පයිතන් භාෂාව

අ. පො . ස උසස් පෙළ තොරතුරු හා සන්නිවේදන තාක්ෂණය (නව නිර්දේශය)

මෙය අ. පො . ස උසස් පෙළ තොරතුරු හා සන්නිවේදන තාක්ෂණ විෂයය ප්‍රතිඵල ඉහල නැංවීම සඳහා මිනුවන්ගොඩ කලාප අධ්‍යාපන කාර්යාලය විසින් මුද්‍රණය කර නොමිලේ බෙදා හරින ලද ප්‍රකාශනයකි. අනවසරයෙන් උපුටා ගැනීම හෝ මුදලට විකිණීම සපුරා තහනම්.

අ. පො . ස උසස් පෙළ තොරතුරු හා සන්නිවේදන තාක්ෂණ විෂයය ගුරුබවතුන්ගේ විෂය නිර්දේශිත මඟ පෙන්වීම ඔපවත් කිරීමේ මෙහෙවරෙහි තවත් උදාර කාර්යයක ප්‍රතිඵලයක් ලෙස සම්පාදනය වූ මෙම පුස්තකය ඔබ වෙත ගෞරවයෙන් පිරිනමමි.

ඒ වජිර රණරාජා පෙරේරා
කලාප අධ්‍යාපන අධ්‍යක්ෂ
කලාප අධ්‍යාපන කාර්යාලය
මිනුවන්ගොඩ.

අ. පො . ස උසස් පෙළ තොරතුරු හා සන්නිවේදන තාක්ෂණ විෂයය ප්‍රතිඵල ඉහල නැංවීම සඳහා සැම විටම මහත් වූ සංවේදීත්වයෙන් කටයුතු මෙහෙයවන අප කලාප අධ්‍යාපන අධ්‍යක්ෂතුමන්ගේ උපදේශනය මෙම පොත සම්පාදනය කිරීම සඳහා මහත් සවියක් වූ අතර අ. පො . ස උසස් පෙළ තොරතුරු හා සන්නිවේදන තාක්ෂණ විෂයය හදාරණ සිසුන්ගේ ප්‍රතිඵල ඉහල නැංවීමට මෙම පොත ඔබට මහත් ශක්තියක් වේවායි පතමි.

ආර්. ඩි. එච්. මෙන්ඩිස්
සහකාර කලාප අධ්‍යාපන අධ්‍යක්ෂ
(තොරතුරු තාක්ෂණ)
කලාප අධ්‍යාපන කාර්යාලය
මිනුවන්ගොඩ.

ගරු කලාප අධ්‍යාපන අධ්‍යක්ෂතුමන්ගේ උපදේශනයෙන්ද, ගරු සහකාර කලාප අධ්‍යාපන අධ්‍යක්ෂතුමන්ගේ (තොරතුරු තාක්ෂණ) මඟපෙන්වීමද ලද උද්දීපනය මගින් මෙම පොත සම්පාදනය කල අතර අ. පො . ස උසස් පෙළ තොරතුරු හා සන්නිවේදන තාක්ෂණ විෂයය හදාරණ සිසුන්ගේ ප්‍රතිඵල ඉහල නැංවීමට මෙම පොත ඔබට මහත් ශක්තියක් වේවායි පතමි.

තරිඳු රන්දිම බෝපිටිය
පළාත් සම්පත් දායක (DSIS),
පරිගණක උපදේශක
කලාපීය පරිගණක සම්පත්
මධ්‍යස්ථානය - උඩුගම්පොල.

