Loop:

Something that repeats continuously.

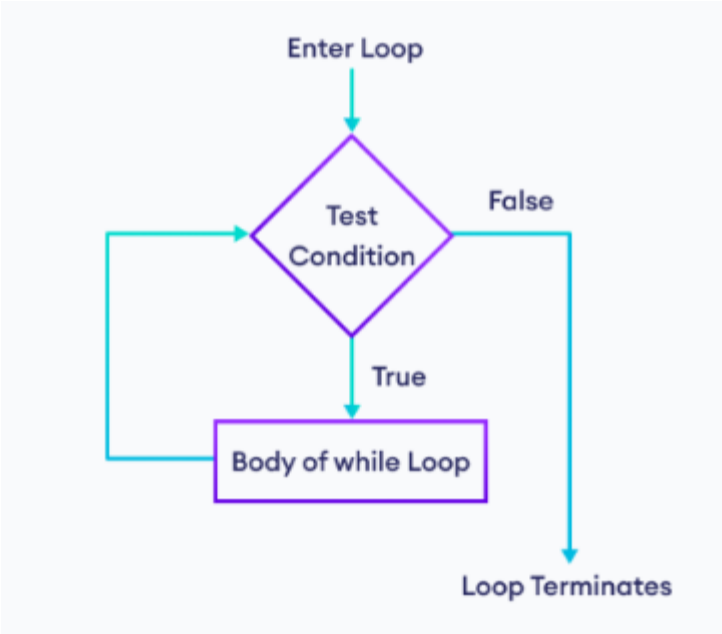
While Loop

Python while loop is used to run a block code until a certain condition is met.

The syntax of while loop is:

```
initialization
while condition:
   #body of while
   --------------
   --------------
   increment/decrement
```

1. A while loop evaluates the condition
2. If the condition evaluates to True, the code inside the while loop is executed.
3. condition is evaluated again.
4. This process continues until the condition is False.
5. When condition evaluates to False, the loop stops.



In [15]:
```
x = 1
while x <= 10:
    print(x,end=" ,")
    x+=1
```

```
1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,
```

In [4]:
```
# Print all the number from 50 to 1.
i = 50
while i >= 1:
    print(i,end=' ')
    i-=1
```

```
50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23
22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
```

In [16]:
```
e = 1
while e <= 200:
    if e%2==0:
        print(e,end=" ")
    e +=1
```

```
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 54 56 58 6
0 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100 102 104 106 108 110 1
12 114 116 118 120 122 124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 1
54 156 158 160 162 164 166 168 170 172 174 176 178 180 182 184 186 188 190 192 194 1
96 198 200
```

Q. Wapp to print the value if an user enters 'rock', 'paper' or 'scissor',Otherwise if an user enters any value other than rock', 'paper' or 'scissor' it should ask for input again.

In [8]:
```
val = 'none'
while val.lower() not in ['paper','rock','scissor']:
    val = input("Enter the value: ")
```

```
print("Your value is:",val)
```

```
Your value is: Paper
```

Infinite while Loop in Python

If the condition of a loop is always True, the loop runs for infinite times (until the memory is full). For example,

In [17]:
```
age = 32
# the test condition is always True
while age > 18:
    print('You can vote')
    break
```

```
You can vote
```

The 👆 condition always evaluates to True. Hence, the loop body will run for infinite times.

For Loop :

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

The syntax of for loop is:

```
for i in collection:
    #body of for
```

In [9]:
```
# for:-
num = [12,34,67,879,34,89,66,42]
for i in num:
    print(i)
```

```
12
34
67
879
34
89
66
42
```

In [10]:
```
# for:-
num = [12,34,67,879,34,89,66,42]
for i in num:
```

```
    if i%2==0:
        print(i,end=" ")
```

```
12 34 34 66 42
```

In [11]:
```
#
for char in 'AEROPLANE':
    print(char,end=" ")
```

```
A E R O P L A N E
```

range() : It is a built-in function generates a sequence of integers between any given interval.

```
range(start,stop+1,step)
```

