அலகு II

பாடம் 6

கட்டுப்பாட்டு கட்டமைப்புகள்

HSQ5E2

கற்றலின் நோக்கங்கள்

இந்தப் பாடப்பகுதியைக் கற்றபின் மாணவர்கள் அறிந்துக் கொள்வது:

- பைத்தான் மொழியில் பல்வேறு பாய்வுக்கட்டுப்பாடுகளைப் பற்றிய அறிவைப் பெறுதல்.
- நிபந்தனை அமைப்பை பயன்படுத்தி நிரலின் பாய்வு செயல் திறனை எவ்வாறு மேம்படுத்துவது என்பதை கட்டளை அமைப்பு மூலம் அறிதல்.
- பன்முறை செயல் அமைப்பை அல்லது மடக்கை பயன்படுத்தி நிரல் பகுதியை குறிப்பிட்ட எண்ணிக்கை வரை அல்லது நிபந்தனை நிறைவேற்றப்படும் வரை திரும்ப செய்யும் குறிமுறையை உருவாக்குதல்.

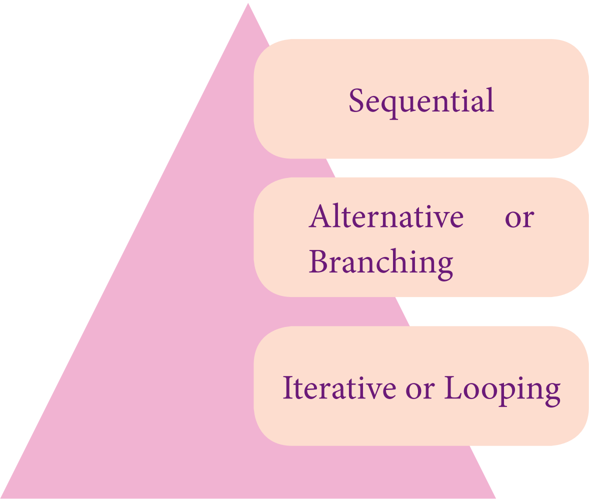
6.1 அறிமுகம்

நிரல்கள் கூற்றுகளின் தொகுதியை கொண்டிருக்கும். இந்த கூற்றுகளே விடைகளைக் கொடுக்கும் இயக்கப் பகுதிகளாகும். பொதுவாக, கூற்றுகள் வரிசையாக, அதாவது கூற்றுகள் ஒன்றன் பின் ஒன்றாக நிறைவேற்றப்படும். ஏதேனும் சில சமயங்களில் நிகழ் நேர நிரல்களை இயக்கும் போது நிரலின் ஒரு பகுதியை அல்லது கூற்றுகளின் தொகுதியை நிறைவேற்றாமல் விட்டு விட்டு நிபந்தனையின் அடிப்படையில் நிரலின் மற்றொரு பகுதியை இயக்க நேரிடும். இதற்கு **"மாற்று"** அல்லது **"கிளைப்பிரிப்பு"** என்று பெயர். மேலும், கூற்றுகளிள் உள்ள ஒரு தொகுதியை பல தடவை நிறைவேற்றவேண்டி இருக்கும். இதற்கு **பன்முறைசெயல்** அல்லது **மடக்கு** என்று பெயர். இந்தப் பாடப் பகுதியில் பைத்தானில் பல்வேறு கட்டுப்பாட்டு கட்டமைப்புகள் அதன் கட்டளை அமைப்புகள் மற்றும் அவற்றைப் பயன்படுத்தி நிரல்கள் உருவாக்குதல் மற்றும் கற்றல் ஆகியவற்றைக் காண்போம்.

6.2 கட்டுப்பாட்டு கட்டமைப்புகள்

கட்டுப்பாட்டு நிரலின் ஒரு பகுதியில் இருந்து இன்னொரு பகுதிக்கு தாவுவதற்கு காரணமான நிரல் கூற்றுகள் **கட்டுப்பாட்டு கட்டமைப்பு** அல்லது **கட்டுப்பாட்டு கூற்றுகள்** எனப்படும். ஏற்கனவே, C++ல் கற்றதைப் போல் இந்த கட்டுப்பாட்டுக் கூற்றுகள் கலப்புக் கூற்றுகளாகும். செயல் முறையின் நிலையைப் பொறுத்து, செயல்முறையின் கட்டுப்பாட்டு பாய்வை அல்லது நிரலின் கட்டுப்பாட்டு பாய்வை மாற்றும்.

பைத்தானில் மூன்று முக்கிய வகையான கட்டுப்பாட்டு அமைப்புகள் உள்ளன.



6.2.1 வரிசை முறை கூற்றுகள் (Sequential Statements)

ஒன்றன் பின் ஒன்றாக நிறைவேற்றப்படும் கூற்றுகளின் வரிசையைக் கொண்டது வரிசை முறை கூற்று ஆகும். உன்னுடைய பெயர், முகவரி மற்றும் தொலைபேசி எண்ணை அச்சிடும் குறிமுறை, வரிசைமுறை கூற்றுக்கு எடுத்துக்காட்டு ஆகும்.

எடுத்துக்காட்டு 6.1

```
# உன்னுடைய பெயர் மற்றும் முகவரியை அச்சிடும் நிரல்- வரிசைமுறை கூற்றுக்கு எடுத்துக்காட்டு
print ("Hello! This is Shyam")
print ("43, Second Lane, North Car Street, TN")
```

வெளியீடு :

```
Hello! This is Shyam
43, Second Lane, North Car Street, TN
```

6.2.2 மாற்று அல்லது கிளைப் பிரிப்புகூற்று (Alternative or Branching Statement)

நம்முடைய அன்றாட வாழ்வில் நாம் பல்வேறு முடிவுகளை எடுக்க வேண்டியுள்ளது. நமது நோக்கத்தை அடைய மாற்றுப் பாதையைத் தேர்ந்தெடுக்கிறோம். தினசரி நாம் செல்லும் சாலை தடை செய்யப்பட்டிருந்தால் மாற்றுப் பாதையை பயன்படுத்தி, நாம் சென்றடைய வேண்டிய இடத்திற்கு சென்று சேர்ந்திடுவோம். இந்த முடிவெடுத்தலை நாம் மாற்று அல்லது கிளைப் பிரிப்பு கூற்று மூலம் கற்க இருக்கின்றோம். கொடுக்கப்பட்ட எண் நேர்மறை எண் அல்லது எதிர்மறை எண்ணா அல்லது ஒற்றைப்படை எண்ணா அல்லது இரட்டைப்படை எண்ணா என்பதை கண்டறிவதற்கு மாற்று அல்லது கிளைப் பிரிப்பு கூற்று பயன்படுகிறது.