පටුන

පරිගණක ක්‍රමලේඛණ භාෂාවක් ඉගනගැනීමේදී දැනගත යුතු පොදු කොටස් පිළිබඳ කෙටි හැඳින්වීමක්. 5
මුර පද (Reserved words / Key words) 5
විවරණ (**comments)** 5
විචල්‍ය**.** (Variables) 5
හඳුන්වන හෙවත් විචල්‍යන්ට නම් ලබාදීම**.** (Identifiers / Variable names) 5
දත්ත ප්‍රරූප. (Data Types) 5
නියත**.**(Constants) 5
මෙහෙයවන පිළිබඳ.(Operators) 6
ගැලීම් පාලන ව්‍යුහයන්.(Control Flow Statements) 6
- අනුක්‍රමය **(**Sequence) 6
- වර්ණය (Selection**)** 7
- සරල වර්ණයක ව්‍යුහය(Simple selection) 7
- බහුවිධ වර්ණයක ව්‍යුහය(Multiple selection) 8
- පුනර්කරණය**(** Iteration) 8

පයිතන් භාෂාවේ මුර පද 10
විවරණ 10
තනි පද පේළි විවරණ 10
බහු පද පේළි විවරණ 11
හඳුන්වන හෙවත් විචල්‍ය නාම 11
පයිතන් භාෂාව තුළ "හඳුන්වන හෙවත් විචල්‍ය නාම" මගින් විචල්‍යන් නිර්මාණය කිරීම. 11
විචල්‍ය නිර්මාණය කිරීමට අදාල උදාහරණ කිහිපයක්. 12
පයිතන් දත්ත ප්‍රරූප හෙවත් දත්ත වර්ග. 13
Numbers 13
Integer 13
Floating point 14
Complex 14
Sequences 14
String 14
List 15
Tuple 15

SET 16
Dictionary 16
None 17
වෙනස් වන සහ වෙනස් නොවන දත්ත ප්‍රරූප (Mutable vs Immutable). 18
වෙනස් වන දත්ත ප්‍රරූප (Mutable)**.** 18
වෙනස් නොවන දත්ත ප්‍රරූප (Immutable). 18
පයිතන් භාෂාවේ නියත. 19
පයිතන් භාෂාවේ ආදාන හා ප්‍රතිදාන ක්‍රියාත්මක කිරීම. 20
ආදානය සඳහා 20
ප්‍රතිදානය සඳහා 20
ක්‍රමලේඛන ව්‍යුහය 21
මෙහෙයවන (Operators) 21
ගණිතමය මෙහෙයවන (Python Arithmetic Operators) 21
සංසන්දනාත්මක මෙහෙයවන (Python Comparison Operators) 23
පැවරුම් මෙහෙයවන (Python Assignment Operators) 25
තර්කානුකූල මෙහෙයවන (Python Logical Operators) 26
බිටුමය මෙහෙයවන. (Python Bitwise Operators) 27
සාමාජිකත්ව මෙහෙයවන (Python Membership Operators)(උ.පෙ විෂය නිර්දේශය තුළ අන්තර්ගත නොවේ. අමතර දැනුම සඳහා ඉදිරිපත් කර ඇත) 28
හැඳුනුම් මෙහෙයවන (Python Identity Operators) (උ.පෙ විෂය නිර්දේශය තුළ අන්තර්ගත නොවේ. අමතර දැනුම සඳහා ඉදිරිපත් කර ඇත) 28
මෙහෙයවන ප්‍රමුකතාවය (Precedence Rules) 29
පයිතන් භාෂාව තුළ ගැලීම් පාලන ව්‍යුහ පිහිටවීම 30
තේරීම ව්‍යුහය සඳහා.......... 30
if ව්‍යුහය 30
if – else ව්‍යුහය. 31
if – elif ව්‍යුහය. 31
පුනර්කරණය ව්‍යුහය සඳහා 32
while ව්‍යුහය 32
for ව්‍යුහය 32
ශ්‍රිතයන්.(Functions) 34
අවශ්‍යතා මත ශ්‍රිතයන් නිර්මාණය කර ගැනීම. 35
පරාමිතීන් රහිත ශ්‍රිත. 35