Syntax:

```
for i in range(start,stop+1,step):
    body of for loop
```

In [12]:
```
for i in range(1,11):
    print(i,end=" ")
```

```
1 2 3 4 5 6 7 8 9 10
```

In [13]:
```
# Reverse numbers from 500 ~ 200.
for num in range(500,199,-1):
    print(num,end=" ")
```

```
500 499 498 497 496 495 494 493 492 491 490 489 488 487 486 485 484 483 482 481 480
479 478 477 476 475 474 473 472 471 470 469 468 467 466 465 464 463 462 461 460 459
458 457 456 455 454 453 452 451 450 449 448 447 446 445 444 443 442 441 440 439 438
437 436 435 434 433 432 431 430 429 428 427 426 425 424 423 422 421 420 419 418 417
416 415 414 413 412 411 410 409 408 407 406 405 404 403 402 401 400 399 398 397 396
395 394 393 392 391 390 389 388 387 386 385 384 383 382 381 380 379 378 377 376 375
374 373 372 371 370 369 368 367 366 365 364 363 362 361 360 359 358 357 356 355 354
353 352 351 350 349 348 347 346 345 344 343 342 341 340 339 338 337 336 335 334 333
332 331 330 329 328 327 326 325 324 323 322 321 320 319 318 317 316 315 314 313 312
311 310 309 308 307 306 305 304 303 302 301 300 299 298 297 296 295 294 293 292 291
290 289 288 287 286 285 284 283 282 281 280 279 278 277 276 275 274 273 272 271 270
269 268 267 266 265 264 263 262 261 260 259 258 257 256 255 254 253 252 251 250 249
248 247 246 245 244 243 242 241 240 239 238 237 236 235 234 233 232 231 230 229 228
227 226 225 224 223 222 221 220 219 218 217 216 215 214 213 212 211 210 209 208 207
206 205 204 203 202 201 200
```

In [14]:
```
# Reverse numbers from 500 ~ 200.
for num in range(500,199,-2):
    print(num,end=" ")
```

```
500 498 496 494 492 490 488 486 484 482 480 478 476 474 472 470 468 466 464 462 460
458 456 454 452 450 448 446 444 442 440 438 436 434 432 430 428 426 424 422 420 418
416 414 412 410 408 406 404 402 400 398 396 394 392 390 388 386 384 382 380 378 376
374 372 370 368 366 364 362 360 358 356 354 352 350 348 346 344 342 340 338 336 334
332 330 328 326 324 322 320 318 316 314 312 310 308 306 304 302 300 298 296 294 292
290 288 286 284 282 280 278 276 274 272 270 268 266 264 262 260 258 256 254 252 250
248 246 244 242 240 238 236 234 232 230 228 226 224 222 220 218 216 214 212 210 208
206 204 202 200
```

Q. Wapp to print the multiplication table of any entered number.

In [18]:
```
n = int(input("Enter a number: "))
for i in range(1,11):
    print(f"{n} X {i} = {n*i}")
```

```
45 X 1 = 45
45 X 2 = 90
45 X 3 = 135
45 X 4 = 180
45 X 5 = 225
45 X 6 = 270
45 X 7 = 315
45 X 8 = 360
45 X 9 = 405
45 X 10 = 450
```

Q. Wapp to print only the vowels from any entered string value.

In [23]:
```
string = input("Enter a string: ")
for i in string:
    if i.lower() in 'aeiou':
        print(i,end=" ")
```

```
A
```

Jump Statement: A jump statement is used to break the normal flow of the program, it transfers the execution control of the program to another statement if the specific condition is true.

Break Statement: The break is a keyword in python which is used to bring the program control out of the loop.

In [24]:
```
for i in 'python':
    print(i)
```

```
p
y
t
h
o
n
```

In [25]:
```
for i in 'python':
    print(i)
    break
```

```
p
```

In [26]:
```
for i in 'python':
    if i == 'h':
        break
    print(i)
```

```
p
y
t
```

Continue Statement: the continue keyword return control of the iteration to the beginning of the Python for loop or Python while loop. All remaining lines in the prevailing iteration of the loop are skipped by the continue keyword, which returns execution to the beginning of the next iteration of the loop.

