அலகு II

பாடம் 7

பைத்தான் செயற்கூறுகள்

7KXTST

கற்றலின் நோக்கங்கள்

இந்த பாடத்தை கற்றபிறகு, மாணவர்கள்:

- செயற்கூறுகளின் கருத்து மற்றும் அதன் வகைகளைப் பற்றி அறிந்து கொள்ள முடியும்.
- பயனர் வரையறுக்கும் மற்றும் உள்ளிணைந்த செயற்கூறுகளுக்கு இடையே உள்ள வேறுபாட்டை அறிந்து கொள்ள முடியும்.
- ஒரு செயற்கூறினை எவ்வாறு அழைப்பது என்பதை தெரிந்து கொள்ள முடியும்.
- செயற்கூறின் அளபுருக்களைப் புரிந்து கொள்ள முடியும்.
- பெயரில்லாத செயற்கூறுவை அறிந்து கொள்ள முடியும்.
- கணித மற்றும் சரங்களின் செயற்கூறுகளைப் பற்றி அறிந்து கொள்ள முடியும்.

7.1 அறிமுகம்

ஒரு குறிப்பிட்ட செயலினை செய்வதற்காக வடிவமைக்கப்பட்டு பெயரிடப்பட்ட குறிமுறையின் தொகுதியே செயற்கூறு எனப்படும். செயற்கூறில் வரையறுத்த குறிப்பிட்ட செயலினை செய்ய விரும்பினால் அதற்கு பொருத்தமான செயற்கூறின் பெயரை கொண்டு அழைக்க வேண்டும். நிரலில், ஒரே செயலை பல தடவை மீண்டும் மீண்டும் செய்ய நேரிட்டால் அதற்கான குறிமுறையை மீண்டும் எழுதத் தேவையில்லை. அதற்கான செயற்கூறுவை அழைத்தால் மட்டும் போதும், அந்த அழைப்பு பைத்தானின் செயற்கூற்றின் உள்ளேயுள்ள குறிமுறையை இயக்க செய்யும். செயற்கூறுவின் பயன்பாடு ஒரு நிரலை, எழுத, படிக்க, சோதிக்க மற்றும் பிழை திருத்தும் வேலைகளை எளிதாக்குகிறது.

செயற்கூறுவின் முதன்மை நன்மைகள்

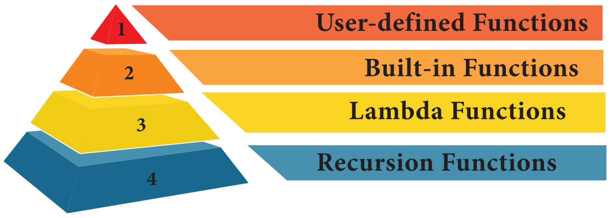
- குறிமுறையை மீண்டும் எழுதுவதை தவிர்த்து குறிமுறையின் மறு பயனாக்கத்திற்கு உதவுகிறது.
- நமது பயன்பாட்டிற்குச் சிறந்த கூறுநிலையை வழங்குகிறது.

குறிப்பு

செயற்கூறுகள் என்பது குறிப்பிட்ட செயலை செய்வதற்கான தொடர்புடைய கூற்றுகளின் தொகுதி ஆகும்.

7.1.1 செயற்கூறுகளின் வகைகள்

அடிப்படையாக செயற்கூறுகளை பின்வருமாறு வகைப்படுத்தலாம்:



படம் – 7.1 – பைத்தான் செயற்கூறுவின்வகைகள்

செயற்கூறுகள்	விளக்கம்
பயனர் வரையறுக்கும் செயற்கூறுகள்	பயனர்கள் தாங்களாகவே வரையறுக்கும் செயற்கூறுகள்.

உள்ளிணைந்த செயற்கூறுகள்	பைத்தானில் உள்ளடக்கப்பட்ட செயற்கூறுகள்
லாம்டா செயற்கூறுகள்	பெயரில்லாத செயற்கூறுகள்
தற்சுழற்சி செயற்கூறுகள்	தற்சுழற்சியாக தன்னைத் தானே அழைத்துக் கொள்ளும் செயற்கூறுகள்

அட்டவணை – *7.1* – பைத்தானின் செயற்கூறுகள் மற்றும் அவற்றின் விளக்கமும்

7.2 செயற்கூறுவை வரையறுத்தல்

ஒரு செயல்பாட்டை உருவாக்கி அதை பயன்படுத்துவதற்கு செயற்கூறினை வரையறுக்க வேண்டும். பைத்தான் மொழியில் பல உள்ளிணைந்த செயற்கூறுகள் உள்ளன. (எடுத்துக்காட்டாக, print() செயற்கூறு), இருப்பினும் பயனர் அவருக்குத் தேவையான செயற்கூறினையும் வரையறுக்க முடியும். செயற்கூறினை வரையறுக்கும் போது நினைவில் கொள்ள வேண்டியவை.

- செயற்கூறு தொகுதி def என்ற சிறப்புச் சொல்லுடன் தொடங்கி செயற்கூறுவின் பெயர் மற்றும் () அடைப்புக்குறியுடன் முடிய வேண்டும்.
- ஏதேனும் உள்ளீட்டு செயலுருப்புகள் அல்லது அளபுருக்கள் இருப்பின் அவற்றை செயற்கூற்றை வரையறுக்கும் போதே ( ) என்ற அடைப்புக்குறிக்குள் கொடுக்க வேண்டும்.
- குறிமுறை தொகுதியானது எப்பொழுதும் முக்காற்புள்ளிக்கு பிறகு உள்தள்ளி வர வேண்டும்.
- "return [கோவை]" கூற்று செயற்கூறுவை முடித்து வைக்கும். விருப்பப்பட்டால் கோவையின் மதிப்பை, அழைக்கும் கூற்றுக்கு திருப்பி அனுப்பும். செயலுருபுகள் இல்லாத **return, return None**-க்கு நிகரானது.

குறிப்பு

பைத்தான் சிறப்புச் சொற்கள் செயற்கூறுவின் பெயர்களாக பயன்படுத்தக் கூடாது.

7.2.1 பயனர் வரையறுக்கும் செயற்கூறுவின் தொடரியல்

```
def <function_name ([parameter1, parameter2…] )> :
        <Block of Statements>
        return <expression / None>
```

குறிப்பு

மேலே உள்ள பொது வடிவத்தில் சதுர அடைப்புக் குறிக்குள் [] உள்ளவை அனைத்தும் விருப்பத் தேர்வாகும்

தொகுதி:

தொகுதி என்பது ஒன்று அல்லது ஒன்றிற்கு மேற்பட்ட வரிகளையுடைய குறிமுறையைக் குழுவாக ஒன்றிணைத்து, அதனை நிறைவேற்றும் போது ஒரு பெரிய வரிசையின் கூற்றுகளாக எடுத்துக் கொள்கிறது. பைத்தானில், தொகுதியில் உள்ள கூற்றுகளை உள்தள்ளல் மூலம் எழுதப்படுகிறது. பொதுவாக, ஒரு வரி உள்தள்ளப் படும்போது (4 இடைவெளி) தொகுதியானது தொடங்கும். தொகுதியின் உள்ளே உள்ள அனைத்து கூற்றுகளையும் ஒரே மாதிரியாக உள்தள்ள வேண்டும்.