අනිවාර්ය පරාමිතික ශ්‍රිත (Required Arguments). 36
මුරපද පරාමිතික ශ්‍රිත (Keyword Arguments)...36
මූලික පරාමිතික ශ්‍රිත (Default Arguments). 37
පරාමිතක අගයන් නැතිව කැඳවා ඇත. 37
ශ්‍රිතයන් තුලින් ප්‍රතිදානයන් ලබා ගැනීම...37
print() නිර්මිත ශ්‍රිතය භාවිතයෙන්. 37
return () නිර්මිත ශ්‍රිතය භාවිතයෙන්...38
ශ්‍රිතයන් තුළ විචල්‍ය 38
ගෝලීය විචල්‍ය 39
ගොණු සමඟ මෙහෙයුම් සිදු කිරීම...40
read නම් ගොණුව විවර කර එහි දත්ත කියවීම...40
read නම් ගොණුව තුළ දත්ත ඉවත් කර අලුත් දත්ත ඇතුළු කිරීම. 41
read නම් ගොනුවේ අවසානයට දත්ත එකතු කිරීම. 41
rstrip()/lstrip()/strip() / split(" ")ශ්‍රිතයන්හි හැසිරීම. 41

උපුටා ගැනීම්

වාසනා ප්‍රකාශන විසින් මුද්‍රිත, තරිඳු රන්දිම බෝපිටිය විසින් රචිත පයිතන් භාෂාව පොත සහ අන්තර්ජාලය ඇසුරෙන්.

අදහස් සහ යෝජනා

- 0702282504
- bopitiyaya@gmail.com
- eduplanminu@gmail.com

පරිගණක ක්‍රමලේඛණ භාෂාවක් අධ්‍යනයේදී දැනගත යුතු පොදු කොටස් පිළිබඳ කෙටි හැඳින්වීමක්.

මුර පද (Reserved words / Key words)

ක්‍රමලේඛයන භාෂාවකට අනන්‍ය වූ වචන මුරපද ලෙස හඳුන්වයි. එම එක් එක් වචන සඳහා අදාල පරිගණක ක්‍රමලේඛන භාෂාව හඳුනන ක්‍රියාකාරකමක් ඇත. එසේම එම වචන එම ක්‍රමලේකන භාෂාව භාවිතයෙන් ක්‍රමලේඛන ගොඩනැගීමේදී වෙනත් කිසිවක් හැඳින්වීමට භාවිතා කල නොහැක.

විවරණ (comments)

යම් ක්‍රම ලේඛනයක් නිර්මාණයේ දී විශේෂිත කරුණු හෝ යම් පැහැදිලි කිරීම් අන්තර් ගත කිරීම විවරණ මගින් සිදුකරයි . මෙම කරුණු අන්තර් ගත කිරීම ක්‍රම ලේඛනයේ ඕනෑම ස්ථානයක සිදු කල හැකි අතර එය පරිගණකය විධානයක් ලෙස සලකනු නොලැබේ.

විචල්‍ය. (Variables)

විචල්‍යක් යනු ක්‍රමලේඛයක් තුළ භාවිතා වන දත්ත රඳවා ගැනීමට වෙන් කරණු ලබන ප්‍රධාන මතකයේ කොටසකි. මෙම මතක කොටස ක්‍රමලේඛනය ක්‍රියාත්මක වන අවස්ථාවේ පමණක් නිර්මාණය වන තාවකාලික එකක් වන අතර ඒ තුළ ගබඩා කරගනු ලබන දත්තයද නිරන්තරයෙන් වෙනස් වේ.

හඳුන්වන හෙවත් විචල්‍යන්ට නම් ලබාදීම. (Identifiers / Variable names)

ක්‍රමලේඛයක් තුළ විචල්‍යන් හඳුන්වා දීමට හෙවත් විචල්‍ය සඳහා ලබා දෙන නම් හඳුන්වන ලෙස කියනු ලැබේ. එක් එක් ක්‍රමලේඛන භාෂාවන්ට අදාලව මෙම විචල්‍ය නාම හෙවත් හඳුන්වනයන් නම් කිරීමේදී අනුගනමය කල යුතු නීති රාමුවක් පවතී.

දත්ත ප්‍රරූප. (Data Types)

ක්‍රමලේඛනයක් තුළ නිර්මාණය කරනු ලබන විචල්‍යන් තුලට ඇතුලත් කරනු ලබන දත්ත ඒවායේ ස්භාවය මත මූලිකව කාණ්ඩ කරණු ලැබේ. එම කාණ්ඩ ගතකිරීමේ ස්භාවය එක් එක් ක්‍රමලේඛණ භාෂාවට අනුරූපීව සුළු වශයෙන් වෙනස් වේ. මෙසේ දත්ත ඒවායේ ස්භාවය මත කාණ්ඩගත කර විචල්‍යන්ට ඇතුලත් කිරීම මගින් විචලයන් අතර සිදුවන ක්‍රියාකාරම් නිවැරදිව ක්‍රියාත්මක කීරීමට හැකියාව ලැබේ.

නියත.(Constants)

නියත යනු විචලයක ව්‍යුහයට තරමක් සමාන වේ. ඒවා විචල්‍යයන්ගෙන් වෙනස්වන්නේ ගබඩා වන දත්ත විච්ල්‍ය තුළ මෙන් නොව ක්‍රමලේඛනය තුළ ස්ථිරව නොවෙනස්ව පවතීමෙනි. එමනිසා

යම් ක්‍රමලේඛනයක් නිර්මාණය කිරීමේදී යම් දත්තයක් නියතව පවත්වාගත යුතු නම් ඒවා ගබඩා කිරීමට විචල්‍ය භාවිතා කල යුතු වේ. නියත නිර්මාණය කිරීමද එක් එක් ක්‍රමලේඛණ භාෂාවට අනුරූපීව සුළු වශයෙන් වෙනස් වේ.

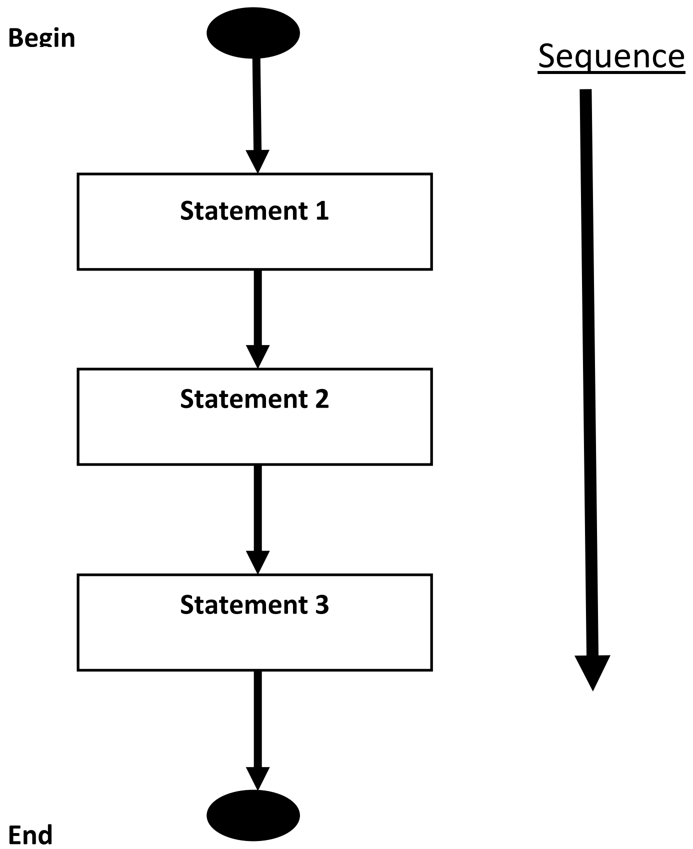
මෙහෙයවන පිළිබඳ.(Operators)

විචල්‍ය / නියත තුළ ඇති දත්තයන් මෙහෙයවමින් විවිධ ප්‍රතිදානයන් සැකසීමේදී ඒ සඳහා අදාල මෙහෙයවනයන් යොදා ගනී. එක් එක් මෙහෙයවනයන්ගේ කාර්යභාරයට අනුව ඒවා ප්‍රධාන කන්ඩ කිහිපයකට වෙන්කර දක්වා ඇත.

ගැලීම් පාලන ව්‍යුහයන්.(Control Flow Statements)

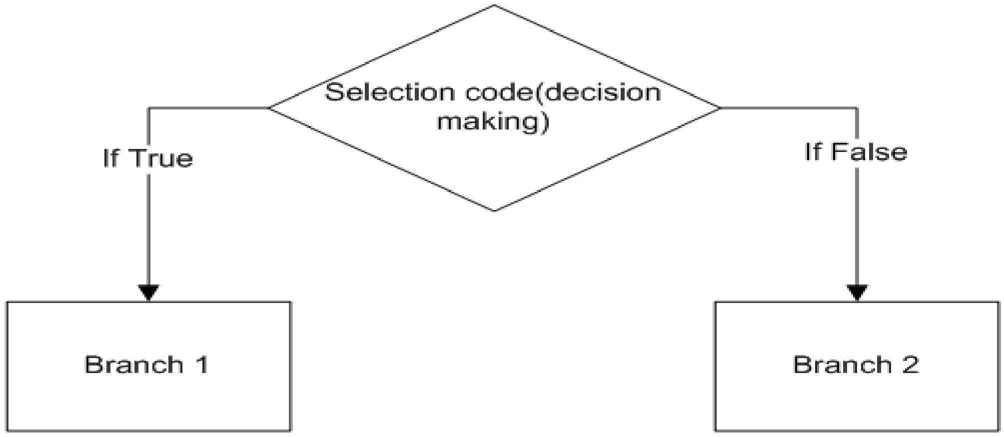
ක්‍රමලේඛයනයක් පරිගණකය මත ක්‍රියාත්මකවීමේදී එහි ඇති උපදෙස් වමේ සිට දකුණටත්, ඉහල සිට පහළටත් කියවමින් ක්‍රියාත්මක කරනු ලැබේ. එසේ ක්‍රමලේකනයක අඩංගු උපදෙස් ක්‍රියාත්මක වීම හෙවත් ක්‍රමලේඛනයේ ගලායාම ප්‍රධාන ව්‍යුහ තුනකින් සමන්විත වේ.

- අනුක්‍රමය (Sequence)



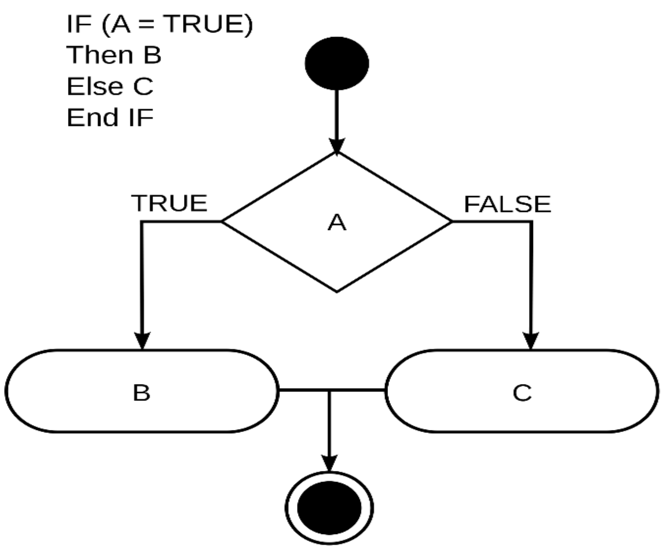
මෙහිදී ක්‍රමලේඛනය අරම්භයේ සිට අවසානය දක්වා අනුපිළිවෙලින් ගලාගෙන යයි.

- <u>වරණය (Selection)</u>

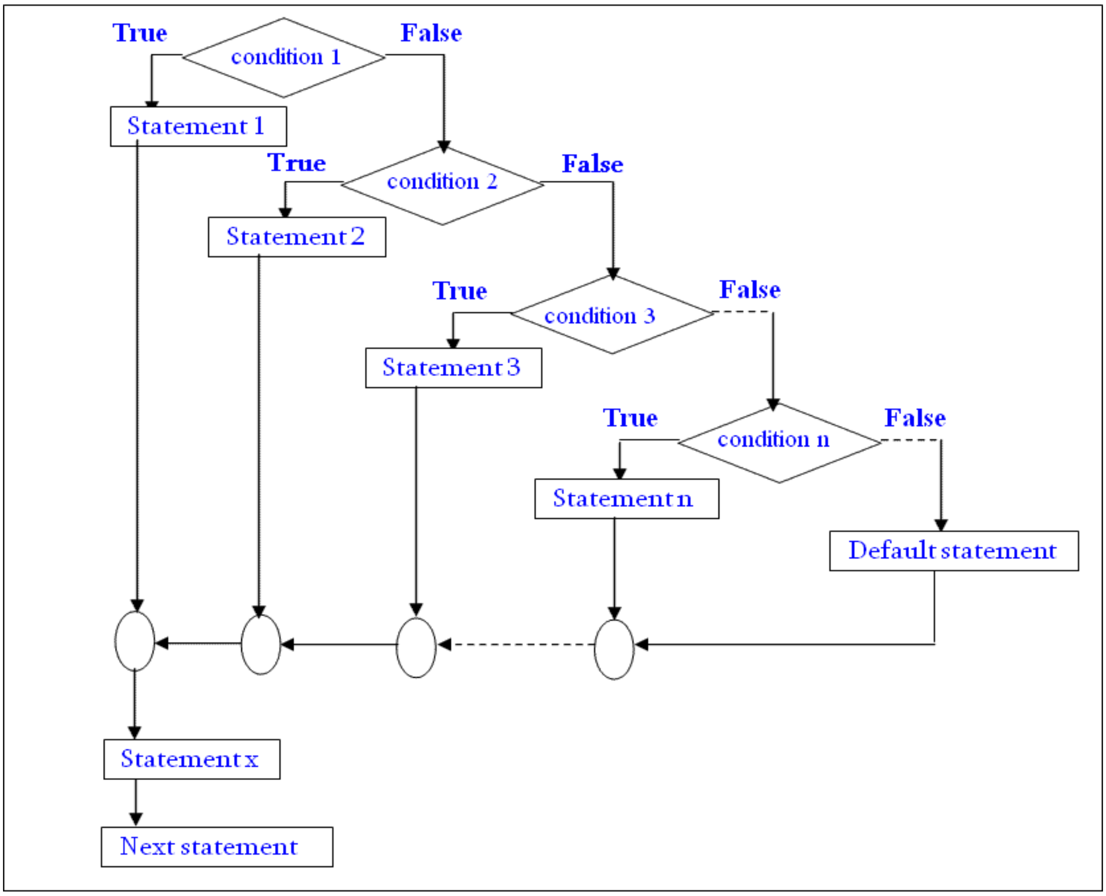


යම් කොන්දේසියක් මත ක්‍රම ලේඛනයේ ගලායාම තෝරාගනු ලබයි.මෙය සරල සහ බහුවිධ (Simple/Multiple)තෝරා ගැනීම් ලෙස වර්ගකල හැක .