In [27]:
```
for i in 'python':
    print(i)
```

```
p
y
t
h
o
n
```

In [28]:
```
for i in 'python':
    print(i)
    break
```

```
p
```

In [29]:
```
for i in 'python':
    print(i)
    continue
```

```
p
y
t
h
o
n
```

In [30]:
```
for i in 'python':
    continue
```

```
    print(i)
```

In [31]:
```
for i in 'python':
    if i == 'o':
        continue
    print(i)
```

```
p
y
t
h
n
```

Pass Statement: The Python pass statement to write empty loops. Pass is also used for empty control statements, functions, and classes.

In [33]:
```
for i in 'python':
    pass
```

In [34]:
```
for i in 'python':
    pass
    print(i,end=" ")
```

```
p y t h o n
```

Functions: A reusable piece of code. Block of statements that does some specific task and returns something.

- By including functions, we can prevent repeating the same code block repeatedly in a program.
- Python functions, once defined, can be called many times and from anywhere in a program.
- If our Python program is large, it can be separated into numerous functions which is simple to track.
- The key accomplishment of Python functions is we can return as many outputs as we want with different arguments.

Syntax of Python Function:

```
def function_name( parameters ):
    #code block
```

In [36]:
```
# Creating a function:
def Function1():    # function declaration
    print("Hello Everyone")  # Function defination

# Function call
Function1()
```

```
Hello Everyone
```

In [37]:
```
Function1()
```

```
Hello Everyone
```

In [38]:
```
# Function With Return:
def WithRet():
    return "Hello"

WithRet()
```

Out[38]:
```
'Hello'
```

In [40]:
```
WithRet() + ' How are you'
```

Out[40]:
```
'Hello How are you'
```

In [39]:
```
# Function without return:
def WithoutRet():
    print("Hello")

WithoutRet()
```

```
Hello
```

In [41]:
```
WithoutRet() + ' How are you'
```

```
Hello
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[41], line 1
----> 1 WithoutRet() + ' How are you'

TypeError: unsupported operand type(s) for +: 'NoneType' and 'str'
```

In [42]:
```
# Function with Arguements:-
# Single Arguement:
def Single(x):
    print(f"Value of x is {x}")

Single(50)
```

```
Value of x is 50
```

In [43]:
```
Single('apple')
```

```
Value of x is apple
```

In [44]:
```
# Double Arguement:-
def Double(a,b):
```

```
    print(f"a -> {a}, b -> {b}")

Double(45,86)
```

```
a -> 45, b -> 86
```

In [45]:
```
Double(23)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[45], line 1
----> 1 Double(23)

TypeError: Double() missing 1 required positional argument: 'b'
```

In [46]:
```
Double(45,78,36)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[46], line 1
----> 1 Double(45,78,36)

TypeError: Double() takes 2 positional arguments but 3 were given
```

In [47]:
```
Double('apple','kiwi')
```

```
a -> apple, b -> kiwi
```

In [52]:
```
# Multiple arguements:
def Multi(*arg):
    print(f'value/s in arg -> {arg}')
    print(type(arg))
```

In [53]:
```
Multi(15)
```

```
value/s in arg -> (15,)
<class 'tuple'>
```

In [54]:
```
Multi(4,5,8,6,3,2,7)
```

```
value/s in arg -> (4, 5, 8, 6, 3, 2, 7)
<class 'tuple'>
```

In [55]:
```
# Create a function to check whether a number is even or odd.
def EvOd(num):
    if num%2==0:
        print('Even')
    else:
        print('Odd')
```

In [56]:
```
EvOd(45)
```

```
Odd
```

In [57]:
```
EvOd(782)
```

```
Even
```

Q. Create a function to print all the factors of an entered number.

In [ ]: