නිපුනණතාව 9

ගැටලු විසඳීමට ඇල්ගොරිතම සංවර්ධනයය කරා ඒවා ආකේතනය (encoding) කිරීම සඳහා Python ක්‍රමලේඛ භාෂාව භාවිත කිරීම

9.1 ගැටලු විසඳීමේ ක්‍රියාවලිය (Problem-solving process)

9.2 ගැටලු විසඳීම සඳහා මුදුන් බිම් පියවරාකාර පිරිපහදු ක්‍රමවේද ගවේෂණය

9.3 ගැටලු විසඳීම සඳහා ඇල්ගොරිතමික ප්‍රවේශය යොදා ගැනීම

9.4 ක්‍රමලේඛනය කිරීමේ විවිධ සුසමාදර්ශ (Paradigms) සැසඳීම

9.5 ක්‍රමලේඛ පරිවර්තනය කිරීමේ අවශ්‍යතාව සහ ක්‍රමලේඛ පරිවර්තක ප්‍රරූප ගවේශනය කිරීම

9.6 සමෝධානික සංවර්ධන පරිසරයේ (IDE) මූලික ලක්ෂණ හඳුනාගැනීමට එය ගවේෂණය කිරීම

9.7 ඇල්ගොරිතම ආකේතනය කිරීම සඳහා විධානාත්මක ක්‍රමලේඛ භාෂාවක් (Python) භාවිත කිරීම

9.8 කුමලේඛ සංවර්ධනයෙහි පාලන ව්‍යුහ භාවිත කිරීම

9.9 ක්‍රමලේඛනයේ දී උපක්‍රම ලේඛ (subprograms)භාවිත කිරීම.

9.10 ක්‍රමලේඛවල දී දත්ත ව්‍යුහ යොදා ගැනීම

9.11 ක්‍රමලේඛවල දී ගොනු සහ දත්ත සමුදාය හැසිරවීම

9.12 දත්ත සමුදායක දත්ත කළමනාකරණය කිරීම

9.13 දත්ත සෙවීම සහ තේරීම (Searching and sorting)

සැකසුම

සමන් ප්‍රියන්ත බාලසූරිය
මහසෙන් ජාතික පාසල
නිකවැරටිය

9.1 ගැටලු විසඳීමේ ක්‍රියාවලිය (**Problem solving process**)

- ගැටලුව හඳුනා ගැනීම (Understanding the problem)
- ගැටලුව හා එහි සීමාවන් අර්ථ දැක්වීම (Defining the problem and boundaries)
- විසඳුම සැලසුම් කිරීම (Planning solution)
- විසඳුම ක්‍රියාත්මක කිරීම (Implementation)

ගැටලු විසඳීම යන යෙදුම ඇසූ පමණින් ඔබට වැටහෙන්නේ යම් ගණිතමය ගැටලුවක් විසඳීමක් විය හැක. එසේ නැතිනම් ඔබ මුහුණ දුන් යම් ප්‍රායෝගික ගැටළුවක් විසඳීම විය හැක. කෙසේ වෙතත් මෙම පාඩමෙන් අරමුණු වෙන්නේ තොරතුරු තාක්ෂණික යෙවුම් භාවිතය භාවිතයෙන් ප්‍රායෝගික ජීවිතයේ දී මුහුණ පාන ගැටලු විසඳීම පිළිබඳවයි. වර්තමානයේ දී බොහොමයක් හස්තම පද්ධති පරිගණකගත පද්ධති බවට පරිවර්තනය වී ඇත්තේ හස්තමය පද්ධති සම්බන්ධව තිබූ ගැටලු, දුෂ්කරතා, අකාර්යක්ෂමතා ඉවත් කර ගනිමින් පරිගණක පද්ධති මගින් කාර්යක්ෂම සේවාවක් ලබා ගැනීම් අරමුණු කරගෙනය. පහත් සඳහන් වන්නේ ද තාක්ෂණය මත පදනම් වූ ගැටලු විසඳා ගත් අවස්ථා කිහිපයකි.

	මෙම පින්තූරයෙන් දැක්වෙන්නේ ස්වයංක්‍රීයව ඇරෙන වැහෙන දොරවල් සහිත පද්ධතියකි. බොහෝ විට වර්තමානයේ විවිධ ආයතනවල මෙවැනි පද්ධති ස්ථාපිත කර ඇති බව දැකිය හැක .
	ස්වයංක්රීය ටෙලර් යන්ත්‍ර භාවිතය මගින් වර්තමානයේ දී ගණුදෙනු සැකසුම් ඉතා වේගවත්ව කාර්යක්ෂමව සිදු කරනු ලැබේ. මෙය ද තොරතුරු තාක්ෂණයේ දියුණු යෙදුමකි.
	ආයතන සේවකයින්ගේ පැමිණීම හා පිටවීම වාර්තාකරණ වාර්තා පොත් වෙනුවට වර්තමානයේ ඇඟිලි සලකුණු යන්ත්‍ර භාවිත කරනු ලැබේ. මෙයින් ද ඉතා කාර්යක්ෂම සේවාවක් ආයතනයට ලැබේ.
	මෙවැනි ගනුදෙනු සැකසුම් පද්ධති රහිත සුපිරි වෙළඳ සලක් වර්තමානයේ දැකිය නොහැකි තරම්ය. ගනුදෙනු සැකසුම් කාර්යක්ෂමව සිදුකිරීමට බොහෝ ව්‍යාපාරිකයන් මෙවැනි යෙදුම් භාවිතයට වර්තමානය පෙළඹී තිබේ.

මෙම සියලු අවස්ථා වලින් නිරූපණය වන්නේ ප්‍රායෝගික ජීවිතයේ ගැටලු විසඳීම් ය. ඒම ගැටලු විසඳීම සඳහා යොදාගෙන ඇත්තේ තොරතුරු තාක්ෂණයික මෙවලම් ය. පරිගණකය භාවිතයෙන් ගැටළු විසඳීමේ ක්‍රියාවලිය පිළිබඳව අධ්‍යයනය කිරීම මෙම ඒකකයේ අරමුණයි.

පරිගණක යෙදවුම් භාවිතයෙන් ගැටලු විසඳීම අදියර කිහිපයක් ඔස්සේ සිදු වේ. ඒවා ඒවා නම්, ගැටලුව හඳුනාගැනීම, ගැටලුව හා එහි සීමාවන් අර්ථ දැක්වීම, විසඳුම් සැලසුම් කිරීම, හා විසඳුම ක්‍රියාත්මක කිරීම්යි. ගැටලු හඳුනාගැනීමේ අදියරේ දී , ගැටලුව හා ඊට පාදකව ඇති සියලුම ක්‍රියාවලි සුවිශේෂීව හඳුනාගනී. ගැටලුව විසඳීමට පෙර ගැටළුව කුමක්දැයි හඳුනාග ගත යුතුය. ගැටළුවට පදනම් වී ඇති හේතු හඳුනාග ගත යුතුය. එවිට ගැටලුව විසඳීම සඳහා සාර්ථක ප්‍රවේශයක් ලැබේ. ගැටළුව මැනවින් හඳුනාගත් පසු ගැටලු විසඳන ආකාරය සුවිශේෂී අර්ථ දැක්විය යුතුය.

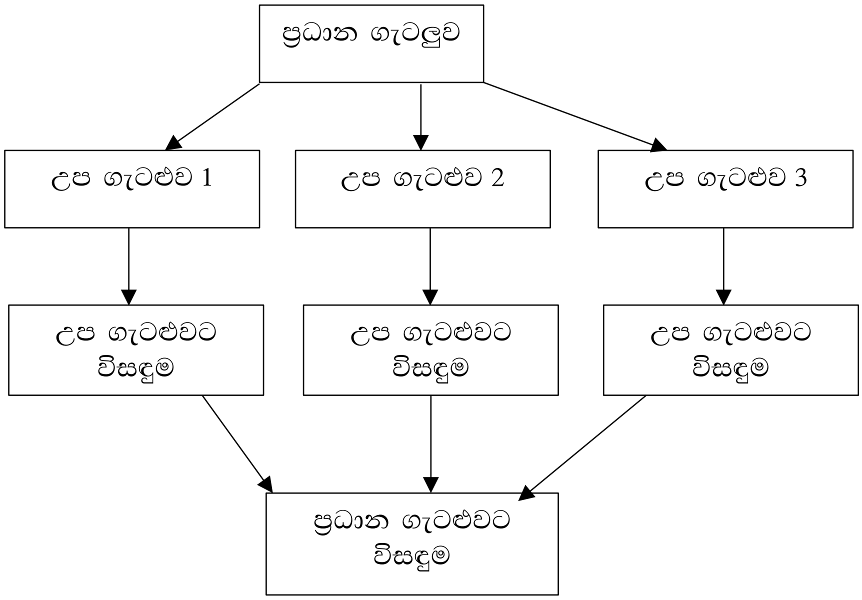
ප්‍රධාන ගැටලුවක් විසඳන විට එයට සම්බන්ධව ඇති අනෙකුත් ගැටලු විසඳී ම මගින් ප්‍රධාන ගැටළුවට විසඳීම ද සිදු වේ. ගැටලුව හා එම ගැටලුව සීමාවන් නිර්ණය කිරීම ගැටලුවක් විසදිමේ දෙවන අදියරයි. ගැටලුව විසඳීමට පෙර සුදුසු විසුදුසු විසඳුමක් සැලසුම් කරගත යුතුය. බොහෝ විට එකම ගැටලුව විසඳීමට විකල්ප විසඳුම් ද තිබිය හැක ඒවායින් වඩාත් කාර්‍යක්ෂම හා වඩාත් උචිත විසඳුම තීරණය කිරීම හා එය ක්‍රියාත්මක කරන ආකාරය විසඳුම් සැලසුම් කිරීමේ අධදියරේදී සිදු වේ. විසඳුම් ක්‍රියාත්මක කිරීමේ අදියරේ දී සිදුවන්නේ තෝරාගත් විකල්ප විසඳුම් අතරෙන් වඩාත් වඩාත් උචිත විසඳුම් ක්‍රියාත්මක කරමින් ගැටලුව විසඳා ගැනීමයි.

9.2 ගැටලු විසඳීම සඳහා මුදුන් බිම් පියවරාකාර පිරිපහදු ක්‍රමවේද ගවේෂණය

- මොඩියුලකරණය (Modularization)
- මුදුන් බිම් සැලසුම (top down design) හා පියවරාකාර පිරිපහදු (Stepwise refinement) ක්‍රමවේදය
- ව්‍යුහ සටහන් (Structure charts)

මොඩියුලකරණය (Modularization)

මොඩියුලකරණය යන්නෙන් අදහස් වන්නේ , ප්‍රධාන ගැටළුවක් විසඳීමට ඇති විට එම ගැටළුව උප ගැටළුවලට විභේදනය කර, එම උප ගැටළු විසඳිම මගින් ප්‍රධාන ගැටළුව විසඳා විසඳා ගැනීමයි. ප්‍රධාන ගැටළුවක් විභේදනය කර ගොඩනගාගන්නා තවදුරටත් විභේදනය නොවන ඒකක මොඩියුල ලෙස හඳුන්වයි. මෙලෙස හඳුනාගත් මොඩිවුල වලට අදාල විසඳුම් ගොඩනැගීම් මගින් ප්‍රධාන ගැටළුවට විසඳුම් ගොඩනැගීම් මොඩියුලකරණය මගින් සිදු කෙරේ.

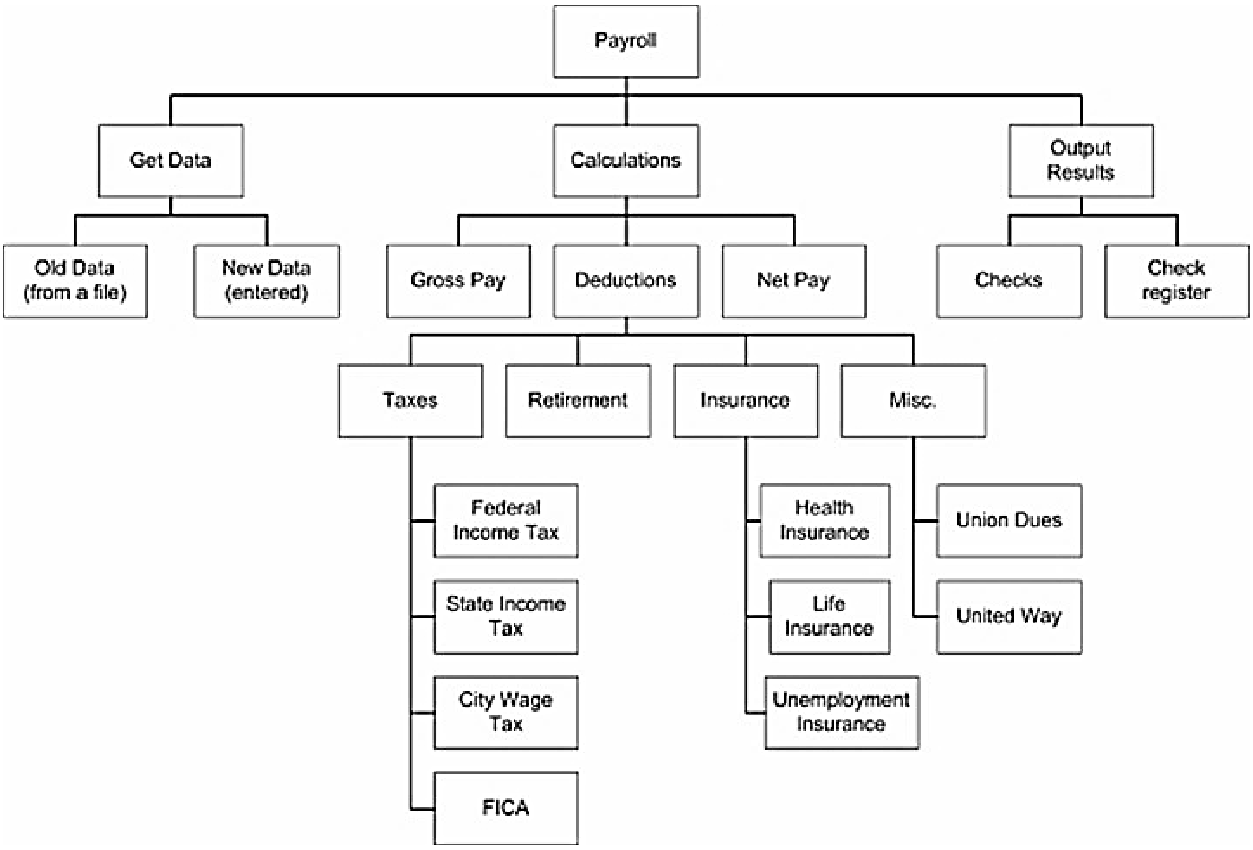


මුදුන් බිම් සැලසුම (top down design) හා පියවරාකාර පිරිපහදු (Stepwise refinement) ක්‍රමවේදය

මුදුන් බිම් සැලසුම් මගින් සමස්ත ගැටළුයේ සිට අනු කොටස් දක්වා ගැටලුවක් විශ්ලේෂණය සිදුවේ. එළුමගින් ගැටලුවේ ව්‍යුහය හා විසඳුම සංවිධානය කර ගන්නා ආකාරය පිළිබඳව පරිපූර්ණ අදහසක් ලබා ලබාගැනීමට හැකිවේ. පියවරාකාර පිරිපහදු ක්‍රමවේදය ද මේ හා සම්බන්ධ ක්‍රියාවලියකි. බොහෝ විට ඔබ අත් දැක ඇති පරිදි, ගැටලුවක් විසඳීම පියවර වශයෙන් සිදුවේ. එක් එක් පියවර නිවැරදිව විසඳීම සමස්ත ගැටලුව විසඳීමේ නිරවද්‍යභාවය රඳා පවතී. සමස්ත ගැටලුව එක් එක් පියවර ඔස්සේ විශ්ලේෂණය කිරීම පියවරාකාර පිරිපහදු ක්‍රමවේදය වේ.

ව්‍යුහ සටහන් (Structure charts)

ව්‍යුහ සටහන් යනු ගැටලළුවකට විසඳුම මුදුන් බිම් සැලසුම් අනුව පියවරෙන් පියවර නිරූපණය කරන රූපමය සටහනකි. සමස්ත ගැටලුව උප ගැටලුවලට හෙවත් මොඩියුලවලට විභේදනය කර, එම එක් එක් මොඩියුල විසඳන ආකාරය නිරූපණය කරමින්, සමස්තයේ සිට අනු කොටස් දක්වා ගැටලුව විභේදනය කරමින් විසඳුම් රූපමය ලෙස නිරූපණය කිරීමට ව්‍යුහ සටහන් යොදා ගනී.



9.3 ගැටලු විසඳීම සඳහා ඇල්ගොරිතමික ප්‍රවේශය යොදා ගැනීම

- ඇල්ගොරිතම
  - ගැලීම් සටහන් (Flow charts)
  - ව්‍යාජ කේත (Pseudo code)

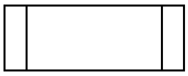
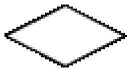
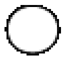
ගැටලු විසඳීම සඳහා ඇල්ගොරිතම ගොඩනැඟීම

ගැටලුවක් විසඳීම සඳහා අනුගමනය කළයුතු පියවර සියල්ල අනුපිළිවෙළින් දක්වූ ක්‍රමවේදයක් ඇල්ගොරිතමයකි. මෙම කාර්යය සඳහා රූපමය (Graphical) හෝ ලිඛිත (Textual) නිරූපණයන් උපයෝගී කරගත හැකි ය. ගැලීම් සටහන් (Flowcharts) මේ සඳහා බහුලව භාවිතවන රූපමය නිරූපණයක් වන අතර ව්‍යාජ කේත (Pseudocode) මගින් ඇල්ගොරිතම ලිඛිතව නිරූපණය කළ හැකි ය.

ගැලීම් සටහන් (Flowcharts)

ඇල්ගොරිතමය රූපමය ආකාරයෙන් පියවරෙන් පියවර ගොඩ නැගෙන හෝ ගොඩ නැගී ඇති ආකාරය හෝ ඉදිරිපත් කිරීම සඳහා ගැලීම් සටහන් යොදා ගැනේ. මෙහි දී එක් එක් ක්‍රියාව දැක්වීම සඳහා පහත දැක්වෙන සංකේත භාවිත කෙරේ.

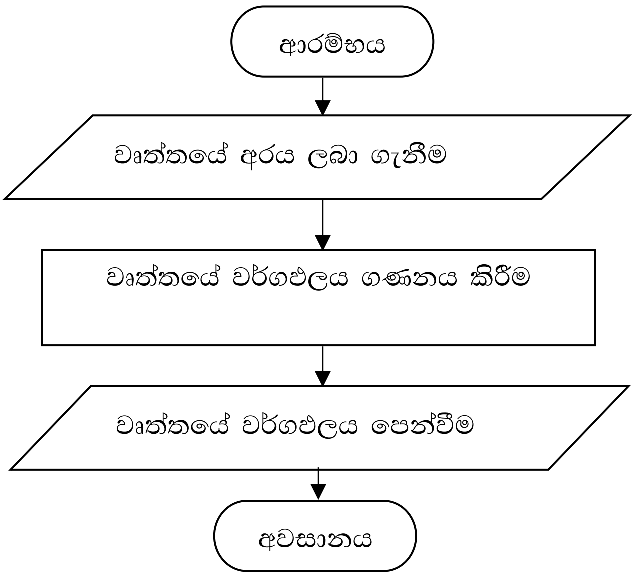
ගැලීම් සටහන් (Flowcharts) මේ සඳහා භාවිතා කරනු ලබන සංකේත

	Terminator	ආරම්භය / අවසානය (Start / Stop)
	Process	විධානයක් හෝ අනුපිළිවෙළින් ක්‍රියා කළ යුතු විධාන එකතුවක්
	Subroutine	උප ක්‍රියාකාරකම
	Data Input / Output	දත්ත ආදාන / ප්‍රතිදාන
	Decision	තීරණ
	Connector	සම්බන්ධක
	Flow Lines	දත්ත ගැලීමේ දිශාවන්

අනුක්‍රමය සහිත ගැලීම් සටහන් :

ආරම්භක පියවරේ සිට අවසාන පියවර දක්වා ඇති පියවර සියල්ල ම එකින් එක අනුපිළිවෙළකට සිදුවීම අනුක්‍රමයක ලක්ෂණය වේ.

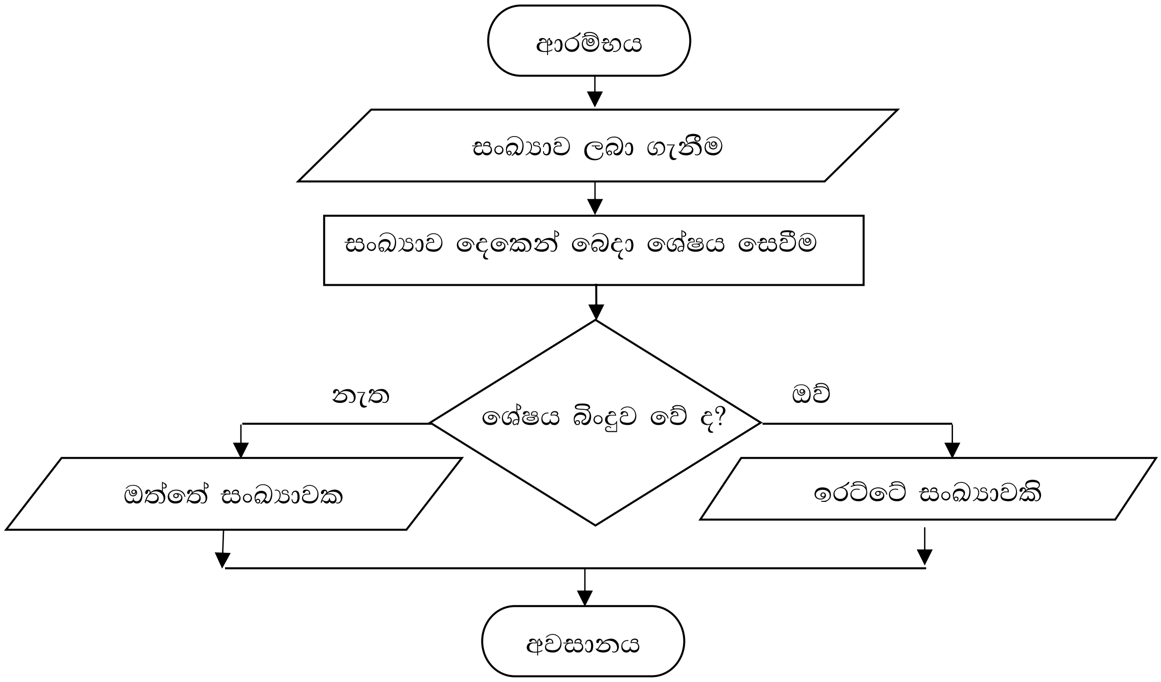
උදා. වෘත්තයක වර්ගඵලය ගණනය කිරීම



වරණය සහිත ගැලීම් සටහන්

කොන්දේසියක් තෘප්ත වීම හෝ නොවීම හෝ අනුව ගැලීම් දිශාව තීරණය වීම වරණයක ඇති විශේෂත්වය යි.

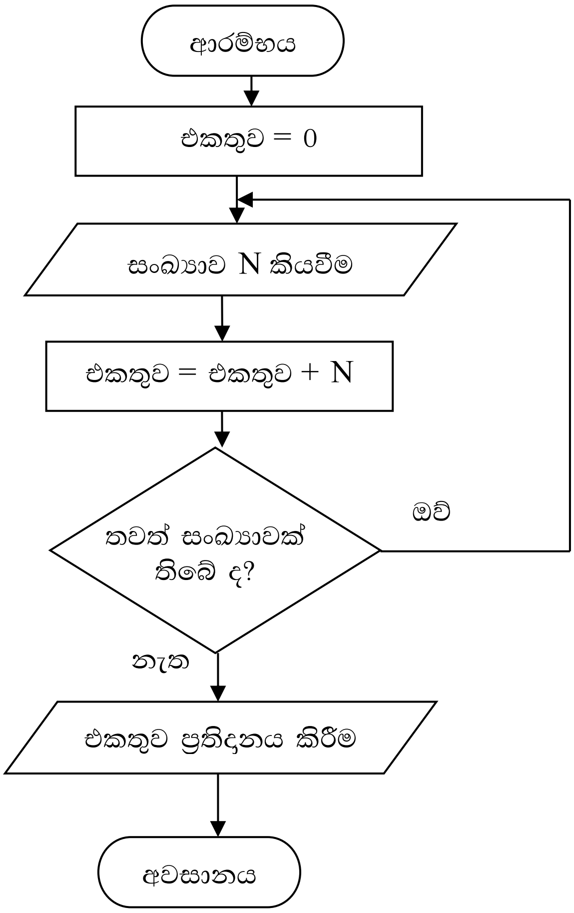
උදා - සංඛ්‍යාවක් ඔත්තේ ද, ඉරට්ටේ ද යන්න සෙවීම



පුනර්කරණ සහිත ගැලීම් සටහන් :

පියවර කිහිපයක්, කොන්දෙසියක් තෘප්ත වන තෙක් හෝ තෘප්තව පවතිනතාක් කල් හෝ නැවත නැවත සිදුවීම මෙහි ඇති විශේෂත්වයයි.

සංඛ්‍යා සමූහයක එකතුව සෙවීම



ව්‍යාජ කේත (Pseudocode)

ඇල්ගොරිතමයක් සරල ඉංග්‍රීසි වචන යොදා ගෙන ලිඛිත ව දැක්වීම ව්‍යාජ කේත මගින් සිදු වේ. මෙසේ ලියනු ලබන ව්‍යාජ කේත පරිගණක භාෂාවකින් ස්වායත්ත වේ. ව්‍යාජ කේත ඕනෑ ම පරිගණක භාෂා උපදෙස් බවට පරිවර්තනය කළ හැකි ය. එබැවින් ව්‍යාජ කේත ලිවීමෙන් පරිගණක ක්‍රමලේඛනය පහසු කරයි.

ඇල්ගොරිතමයේ එක් එක් ප්‍රකාශය සරල ඉංග්‍රීසි වචනවලින් දක්වන ආකාරය විමසා බලමු.

ආරම්භය BEGIN
අවසානය END
ආදානය INPUT, READ, GET
ප්‍රතිදානය OUTPUT, DISPLAY, SHOW
ක්‍රියාවලිය PROCESS, CALCULATE
වරණය IF … THEN .. .ELSE … ENDIF
පුනර්කරණය FOR – DO , WHILE – ENDWHILE ,REPEAT - UNTIL

ව්‍යාජ කේත ලිවීම

අනුක්‍රමය (Sequence)

සංඛ්‍යා දෙකක එකතුව සෙවීම

```
Begin
        Input first number and second number
        Total = first number+ Second number
        Output Total
End
```

ලකුණු3 ක එකතුව සෙවීම

```
Begin
        Input 1st mark, 2nd mark, 3rd mark
        Total= 1st mark+2nd mark+3rd mark
        Output Total
End
```

ලකුණු 3 ක එකතුව සහ සාමාන්‍ය අගය සෙවීම

```
Begin
        Input 1st mark, 2nd mark, 3rd mark
        Average= (1st mark+ 2nd mark+3rd mark)/number of subjects
        Output Average
 End
```

සෘජුකෝණාස්‍රයක වර්ගඵලය සෙවීම

```
Begin
        Input length , width
        Area of rectangle= length X width
        Output Area
 End
```

වෘත්තයක වර්ගඵලය ගණනය කිරීම

```
BEGIN
        INPUT Radius
        CALCULATE Area = 22/7 x Radius x Radius
        DISPLAY Area
 END.
```

වරණය (Sequence)

If Then

“Pass” හෝ “Fail” ලෙස ප්‍රතිදානය

```
Begin
    Input Marks
        If marks > 50 Then
        Display "Pass"
        Else
        Display "Fail"
        Endif
End
```

සාමාර්ථ ප්‍රතිදානය

```
Begin
    Input Marks
        If marks >=  75 Then
                Display "A"
        Else
        If marks >=  65 Then
                Display "B"
        Else
        If marks >=  55 Then
                Display "C"
        Else
        If marks >=  40 Then
                Display "S"
        Else
                Display "S"
        Endif
        Endif
        Endif
        Endif
End
```

සංඛ්‍යා දෙකකින් විශාලතම සංඛ්‍යාව ප්‍රතිදානය

```
Begin
    Input num1, num2
        If num1 > num2 Then
            Display num1
        Else
            Display num2
        End if
End
```

සංඛ්‍යාවක් ඔත්තේ ද, ඉරට්ටේ ද යන්න සෙවීම

```
BEGIN
        READ number as N
        CALCULATE Remainder after number divided by 2
        IF Remainder = 0 THEN
                DISPLAY "Even number"
        ELSE
                DISPLAY "Odd number"
        ENDIF
END.
```

පුනර්කරණය (Repetition)

a. For Next

උදාහරණ 1 - 1 සිට 10 තෙක් ඇති සංඛ්‍යාවල එකතුව ලබා ගැනීමට

```
Begin
        For Count = 1 to 10
                Input num
                Total = Total + num
        Next Count
        Display Total
End
```

b. While Do

උදාහරණය 1 - සංඛ්‍යා ආදානය කර ඒවායේ එකතුව ලබා ගැනීමට

```
Begin
        While num &gt; 0 Do
        Input num
        Total = Total + num
        End While
        Output total
End
```

උදාහරණය 2 - සංඛ්‍යා 5ක් ආදානය කර ඒවායේ එකතුව ලබා ගැනීමට

```
Begin
        Total = 0
        Count = 0
        While count &lt; 5 Do
        Input num
        Total = Total + num
        Count = Count + 1
        End while
End
```

c. Repeat until

උදාහරණය 1 - සංඛ්‍යා ආදානය කර ඒවායේ එකතුව ලබා ගැනීමට

```
Begin
        Repeat
        Input num
        Total = Total + num
        Until num &lt; 0
End
```

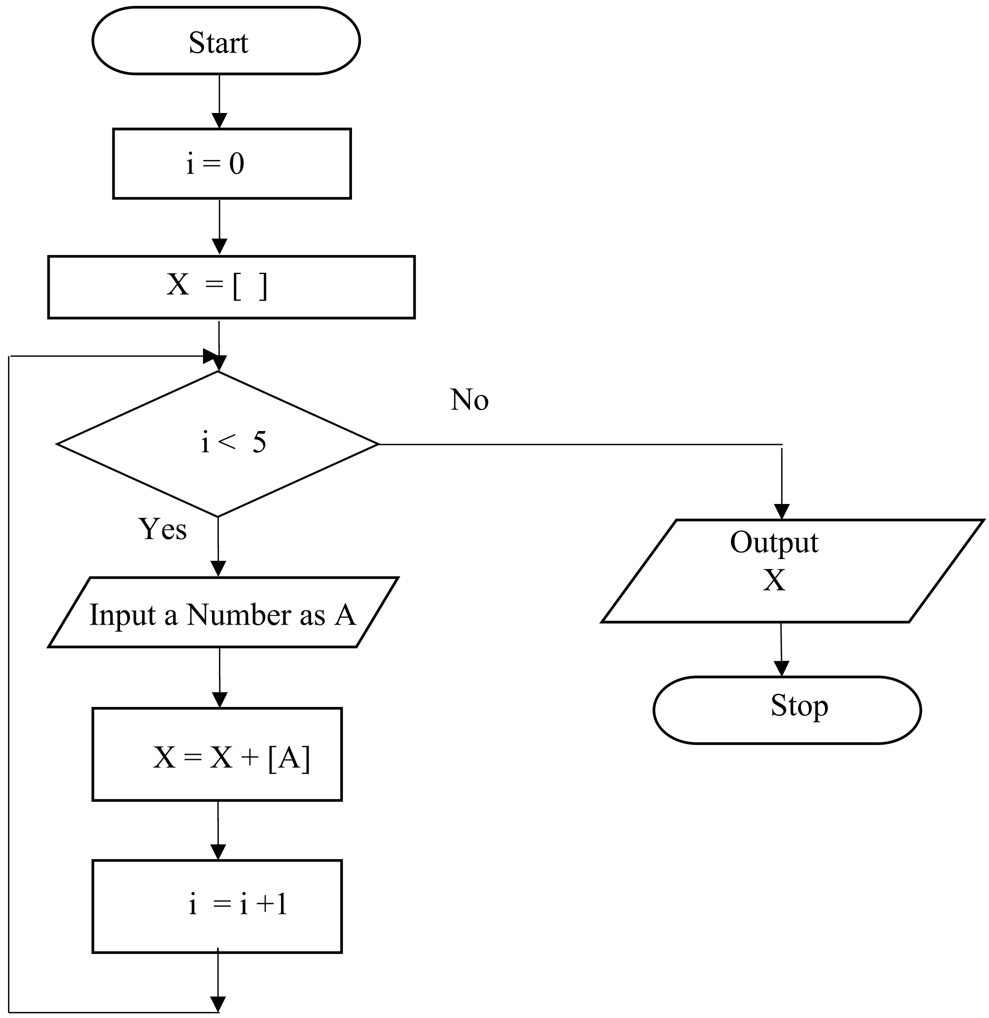
උදාහරණය 2 - සංඛ්‍යා 5ක් ආදානය කර ඒවායේ එකතුව ලබා ගැනීමට

```
Begin
        Total = 0
        Count = 0
                Repeat
                Input num
                Total = Total + num
                Count = Count + 1
                Until Count = 5
        Display Total
End
```

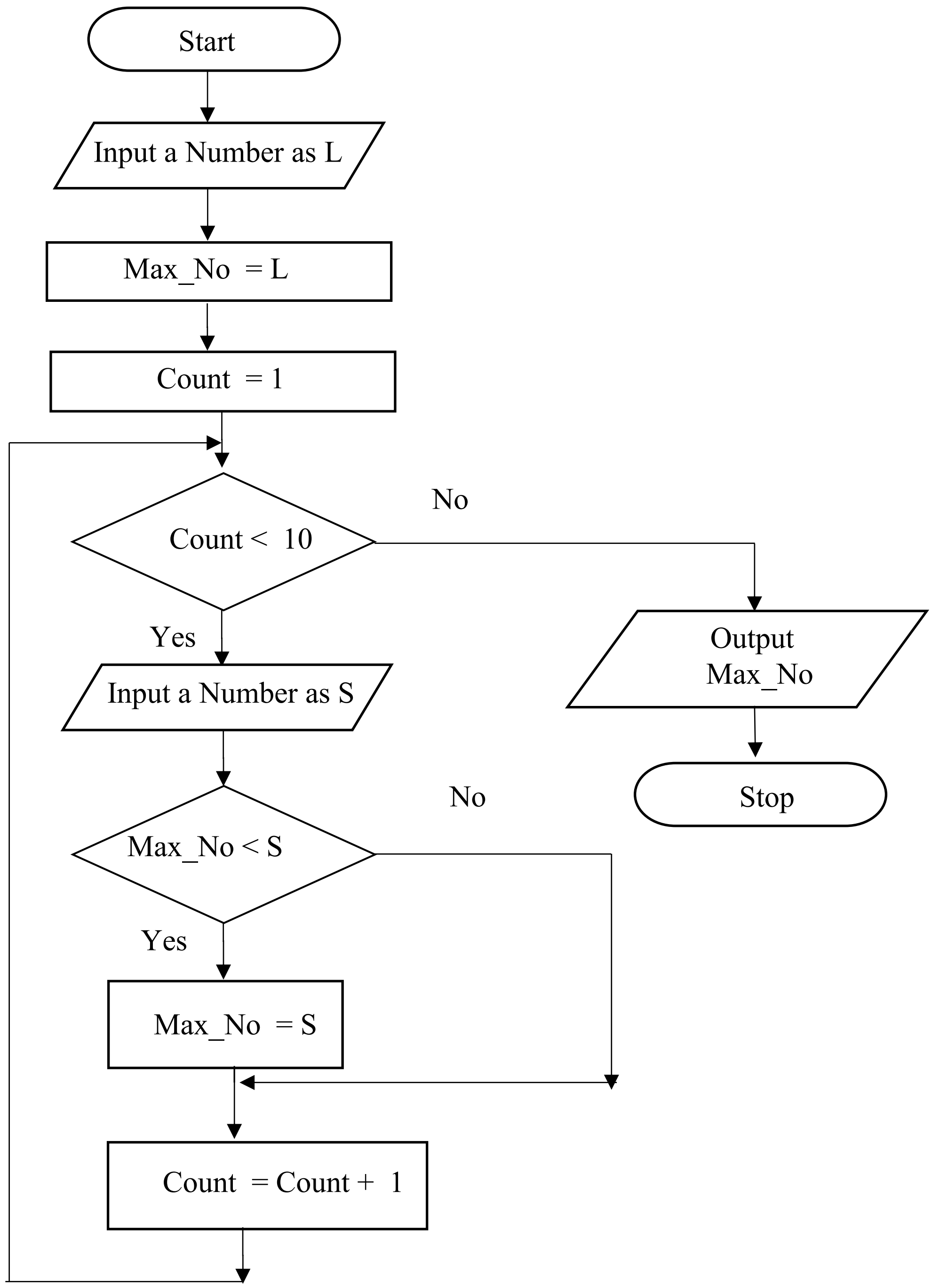
සංඛ්‍යා 10ක එකතුව සහ සාමාන්‍යය සෙවීම

```
BEGIN
    Total = 0
    Average = 0
    n = 1
    WHILE n <= 10
        READ number
        CALCULATE Total = Total + number
        n = n + 1
    ENDWHILE
    CALCULATE Average = Total/(n-1)
    DISPLAY Total, Average
END.
```

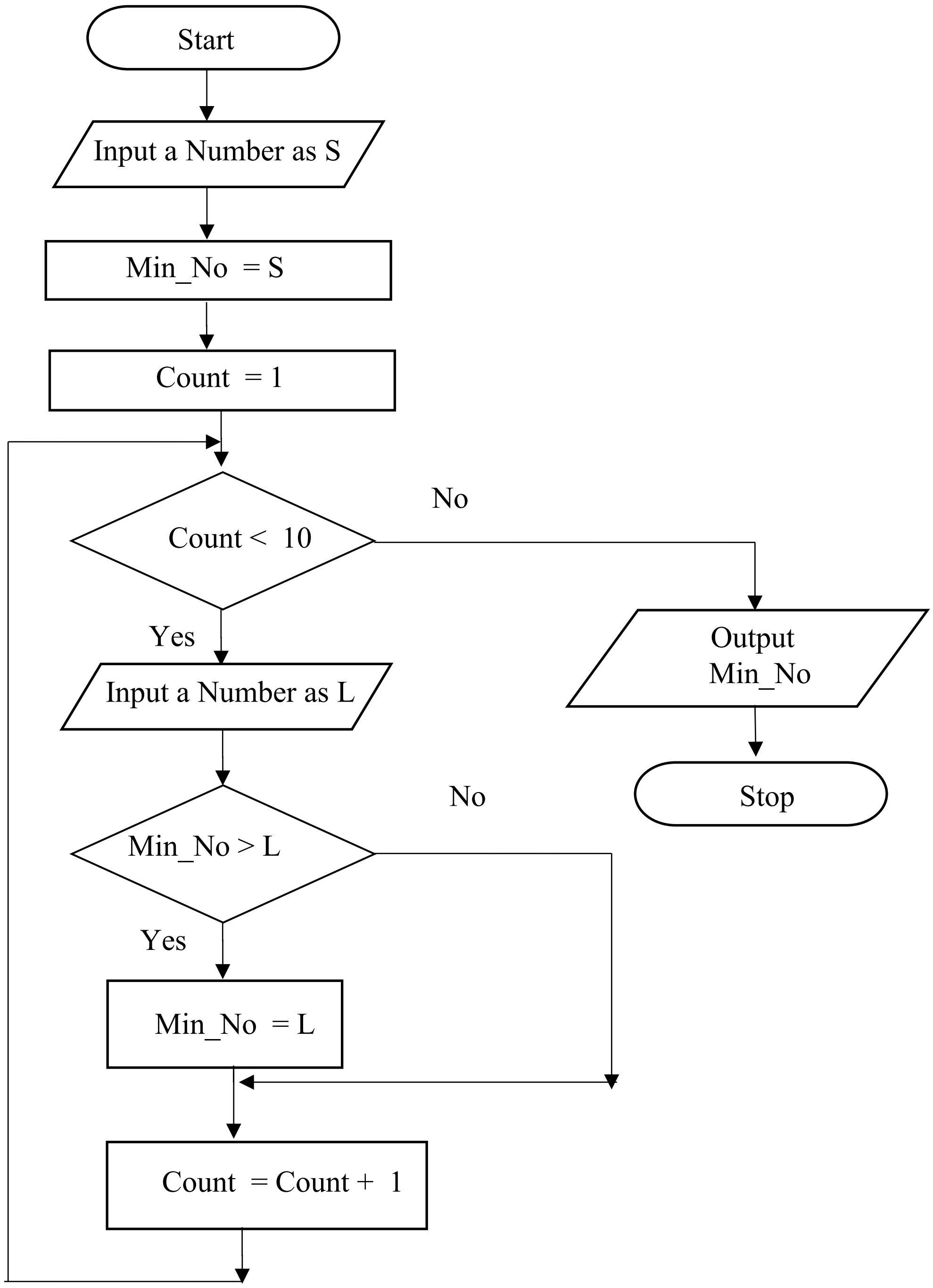
සංඛ්‍යා 5 ක් ආදානය කර ලැයිස්තුවක් සැකසීම



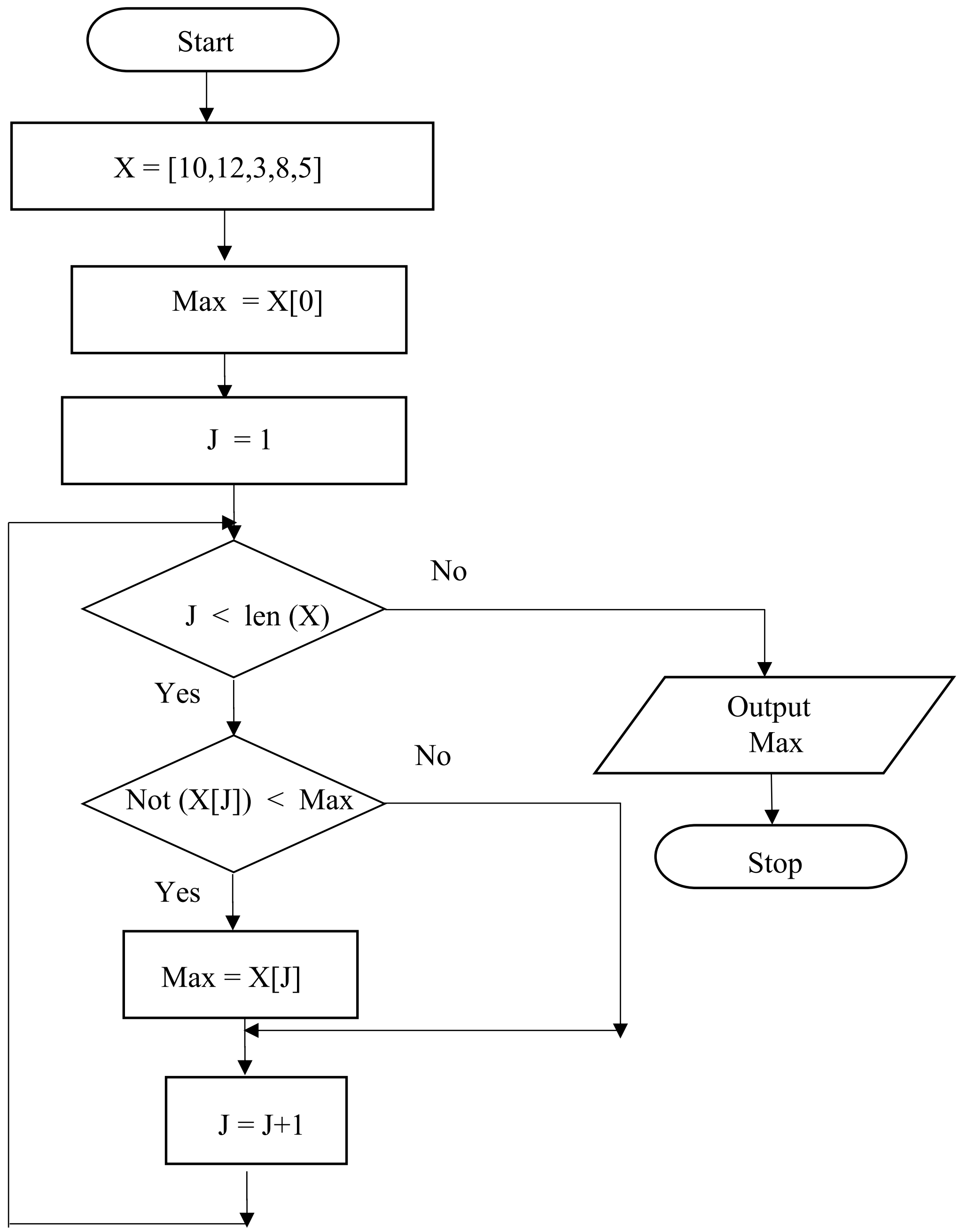
සංඛ්‍යා 10 ක් ආදානය කර විශාලතම සංඛ්‍යාව සෙවීම සඳහා



සංඛ්‍යා 10ක් ආදානය කර කුඩාම සංඛ්‍යාව සෙවීම සඳහා



ලැයිස්තුවක ඇති සංඛ්‍යාවලින් විශාලතම සංඛ්‍යාව සෙවීම සඳහා



9.4 ක්‍රමලේඛනය කිරීමේ විවිධ සුසමාදර්ශ (Paradigms) සසඳා බලා වෙන් කර දක්වයි.

- පරිගණක භාෂාවන්ගේ පරිණාමය
- ක්‍රමලේඛකරණ සුසමාදර්ශ
    - විධානාත්මක (Imperative) භාෂා
    - ප්‍රකාශාත්මක භාෂා (declarative) භාෂා
    - වස්තු නැඹුරු (Object Oriented) භාෂා

ක්‍රමලේඛ භාෂාවල පරිණාමය

ක්‍රමලේඛ භාෂාවක අවශ්‍යතාව

ක්‍රමලේඛයක් යනු පරිගණකය විසින් කළ යුතු විශේෂ කාර්යයක් ඉටුකරන ආකාරය දැක්වෙන උපදෙස් අනුක්‍රමයකි. මෙම උපදෙස් ලබා දීම සඳහා භාෂාවක් අවශ්‍ය වේ.

පහළ තලයේ භාෂා (Low level languages)

යන්ත්‍ර භාෂාව (Machine language)

පරිගණකය තුළ ඍජු ව ම ක්‍රියාත්මක කළ හැකි භාෂාවකි. උපදෙස් ලබා දීම සඳහා 0 සහ 1 යන ද්විමය සංඛ්‍යාංක (බිටු) යොදා ගන්නා ලදි. එබැවින් යන්ත්‍ර භාෂාවෙන් ලියන ලද ක්‍රමලේඛයක් ඍජු ව ම සකසනයට ධාවනය කළ හැකි විය.

පහත සඳහන් වන්නේ යන්ත්‍ර භාෂාවෙන් ලියන ලද ක්‍රමලේඛයකි.

A *machine code* program for adding 1234 and 4321. This is the lowest level of programming: direct manipulation of the digital electronics. (The right column is a continuation of the left column).

10111001	00000000
11010010	10100001
00000100	00000000
10001001	00000000
00001110	10001011
00000000	00011110
00000000	00000010
10111001	00000000
11100001	00000011
00010000	11000011
10001001	10100011
00001110	00000100
00000010	00000000

යන්ත්‍ර භාෂාවෙන් ලියන ලද ක්‍රමලේඛයක ලක්ෂණ

- ක්‍රියාත්මක වීම ඉතා ම වේගවත් වීම
- භාෂා පරිවර්තක වැඩසටහන් අවශ්‍ය නොවීම
- යන්ත්‍රය මත යැපීම (එක් පරිගණකයකට ලියන ලද ක්‍රමලේඛයක් වෙනත් පරිගණකයක ධාවනය නොවීම)
- 0 සහ 1 පමණක් භාවිතයෙන් ලියා ඇති නිසා මිනිසාට තේරුම් ගැනීම සංකීර්ණ වීම

එසෙම්බ්ලි භාෂාව (Assembly language)

ඇසෙම්බ්ලි භාෂාව යනු පරිගණක ක්‍රමලේඛන ලිවීමට භාවිතා කරන ප්‍රාථමික මට්ටමේ පරිගණක භාෂාවකි. මෙහි කෙටි යෙදුම asm ලෙස භාවිතා කරයි. ඇසෙම්බ්ලි පරිගණක භාෂාව මූලික වශයෙන් පරිගණක මෙහෙයුම් පද්ධති හෝ එවැනි වැඩසටහන් ලිවිය හැකි උපාංග සඳහා වැඩසටහන් සකස් කිරීමට භාවිතා කරනු ලබයි.

උදාහරණ

පහත සඳහන් වන්නේ සංඛ්‍යා දෙකක් ආදානය සඳහා එසෙම්බ්ලි භාෂාවෙන් ලියන ලද කේතයකි.

2000	MOV	CX, 0000	[CX] <- 0000
2003	MOV	AX, [3000]	[AX] <- [3000]
2007	MOV	BX, [3002]	[BX] <- [3002]
200B	ADD	AX, BX	[AX] <- [AX] + [BX]
200D	JNC	2010	Jump if no carry
200F	INC	CX	[CX] <- [CX] + 1
2010	MOV	[3004], AX	[3004] <- [AX]
2014	MOV	[3006], CX	[3006] <- [CX]
2018	HLT		

යන්ත්‍ර භාෂාවෙන් 0 සහ 1 මත පදනම් ව ලියන ලද විධාන වෙනුවට (ADD, SUB) වැනි සරල සංකේත නාම භාවිත කර එසෙම්බ්ලි භාෂාව නිර්මාණය කර ඇත.

එසෙම්බ්ලි භාෂාවෙන් ලියන ලද ක්‍රමලේඛයක ලක්ෂණ

- ක්‍රියාත්මක වීම යන්ත්‍ර භාෂා ක්‍රමලේඛයකකට සාපේක්ෂ ව වේගවත් බව අඩු ය.
- ඇසෙම්බ්ලර් නම් වූ භාෂා පරිවර්තක වැඩසටහන මගින් යන්ත්‍ර භාෂා උපදෙස් බවට පරිවර්තනය කළ යුතු වීම
- යන්ත්‍රය මත යැපීම (එක් පරිගණකයකට ලියන ලද ක්‍රමලේඛයක් වෙනත් පරිගණකයක ධාවනය නොවීම)
- 0 හා 1 මත පදනම් වූ විධාන වෙනුවට සංකේත නාම භාවිත කිරීම නිසා සංකේත භාෂා ලෙස ද හඳුන්වනු ලැබේ.

බොහෝ උසස් ගණයේ පරිගණක භාෂා සඳහා ඉන්ටර්ප්‍රිටර් (Interpreter) හෝ සම්පාදක (Compiler) අවශ්‍ය වේ. එයට හේතුව වන්නේ එම පරිගණක භාෂාවන් සංකේතමය පරිගණක භාෂා වීමයි. එනම් 1010 වැනි ද්විමය කේත අඩංගු යාන්ත්‍ර භාෂා නොවන නිසා ඒවා උපාංග වලට සෘජුවම ක්‍රියාත්මක කරවිය නොහැකි නිසා එවැනි භාෂා වෙනත් යෙදවුමක් හරහා යාන්ත්‍ර භාෂාවට හැරවීමට ඉහත කී දෙවර්ගයේ යෙදවුම් අවශ්‍ය වේ.ඒ අනුව ඇසෙම්බ්ලි භාෂාව ද සංකේතමය පරිගණක භාෂාවක් වේ. නමුත් ඇසෙම්බ්ලි භාෂාව යාන්ත්‍ර භාෂාවට හැරවීමට භාවිතා කරනුයේ ඇසෙම්බ්ලර් (Assembler) නම් යෙදවුමයි.

ඉහළ තලයේ භාෂා (High level languages)

ඉංග්‍රීසි භාෂාවේ සරල වචන යොදා ගනිමින් ක්‍රමලේඛකයාට වඩා පහසුවෙන් තේරුම් ගැනීමට හැකි වන පරිදි නිර්මාණය කරන ලද භාෂා ඉහළ තලයේ භාෂා ලෙස හැඳින්වේ.

ඉහළ තලයේ භාෂා සඳහා උදාහරණ

FORTRAN - (Formula Translation)

BASIC – (Beginner's All-purpose Symbolic Instruction Code)

COBOL - (COmmon Business-Oriented Language)

Pascal, C, Java, Python

පහත සඳහන් වන්නේ සංඛ්‍යා දෙකක් ආදානය සඳහා Java භාෂාවෙන් ලියන ලද කේතයකි.

```
public class AddTwoNumbers {
   public static void main(String[] args) {
      int num1 = 5, num2 = 15, sum;
      sum = num1 + num2;
      System.out.println("Sum of these numbers: "+sum);
   }
}
```

ඉහළ තලයේ භාෂාවෙන් ලියන ලද ක්‍රමලේඛයක ලක්ෂණ

- තේරුම් ගැනීම පහසු ය.
- ධාවනය කිරීමට ප්‍රථමයෙන් යන්ත්‍ර භාෂා උපදෙස් බවට පරිවර්තනය කළ යුතු ය.
- යන්ත්‍රය මත යැපෙන භාෂාවක් නො වේ.

ක්‍රමලේඛ භාෂා වර්ග

ක්‍රමලේඛය කිරීම යනු කිසි යම් කාර්යයක් කරන ආකාරය පිළිබඳ ව පරිගණකයට උපදෙස් දෙනු පිණිස පරිගණක ක්‍රමලේඛකයෙක් විසින් සකසනු ලබන නිර්මාණශීලී ක්‍රියාවලියකි. කිසි යම් ගැටලුවකට විසඳුමක් දෙනු පිණිස කුමන කාර්යයක් කළ යුතු ද යන්න ගැන පරිගණකයට උපදෙස් දීමට සකස් කළ උපදෙස් මාලාව ක්‍රමලේඛයක් ලෙස හැඳින්විය හැකි ය.

ක්‍රමලේඛන ක්‍රියාවලියට ප්‍රවේශ වීම පිණිස විකල්ප ප්‍රවේශ ගණනාවක් ඇත. ඒවා ක්‍රමලේඛන පැරඩයිම (Paradigms) ලෙස හැඳින්වේ. ක්‍රමලේඛන භාවිත කොට විශේෂ ගැටලුවලට විසඳුම් ගොඩනැගීම සඳහා මූලික වශයෙන් වෙනස් ආකාර ප්‍රවේශ, විවිධ පැරඩයිම මගින් නියෝජනය වේ. බහුතර ක්‍රමලේඛන භාෂා එක පැරඩයිම වර්ගයක් යටතට ගැනෙන නමුත් සමහර භාෂාවල විවිධ පැරඩයිමවලට අයත් මූලිකාංග දැකිය හැකි ය.

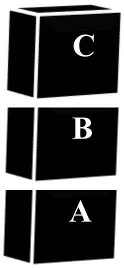
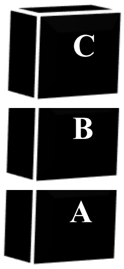
පැරඩයිම (Paradigms)					
විධානාත්මක (Imperative)			ප්‍රකාශන (Declarative)		
පටිපාටිගත (Procedural) උදා. C භාෂාව	වස්තු නැඹුරු (Object Oriented) උදා. Java, Smalltalk, Simula, C++	සමාන්තරගත සැකැස්ම (Parallel Processing) උදා. Java භාෂාව	තාර්කික (Logic) උදා. Prolog භාෂාව	ශ්‍රිතමය දත්ත ගැලීම් (Functional Dataflow) උදා. Lisp භාෂාව	දත්ත ගබඩා (Database) උදා. SQL

විධානාත්මක පරිගණක භාෂා (Imperative Programming Languages)

- විධානාත්මක පරිගණක භාෂා , පරිගනකය සඳහා දෙන විධානයන්ගෙන් සමන්විත වේ. එම විධාන පියවරෙන් පියවර සිදුවිය යුතු ආකාරය දක්වයි.
- විධානාත්මක පරිගණක භාෂා නිර්මාණය වී ඇත්තේ පරිගණක වැඩසටහන් ක්‍රියාකරන්නේ කෙසේද (How a program operate) යන්න පදනම් කරගෙනය.
- එනම් ගැටලුවකට විසඳුම් සෙවීමේදී, විසඳුම ලබා ගන්නා ආකාරය පියවරෙන් පියවර දක්වයි.

ප්‍රකාශන පරිගණක භාෂා (Declarative Programming Languages)

- පාලන ගැලීම (Control Flow) විස්තර කිරීමකින් තොරව ගණනය කිරීමේ තාර්කික න්‍යාය පමණක් දක්වන ක්‍රමලේඛන භාෂා, ප්‍රකාශන පරිගණක භාෂාවන්ට අයත් වේ.
- ප්‍රකාශන පරිගණක භාෂා මගින් අතුරු ප්‍රතිඵල අවම කිරීමට උත්සහ දරණු ලබයි. එනම්, පරිගණකයට ලබාදෙන්නේ ගැටලුව කුමක්ද යන්න පමණි එය සිදුවිය යුතු ආකාරය පියවර ලෙස නොදක්වයි. එනිසා දුවිය යුතු ආකාරය පියවර ලෙස දැක්වීමෙන් ඇතිවන අතුරු ප්‍රතිඵල අවම කරයි.
- ප්‍රකාශන පරිගණක භාෂා නිර්මාණය වී ඇත්තේ මොන ගණනය කිරීමද (What the program must do) අවශ්‍ය යන්න පදනම් කරගෙනය. එම ගණනය කිරීම සිදුවන්නේ කෙසේද යන්න නොදක්වයි.
- එනම් ගැටලුවකට විසඳුම් සෙවීමේදී, ගැටලුව කුමක්ද යන්න පමණක් පරිගනකය වෙත ලබාදෙයි.
- ප්‍රකාශන පරිගණක භාෂා කෘතිම බුද්ධිය මත පදනම්ව ඇත.

පටිපාටිගත ක්‍රමලේඛ **(Procedural Paradigm)**	ප්‍රකාශන ක්‍රමලේඛ **(Declarative Paradigm)**
විසඳුම සොයන ආකාරය පැවසීම Saying how you achieve it 1. A කැටය තබන්න. 2. A කැටය මත B කැටය තබන්න. 3. C කැටය B කැටය මත තබන්න. 	අවශ්‍ය දේ පැවසීම Saying what you want 1.කැට 3කින් සමන්විත කණුවක 

භාෂා පරම්පරාව අනුව පරිගණක භාෂා වර්ගීකරණය

1945 පමණ ජෝන් වොන් නියුමාන්ගේ අදහසකට අනුව පරිගණකයේ භෞතික තත්ව වෙනස් නොකොට ක්‍රමලේඛ මගින් විවිධ කාර්යයන් කරගනීම ඇරඹිණි.එතැන් සිට මෙතෙක් දක්වා නිර්මාණය වූ සියළු පරිගණක භාෂා ප්‍රධාන ආකාර 4කට බෙදා දක්වයි.

පළමු පරම්පරාවේ පරිගණක භාෂා (Machine Language)

මේ වර්ගයට අයත් වන්නේ යන්ත්‍ර භාෂාවයි. එහි පහත ලක්ෂණ පවතී,

- ක්‍රමලේඛ නිර්මාණය සඳහා 0 සහ 1 යන ද්වීමය ස0ඛ්‍යා පමණක් යොදා ගැනීම
- ක්‍රමලේඛ නිර්මාණය ඉතා දුෂ්කර කටයුත්තක් වීම
- ඉතා දියුණු මට්ටමේ ක්‍රමලේඛ නිර්මාණය කළ නොහැකි වීම
- කිසියම් පරිගණකයක තාක්ෂණයක් මූලික කර ගනිමින් නිර්මාණය කර ඇති නිසා වෙනත් තාක්ෂණයන් සහිත පරිගණක වල භාවිත් කළ නොහකි වීම
- දෘඩාංග මත යැපෙන භාෂාවක් නිසා ක්‍රමලේඛකයා පරිගණකයේ දෘඩාංග නිර්මිතිය (Hardware Architecture) පිළිබඳ දැන සිටිය යුතුය.
- පරිවර්තක මෘදුකාංග (Translators) අවශ්‍ය නොවීය එබැවින් වැඩ සටහන ක්‍රියාත්මක වීම වේගවත් ය.

දෙවන පරම්පරාවේ පරිගණක භාෂා (Assembly Language)

- විධාන ක්‍රමය Machine Language එකට සාපේක්ෂව සරළය.
- ඉංග්‍රීසි අකුරු සහ ඉලක්කම්,සංකේත add, sub වැනි යෙදුම්ද භාවිතා කරයි
- විධාන ගණනාවක් වෙනුවට සංකේත භාවිතා කරන නිසා සංකේත භාෂා ලෙස හඳුන්වයි.
- Assembler නම් භාෂා පරිවර්තකය සංකේත භාෂාව යාන්ත්‍රික භාෂාව බවට පරිවර්තනය කිරීම සඳහා යොදා ගන්නා ලදි.
- එයද යන්ත්‍ර මත යැපෙන භාෂාවකි.

- මෙය පළමු පරම්පරාවට වඩා ක්‍රියාත්මක වන වේගය අඩුය.

තුන්වන පරම්පරාවේ පරිගණක භාෂා

- මෙහි හැකියාවන් වැඩිය.ඉතා දියුනු මට්ටමේ ක්‍රමලේඛ නිර්මාණය කිරීම සඳහා යොදා ගනී
- ක්‍රමලේඛ නිර්මාණය කිරීම හා නඩත්තු කිරීම පහසුය
- මේවාට Procedural භාෂා යැයිද කිව හැකිය. උදා - Fortran, Cobol,Pascal, C, Java, Python
- වැරදි හඳුනාගැනීමේ පණිවිඩ දෙනු ලබන අතර ක්‍රම ලේඛය පරික්ෂා කිරීම පහසු වේ.
- යන්ත්‍රය මත යැපෙන්නක් නොවීය.
- ක්‍රමලේඛකයකු පුහුණු කිරීම පහසු වේ.
- සම්පාදක(Compiler) සහ අර්ථවින්‍යාසක(Interpreter) නම් පරිවර්තක මෘදුකාංග භාවිතා කරයි

හතරවන පරම්පරාවේ පරිගණක භාෂා (Artificial Language)

- මෙම පරම්පරාවේ භාෂා මගින් වැඩසටහන් සංවර්ධනයට ගතවන කාලය, ශ්‍රමය සහ පිරිවැය අවම කිරීම සිදුවේ
- මෙම භාෂා යොදා ගන්නා ප්‍රධාන ක්ෂේත්‍ර වන්නේ : දත්තපාදක විමසුම් කිරීම (database queries), වාර්තා උත්පාදනය (report generators), දත්ත කළමනාකරණය, විශ්ලේෂණය සහ වාර්තාකරණය (data manipulation, analysis and reporting), වෙබ් අඩවි සංවර්ධනය (web development) යනාදියයි.
- පරිගණක වලට මිනිසුන් භාවිතා කරන භාෂා තේරුම් ගැනීමට පහසු වන මෘදුකාංග නිර්මාණය කිරීමටද භාවිතා වේ.
- කෘතිම බුද්ධියක් සහිත ක්‍රමලේඛ නිර්මාණය කිරීම සඳහා භාවිතා කරයි
- උදාහරණ ලෙස Prolog, LISP, Mercury, දැක්විය හැක

9.5 ක්‍රමලේඛ පරිවර්තනයකිරීමේ අවශ්‍යතාව සහ ක්‍රමලේඛ පරිවර්තක ප්‍රරූප ගවේශනය කිරීම

- ක්‍රමලේඛ පරිවර්තනය කිරීමේ අවශ්‍යතාව
- ප්‍රභව ක්‍රමලේඛය (Source Program)
- වස්තු ක්‍රමලේඛය (Object Program)
- ක්‍රමලේඛ පරිවර්තක (Programme Translators)
  - අර්ථවින්‍යාසක (Interpreters)
  - සම්පාදක (Compilers)
  - දෙමුහුම් ප්‍රවේශය (Hybrid approach)
- සන්ධාරක (linkers)

පරිගණක භාෂා පරිවර්තනය (Programming Language Translation)

පරිගණකයට ඍජු ව ම ක්‍රියාකරවීමට හැකිවන්නේ 0 හා 1 යන ද්විමය කේතයන් මත පදනම් වූ යාන්ත්‍රික භාෂාවක් පමණක් බව අප දුටුවෙමු. එබැවින් යාන්ත්‍රික නොවන එසෙම්බ්ලි වැනි දෙවෙනි හා තුන්වෙනි පරම්පරාවේ භාෂාවන්ගෙන් ලියවුණු ක්‍රමලේඛ හෝ විධාන

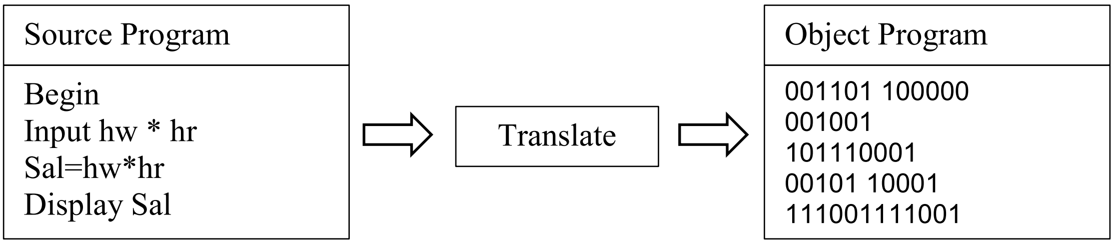
පරිගණකයක් තුළ ක්‍රියාත්මක කිරීමට පෙර පරිගණකයට තේරුම් ගත හැකි යාන්ත්‍රික භාෂා බවට සුදුසු පරිගණක මෘදුකාංගයන් මගින් පරිවර්තනය කළ යුතු ය.

ප්‍රභව ක්‍රමලේඛ (Source Program)

පරිගණක භාෂාවක් යොදාගෙන යම් ගැටළුවක් විසඳා ගැනීම සඳහා ලියන ලද කේතය

විෂය ක්‍රමලේඛ (Object Program)

පරිගණක භාෂා පරිවර්තකයක් මගින් යාන්ත්‍රික භාෂාවට පරිවර්තනය කල ක්‍රමලේඛ විෂය ක්‍රමලේඛ Object Program



මෙසේ එක් පරිගණක භාෂාවක් තවත් භාෂාවකට පරිවර්තනය කිරීමේ ක්‍රියාව පරිගණක භාෂා පරිවර්තනය (Translation) ලෙස හැඳින්වෙයි. මෙම කාර්යය සඳහා භාවිත කළ හැකි ප්‍රධාන ක්‍රමවේද තුනකි.

(1) සම්පාදකය (Compiler) භාවිතය

(2) අර්ථ වින්‍යාසකය (Interpreter) භාවිතය

(3) ඉහත (1) හා (2) හි සංකලනයක් භාවිතය

අර්ථ වින්‍යාසක (Interpreters)

අර්ථ වින්‍යාසක මෘදුකාංග මගින් මුළු ක්‍රමලේඛය ම එකවර යාන්ත්‍රික භාෂාවකට පරිවර්තනය කිරීම සිදු නොවේ. මෙහි දී සිදුවන්නේ ක්‍රමලේඛයේ අන්තර්ගත විධාන එකිනෙක ක්‍රියාත්මක කිරීමේ අවස්ථාවේ දී අදළ යාන්ත්‍රික භාෂා විධාන සමූහයකට පරිවර්තනය කිරීම යි. මෙහි ප්‍රතිඵලය වන්නේ මෙම වර්ගයේ ක්‍රමලේඛයක් ක්‍රියා කරවන සෑම අවස්ථාවක දී ම මෙම පරිවර්තන ක්‍රියාවලිය නැවත නැවත සිදු කිරීමට අවශ්‍ය වීම යි. එබැවින් සාමාන්‍යයෙන් මෙම වර්ගයේ ක්‍රමලේඛයක් ක්‍රියා කිරීමට ගන්නා කාලය සම්පාදනය කළ ක්‍රමලේඛයට ගන්නා කාලයට වඩා වැඩි වෙයි. VB, Python මෙම වර්ගයට අයත් පරිගණක භාෂා වෙයි.

සම්පාදක (Compilers)

සම්පාදක වර්ගයේ මෘදුකාංග මගින් ක්‍රමලේඛයේ ඇති සියලු විධාන එකවර යාන්ත්‍රික භාෂාවට හැරවීම සිදු කෙරේ. මෙසේ ක්‍රමලේඛය යාන්ත්‍රික භාෂාවකට හැරවූ පසු එම යාන්ත්‍රික

භාෂාවෙන් වූ ක්‍රමලේඛය නැවත පරිවර්තනයකින් තොර ව විවිධ දත්ත සඳහා ක්‍රියාත්මක කළ හැකි ය. C, FORTRAN පරිගණක භාෂා සම්පාදක මේ සඳහා නිදසුන් කිහිපයකි.

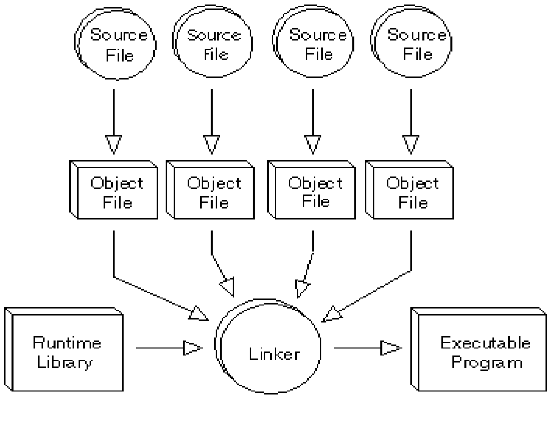
සම්පාදක මගින් භාෂා පරිවර්තනයේ දී පරිවර්තනය සඳහා යොදගන්නා මුල් ක්‍රමලේඛය ප්‍රභව ක්‍රමලේඛය (Source Programme) ලෙස ද පරිවර්තනයෙන් පසු ලබාගන්නා ක්‍රමලේඛය විෂය ක්‍රමලේඛය (Object Programme) ලෙස ද හැඳින්වෙයි. භාෂා සම්පාදකයක් මගින් සිදුවන ප්‍රධාන ක්‍රියාවලිය වන්නේ ප්‍රභව ක්‍රමලේඛය විෂය ක්‍රමලේඛයට පරිවර්තනය කිරීමයි. භාෂා පරිවර්තන ක්‍රියාවලියේ දී පරිගණක පරිවර්තකයා (Translator) සිදුකරන තවත් ප්‍රධාන කාර්යයක් නම් ප්‍රභව ක්‍රමලේඛයෙහි ඇති වැරදි හඳුනා ගැනීමයි. වැරදි සහිත ක්‍රමලේඛ පරිවර්තකයන් මගින් විෂය ක්‍රමලේඛනයනට පරිවර්තනය කළ නොහැකි ය. එබැවින් ක්‍රමලේඛනයන් ගේ ඇති සියලු වැරදි හඳුනා ගැනීම හා නිදෙස් කිරීම මෙම ක්‍රියාවලියේ දී සිදුකළ යුතු ය. මෙසේ ක්‍රමලේඛයන් ගේ ඇති වැරදි හඳුනා ගැනීම සහ එම වැරදි නිදෙස් කිරීමේ ක්‍රියාවලිය 'Debugging' ලෙස ද ක්‍රමලේඛයන් ගේ ඇති වැරදි 'Bugs' ලෙස ද හැඳින්වෙයි.

සම්පාදක හා අර්ථ වින්‍යාසක හි සංකලනයක් භාවිතය

සම්පාදක හා අර්ථ වින්‍යාසක හි සංකලනයක් භාෂා පරිවර්තනය සඳහා යොද ගැනීමේ දී මුලින් ම ක්‍රමලේඛයේ ඇති විධාන අතරමැදි පරිගණක භාෂාවකට පරිවර්තනය කෙරේ. ඉන්පසු මෙම අතරමැදි භාෂාවෙන් වූ ක්‍රමලේඛය අර්ථ වින්‍යාසක මාර්ගයෙන් පරිගණකයේ ක්‍රියාත්මක කෙරේ. 'Java' පරිගණක භාෂාව මෙයට කදිම නිදසුනකි.

සන්ධාරක (linkers)

සම්පාදකයක් මගින් ජනනය කරන ලද විවිධ විෂය කේත එකට සම්බන්ධ කර එක් ක්‍රියාකාරිත්ව ගොනුවක් (executable program) සකසන වැඩසටහනක් සන්ධාරකයක් (linker) වේ.



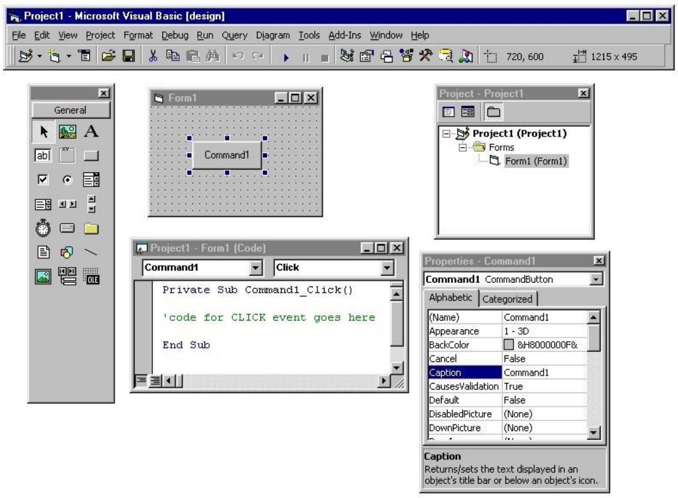
9.6 සමෝධානික සංවර්ධන පරිසරයේ (IDE) මූලික ලක්ෂණ හඳුනා ගැනීමට එය ගවේෂණය කිරීම

- සමෝධානික සංවර්ධන පරිසරයේ මූලික ලක්ෂණ

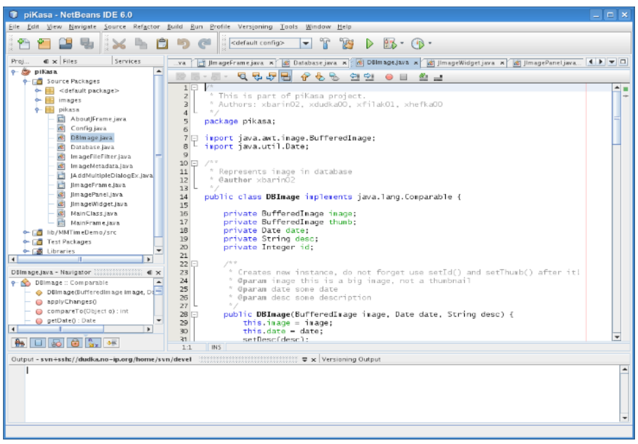
සමෝධානික සංවර්ධන පරිසරයක් යනු වැඩසටහන් සම්පාදක මෙවලමකි. මෙවැනි වැඩසටහන් සම්පාදක මෙවලම්වල ප්‍රධාන කාර්යය වන්නේ ක්‍රමලේඛ පහසුවෙන් ගොඩනැඟීම සඳහා පහසු පසුබිමක් නිර්මාණය කරදීමයි. එමගින් පරිගණක භාෂාවන් පිළිබඳ මනා දැනුමක් නොමැතිව වුවද සරල වැඩසටහන් ගොඩනගා ගැනීමට අවශ්‍ය පහසු පසුබිමක් ලබාදෙයි.

සමෝධානික සංවර්ධන පරිසරයක් මගින් වැඩසටහනේ කේත යතුරු ලියනයට පහසුකම, සකසනලද කේතය ක්‍රියාත්මක කිරීම සඳහා පරිවර්තනය කිරීමේ පහසුකම, දෝෂ හඳුනා ගැනීමේ පහසුකම සලසයි. උදාහරණයක් ලෙස 'Visual Basic' පරිගණක භාෂාව භාවිතකර ක්‍රමලේඛ ලිවීම සඳහා 'VB' හා අනුබද්ධිත 'IDE' දැක්විය හැක.

Visual Basic සංගෘහිත සංවර්ධන පරිසරය

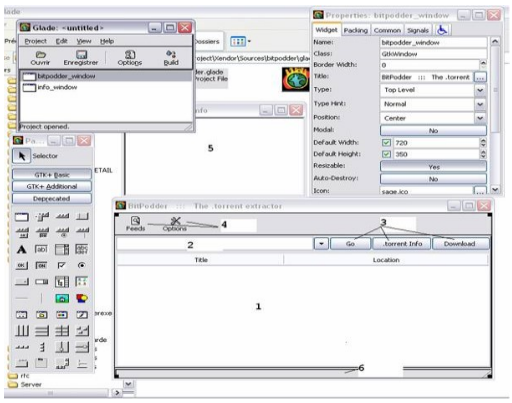


Netbeans යනු Java නම් පරිගණක භාෂාවෙන් පරිගණක වැඩසටහන් සංවර්ධනය සඳහා යොදා ගත හැකි සංගෘහිත සංවර්ධන පරිසරය කි.

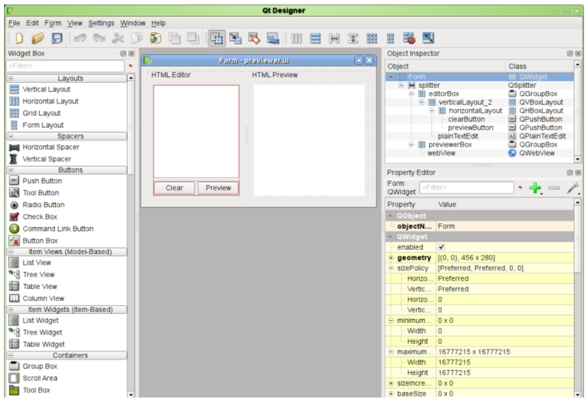


Python භාෂාවෙන් පරිගණක වැඩසටහන් සැකසීම සඳහා Python Glade , QT designer වැනි සංගෘහිත සංවර්ධන පරිසරයක් (IDE) යොදා ගත හැක.

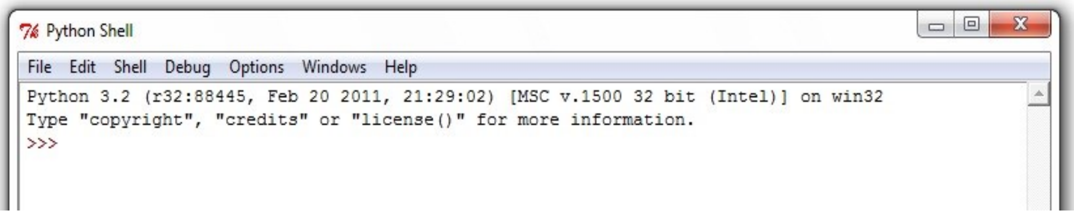
Glade IDE



QT designer

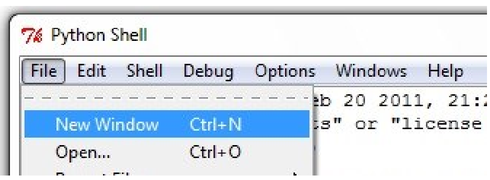


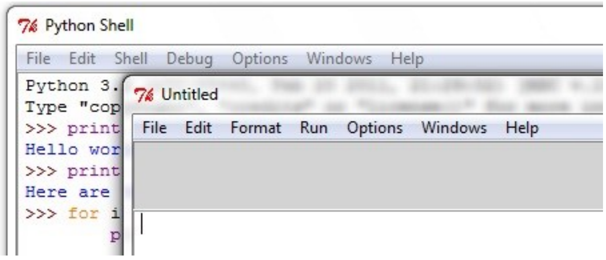
IDLE යන්නෙන් කියැවෙන්නේ පයිතන් භාෂාවෙන් වැඩසටහන් ලිවීමට යොදා ගන්නා Integrated Development Environment (IDE) යන්නයි. මෙම සමොදානික සංවර්ධන පරිසරයක් මගින් පරිගණක වැඩසටහන් සංස්කාරකය (program editor) භාෂා පරිසරය (language environment) සම්බන්ධ කර දෙමින් පරිගණක වැඩසටහන් ලියන්නාට පහසු පරිසරයක් නිර්මාණය කර දෙයි. පහත දැක්වෙන්නේ Python භාෂාවෙන් වැඩසටහන් සැකසීම සඳහා යොදා ගත හැකි සංගෘහිත සංවර්ධන පරිසරයක (IDLE) රූප සටහනකි



මෙය පයිතන් IDLE හි ප්‍රධාන කවුලුවයි. මෙය "Interpreter" හෝ ("shell") ලෙස හඳුන්වනු ලබයි. මෙහි '>>>' සලකුණු මගින් ප්‍රේරකය දක්වයි. ප්‍රේරකය යනු විධාන ලබා දිය හැකි ස්ථානයයි.

පයිතන් වැඩසටහන් ලිවීම සඳහා පයිතන් IDLE හි editor ද යොදා ගත හැකිය. මේ සඳහා Python Shell හි, File මෙනුවේ "New Window" මගින් editor ලබා ගත හැක.





Python editor

Python සමෝධානික සංවර්ධන පරිසරය (IDE) භාවිතයෙන් පහත විධාන ක්‍රියාත්මක කිරීම සඳහා අනුගමනය කළ යුතු ක්‍රමවේදය ලියන්න

1. සකසනලද Python කේතයක් පරිගණකයෙහි තැම්පත් කිරීම

………………………………………………………………………………………………

……………………………………………………………………………………………

2. තැම්පත් කර ඇති Python කේතයක් විවෘත කර ගැනීම

………………………………………………………………………………………………

……………………………………………………………………………………………

3. විධාන පේළි කිහිපයකින් යුතු පරිගණක වැඩසටහනක් සැකසීම සඳහා අවශ්‍ය Python අතුරු මුහුණත ලබා ගැනීම

………………………………………………………………………………………………

……………………………………………………………………………………………

4. විධාන පේළි කිහිපයකින් යුතු පරිගණක වැඩසටහනක් සකසා ඔබ තෝරාගත් ස්ථානයක තැම්පත් (save) කිරීම

………………………………………………………………………………………………

……………………………………………………………………………………………

5. විධාන පේළි කිහිපයකින් යුතු පරිගණක වැඩසටහනක් සකසා ඔබ තෝරාගත් ස්ථානයක තැම්පත් (save) කළ පසු වැඩසටහන ක්‍රියාත්මක කිරීම

………………………………………………………………………………………………

……………………………………………………………………………………………

Python ක්‍රමලේඛ භාෂාව යොදාගෙන පහත විධාන ක්‍රියාත්මක කරන්න. ලැබෙන ප්‍රතිදානය ඉදිරියෙන් සටහන් කරන්න

>>> print("Hello")
>>> print(2+3)
>>> x= 5
>>> y = 3
>>> x+y
>>> print(x+y)
>>> P = "Mahasen"
>>> Q = "School"
>>> P+Q
	……………………………
	……..
	……………………………
	……..
	
	……………………………
	……..
	……………………………

පහත විධාන Python වැඩසටහන මගින් ලබා දී එය Myschool.py ලෙස ඔබගේ Python වැඩසටහන් ගොනුවෙහි තැම්පත් කර ක්‍රියාත්මක කරන්න

ලැබෙන ප්‍රතිදානය ඉදිරියෙන් ලියන්න

```
print("MY School")                                   ……………………………………….
print("MY school is Mahasen national school")        ……………………………………….
print("It is in Nikaweratiya")                       ……………………………………….
print("My school is a National school")              ……………………………………….
print("There are 2000 students in my school")        ……………………………………….
print("There are 150 teachers in my school")         ……………………………………….
print("I love my school very much")                  ……………………………………….
```

ඔබ Python වැඩසටහන් සකසන Python සංගෘහිත සංවර්ධන පරිසරය (IDE) තුළ පහත කටයුතු සිදු කරන ආකාරය ලියන්න

1. ක්‍රමලේඛය පරිවර්තනය (Compiling) සහ ක්‍රියාත්මක කිරීම (Execution) සිදු කරන ආකාරය

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

2. ක්‍රමලේඛයේ ඇති දෝෂ හඳුනා ගැනීම

……………………………………………………………………………………………

……………………………………………………………………………………………

9.7 ඇල්ගොරිතම ආකේතනය කිරීම සඳහා විධානාත්මක ක්‍රමලේඛ භාෂාවක් (Python) භාවිත කිරීම

- ක්‍රමලේඛයක ව්‍යුහය
- විවරණ (Comments)
- නියතයන් සහ විචල්‍යයන්හි භූමිකාව (Role of Constants & Variables)
- ප්‍රාථමික දත්ත වර්ග (Primitive data types)
- කාරක ප්‍රවර්ග (Operator Categories)
    - ගණිතමය (Arithmetical)
    - සසැඳුම් කර්ම (Comparision)
    - තාර්කික (Logical)
    - බිට් අනුසාරිත (Bitwise)
- කාරක ප්‍රමුඛතා (Operator Precedence)
- ආදාන/ ප්‍රතිදාන
    - යතුරු පුවරුවෙන් ආදානය
    - සම්මත උපාංගවලට ප්‍රතිදානය

<u>**ක්‍රමලේඛයක ව්‍යුහය**</u>

පයිතන් භාෂාව හැඳින්වීම

Keywords in Python programming language				
False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

පයිතන් භාෂාවෙන් සකසන වැඩසහෙනක කේත ඇතුලත් කරන ආකාරය පහත පරිදි වේ.

- ප්‍රකාශණ (Statements)

ප්‍රකාශනයක් යනු තනි උපදේශයකි. උ.දා. print (“Hello”)

ප්‍රකාශනයක් පේළි කිහිපයකින් සටහන් කිරීම පහත පරිදි සිදු කළ හැක.

1. Explicit line joining (“\” සංකේතය භාවිතයෙන්)

උදා.

```
>>> y = 5
>>> y = y+\
        2
>>> y
7
>>>
```

2. Implicit line joining – (),[],{} ආදී වරහන් තුළ දක්වන ප්‍රකාශන මෙලෙස යොදා ගනී.

```
>>> month_names = ['Januari', 'Februari', 'Maart',      # These are the
            'April',   'Mei',      'Juni',       # Dutch names
            'Juli',    'Augustus', 'September',  # for the months
            'Oktober', 'November', 'December']   # of the year
>>> month_names
['Januari', 'Februari', 'Maart', 'April', 'Mei', 'Juni', 'Juli', 'Augustus', 'September', 'Oktober', 'November', 'December']
```

ප්‍රකාශන කිහිපයක් තනි පේළියක අඩංගු කිරීම

මේ සඳහා " ; " ලකුණ යොදා ප්‍රකාශන සම්බන්ධ කරනු ලබයි.

උදා.

```
>>> y = 5
>>> print (y)
5
>>> y=5; print(y)
5
>>>
```

විවරණ (Comments)

ක්‍රමලේඛයක් පිළිබඳ සවිස්තර ක්‍රමලේඛය තුලම ලේඛනගත කිරීම සඳහා විවරණ උපයෝගී කරගැනේ. විවරණ ක්‍රමලේඛයක ඇති අනෙකුත් විධාන මෙන් ක්‍රියාත්මක කළ නොහැකි ය. ක්‍රමලේඛයක් තුල විවරණ අන්තර්ගත කිරීමේ මූලික පරමාර්ථය වන්නේ එම ක්‍රමලේඛය පහසුවෙන් තේරුම් ගැනීමට අවශ්‍ය විස්තර සපයා දීමයි.

Python ක්‍රමලේඛයක විවරණ ඇතුලත් කිරීම සඳහා # යන විශේෂ සංකේත යොද ගැනේ. මෙම සංකේතය දක් වූ ස්ථානයේ සිට එම පේළියේ අවසානය තෙක් වූ සියලු විස්තර විවරණයෙහි අන්තර්ගතය ලෙස සැලකෙයි.

යම් විස්තරයක් යාබද පේළි කිහිපයක් තුළ විහිදී යන අවස්ථාවන් හිදී " " " හා " " " යන සලකුණු යොදා ගනී

උදාහරණ

Single line Comments

```
>>> # sample python codes
>>> print ("I started learning
python")
I started learning python
```

Multiline Comments (පයිතන් editor හි)

```
""" This is
a multiline comment used to
describe the python code"""

print("Python language")
print("it is a easy language to learn")
print("Instructions can be given in short codes")
```

නියතයන් සහ විචල්‍යයන්හි භූමිකාව (Role of Constants & Variables)

විචල්‍ය (Variables)

ක්‍රමලේඛයක් දත්ත මත ක්‍රියාකරවීමේ දී අදාළ දත්තයන් ප්‍රධාන මතකයේ (RAM) තබා ගැනීමට අවශ්‍ය වෙයි. මෙසේ දත්තයන් මතකයේ තාවකාලික ව තබා ගැනීම සඳහා විචල්‍ය භාවිත කළ හැකි ය. පරිගණකයේ ප්‍රධාන මතකයේ නිශ්චිත කොටසකට ප්‍රවේශ වීම සඳහා උපයෝගි කර ගත හැකි සංකේත නාමයක් ලෙෂ විචල්‍යයක් දැක්විය හැකි ය. මෙලෙස විචල්‍යයක් මතකයේ කිසියම් කොටසකට අනුබද්ධ කළ විට එම විචල්‍ය උපයෝගි කරගෙන එම අදළ මතක කොටසේ විවිධ දත්ත ගබඩා කිරීම හා නැවත ලබා ගැනීම සිදු කළ හැකි ය.

උදහරණ : -

A = 25 යන ප්‍රකාශනය සලකා බලමු. මෙවැනි ප්‍රකාශනයක් ක්‍රමලේඛයක අඩංගු කළ විට එම පරිගණකයට කිසියම් විධානයක් ලබා දෙනු ලබයි. මෙම විධානය ක්‍රියාකරවීමේ දී සිදුවන කාර්යයන් පහත දැක්වෙයි.

01. පරිගණකයේ ප්‍රධාන මතකයේ කොටසක් A නම් වූ විචල්‍ය සඳහා ලබා ගැනීම.

02. මෙම කොටසට A යන සංකේත නාමය තාවකාලික ව අනුබද්ධ කිරීම.

03. A යනුවෙන් හඳුන්වා දුන් මතක කොටසේ 25 යන අගය ගබඩා කිරීම.

උදාහරණ 2 : -

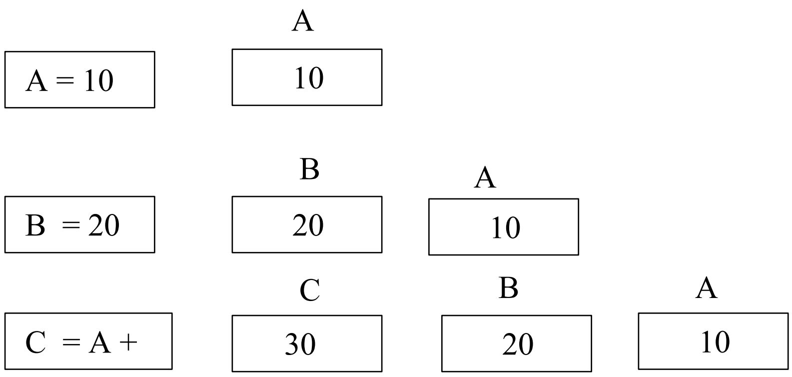
A = 10 B = 20 C = A + B

ඉහත ප්‍රකාශන පරිගණක ක්‍රමලේඛයක් ලෙස ක්‍රියාත්මක කිරීමේ දී සිදුවුණ ක්‍රියාවලිය පහත දැක්වෙයි.

01. ඉහත දැක්වූ පරිදි පරිගණකයේ ප්‍රධාන මතකයේ කොටසකට A යන සංකේත නාමය තාවකාලික ව අනුබද්ධ කිරීම හා එම කොටසේ 10 යන අගය ගබඩා කිරීම.

02. මෙලෙස ම මතකයේ වෙනත් කොටසකට B යන සංකේත නාමය අනුබද්ධ කිරීම හා එම කොටසේ 20 යන අගය ගබඩා කිරීම.

03. A කොටසේ ගබඩා කර ඇති අගයට B කොටසේ ගබඩා කර ඇති අගය එකතු කර ප්‍රතිඵලය පරිගණකයේ වෙනත් කොටසක ගබඩා කිරීම හා එම කොටසට C යන සංකේත නාමය අනුබද්ධ කිරීම.



විචල්‍ය පිළිබඳ මතක තබාගත යුතු මූලික තොරතුරු

1. පරිගණකයේ ප්‍රධාන මතකයේ දත්ත ගබඩා කිරීම සඳහා වෙන්කරගත් කොටසක සංකේත නාමයක් ලෙස විචල්‍යයක් දැක්විය හැකි ය.

2. විචල්‍යයක් හා අනුබද්ධිත මතකයේ විවිධ අවස්ථාවන්වල දී විවිධ අගයන් ගබඩා කළ හැකි ය. මෙලෙස විවිධ අවස්ථාවන්හි විවිධ අගයන් ගබඩා කිරීමේ දී අවසාන වශයෙන් ගබඩා කළ අගය මගින් ඊට පෙර ගබඩා කළ අගය ප්‍රතිස්ථාපනය වෙයි.

උදාහරණයක් ලෙස පහත ක්‍රමලේඛය සලකන්න.

A = 10

A = 20

මෙම ක්‍රමලේඛය ක්‍රියාත්මක කිරීමේ දී පළමු ව මතකයේ කොටසකට A සංකේත නාමය අනුබද්ධ කර එම කොටසේ 10 යන අගය ගබඩා කරනු ලබයි. දෙවැනි විධානය ක්‍රියාත්මක කිරීමේ දී A නම් වූ මතක කොටසේ ඇති අගය 20 යන අගයෙන් ප්‍රතිස්ථාපනය කරනු ලබයි.

3. මතකයේ කොටස් විචල්‍ය සඳහා අනුබද්ධ කිරීම තාවකාලික ක්‍රියාවලියකි. ක්‍රම ලේඛයක් ක්‍රියාත්මක කර අවසන් වූ විට එම ක්‍රමලේඛයේ විචල්‍ය සඳහා ලබාගත් මතකයේ සියලු කොටස් නැවතත් පරිගණකය ලබාගනී.

4. විදුලිය විසංධි වීමකදී විචල්‍යය ප්‍රධාන මතකයෙන් ඉවත්ව යයි.

Python භාෂාවෙහි විචල්‍ය නම් කිරීම

විචල්‍ය නාමයන් සඳහා

- අකුරු (A- Z , a - z)
- අකුරු හා ඉලක්කම්
- _ (Underscore)

ආදිය යොදා ගනී. නමුත් විචල්‍ය නාමයක් කිසි විටකත් ඉලක්කමකින් ආරම්භ නොවේ.

උදාහරණ

```
>>> Number_1 = 45
>>> Number_2 = 55
>>> Total = Number_1 + Number_2
>>> print(Total)
100
>>>
```

විචල්‍යයන් කීපයක් සඳහා ඇත්තේ එකම අගයක් වන විට

```
>>> A = B = C = 10
>>> print(A+B+C)
30
>>>
```

පහත විධාන ක්‍රියාත්මක කරන්න. ලැබෙන ප්‍රතිදානය ඉදිරියෙන් සටහන් කරන්න

>>> 5+6
>>> _+22
>>> AX= 10
>>> AX =20
>>> AX+AX
	………………
	…………
	………………
	……….

නියතයන් (Constant)

නියතයක් යනු ඩැසටහන තුළ යොදා ගැනීමට අවශ්‍ය ස්ථාවර අගයක් ගබඩා කොට තබාගැනීමට යොදා ගන්නා බේත කණ්ඩයකි.

උදා. :-

Pi = 3.142

පයිතන් හි ප්‍රාථමික දත්ත වර්ග (Primitive data types)

කලින් නිර්වචනය කළ ලක්ෂණ සහිත මෙහෙයවන සහිත අගයයන් සමූහයක් දත්ත ප්‍රරූපයක් ලෙස හැඳින්විය හැකිය. දත්ත භාවිතයට පෙර ඒවායේ ප්‍රරූප දැනගත යුතුය.

Python හි සම්මත දත්ත ප්‍රරූප

- Numbers
    - o Integral
        - ▪ Integer
        - ▪ Boolean
    - o Real
    - o Complex
- Sequences
    - o Immutable sequences
        - ▪ Strings
        - ▪ Tuples
        - ▪ Bytes
    - o Mutable sequences
        - ▪ Lists
        - ▪ Byte Arrays
- Set types
    - o Sets
    - o Frozen sets
- Mappings
    - o Dictionaries

Python භාෂාවේ දත්ත ප්‍රරූප Mutable හා Immutable ලෙස වර්ග කළ හැක. Mutable ගණයට වැටෙන දත්ත ප්‍රරූපයන්ට අයත් වස්තූන් ගේ අගයයන් එම වස්තු නිර්මාණයෙන් පසු ඕනෑම අවස්ථාවක වෙනස් කළ හැකිය. එනමුදු Immutable ගණයට අයත් වස්තූන් නිර්මාණය කළ පසු ඒවායේ අගයයන් වෙනස් කළ නොහැකි ය.

Numbers, Strings, හා Tuple යනාදිය Immutable ලණයට අයත් දත්ත ප්‍රරූප වන අතර Lists හා Dictionaries යන ප්‍රරූප Mutable ගණයට අයත් දත්ත ප්‍රරූප වෙයි.

Python භාෂාවේ ඇති එක වරකට එක අවයවයක් බැගින් මුදා හැරීමට හැකියාවක් ඇති දත්ත එකතුවක් Iterable object ලෙස හැඳින්වෙයි. Lists, Strings හා Tuples යන දත්ත ප්‍රරූප මෙවැනි Iterable දත්ත ප්‍රරූප සඳහා නිදසුන් කීපයකි.

Numbers

මෙමගින් ගණිතමය සංඛ්‍යා නිරූපනය වෙයි.තබා ගත හැකි කුඩාම හා විශාලතම සංඛ්‍යාව පරිගණකයේ සංඛ්‍යා නිරූපනය සඳහා ඇති සීමාවන් මත රඳා පවතී.

Data Type	Description
Integers	These represent numbers in an unlimited range, subject to available (virtual) memory. Examples : 23, -20
Boolean	Possible values are 'True' or 'False' Behave like the values 0 and 1 for the values 'True' and 'False' respectively. When converted to a string, the strings "False" or "True" are returned, respectively. Examples : True + 2 = 3
Bytes	Array of 8-bit bytes. Each byte is represented by integers in the range 0 <= x < 256.

Sequences

මෙමගින් සීමාවක් සහිත අනුක්‍රමයකි නිරූපනය කරයි. මෙම අනුක්‍රමයේ සෑම අගයකටම ඍන නොවන සංක්‍යාවක් මගින් ප්‍රවේශ විය හැකිය. මෙම අගයයන් index අගයයන් ලෙස හැඳින්වෙයි. මෙම වර්ගයේ අනුක්‍රමයක ඇති අගයයන් ගණන len() යන ශ්‍රිතය ක්‍රියාත්මක කිරීමෙන් ලබා ගත හැකිය.යම් අනුක්‍රමයක ඇති අගයයන් ගණන n නම් index කුලකයේ අගයයන් 0 සිට n-1 දක්වා වෙයි.මෙම ගණයට ගැනෙන දත්ත ප්‍රරූප immutable හෝ mutable විය හැක.

Set types

මෙමගින් පිළිවෙලක් රහිත සීමාවක් සහිත අගයයන් කුලකයක් නිරූපනය වෙයි. මෙම කුලකයේ අගයයන්ට ප්‍රවේශ වීම සඳහා index අගයයන් භාවිතා කළ නොහැකිය

Mappings

මෙමගින් සීමාවක් සහිත වස්තු කුලකයක් නිරූපනය වෙයි. මෙම කුලකයේ අගයයන්ට ප්‍රවේශ වීම සඳහා index දත්ත කුලකයක් යොදාගත හැකිය. a[k] යන අංකනය මගින් a දත්ත ව්‍යුහයේ k යනුවෙන් දැක්වෙන අවයවයට ප්‍රවේශ විය හැක.

<u>Numbers</u>

දශමක (Decimal) සංඛ්‍යා මෙන්ම අෂ්ඨක (Octal) හා ෂඩ් දශමක (Hexa Decimal) සංඛ්‍යා පද්ධතිවල සංඛ්‍යා ද ගණනය කිරීම් සඳහා යොදා ගත හැක.

පහත විධාන ක්‍රියාත්මක කර ලැබෙන ප්‍රතිදාන ඉදිරියෙන් ලියන්න

>>> 2+2 ……………………………………………

>>> 2 + 2 # add two numbers ……………………………………………

>>> (50-6*5)/4 ……………………………………………

>>> 8/5 ……………………………………………

>>> 8//5 ……………………………………………

>>> 3*3.75/1.5 ……………………………………………

>>> 1.0/2.0 ……………………………………………

<u>අෂ්ඨක (Octal) සංඛ්‍යා</u>

පහත විධාන ක්‍රියාත්මක කර ලැබෙන ප්‍රතිදාන ඉදිරියෙන් ලියන්න

>>> 0o13 ……………………………………………

>>> 0o13+0o13 ……………………………………………

>>> 0o13-0o5 ……………………………………………

<u>ෂඩ් දශමක (Hexa Decimal) සංඛ්‍යා</u>

පහත විධාන ක්‍රියාත්මක කර ලැබෙන ප්‍රතිදාන ඉදිරියෙන් ලියන්න

>>> 0x20 ……………………………………………

>>> 0x16 ……………………………………………

>>> 0x10+0x11 ……………………………………………

<u>Sets</u>

මෙය කුලක වර්ගයේ දත්ත ප්‍රරූපයකි. පහත උදාහරණ මගින් මෙම දත්ත ව්‍යුහය යොදා ගන්නා ආකාරය අධ්‍යයනය කරමු

```
>>> s = set()
>>> s.add(1)
>>> s.add(2)
>>> s
{1, 2}
>>> 1 in s
True
>>> 2 in s
True
>>> 3 in s
False
>>>
>>> x = set([])
>>> x = set(['sunil','gamini','nimal',5,2])
>>> y = set(['sunil','kamal'])
>>> z = x.union(y)
>>> p = x.intersection(y)
>>> x
{'sunil', 'gamini', 2, 5, 'nimal'}
>>> y
{'sunil', 'kamal'}
>>> z
{'gamini', 2, 5, 'nimal', 'sunil', 'kamal'}
>>> p
{'sunil'}
>>> x.add('saman')
>>> x
{2, 5, 'nimal', 'sunil', 'gamini', 'saman'}
>>> "saman" in x
True
>>> x.remove('saman')
>>> x
{2, 5, 'nimal', 'sunil', 'gamini'}
>>> x.clear()
>>> x
set()
```

කාරක කාණ්ඩ (Operator Categories)

- අංක ගණිතමය (Arithmetical)
- සැසඳුම් කර්ම (Comparision)
- තාර්කික (Logical)
- බිට් අනුසාරිත (Bitwise)
- ප්‍රමුඛතාව (Precedence)

Python Arithmetic Operators

a = 10 හා b = 20 නම්

Operator	Description	Example	Output
+	Addition	a + b	
-	Subtraction	a - b	
*	Multiplication	a * b	
/	Division	b / a	
%	Modulus	b % a	
**	Exponent	a**b	
//	Floor Division	9//2	
		9.0//2.0	

Python Comparison Operators

a =10 හා b = 20 නම්

Operator	Description	Example	Output
==	Equal to	(a == b)	
!=	Not equal to	(a != b)	
>	greater than	(a > b)	
<	less than	(a < b)	
>=	greater than or equal to	(a >= b)	
<=	less than or equal to	(a <= b)	

Python Logical Operators

A=10 හා B=20 නම්

Operator	Description	Example	Output
and	Logical AND	A>5 and B> 10	
or	Logical OR	A>5 or B> 10	
not	Logical NOT	not A>10	

Python Assignment Operators:

a = 10 හා b = 20 යයි සිතමු

Operator	Description	Example
=	Simple assignment operator,	c = a + b will assign value of a + b into c
+=	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-=	Subtract AND assignment operator,	c -= a is equivalent to c = c - a
*=	Multiply AND assignment operator	c *= a is equivalent to c = c * a
/=	Divide AND assignment operator,	c /= a is equivalent to c = c / a
%=	Modulus AND assignment operator	c %= a is equivalent to c = c % a
**=	Exponent AND assignment operator, Performs exponential (power) calculation.	c **= a is equivalent to c = c ** a
//=	Floor Division and assigns a value	c //= a is equivalent to c = c // a

Python Bitwise Operators:

Bitwise operator works on bits and perform bit by bit operation.

බිටු අනුසාරිත කර්මයන් මගින් බිටුවෙන් බිටුව අදාළ මෙහෙයුමට ලක් කරයි.

a = 60 හා b = 13 යැයි සිතමු

a = 00111100

b = 00001101

There are following Bitwise operators supported by Python language

<table>
<tr><th>Operator</th><th>Description</th><th>Example</th></tr>
<tr><td>&</td><td>Binary AND Operator</td><td>(a & b) will give 12 which is
0000 1100</td></tr>
<tr><td>|</td><td>Binary OR Operator</td><td>(a | b) will give 61 which is
0011 1101</td></tr>
<tr><td>^</td><td>Binary XOR Operator</td><td>(a ^ b) will give 49 which is
0011 0001</td></tr>
</table>

~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~a) will give -60 which is 1100 0011 (The bitwise inversion of *x* is) -(x + 1)
<<	Binary Left Shift Operator	a << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator	a >> 2 will give 15 which is 0000 1111

Python Identity Operators:

Identity operators compare the memory locations of two objects.

අනන්‍යතා කර්මයන් මගින් අගයයන් දෙකක් සංසන්දනය කිරීම සිදු වේ.

Operator	Example

is	>> >x = 5
	>>> y = 5
	>>> x is y
	True
	>>> x is not y
	False

is not	>>> x = 5
	>>> y = 5
	>>> x is not y
	False

Python Membership Operators:

Operator	Example
in	>>> S = ([1,2]) >>> 1 in S
not in	>>>S = ([1,2]) >>>3 not in s

Python Operators Precedence කර්ම ප්‍රමුඛතා

පහත ගණිතමය ප්‍රකාශය සලකා බලන්න.