பைத்தானில் கீழ்க்கண்ட மாற்று (அல்லது) கிளைப்பிரிப்பு கூற்று வகைகள் உள்ளன:

- Simple if கூற்று
- if..else கூற்று
- if..elif கூற்று

6.2.2.1 Simple if கூற்று

Simple if கூற்று என்பது அனைத்து தீர்மானிப்பு கூற்றுகளிலும் மிக எளிதான கூற்றாகும். நிபந்தனையானது ஒப்பீட்டு கோவையாகவோ அல்லது தருக்க சேவையாகவோ இருத்தல் வேண்டும்.

பொதுவடிவம்:

```
    if <condition>:
            statements-block1
```

மேலே உள்ள கூற்றில் நிபந்தனை சரி என்றால் **statements-block1** நிறைவேற்றப்படும்.

எடுத்துக்காட்டு 6.2:

```
# வயதை சரி பார்த்து வாக்களிக்க தகுதியா என அச்சிடும் நிரல்
x=int (input("Enter your age :"))
if x>=18:
        print ("You are eligible for voting")
```

வெளியீடு1:

```
        Enter your age :34
        You are eligible for voting
```

வெளியீடு 2:

```
        Enter your age :16
        >>>
```

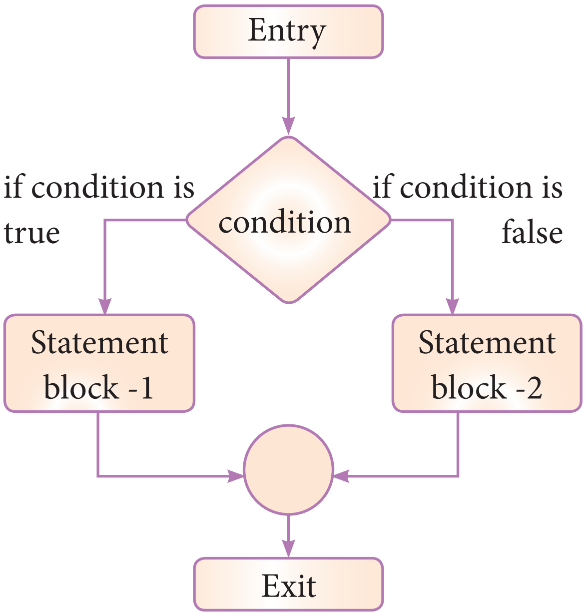
இரண்டாவது வெளியீட்டில் வெளியீடு அச்சிடப்படவில்லை, பைத்தானில் **தூண்டு குறி** மட்டுமே காண்பிக்கிறது. ஏனென்றால், நிபந்தனை தவறாகும் போது நிரலில் என்ன செய்ய வேண்டும் என்பது கொடுக்கப்படவில்லை.

6.2.2.2 if..else கூற்று

if .. else கூற்றானது சரி தொகுதி மற்றும் தவறு தொகுதி இரண்டையுமே சரிபார்ப்பதற்கான கட்டுப்பாட்டை வழங்குகிறது. **'if..else'** கூற்றின் தொடரியல்.

தொடரியல்:

```
if <condition>:
        statements-block 1
else:
        statements-block 2
```



படம் 6.1 *if..else* கூற்றின் செயல்பாடு

if..else கூற்று இரு மாற்று வழிகளை வழங்குகிறது. நிபந்தனையானது எந்த தொகுதியை நிறைவேற்றப்பட வேண்டும் என்பதைத் தீர்மானிக்கிறது.

எடுத்துக்காட்டு 6.3:

```
#உள்ளிடப்பட்ட எண் ஒற்றைப்படை அல்லது இரட்டைப்படை எண்ணா என்பதைக்
கண்டறியும் நிரல்
a = int(input("Enter any number :"))
if a%2==0:
        print (a, " is an even number")
else:
        print (a, " is an odd number")
```

வெளியீடு 1:

```
Enter any number :56
56  is an even number
```

வெளியீடு 2:

```
Enter any number :67
67  is an odd number
```

மேற்கண்ட நிரலுக்கு மாற்று முறையில் வேறு மாதிரியாக பைத்தானில் எழுத முடியும். முழுமையான **if..else** கூற்று பின்வருமாறு எழுதலாம்:

தொடரியல்:

variable = variable1 if condition else variable 2

குறிப்பு:

if ல் குறிப்பிடப்பட்டுள்ள நிபந்தனை பரிசோதிக்கப்படும் நிபந்தனை சரி எனில் மாறி1-ன் (variable1) மதிப்பு, இடது பக்கத்திலுள்ள மாறியில் இருத்தப்படும். இல்லையெனில், மாறி2-ன் (variable2) மதிப்பு இருத்தப்படும்.

எடுத்துக்காட்டு 6.4: # உள்ளிடப்பட்ட எண் ஒற்றைப்படை அல்லது இரட்டைப்படை எண்ணா என்பதைக் கண்டறியும் நிரல் (மாற்று முறையைப் பயன்படுத்தி)

```
a = int (input("Enter any number :"))
x="even" if a%2==0 else "odd"
print (a, " is ",x)
```

வெளியீடு 1:

```
Enter any number :3
3  is  odd
```

வெளியீடு 2:

```
Enter any number :22
22  is  even
```

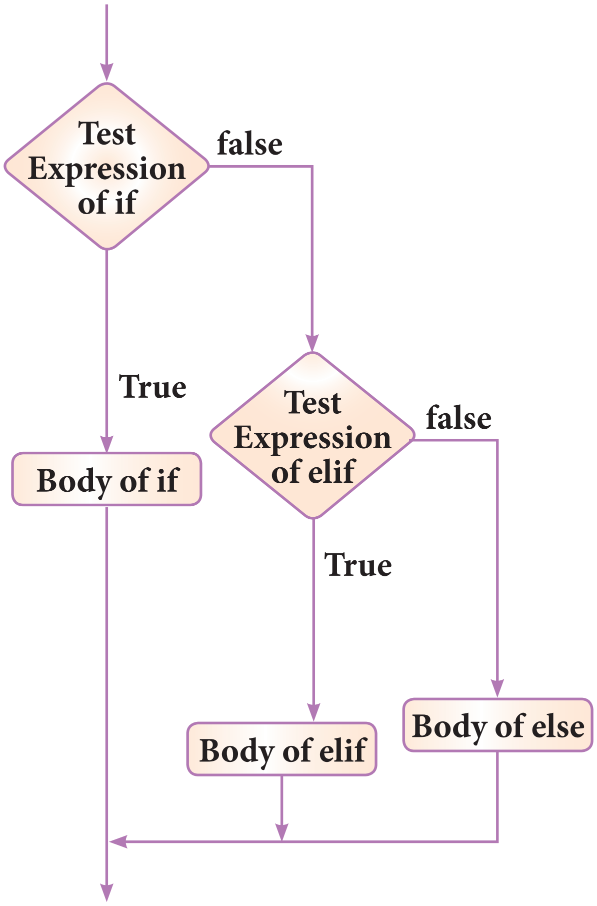
6.2.2.3 பின்னலான if..elif...else கூற்று:

if கூற்றுகளைத் தொடர் கூற்றுகளாக அமைக்க விரும்பும் போது ‘else’ பகுதிக்குப் பதிலாக ‘elif’ பகுதி பயன்படுத்தலாம்.

தொடரியல் :

```
if <condition-1>:
        statements-block 1
elif <condition-2>:
        statements-block 2
else:
        statements-block n
```

மேற்கண்ட if..elif..else பொதுவடிவத்தில், condition-1 பரிசோதிக்கப்படும். condition -1 சரி எனில் statements-block 1 செயல்பாட்டுத் condition-1 நிறைவேற்றப்படும். இல்லையெனில் கட்டுப்பாடு, condition-2யைப் பரிசோதிக்கும். condition-2 சரி எனில் (statements-block-2) நிறைவேற்றப்படும். இல்லையெனில் else பகுதியில் உள்ள (statements-block n) நிறைவேற்றப்படும்.



படம் *6.2 if..elif..else* கூற்றின் செயல்பாடு

'elif' பகுதி **if..else-if..else** கூற்றுகளை இணைக்கு ஒன்றிணைத்து ஒரே **if..elif…else. 'elif'** ன் விரிவாக்கம் **'else if'. 'if'** கூற்றில் பயன்படுத்தப்படும் **'elif'** பகுதிக்கு வரம்பு இல்லை, ஆனால் **else** பகுதி பயன்படுத்தும்போது கூற்றின் இறுதியில் பயன்படுத்த வேண்டும்.

குறிப்பு

C++ல் படித்த பின்னலான **if** கூற்றுக்கு நிகரானது பைத்தானில் உள்ள **if..elif…else** கூற்று.

எடுத்துக்காட்டு 6.5:

கீழே கொடுக்கப்பட்ட நிபந்தனையின் படி மாணவர்கள் பெற்ற தர வரிசையை அச்சிட வேண்டிய நிரல்

சராசரி	தரம்
>=80 and above	*A*
>=70 and <80	*B*
>=60 and <70	*C*
>=50 and <60	D
Otherwise	E

```
#பின்னலான if கூற்றின் பயன்பாட்டை விளக்கும் நிரல்
m1=int (input(“Enter mark in first subject : ”))
m2=int (input(“Enter mark in second subject : ”))
avg= (m1+m2)/2
if avg>=80:
        print (“Grade : A”)
elif avg>=70 and avg<80:
        print (“Grade : B”)
elif avg>=60 and avg<70:
        print (“Grade : C”)
elif avg>=50 and avg<60:
        print (“Grade : D”)
else:
        print (“Grade : E”)
```

வெளியீடு 1:

```
Enter mark in first subject : 34
Enter mark in second subject : 78
Grade : D
```

வெளியீடு 2 :

```
Enter mark in first subject : 67
Enter mark in second subject : 73
Grade : B
```

குறிப்பு

கொடுக்கப்பட்ட எடுத்துக்காட்டில் if-பகுதிக்கும் நான்கு இடைவெளிகள் உள் தள்ளப்பட்டுள்ளது. இது பைத்தானில் பொதுவாக உள் தள்ளப்பட வேண்டிய அளவாகும். பிற நிரலாக்க மொழிகளில் உள் தள்ளல் என்பது நிரலை அழகாக காண்பிப்பதற்காக பயன்படுகிறது. ஆனால் பைத்தானில் எந்த தொகுதியினுடைய கூற்றுகள் என்று குறிப்பதற்காக உள் தள்ளல் அவசியம் தேவைப்படுகிறது.

எடுத்துக்காட்டு 6.5.a:

```
# if  கூற்றில் 'in' மற்றும் 'not in' பயன்பாட்டை விளக்கும் நிரல்
ch=input ("Enter a character :")
# to check if the letter is vowel
if ch in ('a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U'):
        print (ch,' is a vowel')
# to check if the letter typed is not 'a' or 'b' or 'c'
if ch not in ('a', 'b', 'c'):
        print (ch,'  the letter is not a/b/c')
```

வெளியீடு1:

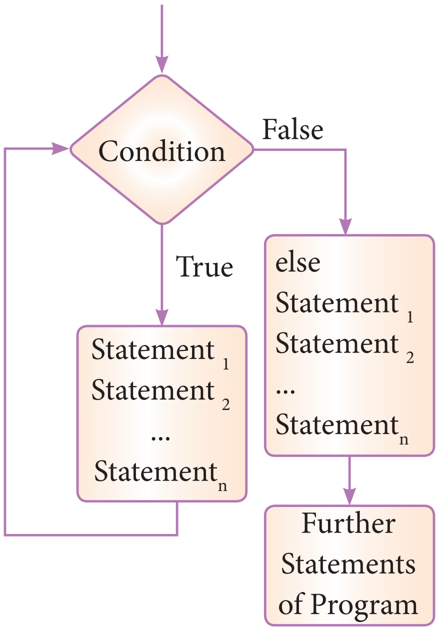
```
        Enter a character :e
        e is a vowel
```

வெளியீடு 2:

```
        Enter a character :x
        x the letter is not a/b/c
```

6.2.3. பன்முறைச் செயல் அல்லது மடக்கு அமைப்பு

பன்முறைச் செயல் அல்லது மடக்கு என்பது பயனர் விரும்பும் குறிமுறைத் தொகுதியை குறிப்பிட்ட எண்ணிக்கை வரை அல்லது நிபந்தனை நிறைவேற்றப்படும் வரை இயக்குவதாகும். ஒரு கூற்று அல்லது பல கூற்றுகளை பல முறை நிறைவேற்ற கூற்று அனுமதிக்கிறது.



படம் 6.3 மடக்கு அமைப்பு எவ்வாறு இயங்குகிறது என்பதை விளக்கும் படம்

WE6I7K

பைத்தான் இரண்டு வகையான மடக்கு அமைப்புகளை வழங்குகிறது:

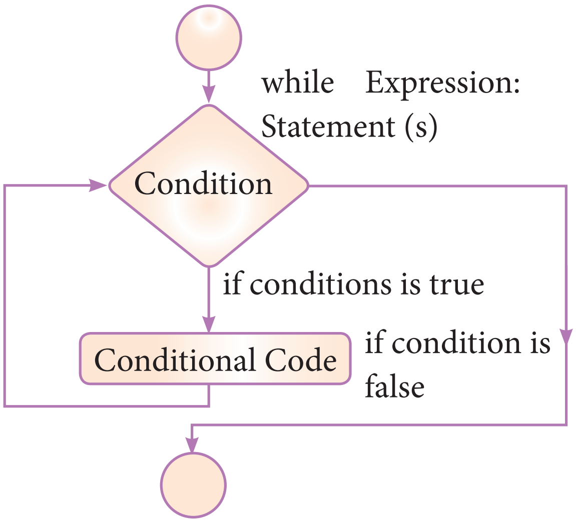
- while மடக்கு
- for மடக்கு

6.2.3.1 while மடக்கு

பைத்தானில் **while** மடக்கு பின்வரும் தொடரியல் கொண்டிருக்கும்.

தொடரியல்:

```
while <condition>:
        statements block 1
[else:
        statements block2]
```



படம் 6.4 மடக்கின் செயல்பாடு

while மடக்கில், நிபந்தனையானது ஏதாவது ஒரு தகுதியான பூலியன் கோவை ஆகும். இது சரி அல்லது தவறு என்ற மதிப்பை தரும். இதன் **else பகுதி கட்டாயப் பகுதி அல்ல, நிபந்தனை சரி என்று இருக்கும் வரை, செயல்பாட்டுத் தொகுதி 1, (statements block 1) நிறைவேற்றப்படும்**. **else பகுதி எழுதப்பட்டிருந்தால் நிபந்தனை தவறு** எனில் **else பகுதி நிறைவேற்றப்படும்**. **while மடக்கானது** நுழைவு சோதிப்பு மடக்கு என்பதை நினைவில்கொள்க. மடக்கின் உள்ளே நுழையும் போது நிபந்தனை தவறு என்றால் மடக்கில் உள்ள கூற்றுகள் ஒரு முறை கூட நிறைவேற்றப்படாது.

எடுத்துக்காட்டு 6.6: while மடக்கை பயன்படுத்தி 10லிருந்து 15 வரை அனைத்து எண்களையும் அச்சிடுவதை விளக்கும் நிரல்.

```
i=10                        # intializing part of the control variable
while (i<=15):              # test condition
        print (i,end='\t')  # statements - block 1
        i=i+1               # Updation of the control variable
```

வெளியீடு:

```
10   11   12   13   14   15
```

கட்டுப்பாட்டு மாறியான iன் தொடக்க மதிப்பு 10 ஆகும். i<=15 என்ற சோதனை பரிசோதிக்கப்பட்டு i-ன் மதிப்பு சரியாக இருந்தால் **i** ன் மதிப்பு அச்சிடப்படும். பிறகு, i ன் மதிப்பு i=i+1 என புதுப்பிக்கப்படும் (இதை i+=1 என குறுக்கு வழி மதிப்பிருந்து செயற்குறியை பயன்படுத்தியும் எழுதலாம். i=16 என ஆகும் போது நிபந்தனைதவறாகும், அப்பொழுது மடக்கானது நிறுத்தப்படும்.

குறிப்பு

print கூற்று **end** மற்றும் **sep** அளபுருக்களை கொண்டிருக்கும். **end** அளபுருவை விடுபடு வரிசையை கொடுக்க விரும்பும்போது ('**\t**'தத்தல், '**\n**' புதிய வரி மற்றும் பல) பயன்படுத்தலாம். **sep** அளபுருசிறப்பு குறியீடுகளான, (காற்புள்ளி) ;(அரைப்புள்ளி) போன்ற மதிப்புகளை பிரிக்க பயன்படும் **print**() வடிவூட்டல் விருப்பங்களை இதற்கு முன் பாடத்தில் படித்ததை நினைவு கூறுக.

பின்வரும் எடுத்துக்காட்டு **while** மடக்கில் **else** பயன்பாட்டை காட்டுகிறது.

எடுத்துக்காட்டு 6.7: # else பகுதியுடன் கூடிய while மடக்கை விளக்கும் நிரல்

```
i=10                                        # intializing part of the control variable
while (i<=15):                              # test condition
        print (i,end='\t')                  # statements - block 1
        i=i+1                               # Updation of the control variable
else:
      print ("\nValue of i when the loop exit ",i)
```

வெளியீடு: 1

```
10    11    12    13    14    15
Value of i when the loop exit  16
```

6.2.3.2 for மடக்கு

for மடக்கு சுலபமாக பயன்படுத்தக் கூடிய ஓர் மடக்காகும். இது ஒரு நுழைவு சோதிப்பு மடக்காகும். நிபந்தனை முதலிலேயே சோதிக்கப்பட்டு சரி எனில் மடக்கின் உடற்பகுதி செயல்பாட்டுத் தொகுதியை நிறைவேற்றப்படும். இல்லையெனில் மடக்கு நிறைவேறாமல் வெளியேறும்.

தொடரியல்:

```
for counter_variable in sequence:
        statements-block 1
[else:                          # optional block
        statements-block 2]
```

தொடரியலில் குறிப்பிட்டுள்ள தொடரியல் மாறியானது (counter_variable), C++-ல் உள்ள for மடக்கில் பயன்படுத்தும் கட்டுப்பாட்டு மாறியை ஒத்ததாகும். வரிசை என்பது தொடக்க, இறுதி மற்றும் மிகுப்பு மதிப்புகளைக் குறிப்பதாகும். பொதுவாக, பைத்தானில் for மடக்கில் வரிசையில் உள்ள தொடக்க, இறுதி, மதிப்புகளைக் குறிப்பதற்காக range( ) செயற்கூறு பயன்படுகிறது. range() செயற்கூறு start முதல் stop-1 வரையிலான மதிப்பு பட்டியலை உருவாக்குகிறது.

range ()ன் தொடரியல்:

range(start,stop,[step])

இதில்,

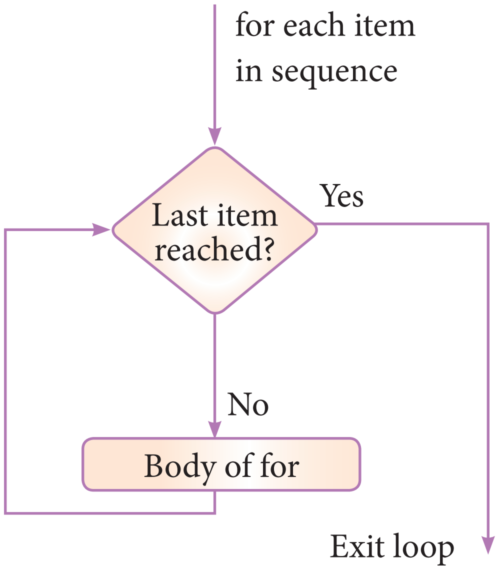
start – தொடக்க மதிப்பைக் குறிக்கும்

stop – இறுதி மதிப்பைக் குறிக்கும்

step – மிகுப்பு மதிப்பைக் குறிக்கும். இது விருப்பப் பகுதியாகும்.

எடுத்துக்காட்டு 6.8: range() எடுத்துக்காட்டுகள் :

range (1,30,1)	*range*-ன் மதிப்புகள் *1* முதல் தொடங்கி *29* வரை
range (2,30,2)	*range()* மதிப்புகள் *2* முதல் *28* வரை
range (30,3,-3)	*range()* மதிப்புகள் *30* முதல் *6* வரை
range (20)	*20* என்ற மதிப்பை இறுதி மதிப்பாக எடுத்துக்கொண்டு (அல்லதுமேல்வரம்பு) *0* முதல் *19* வரையிலான மதிப்புகளைக் கொடுக்கும் (*range()* எப்பொழுதும் *stop* மதிப்பு-*1* வரை மதிப்பைக் கொடுக்கும் என்பதை நினைவில்கொள்க).



படம் 6.5 *for* மடக்கின் செயல்பாடு

எடுத்துக்காட்டு 6.9: #for மடக்கு பயன்படுத்தி ஒற்றை இலக்க இரட்டைப் படை எண்ணை அச்சிட விளக்கும் நிரல்

```
for i in range (2,10,2):
        print (i, end=' ')
```

வெளியீடு:

```
2 4 6 8
```

பின்வரும் நிரல் **for** மடக்கின் **else** பகுதியை விளக்கும்

எடுத்துக்காட்டு 6.10 : #for மடக்கின் else பகுதியைப் பயன்படுத்தி, ஒற்றை இலக்க இரட்டைப் படை எண்ணை அச்சிட விளக்கும் நிரல்

```
for i in range(2,10,2):
          print (i,end=' ')
else:
          print ("\nEnd of the loop")
```

வெளியீடு:

```
          2 4 6 8
          End of the loop
```

குறிப்பு

பைத்தானில், மடக்கு மற்றும் பிற கட்டுப்பாட்டு கூற்றுகளில் உள் தள்ளல் மிக முக்கியமானதாகும். C, C++ போன்ற மொழிகளில் நெறிவு அடைப்புக் குறியைப்{} பயன்படுத்தி தொகுதிகளை உருவாக்குவதைப் போல, பைத்தான் உள்தள்ளல் மூலம் தொகுதிகளையும் துணை தொகுதிகளையும் உருவாக்குகிறது.

range()யைப் பயன்படுத்தி, 1 முதல் 100 வரையிலான எண்களின் கூட்டுத் தொகையைக் கண்டுபிடிப்பதை விளக்கும் மற்றொரு நிரல் கீழே கொடுக்கப்பட்டுள்ளது.

எடுத்துக்காட்டு 6.11: # 1 முதல் 100 வரையிலான எண்களின் கூட்டுத் தொகையை கணக்கிடும் நிரல்

```
n = 100
sum = 0
for counter in range(1,n+1):
          sum = sum + counter
print("Sum of 1 until %d: %d" % (n,sum))
```

மேலேயுள்ள குறியீட்டில், n-ன் தொடக்க மதிப்பு 100, sum-ன் மதிப்பு தொடரியல் மாறியின் மதிப்புடன் கூட்டி (சேர்த்து) sumல் சேமிக்கப்படுகிறது. for மடக்குச் செயலானது ஒன்றிலிருந்து தொடங்கி இறுதி வரம்பு-1 மதிப்பு வரை நிறைவேற்றப்படும் (அதாவது nன் மதிப்பு 100 ஆகும். அதனால் இந்த மடக்கு, 1 முதல் 99 மதிப்பு வரை மட்டுமே நிறைவேற்றப்படும். இறுதி வரம்பை n+1 என்று அமைப்பதனால் மடக்கின் இறுதி வரம்பு 100 என்ற எண் வரை செயல்படுத்த முடியும்.)

வெளியீடு:

```
          Sum of 1 until 100: 5050
```

குறிப்பு

சரம், பட்டியல், அகராதி முதலியனவற்றில் இருந்தும் **range()** செயற்கூறு மதிப்புகளை எடுத்துக்கொள்ளும். இவற்றைப் பற்றி அடுத்து வரும் படங்களில் படிக்கலாம்.

பின்வரும் எடுத்துக்காட்டு, **range()** செயற்கூறில் சரத்தின் பயன்பாட்டை விளக்குவதாகும்.

எடுத்துக்காட்டு 6.12: #for மடக்கில் range() செயற்கூறில் சரத்தின் பயன்பாட்டை விளக்கும் நிரல்.

```
for word in 'Computer':
    print (word,end=' ')
else:
    print ("\nEnd of the loop")
```

வெளியீடு

```
    C o m p u t e r
    End of the loop
```

6.2.3.3 பின்னலான மடக்கு அமைப்பு:

ஒரு மடக்கின் உள்ளே மற்றொரு மடக்கு இடம் பெற்றிருந்தால் அது பின்னலான மடக்கு அமைப்பாகும். **while** மடக்கின் உள்ளே மற்றொரு **while** மடக்கு, **for மடக்கின்** உள்ளே மற்றொரு **for** மடக்கு, **for** மடக்கினுள்ளே **while** மற்றும் **while** மடக்கினுள்ளே **for** மடக்கு என இது போன்ற பின்னலான மடக்குகளை உருவாக்கலாம்.

பின்வரும் எடுத்துக்காட்டானது, **for** மடக்கைப் பயன்படுத்தி பின்வரும் அமைப்பில் எண்களை அச்சிடுவதை விளக்குவதாகும்.

```
1
1    2
1    2   3
1    2   3    4
1    2   3    4     5
```

எடுத்துக்காட்டு 6.13: # பின்னலான மடக்கு –for மடக்கின் உள்ளே while மடக்கை விளக்கும் நிரல்

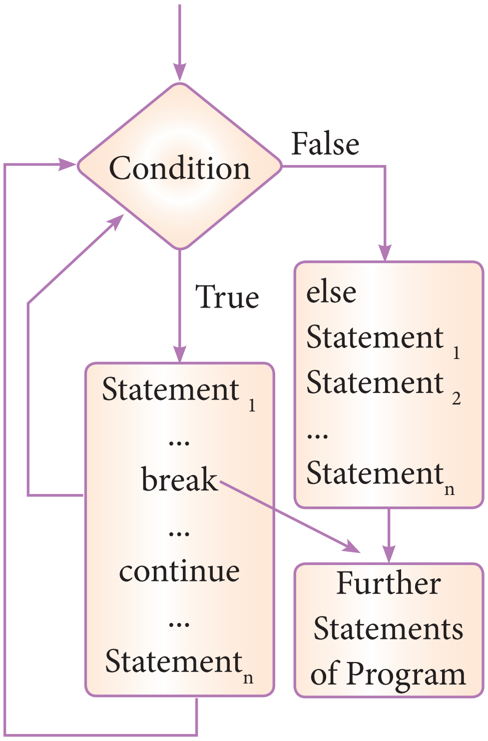
```
i=1
while (i<=6):
   for j in range (1,i):
       print (j,end='\t')
   print (end='\n')
   i +=1
```

வெளியீடு:

```
    1
    1    2
    1    2   3
    1    2    3    4
    1    2    3    4     5
```

6.2.4 பைத்தானில் JUMP கூற்றுகள்:

பைத்தானிலுள்ள **JUMP** கூற்று, கட்டுப்பாட்டை எந்தவொரு நிபந்தனையுமின்றி, நிரலின் ஒரு பகுதியில் இருந்து மற்றொரு பகுதிக்கு இடமாற்றம் செய்ய பயன்படுகிறது. பைதானில் **Jump** கூற்றை பயன்படுத்த மூன்று சிறப்புச் சொற்கள் உள்ளன. அவை: **break, continue, pass**. பின்வரும் பாய்வுப்படம் **break** மற்றும் **continue** பயன்பாட்டை விளக்கும்.



படம் 6.6 மடக்கு அமைப்பில் *break* மற்றும் *continue* கூற்றுகளின் பயன்பாடு

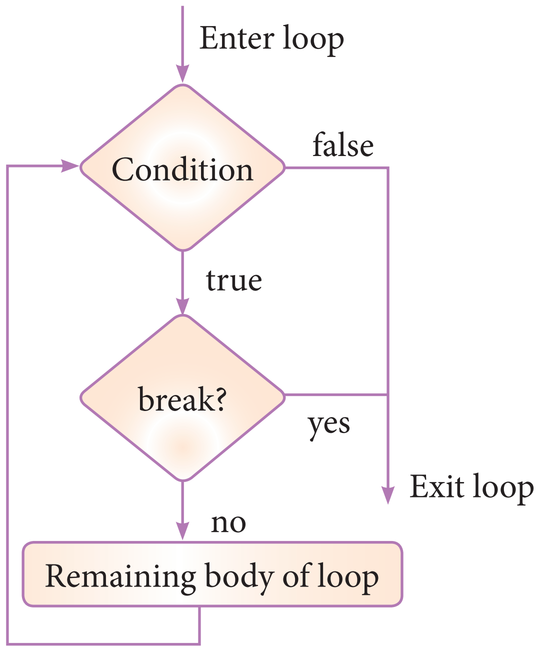
6.2.4.1 break கூற்று

break கூற்றானது அதை உள்ளடக்கிய மடக்கை விட்டு வெளியேறச் செய்கிறது. நிரலின் கட்டுப்பாடானது, மடக்கின் உடற்பகுதியைத் தொடர்ந்து உடனடியாக இருக்கும் கூற்றுக்கு பாய்கிறது.

நிபந்தனையானது தவறு என்று பரிசோதிக்கும் வரை **while** அல்லது **for** மடக்கு செயல்படுத்தப்படும். ஆனால், **break** கூற்றைப் பயன்படுத்தி கட்டுப்பாட்டை மடக்கை விட்டு நிறுத்தி வெளியேறச் செய்யமுடியும். **break** கூற்று நிறைவேற்றப்படும் போது, நிரலின் கட்டுப் பாட்டுபாய்வு, மடக்கை விட்டு வெளியேறி, மடக்கு அமைப்பிற்கு **இறுதிக்கு அடுத்து** உள்ள குறி முறை பகுதியில் இருந்து நிறைவேற்றத் தொடங்கும்.

break கூற்று பின்னலான மடக்கின் உள்ளே இருந்தால், **break** கூற்று மிகவும் உள்ளே உள்ள மடக்கை விட்டு வெளியேறும்.

> **தொடரியல்:**
>
> *break*



படம் 6.7 *break* கூற்றின்செயல்பாடு

for மடக்கு மற்றும் while மடக்கினுள் break கூற்றின் செயல்பாடு கீழே கொடுக்கப்பட்டுள்ளது.

```
for var in sequence:
if condition:
        break
        #code inside for loop
        #code outside for loop
while test expression:
#code inside while loop
if condition:
        break
        #code inside while loop
        #code outside while loop
```

எடுத்துக்காட்டு 6.14: #for மடக்கின் உள்ளே break கூற்றின் பயன்பாட்டை விளக்குகிறது

```
for word in "Jump Statement":
    if word = = "e":
          break
print (word, end='')
```

வெளியீடு:

```
Jump Stat
```

மேற்கண்ட நிரல், கொடுக்கப்பட்ட **"Jump statement"** என்ற சரத்தை மடக்குச் செயலாக மீண்டும் மீண்டும் இயக்குகிறது. கொடுக்கப்பட்ட சர வரிசையில் **'e'** என்ற எழுத்து வரும் வரை ஒவ்வொரு எழுத்தாக பரிசோதிக்கப்படும். அந்த எழுத்து வரும்போது கட்டுப்பாடானது மடக்குக்கு வெளியே செல்லும் அல்லது மடக்கை நிறுத்திவிடும். வெளியீட்டில் காட்டியுள்ளவாறு, கொடுக்கப்பட்ட சரத்தில் **'e'** என்ற எழுத்து வரும் வரை ஒவ்வொரு எழுத்தாக அச்சிடப்பட்டு மடக்கு நிறுத்தப்படும்.