பின்னலான தொகுதி

ஒரு தொகுதி மற்றொரு தொகுதியைக் கொண்டிருந்தால் அது பின்னலான தொகுதியாகும். முதல் தொகுதியின் கூற்றுகளை ஒற்றை தத்தல் இடைவெளி அளவிற்கு உள்தள்ளினால், இரண்டாவது தொகுதியின் கூற்றுகளை இரண்டு தத்தல் இடைவெளி அளவிற்கு உள்தள்ள வேண்டும்.

செயற்கூற்றை வரையறைக்கும் எடுத்துக்காட்டு,

```
def Do_Something( ):
value =1          #Assignment கூற்று
return value      #Return கூற்று
```

இப்பொழுது செயற்கூறின் செயல்பாட்டை சோதிப்பின் மூலம் அது நிரலில் எவ்வாறு வேலை செய்கிறது என்பதை காணலாம். பின்வரும் எடுத்துக்காட்டில் ஒரு சாதாரண செயற்கூறு சரத்தை வெளியிடுவதைக் காணலாம்.

எடுத்துக்காட்டு

```
def hello():
        print ("hello - Python")
        return
```

7.2.2 பயனர் வரையறுக்கும் செயற்கூறுகளின் நன்மைகள்

1. செயற்கூறுகள் ஒரு நிரலை சிறு தொகுதியாக பிரிக்க உதவுகிறது. இது குறிமுறையை எளிதாக கையாள உதவுகிறது.
2. குறிமுறையின் மறுபயனாக்கத்திற்கு உதவுகிறது. ஒவ்வொரு முறையும் கூற்றுகளின் வரிசைகளை நிறைவேற்றும் போது, நாம் அந்த செயற்கூற்றினை அழைத்தால் போதுமானது.
3. செயற்கூற்றின் செயல்பாடுகளை மாற்றம் செய்வது எளிதாகிறது. வெவ்வேறு நிரலர்கள் வெவ்வேறு செயற்கூற்றில் வேலை செய்ய முடியும்.

7.3 செயற்கூறினை அழைத்தல்

எடுத்துக்காட்டு 7.2.1 ல் hello() செயற்கூறினை அழைக்க இந்த குறிமுறை பயன்படும். "hello()" செயற்கூறினை அழைக்கும் பொழுது நிரல் பின்வரும் சரத்தை வெளியீடுகிறது.

வெளியீடு

```
hello – Python
```

மாற்றாக, "hello()" செயற்கூறினை print() செயற்கூற்றிலிருந்து அழைப்பதை பின்வரும் எடுத்துக்காட்டில் காணலாம்.

எடுத்துக்காட்டு:

```
def hello():
        print ("hello - Python")
        return
print (hello())
```

return கூற்றில் எந்த அளபுருவும் இல்லை எனில் வெளியீட்டின் இறுதிக் கூற்றாக, "None" வெளியிடப்படும்.

மேலே உள்ள செயற்கூறு பின்வரும் வெளியீட்டைக் கொடுக்கும்.

வெளியீடு

```
hello – Python
None
```

7.4 செயற்கூறினுள் அளபுருக்களை அனுப்புதல்

அளபுருக்கள் அல்லது செயலுருப்புகளை செயற்கூறினுக்கு அனுப்பலாம்.

def function_name (parameter(s) separated by comma):

அளபுருக்களின் பயன்பாட்டினை செயற்கூறு வரையறையில் பார்க்கலாம். அடைப்புக் குறிக்குள் கொடுக்கப்படும் அளபுருக்கள் அந்த செயற்கூறினுள்ளேயே பயன்படுத்தப்படும். அனைத்து விதமான தரவுகளையும் செயற்கூறினுள் அனுப்ப முடியும். அளபுருக்களை செயற்கூறினுக்கு அனுப்பும் செயற்கூறு வரையறை எடுத்துக்காட்டு நிரலில் கொடுக்கப்பட்டுள்ளது.

எடுத்துக்காட்டு :

```
# assume w = 3 and h = 5
        def area(w,h):
                return w * h
        print (area (3,5))
```

மேலே உள்ள குறிமுறையில் அகலம் மற்றும் உயரத்தின் மதிப்புகள் முறையே w மற்றும் h அளபுருக்களில் இருத்தப்படுகிறது. “area” என்ற செயற்கூறினை உருவாக்குவதற்கு இந்த அளபுருக்கள் பயன்படுகின்றன. மேலே குறிப்பிட்ட செயற்கூறினை அழைக்கும் போது வெளியீடாக அகலம் மற்றும் உயரத்தின் பெருக்கல் மதிப்பினை திருப்பி அனுப்புகிறது. w மற்றும் h அளபுருக்களுக்கு முறையே 3 மற்றும் 5 மதிப்புகளை அனுப்பினால் வெளியீடாக 15 என்ற மதிப்பதை திருப்பி அனுப்பும்.

உங்களுக்குத் தெரியுமா?

நாம் அடிக்கடி அளபுருக்கள் மற்றும் செயலுருபுக்கள் என்ற வார்த்தையை மாற்றி மாற்றி உபயோகப்படுத்துகிறோம். எனினும், இரண்டிற்கும் இடையே ஒரு சிறிய வித்தியாசம் உள்ளது. அளபுருக்கள் என்பவை செயற்கூறு வரையறையில் பயன்படுத்தப்படும் மாறிகள். ஆனால் செயலுருப்புகள் என்பது செயற்கூறின் அளபுருக்களுக்கு அனுப்பப்படும் மதிப்புகள் ஆகும்.

7.5 செயற்கூறு செயலுருபுகள் (Function Arguments)

செயலுருபுகள் செயற்கூறினை அழைக்க பயன்படுகிறது. செயலுருபுகள் நான்கு வகைப்படும் தேவைப்படும் செயலுருபுகள் (Required arguments), சிறப்பு சொல் செயலுருபுகள் (Keyword arguments), தானமைவு செயலுருபுகள் (Default arguments), மாறும் நீள செயலுருபுகள் (variable length arguments).

செயற்கூறு செயலுருபுக்கள்
(Function Arguments)

1. தேவைப்படும் செயலுருபுக்கள் (Required arguments)
2. சிறப்புச் சொல் செயலுருபுக்கள் (Keyword arguments)
3. தானமைவு செயலுருபுக்கள் (Default arguments)
4. மாறும்-நீள செயலுருபுக்கள் (Variable-length arguments)

7.5.1 தேவைப்படும் செயலுருபுகள்

செயலுருபுகளை சரியான இடவரிசையில் செயற்கூற்றுக்கு அனுப்புவதே **தேவைப்படும் செயலுருபுக்கள்** எனப்படும். இங்கு, செயற்கூறு அழைப்புக் கூற்றில் உள்ள செயலுருபுகளின் எண்ணிக்கை, செயற்கூறு வரையறையோடு சரியாக பொருந்த வேண்டும். தொடரியல் பிழையை தவிர்த்து, தேவையான வெளியீட்டை பெறுவதற்கு குறைந்தது ஒரு செயலுருபாவது தேவை.

எடுத்துக்காட்டு

```
def printstring(str):
        print ("Example - Required  arguments ")
        print (str)
        return
# Now you can call printstring() function
printstring()
```

மேலே உள்ள குறிமுறை இயக்கப்படும் போது இது பின்வரும் பிழையை கொடுக்கிறது.

```
Traceback (most recent call last):
File "Req-arg.py", line 10, in <module>
printstring()
     TypeError: printstring() missing 1 required positional argument: 'str'
```

மேலே உள்ள குறிமுறையில் print string( )க்கு பதிலாக printstrin("Welcome") பயன்படுத்தினால் இதன் வெளியீடு,

வெளியீடு

```
        Example - Required arguments
        Welcome
```

7.5.2 சிறப்புச் சொல் செயலுருபுகள்

அளபுருக்களின் பெயரை அடையாளம் கண்ட பின்பு சிறப்புச் சொல் செயலுருபானது செயற்கூறினை அழைக்கிறது. சிறப்புச் சொல் செயலுருபின் மதிப்பு அளபுரு பெயருடன் பொருந்த வேண்டும். எனவே, முறையற்ற வரிசையிலும் செயலுருபுகளைக் கொடுக்கலாம்.

எடுத்துக்காட்டு:

```
def printdata (name):
        print ("Example-1 Keyword arguments")
        print ("Name :",name)
        return
# Now you can call printdata() function
printdata(name = "Gshan")
```

மேற்கண்ட குறிமுறை இயக்கப்படும் போது பின்வரும் வெளியீடு கிடைக்கும்.

வெளியீடு

```
Example-1 Keyword arguments
Name :Gshan
```

எடுத்துக்காட்டு:

```
def printdata (name):
        print ("Example-2 Keyword arguments")
        print ("Name :", name)
        return
# Now you can call printdata() function
printdata (name1 = "Gshan")
```

மேற்கண்ட குறிமுறை இயக்கப்படும் போது பின்வரும் வெளியீடு கிடைக்கும்.

```
TypeError: printdata() got an unexpected keyword argument 'name1'
```

எடுத்துக்காட்டு:

```
def printdata (name, age):
        print ("Example-3 Keyword arguments")
        print ("Name :",name)
        print ("Age :",age)
        return
# Now you can call printdata()  function
printdata (age=25, name="Gshan")
```

மேற்கண்ட குறிமுறை இயக்கப்படும் போது பின்வரும் வெளியீடு கிடைக்கும்.

வெளியீடு

```
Example-3 Keyword arguments
Name : Gshan
Age : 25
```

குறிப்பு

மேலே உள்ள நிரலில் அளபுருக்களின் வரிசைமுறை மாறி உள்ளது.

7.5.3 தானமைவு செயலுருபுகள்

பைத்தானில், செயற்கூறை அழைக்கும் போது எந்த மதிப்பும் கொடுக்கப்படவில்லை எனில், செயலுருபானது தானாகவே மதிப்பை எடுத்துக் கொள்ளும். இதுவே **தானமைவு செயலுருபு** ஆகும்.

பின்வரும் எடுத்துக்காட்டில் செயலுருபுவை அனுப்பாதபோது தானமைவு Salary-யை அச்சிடுவதைக் காணலாம்.

எடுத்துக்காட்டு:

```
def printinfo( name, salary = 3500):
        print (“Name: “, name)
        print (“Salary: “, salary)
        return
printinfo(“Mani”)
```

மேற்கண்ட குறிமுறை இயக்கப்படும் போது பின்வரும் வெளியீடு கிடைக்கும்.

வெளியீடு

```
Name: Mani
Salary: 3500
```

மேற்கண்ட குறிமுறையை printinfo(“Ram”, 2000)என மாற்றும் போது பின்வரும் வெளியீடு கிடைக்கும்.

வெளியீடு

```
Name: Ram
Salary: 2000
```

மேலே உள்ள குறிமுறையில் 2000 என்ற மதிப்பை salary என்ற செயலுருபுக்கு அனப்பினால் ஏற்கனவே உள்ள தானமைவு மதிப்பு நிராகரிக்கப்படும்.

7.5.4 மாறும் நீள செயலுருபுகள்

சில சமயங்களில், ஏற்கனவே குறிப்பிடப்பட்ட செயலுருபுகளை விட அதிகமான செயலுருபுகளை அனுப்ப வேண்டிய தேவை இருக்கும். மீண்டும் செயற்கூறினை மறுவரையறை செய்வது கடினமான செயல்

அதற்கு பதிலாக மாறும் நீள செயலுருபுகளை பயன்படுத்தலாம். செயற்கூறு வரையறையில் இது குறிப்பிடப்படுவதில்லை. மாறாக இந்த செயலுருபுகளை குறிப்பிடுவதற்கு * குறியீடு பயன்படுகிறது. மூன்றுக்கு அதிகமான செயலுருபுகளை sum( ) செயற்கூறுவிற்கு அனுப்பும் பொழுது என்ன நிகழும் என பார்ப்போம்.

எடுத்துக்காட்டு:

```
def sum(x,y,z):
        print("sum of three nos :",x+y+z)
sum(5,10,15,20,25)
```

மேலே உள்ள குறிமுறையை இயக்கும் போது பின்வரும் வெளியீடு கிடைக்கும்.

```
TypeError: sum() takes 3 positional arguments but 5 were given
```

7.5.4.1 மாறும் நீள செயலுருபின் பொதுவடிவம்

```
def function_name(*args):
        function_body
        return_statement
```

எடுத்துக்காட்டு:

```
def printnos (*nos):
        for n in nos:
        print(n)
        return
# now invoking the printnos() function
print ('Printing two values')
printnos (1,2)
print ('Printing three values')
printnos (10,20,30)
```

வெளியீடு

```
Printing two values
        1
        2
Printing three values
        10
        20
        30
```

மதிப்பீடு செய்க

மேலே உள்ள நிரலில், printnos என்ற செயற்கூறின் பெயரை printnames என்ற அனைத்து இடங்களிலும் மாற்றி ஏற்ற தரவுகளை கொடுக்கவும். எடுத்துக்காட்டு, printnos(10,20,30) என்பதை printnames('mala', 'kala', 'bala') என மாற்றி கொடுத்து வெளியீட்டைக் காண். மாறி நீள செயலுருபுகளில், செயலுருபுகளை இரண்டு வழிகளில் அனுப்பலாம்.

In Variable Length arguments we can pass the arguments using two methods.

1. சிறப்புச் சொற்களற்ற மாறி செயலுருபுகள்
2. சிறப்புச் சொல் மாறி செயலுருபுகள்

சிறப்புச் சொற்களற்ற மாறி செயலுருபுகளை tuples என்று அழைக்கிறோம். இதைப்பற்றி விரிவாக பின்வரும் பாடங்களில் காணலாம்.

குறிப்பு

சிறப்புச் சொல் மாறி செயலுருபு இந்த புத்தகத்திற்கு அப்பாற்பட்டது.

உங்களுக்குத் தெரியுமா?

பைத்தானில் print() செயற்கூறு என்பது மாறும் நீள செயலுருபுக்கு ஒரு எடுத்துக்காட்டு ஆகும்.

7.6 பெயரில்லாத செயற்கூறுகள் (Anonymous Function)

பெயரில்லாத செயற்கூறுகள் என்றால் என்ன?

பைத்தானில், பெயரில்லாமல் வரையறுக்கப்படும் செயற்கூறுவுக்கு **பெயரில்லாத செயற்கூறு** என்று பெயர். மற்ற சாதாரண செயற்கூறுகள் **def** என்ற சிறப்பு சொல்லுடன் வரையறுக்கப்படுகிறது. பைத்தானில் பெயரில்லாத செயற்கூறுகள் **லாம்டா** சிறப்புச் சொல்லுடன் வரையறுக்கப்படுகிறது. எனவே, பெயரில்லா செயற்கூறுகளை லாம்டா செயற்கூறுகள் என்றும் அழைக்கலாம்.

லாம்டா அல்லது பெயரில்லா செயற்கூறின் பயன்கள்

- லாம்டா செயற்கூறு பெரும்பாலும் சிறிய மற்றும் ஒரு முறை பெயரில்லாத செயற்கூறை உருவாக்க பயன்படுகிறது.
- filter( ), map() மற்றும் reduce() போன்ற செயற்கூறுகளுடன் சேர்த்து லாம்டா செயற்கூறுகளை பயன்படுத்தலாம்.

குறிப்பு

filter(), map() மற்றும் reduce() செயற்கூறுகள் இந்த புத்தகத்திற்கு அப்பாற்பட்டது.

உங்களுக்குத் தெரியுமா?

லாம்டா செயற்கூறு எண்ணற்ற செயலுருபுகளை எடுத்துக் கொண்டு திருப்பி அனுப்பும் ஒரே ஒரு மதிப்பை கோவை வடிவில் கொடுக்கும். லாம்டா செயற்கூற்றால் முழுதளாவிய மாறிகள் மற்றும் அளபுரு பட்டியலில் உள்ள மாறிகளை மட்டுமே அணுக முடியும்.

7.6.1 பெயரில்லா செயற்கூறின் பொது வடிவம்

lambda [argument(s)] :expression

எடுத்துக்காட்டு:

```
sum = lambda arg1, arg2: arg1 + arg2
print ('The Sum is :', sum(30,40))
print ('The Sum is :', sum(-30,40))
```

வெளியீடு

```
        The Sum is : 70
        The Sum is : 10
```

மேற்கண்ட லாம்டா செயற்கூறு arg 1 செயலுருபுவை arg 2 செயலுருபுடன் கூட்டி விடையை sum மாறியில் சேமிக்கும். **print ( )** செயற்கூறை பயன்படுத்தி வெளியீட்டை வெளியிடும்.

7.7 return கூற்று

- return கூற்று செயற்கூறினை முடித்து வைத்து அழைப்புக் கூற்றுக்கு மதிப்பை திருப்பி அனுப்பும். பொதுவாக செயற்கூறின் நோக்கம் உள்ளீடைப் பெற்று ஏதேனும் ஒரு மதிப்பை திருப்பி அனுப்புவதாகும்.
- ஒரு செயற்கூறு மதிப்பை அழைப்புக்கூற்றுக்கு திருப்பி அனுப்ப தயாராக இருக்கும் போது return கூற்று பயன்படுத்தப்படுகிறது. எனவே, இயக்க நேரத்தில் பல return கூற்றுகள் இருந்தாலும் ஒரே ஒரு return கூற்று மட்டுமே இயக்கப்படும்.
- செயற்கூறில் ஒன்றுக்கு மேற்பட்ட return கூற்றுகள் இருந்தாலும் ஒரே ஒரு return கூற்று மட்டுமே இயக்க நேரத்தில் இயக்கப்படும்.

7.7.1 return கூற்றின் பொது வடிவம்

return [கோவைகளின் பட்டியல்]

இந்த கூற்றில் உள்ள கோவைகள் மதிப்பீடு செய்யப்பட்டு, மதிப்பைத் திருப்பி அனுப்புகிறது. செயற்கூறினுள் உள்ள கூற்றில் கோவைகள் இல்லாமலோ அல்லது return கூற்று இல்லாமலோ இருந்தால் செயற்கூறு “None” பொருளைத் தருபிப் அனுப்பும்.

எடுத்துக்காட்டு :

```
# return statment
def usr_abs (n):
        if n>=0:
                return n
        else:
                return –n
# Now invoking the function
x=int (input("Enter a number :")
print (usr_abs (x))
```

வெளியீடு 1:

```
        Enter a number : 25
                25
```

வெளியீடு 2:

```
        Enter a number : -25
                25
```

7.8 மாறிகளின் வரையெல்லை

இது நிரலின் அணுகக்கூடிய பகுதியைக் குறிப்பதாகும். அதாவது, எந்த பகுதியில் மாறியைப் பயன்படுத்துகிறோமோ அதைக் குறிக்கிறது. வரையெல்லையானது நடப்பு மாறித் தொகுதிகள் மற்றும் அதன் மதிப்புகளைக் கொண்டிருக்கும். இரண்டு வகையான வரையெல்லைகளைப் பார்க்கலாம். உள்ளமை வரையெல்லை மற்றும் குளோபல் வரையெல்லை.

7.8.1 உள்ளமை வரையெல்லை (Local Scope)

ஒரு செயற்கூறுவின் உடற்பகுதியின் உள்ளே அல்லது உள்ளமை வரையெல்லையில் மாறியை அறிவிப்பது உள்ளமை மாறி எனப்படும்.

உள்ளமை மாறியின் விதிமுறைகள்

- உள்ளமை மாறியின் வரையெல்லை அது வரையறுக்கப்பட்டுள்ள தொகுதிக்குள் மட்டுமே பயன்படுத்த முடியும்.
- செயற்கூறினுள் மாறி உருவாக்கப்படும் போது அது உள்ளமைவாக அமையும்.
- செயற்கூறு இயக்கப்படும் போது மட்டுமே உள்ளமை மாறிகள் உருவாக்கப்படும்.

எடுத்துக்காட்டு : உள்ளமை மாறியை உருவாக்குதல்

```
def loc():
        y=0 # local scope
        print(y)
loc()
```

வெளியீடு

```
        0
```

எடுத்துக்காட்டு : வரையெல்லைக்கு வெளியே உள்ளமை மாறிகளை அணுகுதல்

```
def loc():
        y = "local"
loc()
print(y)
```

மேலே உள்ள குறிமுறையை இயக்கும் போது வெளியீட்டில் பின்வரும் பிழை தோன்றும்.

குளோபல் வரையெல்லையில் உள்ளமை மாறியான ‘ y’ யை அணுக முற்பட்டதால் மேற்கண்ட பிழை தோன்றுகிறது.

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

7.8.2 குளோபல் வரையெல்லை (Global Scope)

குளோபல் வரையெல்லை உடைய மாறியை நிரலில் எங்கு வேண்டுமானாலும் அணுக முடியும். எந்த ஒரு செயற்கூறு வரையெல்லைக்கு வெளியேயும் மாறியை வரையறுத்து உருவாக்க முடியும்.

குளோபல் வரையெல்லை சிறப்புச் சொல்லின் விதிமுறைகள்

- செயற்கூறுக்கு வெளியே மாறியை அறிவிக்கும் போது அது தானமைவாக குளோபல் ஆகும். ‘global’ என்ற சிறப்புச் சொல்லை பயன்படுத்த வேண்டியதில்லை.
- செயற்கூறினுள் முழுதளாவிய மாறியை படிக்க மற்றும் எழுத ‘global’ சிறப்புச்சொல் பயன்படுத்த வேண்டும்.
- செயற்கூறுவிற்கு வெளியே ‘global’ என்ற சிறப்புச் சொல் எந்த விளைவையும் ஏற்படுத்தாது.

'Global' சிறப்புச் சொல்லின் பயன்பாடு

எடுத்துக்காட்டு : செயற்கூறுவின் உள்ளிருந்து குளோபல் மாறியை அணுகுதல்

```
c = 1            # global variable
def add():
        print(c)
add()
```

வெளியீடு

```
        1
```

எடுத்துக்காட்டு : செயற்கூறுவின் உள்ளிருந்து முழுதளாவிய மாறியை மாற்றுதல்

மேற்கண்ட நிரலை இயக்கும் போது வெளியீடு பின்வரும் பிழையை காண்பிக்கும்.

```
c = 1           # global variable
def add():
        c = c + 2 # increment c by 2
        print(c)
add()
```

வெளியீடு

```
     Unbound Local Error: local variable 'c' referenced before assignment
```

குறிப்பு

'global' என்ற சிறப்பு சொல்லைப் பயன்படுத்தாமல் குளோபல் மாறியை செயற்கூறின் உள்ளே மாற்றம் செய்ய முடியாது. ஆனால் அணுக மட்டுமே முடியும்.

எடுத்துக்காட்டு : செயற்கூறின் உள்ளிருந்து 'global' சிறப்புச் சொல்லைப் பயன்படுத்தி குளோபல் மாறியை மாற்றுதல்

```
x = 0                          # global variable
def add():
        global x
        x = x + 5              # increment by 2
        print ("Inside add() function  x value is :", x)
add()
print ("In main x value is :", x)
```

மேற்கண்ட நிரலை இயக்கும் போது பின்வரும் வெளியீடு தோன்றும்.

வெளியீடு

```
        Inside add() function  x value is : 5
        In main x value is : 5
```

மேலே உள்ள நிரலில், x என்பது குளோபல் மாறி ஆகும். add() செயற்கூறினுள், 'global' என்ற சிறப்புச் சொல் x மாறிக்கு பயன்படுத்தப்பட்டு அதனுடன் 5 என்ற மதிப்பு கூட்டப்படுகிறது. இப்பொழுது நாம் மாற்றம் செய்யப்பட்ட குளோபல் மாறி xயை செயற்கூறின் வெளியே காணலாம். அதாவது, x ன் மதிப்பு 5 ஆகும்.

7.8.3 குளோபல் மற்றும் உள்ளமை மாறிகள்

இங்கு, நாம் ஒரே குறிமுறையில் குளோபல் மற்றும் உள்ளமை மாறிகளை எவ்வாறு பயன்படுத்துவது என்பதைக் காணலாம்.

எடுத்துக்காட்டு : ஒரே குறிமுறையில் குளோபல் மற்றும் உள்ளமை மாறிகளின் பயன்பாடு

```
x=8                    # x is a global variable
def loc():
        global x
        y = "local"
        x = x * 2
        print(x)
        print(y)
loc()
```

நிரலை இயக்கும் போது வெளியீடு பின்வருமாறு தோன்றும்

வெளியீடு

```
        16
        local
```

மேற்கண்ட நிரலில் x என்பது குளோபல் மாறியாகவும், y என்பது உள்ளமை மாறியாகவும் loc() செயற்கூறில அறிவிக்கப்பட்டுள்ளது. Loc() செயற்கூறினை அழைத்த பிறகு, xன் மதிப்பு 16 என மாறிவிடும். ஏனெனில் x=x*2 என பயன்படுத்தி உள்ளோம். அதன்பிறகு, உள்ளமை மாறி yயின் மதிப்பு அச்சிடப்பட்டுள்ளது.

எடுத்துக்காட்டு : ஒரே பெயரைக் கொண்ட குளோபல் மற்றும் உள்ளமை மாறிகள்

```
x = 5
def loc():
        x = 10
        print ("local x:", x)
loc()
print ("global x:", x)
```

வெளியீடு

```
        local x: 10
        global x: 5
```

மேற்கண்ட நிரலில், ஒரே பெயரைக் கொண்ட x மாறி குளோபல் மற்றும் உள்ளமை மாறியாக பயன்படுத்தப்பட்டுள்ளது. இரண்டு வரையெல்லையிலும் மாறியை அறிவித்துள்ளதால், அச்சிடும் போது வெவ்வேறான விடைக் கிடைக்கிறது. அதாவது, loc() செயற்கூறின் உள்ளே உள்ள x மாறி உள்ளமை வரையெல்லையாகவும் செயற்கூறின் வெளியே உள்ள *x* மாறி குளோபல் வரையெல்லையாகவும் இருக்கும்.

வெளியீடு :- local *x*: 10, is called local scope of variable.

வெளியீடு:- global *x*: 5, is called global scope of variable.

7.9 உள்ளிணைந்த நூலகத்தை பயன்படுத்தும் செயற்கூறுகள்

7.9.1 உள்ளிணைந்த மற்றும் கணித செயற்கூறுகள்

செயற்கூறு	விளக்கம்	பொதுவடிவம்	எடுத்துக்காட்டு
abs ( )	எண்ணின் முழு எண்ணை திருப்பி அனுப்பும். செயலுருபானவது முழு எண்ணாகவோ அல்லது தசம எண்ணாகவோ இருக்கலாம்.	abs (x)	x=20 y=-23.2 print('x = ', abs(x)) print('y = ', abs(y)) **வெளியீடு** x = 20 y = 23.2
ord ( )	கொடுக்கப்பட்ட யுனிக்கோடு எழுத்திற்கு ASCII மதிப்பை திருப்பி அனுப்பும். இது chr() செயற்கூறின் தலைகீழாகும்.	ord (c)	c= 'a' d= 'A' print ('c = ',ord (c)) print ('A = ',ord (d)) **வெளியீடு** c = 97 A = 65
chr ( )	கொடுக்கப்பட்ட ASCII மதிப்பிற்கு யுனிக்கோடு எழுத்தை திருப்பி அனுப்பும். இது ord() செயற்கூறின் தலைகீழாகும்.	chr (i)	c=65 d=43 print (chr (c)) prin t(chr (d)) **வெளியீடு** A +
bin ( )	கொடுக்கப்பட்ட முழு எண்ணிற்கு நிகரான இரும எண்ணை "0b" யை முன்னொட்டமாக கொண்டு திருப்பி அனுப்பும். **குறிப்பு:** இதற்கு பதிலாக format () செயற்கூறுவை பயன்படுத்தலாம்.	bin (i)	x=15 y=101 print ('15 in binary : ',bin (x)) print ('101 in binary : ',bin (y)) **வெளியீடு** 15 in binary : 0b1111 101 in binary : 0b1100101

type ( )	கொடுக்கப்பட்ட பொருளின் தரவின வகையைத் திருப்பி அனுப்பும். **குறிப்பு:** ஒரு அளபுரு பொருளுடன் இது பயன்படுத்தப்படுகிறது.	type (object)	x= 15.2 y= 'a' s= True print (type (x)) print (type (y)) print (type (s)) **வெளியீடு** <class 'float'> <class 'str'> <class 'bool'>
id ( )	கொடுக்கப்பட்ட பொருளின் நினைவக முகவரியைத் திருப்பி அனுப்பும். **குறிப்பு:** x, y நினைவக முகவரி ஒவ்வொரு கணிப்பொறிக்கும் வேறுபடும்.	id (object)	x=15 y='a' print ('address of x is :',id (x)) print ('address of y is :',id (y)) **வெளியீடு** address of x is : 1357486752 address of y is : 13480736
min ( )	கொடுக்கப்பட்ட பட்டியலில் இருந்து மிகச்சிறிய மதிப்பைத் திருப்பி அனுப்பும்	min (list)	MyList = [21,76,98,23] print ('Minimum of MyList :', min(MyList)) **வெளியீடு** Minimum of MyList : 21
max ( )	கொடுக்கப்பட்ட பட்டியலில் இருந்து மிகப்பெரிய மதிப்பைத் திருப்பி அனுப்பும்	max (list)	MyList = [21,76,98,23] print ('Maximum of MyList :', max(MyList)) **வெளியீடு** Maximum of MyList : 98
sum ( )	கொடுக்கப்பட்ட பட்டியலில் உள்ள மதிப்புகளின் கூட்டுத் தொகையை திருப்பி அனுப்பும்	sum (list)	MyList = [21,76,98,23] print ('Sum of MyList :', sum(MyList)) **வெளியீடு** Sum of MyList : 218

format ( )	கொடுக்கப்பட்ட எண்ணை வேறு எண்முறைக்கு மாற்றி திருப்பி அனுப்புகிறது. 1. இருமநிலை வடிவம். வெளியீட்டை இருமநிலை வடிவில் கொடுக்கும் 2. 2. எண்ணிலை வடிவம். வெளியீட்டை எண்ணியை வடிவில் கொடுக்கும் 8. 3. நிலையான புள்ளி வடிவம். வெளியீட்டை நிலையான புள்ளி எண் வடிவில் கொடுக்கும் தானமைவு தசம துல்லியம் 6 ஆகும்.	format (value [, format_spec])	x= 14 y= 25 print ('x value in binary :',format(x,'b')) print ('y value in octal :',format(y,'o')) print('y value in Fixed-point no ',format(y,'f')) **வெளியீடு** x value in binary : 1110 y value in octal : 31 y value in Fixed-point no : 25.000000
round ( )	கொடுக்கப்பட்ட எண்ணிற்கு அருகே உள்ள முழு எண்ணாக மாற்றி திருப்பி அனுப்பும். 1. முதல் செயலுருபு கொடுக்கப்பட்ட எண்ணைக் குறிக்கும் 2. இரண்டாவது செயலுருபு முழு எண்ணிற்கு பிறகு தசம புள்ளிகளின் எண்ணிக்கையைக் குறிக்கும்.	round (number [,ndigits])	x= 17.9 y= 22.2 z= -18.3 print ('x value is rounded to', round (x)) print ('y value is rounded to', round (y)) print ('z value is rounded to', round (z)) **வெளியீடு:1** x value is rounded to 18 y value is rounded to 22 z value is rounded to -18 n1=17.89 print (round (n1,0)) print (round (n1,1)) print (round (n1,2)) **வெளியீடு:2** 18.0 17.9 17.89

pow ( )	கொடுக்கப்பட்ட எண்ணின் ab அடுக்கு பெருக்கத்தை திருப்பி அனுப்பும். (a**b) ன் அடுக்கு b.	pow (a,b)	a= 5 b= 2 c= 3.0 print (pow (a,b)) print (pow (a,c)) print (pow (a+b,3)) **வெளியீடு:** 25 125.0 343

கணித செயற்கூறுகள்

குறிப்பு

அனைத்து கணித செயற்கூறுகளுக்கும் தேவையான முக்கியமான math கூறுகள்

செயற்கூறு	விளக்கம்	பொது வடிவம்	எடுத்துக்காட்டு
floor ( )	x-யை விடக்குறைவான அல்லது x-க்கு நிகரான பெரிய முழு எண்ணைத் திருப்பி அனுப்பும்.	math.floor (x)	import math x=26.7 y=-26.7 z=-23.2 print (math.floor (x)) print (math.floor (y)) print (math.floor (z)) **வெளியீடு:** 26 -27 -24
ceil ( )	x-யை விட பெரிய அல்லது xக்கு நிகரான சிறிய முழு எண்ணைத் திருப்பி அனுப்பும்.	math.ceil (x)	import math x= 26.7 y= -26.7 z= -23.2 print (math.ceil (x)) print (math.ceil (y)) print (math.ceil (z)) **வெளியீடு:** 27 -26 -23

sqrt ( )	*x*- ன் வர்க்கமூலத்தை திருப்பி அனுப்பும். குறிப்பு: *x*-ன் மதிப்பு பூஜ்ஜியத்தை விட பெரிய மதிப்பாக இருத்தல் வேண்டும்.	sqrt (x)	import math a= 30 b= 49 c= 25.5 print (math.sqrt (a)) print (math.sqrt (b)) print (math.sqrt (c)) **வெளியீடு:** 5.477225575051661 7.0 5.049752469181039

7.9.2 செயற்கூறில் தொகுப்பு (Composition)

செயற்கூறுகளின் தொகுப்பு என்றால் என்ன?

செயற்கூறுகளின் தொகுப்பு என்றால் என்ன? செயற்கூறு திருப்பி அனுப்பும் மற்றொரு செயற்கூறிற்கு செயலுருபாக, பின்னலான அமைப்பில் பயன்படுத்தினால் அதற்கு **தொகுப்பு (composition)** என்று பெயர். எடுத்துக்காட்டாக, பயனரிடமிருந்து எண் மதிப்பை அல்லது கோவையை உள்ளீடாகப் பெற விரும்பினால், input() செயற்கூறு மூலம் பயனரிடமிருந்து சரத்தை உள்ளீடாகப் பெற்று eval() செயற்கூறு மூலம் அதன் மதிப்பை மதிப்பீடு செய்ய வேண்டும்.

எடுத்துக்காட்டு :

```
# இந்த நிரல் தொகுப்பை விளக்குகிறது
          >>> n1 = eval (input ("Enter a number: "))
Enter a number: 234
          >>> n1
          234
>>> n2 = eval (input ("Enter an arithmetic expression: "))
          Enter an arithmetic expression: 12.0+13.0 * 2
>>> n2
          38.0
```

7.10 பைத்தான் –தற்சுழற்சி செயற்கூறுகள்

ஒரு செயற்கூறு தன்னைத் தானே அழைத்தால் அதை தற்சுழற்சி என்றழைக்கப்படும். மடக்கினை போன்ற தற்சுழற்சியும் செயல்படும் ஆனால் சில சமயங்களில் மடக்கினை பயன்படுத்துவதற்கு பதில் தற்சுழற்சியை பயன்படுத்தலாம். எந்தவொரு மடக்கினையும் தற்சுழற்சியாக மாற்றமுடியும்.

தற்சுழற்சி செயற்கூறு தன்னைத்தானே அழைக்கும். ஒரு நிபந்தனையில் நிறுத்தப்படாவிட்டால், ஒரு செயல்முறை காலவரையின்றி செயல்படும். இந்த செயற்பாட்டை காலவரையின்றி செயல்படும் தற்சுழற்சி எனப்படும். தற்சுழற்சி செயற்கூறு கொடுக்கப்படும் நிபந்தனையை அடிப்படை நிபந்தனை எனப்படும். ஒவ்வொரு தற்சுழற்சி செயற்கூற்றிக்கும் அடிப்படை நிபந்தனை கொடுக்கப்பட வேண்டும் இல்லையென்றால் மடக்கு காலவரையின்றி இயங்கும்.

தற்சுழற்சி செயற்கூறு எவ்வாறு செயல்படும்

1. தற்சுழற்சி செயற்கூறு வெளிபுர குறிமுறையிலிருந்து அழைக்கப்படும்.

2. அடிப்படை நிபந்தனை நிறைவேற்றப்பட்டால் நிரலானது ஏற்ற வெளியீடு கொடுத்து வெளியேறும்.

3. இல்லையெனில், செயற்கூறானது தேவையான செயற்பாட்டை இயக்கும் மேலும் தற்சுழற்சி முறையில் தன்னைத் தானே அழைத்துக் கொள்ளும்.

தற்சுழற்சி பயன்படுத்தி காரணிபடுத்துதலுக்கு எடுத்துக்காட்டு.

எடுத்துக்காட்டு :

```
def fact(n):
        if n == 0:
                return 1
        else:
                return n * fact (n-1)
print (fact (0))
print (fact (5))
```

வெளியீடு

```
        1
        120
```

உங்களுக்குத் தெரியுமா?

print (fact (2000)) கூற்று Runtime பிழைச் செய்தியைக் கொடுக்கும். அதிகபட்ச மறு நிகழ்வின் சுழற்சியை ஒப்பிடுகையில் இது அதிகம். இது எதனால் எனில், தானமைவாக 1000 அழைப்புகளுக்குப் பிறகு பைத்தான் தன்னைத்தானே அழைப்பதை நிறுத்திவிடும். வரம்புகளை (limit) sys.setrecursionlimit (limit_value) பயன்படுத்தி மாற்றியமைக்க அனுமதிக்கிறது.

எடுத்துக்காட்டு

```
import sys
sys.setrecursionlimit(3000)
def fact(n):
        if n == 0:
        return 1
         else:
        return n * fact(n-1)
print(fact (2000))
```

நினைவில் கொள்க

- செயற்கூறுகள் ஒரு குறிப்பிட்ட செயலினை செய்வதற்காக வடிவமைக்கப்பட்டு பெயரிடப்பட்ட குறிமுறையின் தொகுதியே செயற்கூறு எனப்படும்.
- பயனர் வரையறைத்த, உள்ளிணைந்த, லாம்டா மற்றும் தற்சுழற்சிக்கு செயற்கூறுகளின் வகைகள் ஆகும்.
- செயற்கூறு தொகுதி "def" என்ற சிறப்புச் சொல்லுடன் தொடங்கி செயற்கூறுவின் பெயர் மற்றும் ( ) அடைப்புக்குறியுடன் முடிய வேண்டும்.
- செயலுறுப்புகளில் இல்லாத return, None - ற்க்கு நிகரானது. return கூற்றொரு விருப்பத்தேர்வுயில்லை
- பைத்தானில், ஒரு தொகுதிக்குள் உள்ள கூற்றுகள் உள்தள்ளிருக்க வேண்டும்.
- ஒரு தொகுதிக்குள் வேறொரு தொகுதியிருப்பின் அதை பின்னலான தொகுதி எனப்படும்.
- செயற்கூறை அழைப்பதற்கு செயலுருபுகள் பயன்படுகின்றன. 4 வகையான செயலுருப்புக்கள் செயற்கூறில் பயன்படுத்தாம்: தேவையான செயலுருபுக்கள், சிறப்பு செயல் செயலுருப்புகள், தானமைவு செயலுருபுக்கள் மற்றும் மாறும்-நீள செயலுருப்புகள்.
- தேவையான செயலுருபபுகள் செயற்கூறில் சரியான இடத்திற்கு அனுப்பப்பட வேண்டிய செயலுருபுகளாகும்.
- அளபுருக்களின் பெயரை அடையாளம் கண்ட பின்பு சிறப்புச் சொல் செயலுருபானது செயற்கூறினை அழைக்கிறது.
- பைத்தானில், செயற்கூறை அழைக்கும் போது எந்த மதிப்பும் கொடுக்கப்படவில்லை எனில் செயலுருபானது தானாகவே மதிப்பை எடுத்துக் கொள்ளும்.
- * குறியீடு மாறும்-நீள செயலுருபுகளை வரையறைக்க பயன்படும்.
- பெயரில்லாமல் வரையறுக்கப்படும் செயற்கூறுவுக்கு பெயரில்லாத செயற்கூறாகும்.
- மாறிகளின் வரையெல்லை, நிரலின் அணுகக் கூடிய பகுதியைக் குறிப்பதாகும்.
- செயற்கூறு திருப்பி பின்னலான அமைப்பில் பயன்படுத்தினால் அதற்கு தொகுப்பு. என்று பெயர்.
- ஒரு செயற்கூறு தன்னைத்தானே அழைத்தால் அதை தற்சுழற்சி என்றழைக்கப்படும். மடக்கினை போன்ற தற்சுழற்சியும் செயல்படும். ஆனால் சில சமயங்களில் மடக்கினை பன்படுத்துவதற்கு பதில் தற்சுழற்சியை பயன்படுத்தலாம்.

செய்முறைப் பயிற்சி:1

எடுத்துக்காட்டு 7.5.3-ல் உள்ள நிரலில் பின்வரும் குறிமுறையை முயற்சித்துப் பார்க்கவும்.

வ.எண்	குறிமுறை	வெளியீடு
1	printinfo(“3500”)	
2	printinfo(“3500”,”Sri”)	
3	printinfo(name=”balu”)	
4	printinfo(“Jose”,1234)	
5	printinfo(“ ”,salary=1234)	

செய்முறைப் பயிற்சி :2

பின்வரும் செயற்கூறுகளை மதிப்பீடு செய்து அதன் வெளியீட்டை எழுதுக.

