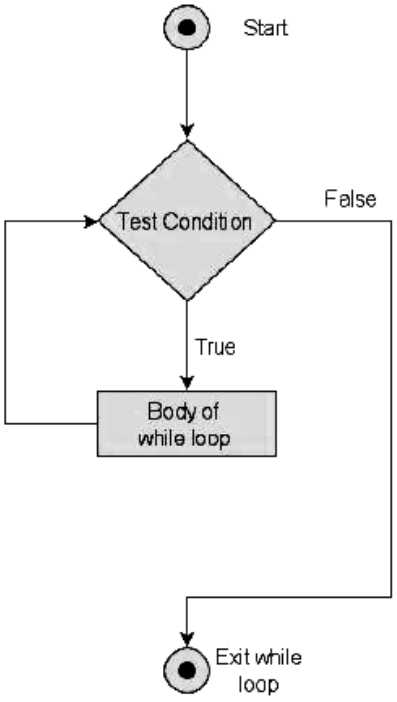
පුනර්කරණය (Repetition / Iteration)

පයිතන් උපදෙස් මාලාවක කිසියම් කේතනයක් හෝ කේතනයන් කිහිපයක් නැවත නැවත කිසියම් වාර ගණනාවක් ක්‍රියාත්මක කළ යුතු නම් එවැනි අවස්ථාවක දී භාවිතා කරන උපදේශනයන් පුනර්කරණය ලෙස හැඳින්වේ. මේ සඳහා භාවිතා කරනු ලබන කේතනයන් ආකාර දෙකකි.

while	ලබා දෙන කොන්දේසියක් සත්‍යව පවතින තාක් කල් කිසියම් උපදේශනයක් හෝ උපදේශන කිහිපයක් නැවත නැවත ක්‍රියාත්මක කිරීම සඳහා භාවිතා කරනු ලැබේ.
for	උපදේශන එකක් හෝ ඊට වැඩි ගණනක් නිශ්චිත වාර ගණනක් හෝ යමක අන්තර්ගතය අවසන් වන තෙක් නැවත නැවත ක්‍රියාත්මක කිරීම සඳහා භාවිතා කරනු ලබයි.
Nested Loops	ලූපයක් තුළ ලූපයක් භාවිතා කිරීමයි.

while පුනර්කරණ ව්‍යුහය



යම් උපදෙස් මාලාවක් කිසියම් ප්‍රකාශනයක් සත්‍ය වන තෙක් නැවත නැවත කිරීම සඳහා මෙම ව්‍යුහය භාවිතා කෙරේ. මෙම ව්‍යුහයේ දැක්වෙන කොන්දේසිය සත්‍යව පවතින තෙක් ප්‍රකාශයන් පුනර්කරණය කෙරේ.

කාරක නීති මාලාව :

```
while expression :
   statement( s )
```

උදා:

```
01.
x = 0
while x<5 :
```

```
   print(" OK ")
   x= x+1
```

……
……
……

02.

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

……

03.

```
count = 0
while (count < 9):
   print('The count is:', count)
   count = count + 1
print("Good bye!")
```

……
……
……
……
……
……
……

04.

```
var = 1
while var == 1 :  # This constructs an infinite loop
   num = input("Enter a number  :")
   print ("You entered: ", num)
print ("Good bye!")
```

……
……
……

05.

```
count = 0
while count < 5:
   print(count, " is  less than 5")
   count = count + 1
else:
   print(count, " is not less than 5")
```

……
……
……

06.
```
flag = 1
while (flag): print ('Given flag is really true!')
print ("Good bye!")
```
.....
.....
.....

for පුනර්කරණ ව්‍යුහය

මෙම ප්‍රකාශනය පුනරාවර්තනය කළ හැකි අනුක්‍රමයක සෑම අවයවයක් මතම කිසියම් උපදෙස් මාලාවක් ක්‍රියාකරවීම සඳහා යොදා ගැනේ.