5- 3 x 2

මෙම ප්‍රකාශය දෙයාකාරයකට සුළු කළ හැකි ය.

5-(3 x 2) = -1   (5 - 3) x 2 = 4

මෙවැනි ප්‍රකාශන විවිධ ආකාරයට සුළු කළ විට ලැබෙන ප්‍රතිඵල බොහෝ විට එකිනෙකට වෙනස් වේ. එබැවින් ප්‍රකාශයක විවිධ වර්ගයේ ගණිතමය කර්මයන් ඇතිවිට එම ප්‍රකාශය සුළු කිරීමේ දී කර්මයන් සඳහා දියයුතු ප්‍රමුඛතා අර්ථ දක්වා ඇත. මෙම ප්‍රමුඛතා 'Operator Precedence' ලෙෂ හැඳින්වෙයි.
ගණිතමය ගැටළු විසඳීමේ දී යොදාගන්නා ගණිතමය කර්මයන් හි ප්‍රමුඛතා පහත දැක්වෙන පරිදි වෙයි.

කර්මය	ප්‍රමුඛතා අනුපිළිවෙළ
(1)	( )
(2)	^
(3)	x , /
(4)	+, -

Python භාෂාවේ භාවිතා කරන කර්මයන් හි ප්‍රමුඛතා අනුපිළිවෙල පහත දැක්වෙන පරිදි වෙයි. පහක දැක්වෙන්නේ ඉහළම ප්‍රමුකතාවේ සිට පහළම ප්‍රමුඛතාව දක්වා ප්‍රමුඛතා අනුපිළිවෙල යි.

පහත වගුවේ දැක්වෙන්නේ පයිතන් භාෂාවේ කර්ම සඳහා ලබා දී ඇති ප්‍රමුඛතා අනුපිළිවෙලය. එය ඉහළම ප්‍රමුඛතාවයේ සිට පහළම පහළම ප්‍රමුඛතා දක්වා සංවිධානය කර ඇත.

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^		Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += |= &= >>= <<= *= **=	Assignment operators
is, is not	Identity operators
in , not in	Membership operators
not ,or, and	Logical operators

- ආදාන/ ප්‍රතිදාන
    - යතුරු පුවරුවෙන් ආදානය
    - සම්මත උපාංගවලට ප්‍රතිදානය

වැඩසටහන ක්‍රියාත්මක වනවිට යතුරු පුවරුවෙන් දත්ත ආදානය කර ප්‍රතිදාන ලබා ගැනීම ආදර්ශනය කෙරෙක පහත උදාහරණ අධ්‍යයනය කර එම වැඩසටහන් ක්‍රියාත්මක කරන්න.

උදාහරණ 1 - වෘත්තයක වර්ගඵලය සෙවීම

```
Radius = float(input('Enter Radius: '))
pi = 22/7
Area = pi*Radius**2
print ('Area of a Circle = ',(Area))
```

උදාහරණ 2 - සෘජුකෝණාස්‍රයක වර්ගඵලය සහ පරිමිතිය යසෙවීම

```
length = float(input('Enter the length: '))
breadth = float(input('Enter the breadth: '))
Area = length*breadth
perimeter = 2*(length+breadth)
print ('Area of the Rectangle = ',(Area))
print ('Perimeter of the Rectangle = ',(perimeter))
```

9.8 ක්‍රමලේඛ සංවර්ධනයෙහි පාලන ව්‍යුහ භාවිත කිරීම

- පාලන ව්‍යුහ (Control structures)
    - අනුක්‍රමය (Sequence)
    - තේරීම (Selection)
    - පුනරුක්තිය (Repetition)
- පුනර්කරණය (Iteration)
- ලූපනය (looping)

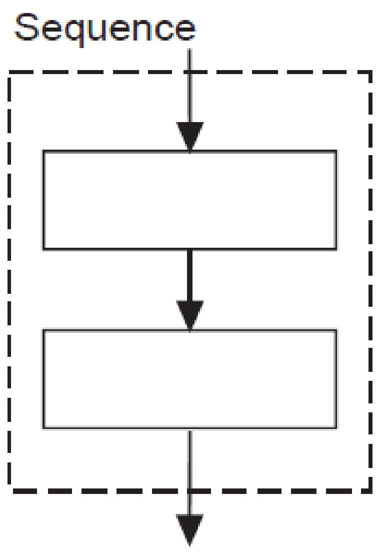
සෑම පරිගණක ක්‍රමලේඛයක ම පහත සඳහන් පාලන ව්‍යුහයන් එකක් හෝ ඊට වැඩි ගණනක් භාවිත කෙරේ.

1. අනුක්‍රමය (sequence)
2. වරණය (selection)
3. පුනර්කරණය (repetition/iteration)

පරිගණක ක්‍රමලේඛ තුළ ඇති වගන්ති ක්‍රියාත්මක කරන අනුපිළිවෙළ පාලන ව්‍යුහයන් මගින් පාලනය කෙරේ.

1. අනුක්‍රමය

ක්‍රමලේඛයක ඇති වගන්ති එකිනෙක අනුපිළිවෙළින් දී ඇති ආකාරයට ක්‍රියාත්මක කිරීම අනුක්‍රමය නමින් හඳුන්වනු ලැබේ.

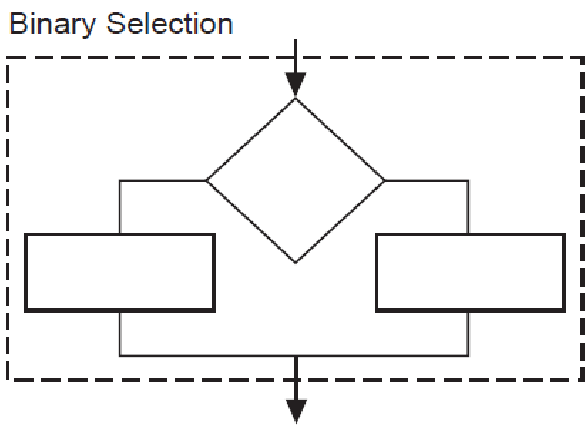


උදාහරණ

```
price=float(input("Enter Price Rs:"))
qty=int(input("Enter Quantity:"))
amount=price*qty
print ("Amount is Rs: ", amount)
```

2. වරණය

මෙහි දි තෝරා ගැනීම සඳහා විකල්ප ප්‍රකාශන සමූහයක් ඇති විට දෙන ලද කොන්දේසියක් පරීක්ෂා කර එක් විකල්පයක් පමණක් තෝරා ගැනීම සිදුවේ



උදාහරණ 1 - Odd numbers and Even Numbers

```
Number = int (input("Enter No: "))
if Number % 2 == 1:
    Result= "Odd Number"
else:
    Result= "Even Number"
print ("Number is:", Result)
```

උදාහරණ 2 - To find the grade of the student

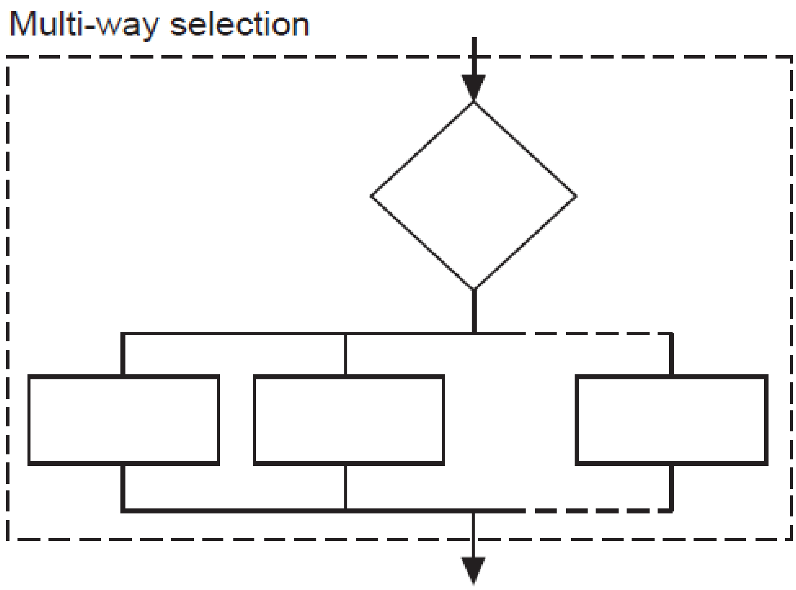
```
Marks = int(input("Enter Marks :"))
if Marks > 100 or Marks < 0:
    Grade ="Invalide Marks"
elif Marks >= 75:
    Grade ="Grade = A"
elif Marks >= 65:
    Grade ="Grade = B"
elif Marks >= 50:
    Grade ="Grade = C"
elif Marks >= 35:
    Grade ="Grade = S"
else:
    Grade ="Grade = W"
print("Student Grade is : ",Grade)
```

උදාහරණ 3 - To check the state of water

```
Temp=float(input('Temperature is in Celcius : '))
if Temp >= 100:
    Result = 'Steam'
elif Temp >= 27:
    Result ='Hot Water'
elif Temp>  0:
    Result ='Cool Water'
else:
    Result = 'Ice'
print ('Status of Water is: ',Result)
```

උදාහරණ 4 To find the maximum number of three numbers you input

```
number1=int(input("Nimber 1 is = "))
number2=int(input("Number 2 is = "))
number3=int(input("Number 3 is = "))
if number1>number2 and number1>number3:
    max_number=number1
elif number2>number3:
    max_number=number2
else:
    max_number=number3
print("Max Number is =",max_number)
```



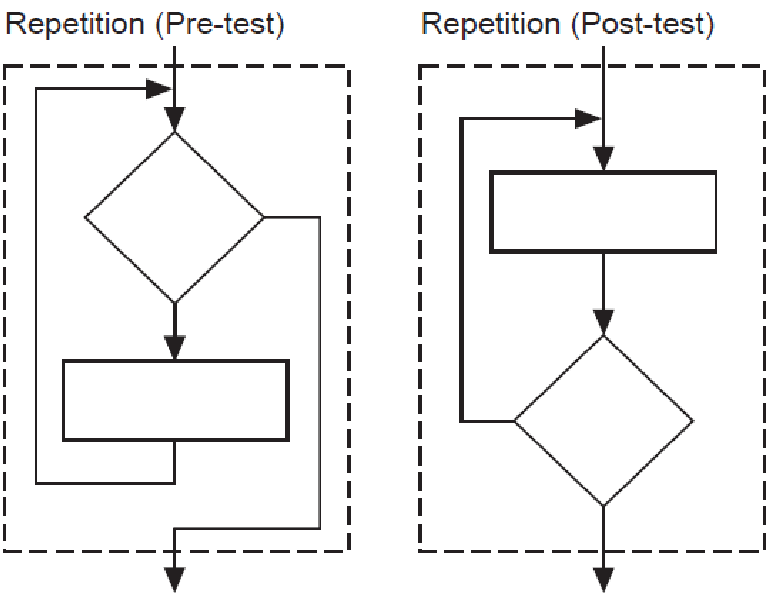
උදාහරණ 5

```
num1=int(input("Enter first number:"))
num2=int(input("Enter second number:"))
print ("1.  Add")
print ("2. Subtract")
print ("3. Multiply")
print ("4. Divide")
choice=int(input("Enter Choice:"))
if choice==1:
    print (num1, "+" , num2, "=", num1+num2)
elif choice==2:
    print (num1, "-" , num2, "=", num1-num2)
elif choice==3:
```

```
    print (num1, "*" , num2, "=",num1*num2)
elif choice==4:
    print (num1, "/" , num2, "=",num1/num2)
else:
    print ("Invalid Entry!")
```

3. පුනර්කරණය

වගන්ති එකක් හෝ කිහිපයක් නැවත නැවත ක්‍රියාත්මක කිරීම පුනර්කරණය ලෙෂ හැඳින්වේ.



a) for ප්‍රකාශනය

වගන්ති එකක් හෝ වැඩිගණනක් හෝ යම් නිශ්චිත වාර ගණනක් පුනර්කරණය කිරීම සඳහා මෙම ප්‍රකාශනය භාවිත කෙරේ. මෙය යොදාගත හැක්කේ අවස්ථා දෙකක දී පමණි.

i) යම් සංඛ්‍යා පරාසයක් තුළ පුනර්කරණය සිදු කිරීම.

කාරක රීති

```
for var-name in range (start-num,stop-num)
```

ii) දත්ත ලැයිස්තුවක් සමග භාවිත කිරීම.

කාරක රීති

```
for list-item in list
statement(s)
```

උදාහරණ 1

```
for num in range(1, 5):
    print (num)
```

මෙම උදාහරණය 1 සිට 4 දක්වා සංඛ්‍යා ප්‍රතිදානය කරයි. මෙම උදාහරණයේ ඇති range යන්න මගින්ලබාදෙන ලද ආරම්භක අගය වන 1 සිට අවසන් අගය වන 5 දක්වා සංඛ්‍යා සමූහයක් ජනනය කරනු ලැබේ. (මෙහි අවසන් අගය වන 5 ඇතුළත් නොවේ.)

උදාහරණ 2

```
for i in range (3,8):
    print(i)
```

උදාහරණ 3

```
for i in range (8,3,-1):
    print(i)
```

උදාහරණ 4

```
numbers=[10,30,40,60,50]
for num in numbers:
    print (num)
```

ඉහත උදාහරණය for ප්‍රකාශනය දත්ත ලැයිස්තුවක් සමග භාවිත කෙරේ. මෙහි දී මුල දී සංඛ්‍යා ලැයිස්තුව යොදා ගනු ලැබේ.

උදාහරණ 5

```
for letter in "Computer":
    print (letter)
```

b) while ප්‍රකාශනය

මෙම ප්‍රකාශනය සඳහා ලබා දෙන කොන්දේසියක් සත්‍යව පවතින තෙක් පුනර්කරණය සිදුවේ.

උදාහරණ 1

```
i = 1
while i <=10:
    print (i)
    i = i+1
```

උදාහරණ 2

```
count = 0
num = int (input ("Enter Number" ))
while num>0:
```

```
    if num%2==0:
        count += 1
    num = int (input ("Enter Number" ))
print ("Total Even numbers", count)
```

ඉහත උදාහරණයේ දී මුලින් ම ලබා ගන්නා සංඛ්‍යාව ධන සංඛ්‍යාවක් නම් num>0 කොන්දේසිය සත්‍ය වේ. එසේ නම් while ප්‍රකාශනය තුළට පැමිණේ. එහි දී එම සංඛ්‍යාව ඉරට්ටේ සංඛ්‍යාවක් දැයි පරීක්ෂා කෙරේ. එසේ නම් count හි අගය එකකින් වැඩි කෙරේ. ඉන්පසු නැවත සංඛ්‍යාවක් ලබා ගැනේ. ලබා දී ඇති කොන්දේසිය (num>0) අසත්‍ය වන තාක් පුනර්කරණය සිදුකෙරේ. ලබාදුන් මුළු ඉරට්ටේ සංඛ්‍යා ගණන මෙම ක්‍රමලේඛය මගින් ප්‍රතිදානය කෙරේ.

උදාහරණ 3

Prgram to input ten numbers and to find the maximum number

```
L= int(input("First Number is = "))
Max_No = L
Count = 1
while Count < 10:
    S=int(input("Next Number is = "))
    if Max_No < S:
        Max_No = S
    Count = Count + 1
print("Maximum Number is = ",Max_No)
```

උදාහරණ 3

To find the Maximum Number

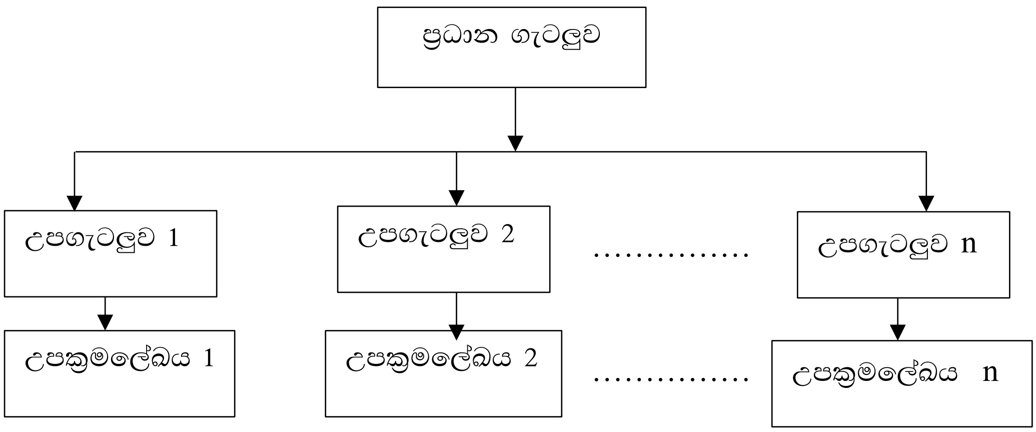
```
First_No= int(input("First Number is = "))
Max_No = First_No
Count = 1
while Count < 10:
    Next_No=int(input("Next Number is = "))
    if Next_No > Max_No:
        Max_No = Next_No
    Count = Count + 1
    print("Maximum Number is = ",Max_No)
```

9.9 ක්‍රමලේඛනයේ දී උපක්‍රම ලේඛ (subprograms)භාවිත කිරීම.