வ.எண்	செயற்கூறு	வெளியீடு
1	1) abs(-25+12.0)) 2) abs(-3.2)	
2	1) ord('2') 2) ord('$')	
3	type('s')	
4	bin(16)	
5	1) chr(13) 2) print(chr(13))	
6	1) round(18.2,1) 2) round(18.2,0) 3) round(0.5100,3) 4) round(0.5120,3)	
7	1) format(66, 'c') 2) format(10, 'x') 3) format(10, 'X') 4) format(0b110, 'd') 5) format(0xa, 'd')	

8	1) pow(2,-3) 2) pow(2,3.0) 3) pow(2,0) 4) pow((1+2),2) 5) pow(-3,2) 6) pow(2*2,2)	

செய்முறைப் பயிற்சி:3

பின்வரும் செயற்கூறுகளை மதிப்பீடு செய்து அதன் வெளியீட்டை எழுதுக

வ.எண்	செயற்கூறு	வெளியீடு
1	eval('25*2-5*4')	
2	math.sqrt(abs(-81))	
3	math.ceil(3.5+4.6)	
4	math.floor(3.5+4.6)	

மதிப்பீடு

பகுதி–அ

சரியான விடையை தேர்ந்தெடுத்து எழுதுக **(1 மதிப்பெண்)**

1. ஒரு குறிப்பிட்ட செயலைச் செய்வதற்காக வடிவமைக்கப்பட்டு, பெயரிடப்பட்ட குறிமுறையின் தொகுதி
   (அ) மடக்கு (ஆ) கிளைப்பிரிப்பு
   (இ) செயற்கூறு (ஈ) தொகுதி
2. தன்னைத்தானே அழைத்துக் கொள்ளும் செயற்கூறை இவ்வாறு அழைப்பர்.
   (அ) உள்ளிணைந்த (ஆ) தற்சுழற்சி
   (இ) லாம்டா (ஈ) return கூற்று
3. எந்த செயற்கூறை பெயரில்லா செயற்கூறு என்று அழைக்கப்படுகிறது?
   (அ) லாம்டா (ஆ) தற்சுழற்சி
   (இ) செயற்கூறு (ஈ) வரையறை

ZSUKEV

4. செயற்கூறு தொகுதியை எந்த சிறப்புச்சொல் தொடங்கிவைக்கிறது?

(அ) define (ஆ) for
(இ) finally (ஈ) def

5. எந்த சிறப்புச்சொல் செயற்கூறு தொகுதியை முடித்து வைக்கிறது?

(அ) define (ஆ) return
(இ) finally (ஈ) def

6. செயற்கூறு வரையிறையில் பின்வரும் எந்த குறியீடு பயன்படுத்தப்படுகிறது?

(அ) ; (அரைப் புள்ளி) (ஆ) . (புள்ளி)
(இ) : (முக்காற் புள்ளி) (ஈ) $ (டாலர்)

7. செயற்கூறுக்கு எந்த செயலுருபு சரியான இட வரிசையில் செயலுருபுகளை அனுப்பும்?

(அ) தேவைப்படும் (ஆ) சிறப்புச்சொல்
(இ) தானமைவு (ஈ) மாறிநீளம்

8. பின்வரும் கூற்றுகளைப் படித்து, சரியான கூற்றுகளைத் தேர்ந்து எடுக்கவும்.

(I) பைத்தானில், செயற்கூறை வரையறுக்கும் போது குறிப்பிட்ட தரவு வகைகளைக் குறிப்பிடத் தேவையில்லை

(II) பைத்தான் சிறப்புச் சொற்களைச் செயற்கூறின் பெயராகப் பயன்படுத்தலாம்.

(அ) I சரி மற்றும் II தவறு (ஆ) இரண்டுமே சரி
(இ) I தவறு மற்றும் II சரி (ஈ) இரண்டுமே தவறு

9. கொடுக்கப்பட்ட கூற்றை வெற்றிகரமாக நிறை வேற்றுவதற்கு, பின்வருவனவற்றுள் சரியான ஒன்றைத் தேர்ந்தெடு

`if _____ : print(x, " is a leap year")`

(அ) x%2=0 (ஆ) x%4==0
(இ) x/4=0 (ஈ) x%4=0

10. testpython() செயற்கூறைவரையறுக்க பின்வரும் எந்த சிறப்புச் சொல்பயன்படுகிறது?

(அ) define (ஆ) pass
(இ) def (ஈ) while

பகுதி–ஆ

அனைத்து வினாக்களுக்கும் விடையளி (2 மதிப்பெண்)

1. செயற்கூறு என்றால் என்ன?
2. செயற்கூறின் வகைகளை எழுதுக.
3. செயற்கூறுவின் முக்கிய நன்மைகள் யாவை?
4. மாறியின் வரையெல்லை என்றால் என்ன? அதன் வகைகளைக் குறிப்பிடுக.

Chapter 7.indd 116 30-01-2020 19:31:00

5. குளோபல் வரையெல்லை-வரையறு

6. தன்னைத் தானே அழைக்கும் செயற்கூறில் அடிப்படை நிபந்தனை என்றால் என்ன?

7. தன்னைத்தானே அழைக்கும் செயற்கூறுக்கு வரம்பை எவ்வாறு அமைக்க வேண்டும்? எடுத்துக்காட்டு தருக.

பகுதி–இ

அனைத்து வினாக்களுக்கும் விடையளி **(3 மதிப்பெண்)**

8. உள்ளமை மாறிகளுக்கான விதிமுறைகளை எழுதுக.

9. பைத்தானிலுள்ள முழுதளாவி சிறப்புச் சொல்லுக்கான அடிப்படை விதிமுறைகளை எழுதுக.

10. செயற்கூறினுள் முழுதளாவி மாறியை மாற்றம் செய்தால் என்ன நிகழும்?

11. ceil() மற்றும் floor() செயற்கூறுகளை வேறுபடுத்துக.

12. கொடுக்கப்பட்ட வருடம் லீப் வருடமா இல்லையா என்பதனைச் சோதிக்கும் பைத்தான் நிரலை எழுதுக.

13. செயற்கூறில் தொகுப்பு என்பது என்ன?

14. தற்சுழற்சி எவ்வாறு செயல்படுகிறது?

15. செயற்கூறினை வரையறுக்கும் போது குறிப்பிடப்பட வேண்டிய குறிப்புகள் யாவை?

பகுதி–ஈ

அனைத்து வினாக்களுக்கும் விடையளி **(5 மதிப்பெண்)**

16. செயற்கூறின் வகைகளை எடுத்துக்காட்டுடன் விவரி

17. மாறியின் வரையெல்லைகளை எடுத்துக்காட்டுடன் விளக்குக.

18. பின்வரும் உள்ளிணைந்த செயற்கூறுகளை விளக்குக.

    (a) id()

    (b) chr()

    (c) round()

    (d) type()

    (e) pow()

19. இரண்டு எண்களின் LCM கண்டுப்பிடிப்பதற்கான பைத்தான் நிரலை எழுதுக.

20. தற்சுழற்சி செயற்கூறு பற்றி எடுத்துக்காட்டுன் விளக்குக.

Reference Books

1. *Python Tutorial book from tutorialspoint.com*
2. *Python Programming: A modular approach by Pearson – Sheetal, Taneja*
3. *Fundamentals of Python –First Programs by Kenneth A. Lambert*