இதில் முக்கியமாக கவனிக்க வேண்டியது என்னவென்றால், மடக்கானது **'break'** கூற்றுடன் முடிந்தால் **'else'** பகுதி நிறைவேற்றப்படாது. இதை விளக்குவதற்கு **else** பகுதியுடன் மேலே குறிப்பிட்ட நிரலை மேம்படுத்தி அதன் வெளியீடு என்னவென்று காண்போம்:

எடுத்துக்காட்டு 6.15: # for மடக்கின் உள்ளே break கூற்றின் பயன்பாட்டை else பகுதியுடன் விளக்கும் நிரல்

```
for word in “Jump Statement”:
    if word = = “e”:
            break
            print (word, end=”)
else:
            print (“End of the loop”)
print (“\n End of the program”)
```

வெளியீடு:

```
Jump Stat
End of the program
```

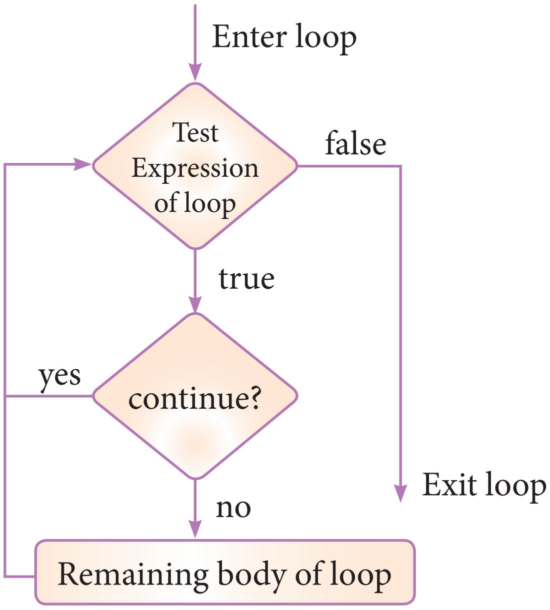
break கூற்று, மடக்கின் 'else' பகுதியைத் தவிர்த்து கட்டுப்பாட்டை மடக்கு தொகுதிக்குப் பிறகுள்ள உள்ளே அடுத்த கூற்றுக்கு கொண்டு செல்கிறது.

6.2.4.2 continue கூற்று

continue கூற்றானது break கூற்றைப் போல் இல்லாமல், மடக்கின் மீதமுள்ள குறிமுறையைத் தவிர்த்து அடுத்த மடக்கு செயலை ஆரம்பிக்கும்.

தொடரியல்:

continue



படம் 6.8 *continue* கூற்றின் செயல்பாடு

for மற்றும் while மடக்கினுள் **continue** கூற்றின் செயல்பாடானது கீழே கொடுக்கப்பட்டுள்ளது.

```
for var in sequence:
        # code inside for loop
        if condition:
                continue
        #code inside for loop
#code outside for loop
while test expression:
        #code inside while loop
        if condition:
                continue
        #code inside while loop
#code outside while loop
```

எடுத்துக்காட்டு 6.16: # for மடக்கினுள் continue கூற்றின் பயன்பாட்டை விளக்கும் நிரல்

```
for word in "Jump Statement":
    if word = = "e":
        continue
    print (word, end='')
print ("\n End of the program")
```

வெளியீடு:

```
Jump Statmnt
End of the program
```

மேலே கொடுக்கப்பட்ட நிரலும் ‘**break**’ கூற்றுக்கு எழுதப்பட்ட நிரலும் ஒன்றே, ஆனால் ‘**break**’க்கு பதிலாக இங்கு ‘**continue**’ கூற்று பயன்படுத்தப்பட்டுள்ளது. ‘**e**’ என்ற எழுத்தைத் தவிர மற்ற அனைத்து எழுத்துக்களும் அச்சிடப்படுகிறது.

6.2.4.3 pass கூற்று

pass கூற்று ஒரு **null** கூற்றாகும். நிரல் பெயர்ப்பி **pass** கூற்றை நிறைவேற்றும் பொழுது, அதே கூற்றை முழுவதுமாக புறக்கணித்துவிடும். **pass** கூற்றை இயக்கும் பொழுது எந்த செயல்பாடும் நடைபெறாது.

‘**if**’ கூற்றிலும், மடக்கினுள்ளும் எந்த கட்டளையையும் அல்லது கூற்றையும் நிறைவேற்ற வேண்டாம் என நினைக்கும் போது **pass** கூற்று இயக்கப்படுகிறது.

தொடரியல்:

pass

எடுத்துக்காட்டு 6.17:# pass கூற்றின் பயன்பாட்டை விளக்கும் நிரல்

```
a=int (input(“Enter any number :”))
if (a==0):
    pass
else:
    print (“non zero value is accepted”)
```

மேற்கண்ட நிரலை இயக்கும் போது உள்ளீட்டின் மதிப்பு 0 எனில் ஒரு செயலும் நடைபெறாது. மற்ற பிற உள்ளீட்டு மதிப்புகளுக்கு பின்வரும் வெளியீடு கிடைக்கும்:

வெளியீடு:

```
    Enter any number :3
    non zero value is accepted
```

குறிப்பு

pass கூற்று பொதுவாக இட ஒதுக்கீட்டிற்காக பயன்படுகிறது. மடக்கு அல்லது செயற்கூற்றைதற் பொழுது செயல்படுத்தாமல் பிறகு செயல்படுத்த விரும்பும் போது, வெற்று உடற்பகுதியாக உருவாக்க முடியாது. ஏனெனில், நிரல்பெயர்ப்பி பிழை செய்தியைத் தரும். இதை தவிர்க்க ‘**pass**’ கூற்று எதையும் நிறைவேற்றாத உடற்பகுதியில் அமைக்கப்படுகிறது.

எடுத்துக்காட்டு 6.18: # for மடக்கினுள் ‘pass’ கூற்றை விளக்கும் நிரல்

```
for val in “Computer”:
        pass
print (“End of the loop, loop structure will be built in future”)
```

வெளியீடு:

```
        End of the loop, loop structure will be built in future.
```

நினைவில் கொள்க:

- நிரல் என்பது வரிசையாக நிறை வேற்றப்படும் கூற்றுகளைக் கொண்டதாகும். அந்த வரிசையை மாற்றுவதற்கு கட்டுப்பாட்டுக் கூற்றுகள் பயன்படுகிறது.
- கட்டுப்பாட்டு நிரலின், ஒரு பகுதியில் இருந்து இன்னொரு பகுதிக்கு தாவுவதற்கு காரணமான நிரல் கூற்றுகள் கட்டுப்பாட்டு கட்டமைப்பு (அல்லது) கட்டுப்பாட்டு கூற்றுகள் எனப்படும்.
- மூன்று வகையான கட்டுப்பாட்டுப் பாய்வுகள் உள்ளன
    - o வரிசை முறை
    - o மாற்று அல்லது கிளை பிரிப்பு
    - o பன்முகச் செயல் அல்லது மடக்குச் செயல்
- பைத்தானில் பல்வேறு வகையான if அமைப்புகளை பயன்படுத்தி கிளைப்பிரிப்பு சாத்தியமாகிறது.
- உள் தள்ளல் பைத்தானில் மிக முக்கிய பங்கு வகிக்கிறது. நெளிவு அடைப்புக் குறியை ({}) பயன்படுத்தாமல் கூற்றுகளை ஒரு தொகுதிக்குள் குறிப்பிட உள்தள்ளல் பயன்படுகிறது.
- பைத்தானில் தவறான உள் தள்ளல்களுக்கு நிரல் பெயர்ப்பிபிழை செய்திகளைக் காண்பிக்கும்.
- print() செயற்கூறு விடுபடு வரிசையைப் பயன்படுத்தி பயனர் விரும்பும் வெளியீட்டை வடிவமைக்க ஆதரவளிக்கிறது.
- range() செயற்கூறு for மடக்கில் வரம்பிற்குட்பட்ட மதிப்புகளை வழங்குகிறது.
- பைத்தானில் break, continue மற்றும் pass ஆகிய கூற்றுகள் jump கூற்றுகளாக செயல்படுகிறது.
- பைத்தானில் pass கூற்று ஒரு null கூற்றாகும். இது பொதுவாக இட ஒதுக்கீட்டிற்காக பயன்படுகிறது.

செய்முறைப் பயிற்சி

1. கொடுக்கப்பட்ட எழுத்து ‘உயிரெழுத்தா இல்லையா’ என கண்டறியும் நிரலை எழுதுக.

2. if..else..elif கூற்றைப் பயன்படுத்தி கொடுக்கப்பட்ட மூன்று எண்களில் மிகச் சிறிய எண்ணை கண்டறியும் நிரலை எழுதுக.

3. கொடுக்கப்பட்ட எண் நேர்மறை எண்ணா அல்லது எதிர்மறை எண்ணா அல்லது பூஜ்யமா எனக் கண்டறியும் நிரலை எழுதுக.

4. கொடுக்கப்பட்ட வருடம் லீப் வருடமா இல்லையா எனக் கண்டறியும் நிரலை எழுதுக.

செய்முறைப் பயிற்சி

1. பைபோனாசி தொடரை (0 1 1 2 3 4 5……n) வரை வெளியிடுவதற்கான நிரலை எழுதுக.
2. n வரையிலான இயல் எண்களின் கூட்டுத் தொகையை வெளியிடுவதற்கான நிரலை எழுதுக.
3. கொடுக்கப்பட்ட எண் பாலின் ரோமா இல்லையா என்பதை சரிபார்க்கும் நிரலை எழுதுக.
4. பின்வரும் அமைப்பை அச்சிடுவதற்கான நிரலை எழுதுக.

```
* * * * *
* * * *
* * *
* *
*
```

SZHFZA

மதிப்பீடு

பகுதி I

I சரியான விடையைத் தேர்ந்தெடுத்து எழுதுக **(1 மதிப்பெண்)**

1. பைத்தானில் எத்தனை முக்கியமான கட்டுப்பாட்டு கட்டமைப்புகள் உள்ளன?

   அ) 3 ஆ) 4

   இ) 5 ஈ) 6

2. elif என்பதன் விரிவாக்கம்

   அ) nested if ஆ) if..else

   இ) else if ஈ) if..elif

3. பைத்தான் நிரலில் எது முக்கிய பங்கு வகிக்கிறது?

   அ) கூற்றுகள் ஆ) கட்டுப்பாடு

   இ) அமைப்பு ஈ) உள்தள்ளல்

4. எந்த கூற்று பொதுவாக இட ஒதுக்கீட்டிற்காகப் பயன்படுகிறது?

   அ) continue ஆ) break

   இ) pass ஈ) goto

5. if கூற்றின் நிபந்தனை பின்வரும் எந்த வடிவில் இருக்க வேண்டும்

   அ) கணித அல்லது ஒப்பீட்டுக் கோவைகள்

   ஆ) கணித அல்லது தருக்கக் கோவைகள்

   இ) ஒப்பீட்டு அல்லது தருக்கக் கோவைகள்

   ஈ) கணித கோவைகள்

6. எது மிகவும் சுலபமான மடக்கு எது?

அ) do..while ஆ) while

இ) for ஈ) if..elif

7. பின்வரும் குறிமுறையின் வெளியீடு என்ன?

```
i=1
while True:
        if i%3 ==0:
                break
        print(i,end='')
        i +=1
```

அ) 12 ஆ) 123

இ) 1234 ஈ) 124

8. பின்வரும் குறிமுறையின் வெளியீடு என்ன?

```
T=1
while T:
        print(True)
        break
```

அ) தவறு ஆ) சரி

இ) 0 ஈ) வெளியீடு இல்லை

9. பின்வருவனவற்றில் எது jump கூற்று கிடையாது ?

அ) for ஆ) pass

இ) continue ஈ) break

10. எந்த நிறுத்தற்குறி பின்வரும் அடிக்கோடிட்ட இடத்தில் இடம் பெற வேண்டும்?

```
if <condition>_
   statements-block 1
else:
   statements-block 2
```

அ) ; ஆ) :

இ) :: ஈ) !

பகுதி–ஆ

II பின்வரும் வினாக்களுக்கு விடையளி (2 மதிப்பெண்)

11. பைத்தானில் உள்ள கட்டுப்பாட்டு கட்டமைப்புகளைப் பட்டியலிடுக.

12. break கூற்றைப் பற்றி குறிப்பு வரைக.

13. if..else கூற்றின் பொது வடிவத்தை எழுதுக?

14. கட்டுப்பாட்டு கட்டமைப்பு என்றால் என்ன?

15. range() செயற்கூறு குறிப்பு வரைக.

பகுதி–இ

III பின்வரும் வினாக்களுக்கு விடையளி (3 மதிப்பெண்)

16. பின்வரும் வெளியீட்டைப் பெற நிரலை எழுதுக

```
A
A B
A B C
A B C D
A B C D E
```

17. if..else அமைப்பைப் பற்றி குறிப்பு வரைக.

18. if..else..elif கூற்றைப் பயன்படுத்தி கொடுக்கப்பட்ட மூன்று எண்களில் பெரிய எண்ணைக் கண்டுபிடிப்பதற்கான பொருத்தமான நிரலை எழுதுக.

19. while மடக்கின் பொதுவடிவம் யாது?

20. break மற்றும் continue கூற்றுகளின் வேறுபாடு யாது?

பகுதி–ஈ

IV பின்வரும் வினாக்களுக்கு விடையளி (5 மதிப்பெண்)

21. for மடக்கைப் பற்றி விரிவான விடையளிக்கவும்.

22. if..else..elif கூற்றை எடுத்துக்காட்டுடன் விளக்குக.

23. அனைத்து மூன்று இலக்க ஒற்றைப்படை எண்களை வெளியிடுவதற்கான நிரலை எழுதுக.

24. கொடுக்கப்பட்ட எண்ணின் பெருக்கல் வாய்ப்பாட்டை வெளியிடும் நிரலை எழுதுக.