- උප ක්‍රමලේඛ ප්‍රරූප
    - තුළබැඳි / තිළැලි (Built in)
    - පරිශීලක නිර්වාචිත (User defined)
- ව්‍යුහය (Structure)
- පරාමිති යැවීම (Parameter Passing)
- ප්‍රත්‍යාගමන අගය (Return Values)
- පෙර නිමි අගය (Default values)
- විචල්‍ය පරාසය (Scope of variables)

උපක්‍රමලේඛ භාවිතය

ක්‍රමලේඛයක ක්‍රියාවලි ප්‍රමාණය වැඩිවීමත් සමග ම එය සංකීර්ණ වන නිසා කියවා තේරුම් ගැනීම, ක්‍රමලේඛය නඩත්තු කිරීම වැනි කටයුතු අපහසු වේ. එබැවින් ක්‍රමලේඛ ලිවීමේ දී හැකිතාක් දුරට උපක්‍රමලේඛ ලෙස ලිවීම සාර්ථක ක්‍රමයකි



උප ක්‍රමලේඛ ප්‍රරූප

තුළබැඳි / තිළැලි (Built in)

Function	Description
bin()	Returns the binary version of a number
bool()	Returns the Boolean value of the specified object
float()	Returns a floating point number
help()	Executes the built-in help system
hex()	Converts a number into a hexadecimal value
input()	Allowing user input
int()	Returns an integer number

len()	Returns the length of an object
list()	Returns a list
max()	Returns the largest item in an iterable
min()	Returns the smallest item in an iterable
object()	Returns a new object
print()	Prints to the standard output device
range()	Returns a sequence of numbers, starting from 0 and increments by 1 (by default)
round()	Rounds a numbers
str()	Returns a string object
tuple()	Returns a tuple
type()	Returns the type of an object

පරිශීලක නිර්වාචිත ශ්‍රිත (User defined functions)

ශ්‍රිතයක් නිර්මාණය

```
def my_function():
  print("Hello from a function")
  print(“Python is easy to learn")
  print(“I’m learning Python functions")
  print(“Learning Python is interesting")
```

ශ්‍රිතයක් කැඳවීම

```
my_function()
```

පරාමිති යැවීම (Parameter Passing)

```
def my_function(text_value):
  print(text_value + "National School")

my_function("Mahasen")
my_function("Wijayaba")
my_function("Panduwasnuwara")
```

ප්‍රත්‍යාගමන අගය (Return Values)

```
def return_sum(x,y):
    c = x + y
    return c
res = return_sum(4,5)
print(res)
```

පෙර නිමි අගය (Default values)

උදාහරණය 1

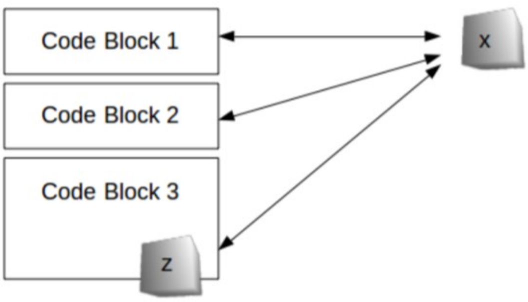
```
def my_function(country = "Sri Lanka"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

උදාහරණය 2

```
def Hello(name="everybody"):
    """ Greets a person """
    print("Hello " + name + "!")
Hello("Peter")
Hello()
```

විචල්‍ය පරාසය (Scope of variables)

ස්ථානීය විචල්‍ය හා ගෝලීය විචල්‍ය (Local variables and global variables)



ස්ථානීය විචල්‍ය (Local variables)

උදාහරණය 1

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


varscope()
```

උදාහරණය 2

```
def sum_test(x,y):
    sum = x + y
    return sum


print(sum_test(8,6))
```

ගෝලීය විචල්‍ය (global variables)

උදාහරණය 1

```
z = 10
def afunction():
    global z
    print(z)

afunction()
print(z)
```

උදාහරණය 2

```
z = 10

def afunction():

    global z

    z = 9

afunction()

print(z)
```

අභ්‍යාස

වෘත්තයක හා සෘජුකෝණාස්‍රයක වර්ගඵලය සෙවීම සඳහා පයිතන් ශ්‍රිත යොදාගෙන ක්‍රමලේඛයක් ගොඩනගන්න

```
def Areaofcircle():
    r = float(input('enter radius:'))
    area1 = (22/7)*r**2
    print('area is',area1)
def Areaofrect():
    L = float(input('enter Length:'))
    B = float(input('enter Breadth:'))
    area2 = L*B
    print('area is',area2)

print ("1. circle")
print ("2. rectangle")

choice = int(input("Enter your choice:"))

if choice == 1:
    Areaofcircle()

if choice == 2:
    Areaofrect()
```

9. 10 ක්‍රමලවල දී දත්ත ව්‍යුහ යොදා ගැනීම

- දත්ත ව්‍යුහ
    - Strings
    - Lists
    - Tuples
    - Dictionaries

Strings - තන්තු

අක්ෂර සංයෝජන නිරූපනය සඳහා මෙම දත්ත ප්‍රරූපය යොදා ගනී

Python භාෂාවේ String දත්ත ප්‍රරූපය යොදා ගන්නා ආකාරය අධ්‍යයනය සඳහා පහත විධාන ක්‍රියාත්මක කර ලැබෙන ප්‍රතිදාන ඉදිරියෙන් ලියන්න

උදාහරණ 1

>>> 'This is a string' ……………………………………………………………….

>>> "This is a string" ……………………………………………………………….

>>> "i\'m learning python" ……………………………………………………………….

>>> 'This text has\
two lines' ……………………………………………………………….

>>> "123"+"456" ……………………………………………………………….

>>>print("Hello"+ "Friends") ……………………………………………………………….

>>> print("_"* 20) ……………………………………………………………….

>>> print("*" * 20) ……………………………………………………………….

උදාහරණ 2

To count the length of a string

```
while True:
    x=input("Enter a String : ")
    char_count = len(x)       # compute the length
    print ("String; " ,x, " length : ", char_count)
```

Tuples

Tuple යනු එක් වරක දී එක් දත්තය බැගින් මුදා හැරිය හැකි දත්ත ව්‍යුහයකි. මෙවැනි වස්තුවක් (Object) තුළ එහි අවයව වෙන වෙනම හඳුනාගැනීම සඳහා දර්ශකයක් (index) පදනම් කර ගනී

උදාහරණ

a = (10,20,30,40,50,60)

Index	0	1	2	3	4	5
Value	10	20	30	40	50	60

මෙහි දර්ශකය 0 වන ස්ථානයේ සඳහන් අගය වන්නේ 10 ය. එමෙන්ම Tuple යනු immutable ගණයට අයත් දත්ත ව්‍යුහයකි. එනම් එක් වරක දී සකසන ලද Tuple එකක් සඳහා නැවත අවයව අළුතින් ඇතුළත් කළ නොහැකි වීම යි.

පහත විධාන ක්‍රියාත්මක කර ලැබෙන ප්‍රතිදාන ඉදිරියෙන් ලියන්න

```
>>> a = (1,2,3,4,5,6)
>>> a                  ………………………………………….
>>> print(a[0])        ………………………………………….
>>> b = (2,15,81,96,75,85,63,85,26)
>>> b                  ………………………………………….
>>> b[0]               ………………………………………….
>>> b[ 1 ]             ………………………………………….
>>> b[2]               ………………………………………….
>>> b[6]               ………………………………………….
>>> b[7]               ………………………………………….
>>> b[8]               ………………………………………….
>>> b[0:3]             ………………………………………….
>>> b[1:5]             ………………………………………….
x = (1,10.2, "abc")
print(x[2])
```

Lists

List එකක් තුළ විවිධ වර්ගයේ දත්ත තිබිය හැක.

උදාහරණ

a = ['Birds','Eggs',100, 1234]

පහත විධාන ක්‍රියාත්මක කර ලැබෙන ප්‍රතිදාන ඉදිරියෙන් ලියන්න

```
>>> numbers =[2,4,6,8,10,12,14,16,18,20]
>>> print(numbers[0:3])     ………………………………………………
>>> print(numbers[0:])      ………………………………………………
>>> print(numbers[1:])      ……………………………………………….
>>> print(numbers[5:])      …………………………………………………
>>> print(numbers[:1])      …………………………………………………
>>> print(numbers[:4])      …………………………………………………
>>> len(numbers)            …………………………………………………
```

```
>>>  x = [1,10.2,"abc"]
>>>  print(x[2])
>>> a = ['abc', (1, 5.6, 'cde'), 100, 10.57, 6, 'abc', 6, 8]
>>> a[1][2]
```

List Methods

1. Append method

මෙම ක්‍රමය භාවිතා කරන්නේ list එකකට අළුතින් දත්තයක් අන්තර්ගත කිරීම සඳහා ය.

උදාහරණ

```
>>> lst = [1,2,3]
>>> lst
[1, 2, 3]
>>> lst.append(4)
>>> lst
[1, 2, 3, 4]
>>>
```

2. Count method

ලැයිස්තුවක ඇති අවයව ගණනය කිරීම සඳහා යොදා ගනී

```
>>> letters = list('pahanma')
>>> letters
['p', 'a', 'h', 'a', 'n', 'm', 'a']
>>> letters.count('a')
3
```

3. Adding sequence

```
>>> [1,2,3]+[4,5,6]
[1, 2, 3, 4, 5, 6]
>>> ['Mahasen']+['National']+['School']
 ['Mahasen', 'National', 'School']
```

Program to find the minimum number of a list

```
x = [10,12,4,5,3]
min  = x[0]
j = 1
while (j< len(x)):
```

```
    if (x[j]< min):
        min = x[j]
    j = j+1
print(min)
```

Program to find the maximum number of a list

```
x = [10,12,4,5,3,25,18,6]
max  = x[0]
j = 1
while (j< len(x)):
    if not(x[j]< max):
        max = x[j]
    j = j+1
print(max)
```

To find the index of the maximum Number

```
x = [10,12,4,5,3,15,56]
index = 0
min1 = x[0]
j = 0
while (j < len(x)):
    if not(x[j]< min1):
        min1 = x[j]
        index = j
    j = j+1
print(index)
```

To find the index of the minimum Number

```
x = [10,12,4,5,3,15,56,1]
index = 0
min1 = x[0]
j = 0
while (j < len(x)):
    if (x[j]< min1):
        min1 = x[j]
        index = j
```

```
    j = j+1
print(index)
```

To create a list of Numbers

```
lt = []
x = input("Enter a Number : ")
while (x != '0'):
    lt = lt+[x]
    x = input("Enter a Number : ")
print(lt);
```

To create a list of Integers

```
lt = []
x = int(input("Enter a Number : "))
while (x != 0):
    lt = lt+[x]
    x = int(input("Enter a Number : "))
print(lt);
```

Dictionary

Dictionary එකක් තුළ දත්ත යුගලයක් පවතී. ඒවා Keys හා Values ලෙස හඳුන්වනු ලබයි.

උදාහරණ 1

```
>>> dict = {'Name': 'Zara','Age':'7','Class':'First'}
>>> print(dict['Name'])
Zara
>>> print(dict['Class'])
First
>>> dict['Age'] = 8
>>> dict
{'Age': 8, 'Name': 'Zara', 'Class': 'First'}
>>> dict['School']= "Mahasen"
>>> dict
{'Age': 8, 'Name': 'Zara', 'School': 'Mahasen', 'Class': 'First'}
 >>>
```

උදාහරණ 2

එකම key එකක් සඳහා Values (වටිනාකම්) දෙකක් පැවරීම සිදු කළ විට

```
>>> dict = {'Name': 'Zara','Age':7,'Class':'First'}
>>> dict
{'Age': 7, 'Name': 'Zara', 'Class': 'First'}
>>> dict = {'Name': 'Zara','Age':7,'Class':'First', 'Name':'Malithi'}
>>> print(dict['Name'])
Malithi
```

උදාහරණ 3

```
>>> x = {'a' : 1 , 'b' :2}
>>> for i in x:
        print(x[i])
```

උදාහරණ 4

```
>>> x = {1:10,"abc":10.5,"c":"nimal"}
>>> print(x["c"])
```

9.11 ක්‍රමලවල දී ගොනු සහ දත්ත සමුදාය හැසිරවීම

- ගොනු හැසිරවීම
    - මූලික ගොනු මෙහෙයු

උදාහරණය 1

```
file = open("H:testfile.txt","w")

file.write("Hello World")
file.write("This is our new text file")
file.write("and this is another line.")
file.write("Why? Because we can.")

file.close()
```

උදාහරණය 2

```
f = open("D:/A/b.text","a")
data=input("Enter data:")
while (data!=""):
   f.write(data+"\n")
   data=input("Enter data:")
f.close()
```

උදාහරණය 3

```
f = open("D:/A/b.text","r")
while True:
   line = f.readline().strip()
   if not line:
      break
   print(line)
f.close()
```

උදාහරණය 4

```
# Program name : example1.py
f = open('D:/A/b.text', 'w')
dataitems = ("Name","Age","Sex","Telephone")
datavalues = ["","","",""]
recordcount = 1

def readData():
   global recordcount
   i = 0
   print('Getting data for record :'+ str(recordcount))
   for value in dataitems:
      datavalues[i] = input(value + ": ")
      i = i + 1
   recordcount +=  1
   print()

def writeData(Name,Age,Telephone,Sex='M'):
   f.write(Name+','+
         Age+','+
         Telephone+','
         +Sex+
         '\n')

datavalues[0] = 'dummay name'
while datavalues[0]:
   readData()
   if(datavalues[0]):
      writeData(datavalues[0],datavalues[1],datavalues[3],datavalues[2])

f.close()
```

9.12 දත්ත සමුදායක දත්ත කළමනාකරණය කිරීම

- දත්ත සමුදායට සම්බන්ධ වීම.
- දත්ත සමුද්ධරණය (Retrieve data)
- දත්ත එක් කිරීම (add)
- නවීකරණය (modify) සහ මැකීම (delete)

දත්ත සමුදායට සම්බන්ධ වීම.

```
import pymysql
conn = pymysql.connect(host="localhost", user="root", password="", db="course_selection")
```

දත්ත පාදකයක දත්ත වගුවක් ගොඩනැගීම

```
import pymysql
conn = pymysql.connect (host="localhost", user="root", password="",
db="course_selection")
myCursor = conn.cursor()

myCursor.execute ("""CREATE TABLE foundation_course
(
stu_id int primary key,
stu_name varchar(40),
test_mark int,
center varchar (20)
) """)
conn.commit()
conn.close()
```

දත්ත වගුවකට දත්ත ආදානය

```
import pymysql
conn = pymysql.connect (host="localhost", user="root", password="",
db="course_selection")
myCursor = conn.cursor()

myCursor.execute(" INSERT INTO foundation_course (stu_id, stu_name, test_mark,
center) VALUES (182, 'Rashen Gimhana ', 55,'Kurunegala');")
print("> Data Inserted!!!! ")
```

```
conn.commit()
conn.close()
```

දත්ත සමුද්ධරණය (Retrieve data)

```
import pymysql
conn = pymysql.connect (host="localhost", user="root", password="",
db="course_selection")

myCursor = conn.cursor()

myCursor.execute("SELECT * FROM foundation_course;")
print(myCursor.fetchall(),end = "\n")
conn.commit()
conn.close()
```

දත්ත එක් කිරීම (add)

```
import pymysql
conn = pymysql.connect (host="localhost", user="root", password="",
db="course_selection")

myCursor = conn.cursor()

myCursor.execute(" INSERT INTO foundation_course (stu_id, stu_name, test_mark,
center) VALUES (182, 'Rashen Gimhana ', 55,'Kurunegala');")

print("> Data Inserted!!!! ")

conn.commit()
conn.close()
```

නවීකරණය (modify) සහ මැකීම (delete)

```
import pymysql
conn = pymysql.connect (host="localhost", user="root", password="",
db="course_selection")

myCursor = conn.cursor()

myCursor.execute("UPDATE foundation_course SET stu_name='Sunanda Kumara'
WHERE stu_id = 128;")
conn.commit()
conn.close()
```

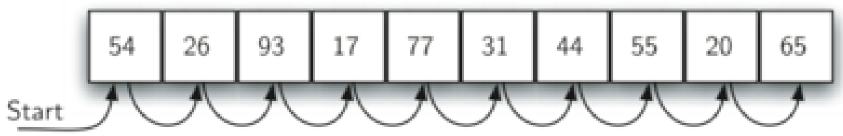
9.13 දත්ත සෙවීම සහ තේරීම (Searching and sorting)

- සෙවුම් ශිල්ප ක්‍රම
  - අනුක්‍රමික සෙවුම (Sequential search)

- තේරුම් ශිල්ප ක්‍රම
    - බුබුළු තේරීම/ යා සැසඳුම් තේරීම් (Bubble sort)

සෙවුම් ශිල්ප ක්‍රම

අනුගාමික සෙවීම (The Sequential Search)



උදාහරණය 1

```
def sequentialSearch(alist, item):

    pos = 0
    found = False
    while pos < len(alist) and not found:
        if alist[pos] == item:
            found = True
        else:
            pos = pos+1
    return found
testlist = [1, 2, 32, 8, 17, 19, 42, 13, 0]
print(sequentialSearch(testlist, 3))
print(sequentialSearch(testlist, 13))
```

උදාහරණය 2

```
def orderedSequentialSearch(alist, item):
    pos = 0
    found = False
    stop = False
    while pos < len(alist) and not found and not stop:
        if alist[pos] == item:
            found = True
        else:
            if alist[pos] > item:
                stop = True
            else:
                pos = pos+1
    return found
testlist = [0, 1, 2, 8, 13, 17, 19, 32, 42,]
print(orderedSequentialSearch(testlist, 3))
print(orderedSequentialSearch(testlist, 13))
```

තේරීමේ ශිල්පක්‍රම (Sorting Techniques)

බුබුළු තේරීම (Bubble Sort)

උදාහරණය 1

```
def bubbleSort(alist):
    for passnum in range(len(alist)-1,0,-1):
        for i in range(passnum):
            if alist[i]>alist[i+1]:
                temp = alist[i]
                alist[i] = alist[i+1]
                alist[i+1] = temp
alist = [54,26,93,17,77,31,44,55,20]
bubbleSort(alist)
print(alist)
```

උදාහරණය 2

```
def shortBubbleSort(alist):
    exchanges = True
    passnum = len(alist)-1
    while passnum > 0 and exchanges:
       exchanges = False
       for i in range(passnum):
           if alist[i]>alist[i+1]:
               exchanges = True
               temp = alist[i]
               alist[i] = alist[i+1]
               alist[i+1] = temp
       passnum = passnum-1
alist = [20,30,40,90,50,60,70,80,100,110]
shortBubbleSort(alist)
print(alist)
```