilizzando il metodo open() all'inizio dello script. Tre righe vengono scritte nel file utilizzando il metodo write(). Successivamente, lo stesso file viene aperto in lettura utilizzando il metodo open() e ogni riga del file viene letta e stampata utilizzando il ciclo for.

c24_1.py - # Definisci il nome file

```
# definisco il nome file e il percorso di ricerca
filename = "languages.txt"
# apertura del file in scrittura
fileHandler = open(filename, "w")
```

c24_2.py - # Riscrivo del testo

```
fileHandler.write("Bash\n")
fileHandler.write("Python\n")
fileHandler.write("PHP\n")

# chiusura del file
fileHandler.close()
```

c24_3.py - # Apertura del file in lettura

```
fileHandler = open(filename, "r")

# legge il file riga per riga
for line in fileHandler:
     print(line)

# chiude il file
fileHandler.close()
```

C25 – Elenco dei files in una directory:

Il contenuto di qualsiasi directory può essere letto utilizzando il modulo os di Python. Il seguente script mostra come ottenere l'elenco di una directory specifica in Python utilizzando il modulo os. Il metodo listdir() viene utilizzato nello script per trovare l'elenco di file e cartelle di una directory. Il ciclo for viene utilizzato per stampare il contenuto della directory.

c25.py

```
# Importa il modulo OS
import os

# formula il path di lavoro
path = 'Z:/Temp_0'

# legge il contenuto della directory
files = os.listdir(path)

# Stampa il contenuto della directory
for file in files:
    print(file)
```

```
>>>    cgi-bin
       documenti
       htdocs
       index.html
       JS_htdocs
```

Il contenuto della directory verrà visualizzato dopo l'esecuzione dello script se esiste il percorso definito della directory.

Sommario

C26 – Leggere e scrivere usando pickle:

Il seguente script mostra i modi per scrivere e leggere i dati utilizzando il modulo pickle di Python. Nello script un oggetto viene dichiarato e inizializzato con cinque valori numerici. I dati di questo oggetto vengono scritti in un file utilizzando il metodo dump(). Successivamente, il metodo load() viene utilizzato per leggere i dati dallo stesso file e memorizzarli in un oggetto.

c26.py

```
# importa il modulo pickle
import pickle

# Dichiara l'oggetto per memorizzare i dati
dataObject = []
```

```
# ripete il ciclo per 5 volte
for num in range(10, 15):
    dataObject.append(num)


# Apre un file per scrivere i dati
file_handler = open('languages', 'wb')

# Scarica i dati nel file
pickle.dump(dataObject, file_handler)

# chiude il file
file_handler.close()

# Apre il file per leggere i dati
file_handler = open('languages', 'rb')

# Carica i dati dal file dopo la deserializzazione
dataObject = pickle.load(file_handler)


# Ripete l'operazione per stampare i dati
for val in dataObject:
    print('The data value : ', val)

>>> The data value :  10
       The data value :  11
       The data value :  12
       The data value :  13
       The data value :  14

# chiude il file
file_handler.close()
```

Approfondimento: How to pickle objects in Python.

Sommario

C27 – Definire classi e metodi:

Lo script seguente mostra come è possibile dichiarare e accedere a una classe e a un metodo in Python. Qui, una classe viene dichiarata con una variabile di classe e un metodo. Successivamente, viene dichiarato un oggetto della classe per accedere alla variabile della classe e al metodo della classe.

c27.py

```
# Definisce la classe
class Employee:
    name = "Mostak Mahmud"
    # Definisce il metodo
    def details(self):
        print("Post: Marketing Officer")
        print("Department: Sales")
        print("Salary: $1000")

# Crea l'oggetto impiegato
emp = Employee()

# Stampa la variabile di classe
print("Name:", emp.name)

# Richiama il metodo di classe
emp.details()
```

Sommario

C28 - Uso della funzione range():

Il seguente script mostra i diversi usi della funzione range in Python. Questa funzione può accettare tre argomenti. Questi sono inizio, fine e passo. **L'argomento stop è obbligatorio**. Quando viene utilizzato un argomento, il valore predefinito di inizio è 0. La funzione range() con un argomento, due argomenti e tre argomenti viene qui utilizzata nei cicli for.

c28.py

...

```
# range() con un solo parametro
for val in range(6):
    print(val, end=' ')
print('\n')
```

…

```
>>> 0 1 2 3 4 5

# range() con due parametri
for val in range(5,10):
    print(val, end=' ')
print('\n')
```

…

```
>>> 5 6 7 8 9

# range() con tre parametri
for val in range(0, 8, 2):
    print(val, end='  ')
print('\n')
```

…

```
>>> 0  2  4  6
```

Sommario

C29 – Uso della funzione map():

La funzione map() viene utilizzata in Python per restituire un elenco utilizzando qualsiasi funzione definita dall'utente e qualsiasi oggetto iterabile. Nello script seguente, la funzione cal_power() viene definita per calcolare x^n e la funzione viene utilizzata nel primo argomento della funzione map(). Una lista denominata numeri viene utilizzata nel secondo argomento della funzione map(). Il valore di x verrà preso dall'utente e la funzione map() restituirà un elenco di valori di potenza di x, in base ai valori degli elementi dell'elenco di numeri.

c29.py

```
# Definizione della funzione per calcolare l’elevamento a potenza
def cal_power(n):
    return x ** n

# chiede il valore di x
x = int(input("Valore di x:"))
# definisce una lista di numeri
numbers = [2, 3, 4]
```

>>> Valore di x: 2

```
# Calcola x elevato a n usando map()
result = map(cal_power, numbers)
print(list(result))
```

>>> [4, 8, 16]

Sommario

C30 – Uso della funzione filter():

La funzione filter() di Python utilizza una funzione personalizzata per filtrare i dati da un oggetto iterabile e creare un elenco con gli elementi per quelli che la funzione restituisce true. Nello script seguente, la funzione SelectedPerson() viene utilizzata nello script per creare un elenco di dati filtrati in base agli elementi di SelectedList.

c30.py

```
# Definisce una lista di elementi
participant = ['Monalisa','Akbar Hossain','Jakir Hasan', 'Zahadur Rahman', 'Zenifer Lopez']

# Definisce la funzione di filtro selected candidates
def SelectedPerson(participant):
    selected = ['Akbar Hossain', 'Zillur Rahman', 'Monalisa']
    if (participant in selected):
        return True

selectedList = filter(SelectedPerson, participant)
print('The selected candidates are:')
for candidate in selectedList:
    print(candidate)
```

>>> The selected candidates are:

```
Monalisa
Akbar Hossain
```

RELATED LINUX HINT POSTS

- Python Subprocess.Popen Examples
- Python String Examples
- How to Resolve the “No Module Named Sklearn” Error in Python
- How to Concatenate Lists in Python
- How to Do the Floor Division in Python to Round Down
- Python Optional Arguments Defined in Functions
- Python Nested Dictionary Usage