කාරක රීතිය :

```
for iterating_var in sequence:
    statements(s)
```

උපදේශයන් එකක් හෝ ඊට වැඩි ගණනක් නිශ්චිත වාර ගණනක් නැවත නැවත ක්‍රියාත්මක කර ගැනීමට අවශ්‍ය වූ විට මෙම උපදේශනය භාවිතයට ගනු ලැබේ. මෙය භාවිතා කළ හැකි අවස්ථාවන් තුනක් පවතියි.

1. කිසියම් සංඛ්‍යා පරාසයක් යොදා ගැනීම මගින්
2. දත්ත ලැයිස්තුවක් (List) සමග එහි ඇති දත්ත අවසන් වන තෙක්
3. අක්ෂර සහිත දත්ත කාණ්ඩයක් (String) අවසන් වන තෙක්

01.
```
for letter in "Python":
   print("Current Letter : ", letter)
```
.....
.....
.....
.....
.....
.....

02.
```
fruits = ['banana', 'apple', 'mango']
for fruit in fruits :
   print("Current fruit :", fruit)
```
.....
.....
.....

03.
```
fruits = ['banana', 'apple', 'mango']
for index in range(len(fruits)) :
   print("Current fruit :", fruits[index])
```
.....
.....
.....

04.
```
var = range(0,10)
for i in var : print(i)
```
……

05.
```
primes = [2, 3, 5, 7]
for prime in primes:
   print(prime)
```
……

06.
```
# Prints out the numbers 0,1,2,3,4
for x in range(5):
   print(x)

# Prints out 3,4,5
for x in range(3, 6):
   print(x)

# Prints out 3,5,7
for x in range(3, 8, 2):
   print(x)
```

07.
```
# Prints out 0, 1,2,3,4
count = 0
while count < 5:
   print(count)
   count += 1  # This is the same as count = count + 1
```

පුනර්කරණ පාලනයන් (Loop Control Statements)

පුනර්කරණ ව්‍යුහයක සාමාන්‍ය ක්‍රියාත්මක කිරීම වෙනස් කළ හැකි ආකාරයන් තුනක් ඇත. මේවා පයිතන් මූල පද (Keywords) වල ද සඳහන් වේ.

Control Statement	Description
break statement	Terminates the loop statement and transfers execution to the statement immediately following the loop.
continue statement	Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
pass statement	The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

බණ්ඩනය (Break)

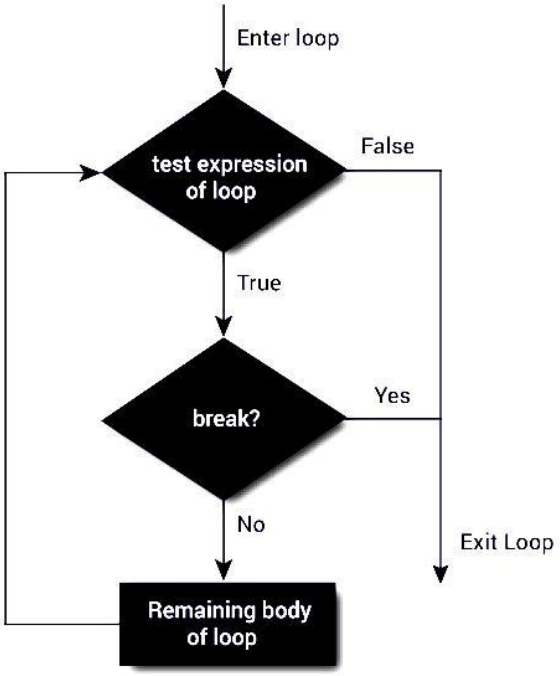
පුනර්කරණයක් තුළ දී ලබා දෙන කොන්දේසියක් සත්‍ය වන තෙක් හෝ කිසියම් වූ නිශ්චිත වාර ගණනක් නැවත නැවත ක්‍රියාත්මක වීමක් සිදුවේ. නමුත් අවශ්‍ය නම් යම් කොන්දේසියක් සත්‍ය වුවහොත් පුනර්කරණය දිගින් දිගටම සිදු නොවී නැවැත්වීමේ හැකියාවක් පවතියි. මේ සඳහා බණ්ඩනය (Break) භාවිතා කරනු ලැබේ.

```
for var in sequence:
    # codes inside for loop
    if  condition:
        break
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        break
    # codes inside while loop

# codes outside while loop
```



01.

```
# Use of break statement inside loop
for val in "string":
   if val == "i":
      break
   print(val)
print("The end")
```

……
……
……

02.

```
for letter in "Python" :
   if letter == 'h':
      break
   print("Current Letter :", letter)
```

……

03.

```
var = 10
while var > 0:
   print("Current Letter :",var)
   var = var – 1
   if var == 5:
      break
```

……
……
……
……

යළි කරගෙන යාම (continue)

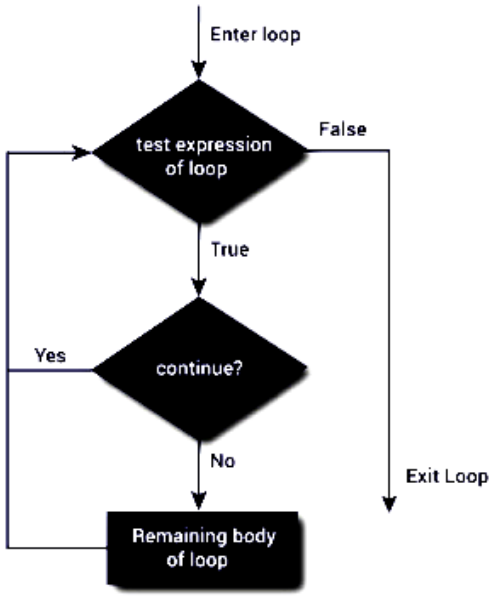
පුනර්කරණයක් ක්‍රියාත්මක වෙමින් පවතින අවස්ථාවක දී break විධානය මගින් එය නැවැත්විය හැකි අතර මෙසේ break විධානයෙන් නැවැත් වූ පසු එම පුනර්කරණය නැවත ක්‍රියාත්මක කිරීමට අවශ්‍ය වූ විටක දී continue විධානය භාවිතා කළ හැකිය.

```
for var in sequence:
    # codes inside for loop
    if  condition:
        continue
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        continue
    # codes  inside while loop

# codes outside while loop
```



01.

```
# Program to show the use of continue statement inside loops
for val in "string":
   if val == "i":
      continue
   print(val)
```

```
print("The end")
```
……
……
……

02.
```
for letter in "Python" :
   if letter == 'h':
      continue
   print("Current Letter :", letter)
```
……
……
……

03.
```
var = 10
while var > 0:
   var = var -1
   if var == 5:
      continue
   print("Current variabe value :", var)
```
……
……
……
……

pass

```
for letter in "Python" :
   if letter == 'h':
      pass
   print("Current Letter :", letter)
```
……
……
……

පහත පයිතන් ක්‍රමලේඛයන්හි ප්‍රතිදානය ලියා දක්වන්න.

# Measure some strings: 	
words = ['Father', 'Brother', 'Daughter']
for w in words:
  print(w, len(w))	

for i in range(5): print(i)	
range(5, 10) range(0, 10, 3) range(-10, -100, -30)	
a = ['Mary', 'had', 'a', 'little', 'lamb'] for i in range(len(a)): print(i, a[i])	

print(range(10))	
list(range(5))	

for num in range(2, 10):
  if num % 2 == 0:
  print("Found an even number", num)
  continue
  print("Found a number", num)	

while True: pass # Busy-wait for keyboard interrupt (Ctrl+C)	

balance = 522
if balance < 0:
  print("Balance is below 0, add funds now or you will be charged a penalty.")
else:
  print("Your balance is 0 or above.")	

count = 0
while True:
  print(count)
  count += 1
  if count >= 5:
  break
 	
for x in range(10):
  # Check if x is even
  if x % 2 == 0:
  continue
  print(x)	