- <u>සරල වර්ණයක ව්‍යුහය(Simple selection)</u>

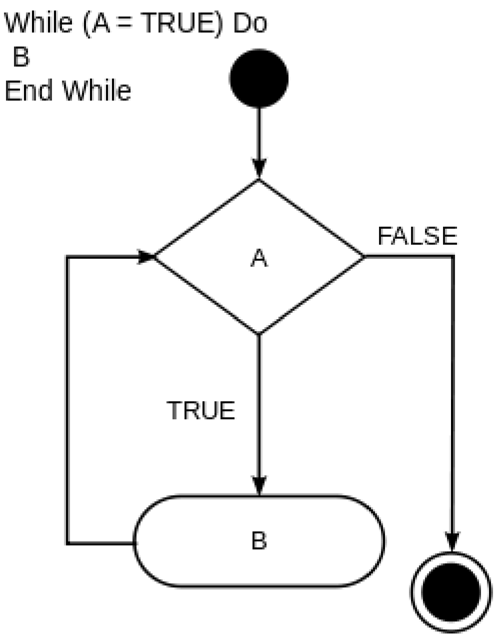


- <u>බහුවිධ වර්ණයක ව්‍යුහය(Multiple selection)</u>

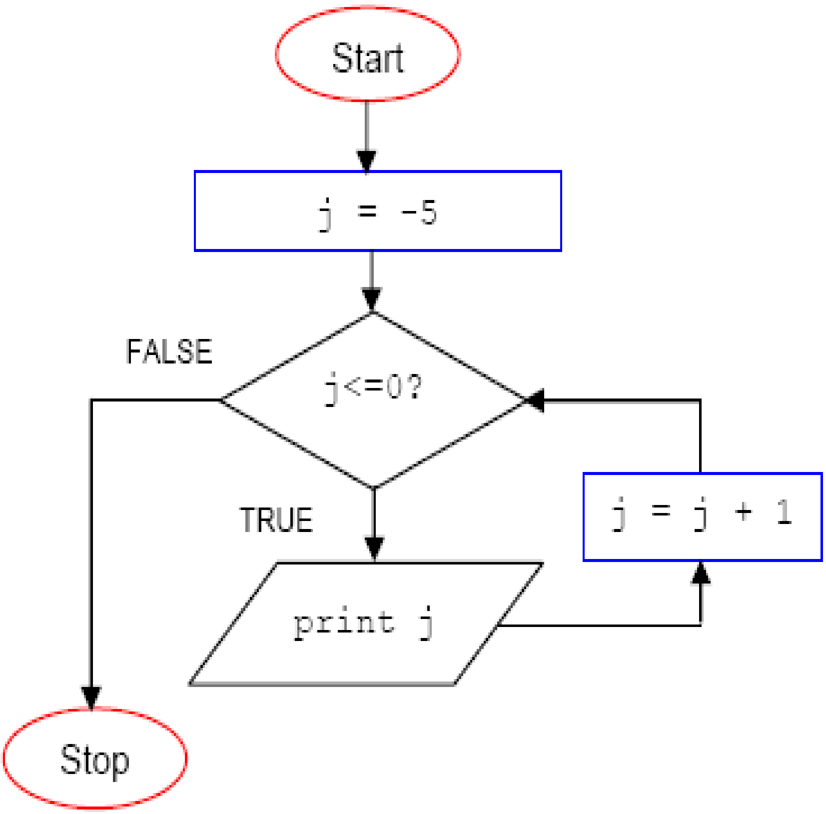


මෙහිදී කොන්දේසි ගණනාවක් මත ගමන් මාර්ග ගණනාවකින් මාර්ගයක් තෝරා ගැනීම සිදු කරයි .

- <u>පුනර්කරණය(Iteration)</u>



යම් කොන්දේසියක් සත්‍යව පවතින තුරු යම් ක්‍රියාවලියක් නැවත නැවත ක්‍රියාත්මක කරවයි.මේවා ද ආකාර දෙකක් පවතී .ඉහත ව්‍යුහය තර්ක පාලන (Logic control) ස්වරූපයයි.පහතින් ගණිත පාලන(Counter control) ස්වරූපයක උදාහරණයක් දක්වයි.මෙහිදී යම් ගණිත අගයක් සහිත කොන්දේසියක් පූර්ණ වෙනතුරු නැවත නැවත සිදුවීම (Iteration) ක්‍රියාත්මක වේ.



ඉහත කෙටියෙන් හඳුන්වාදෙන ලද ක්‍රමලේඛන කොටස් පයිතන් ක්‍රමලේඛන භාෂාව මගින් හසුරවමින් ක්‍රමලේඛන නිර්මාණය කරන අකාරය පිළිබඳ දැන ගනිමු.

පයිතන් භාෂාවේ මුර පද

මේවා කට පාඩම් කලයුතු නොවන අතර ප්‍රයෝගික ක්‍රියාකාරකම් මගින් මතකයේ රැඳෙනු ඇත.

True	False	class	def	return
if	elif	else	try	except
raise	finally	for	in	is
not	from	Import	global	lambda
nonlocal	pass	While	break	continue
and	with	As	yield	del
or	assert	None		

විවරණ

පයිතන් බස තුළ විවරණ ඇති කිරීම අකාර දෙකකින් සිදුකල හැක.

<u>තනි පද පේළි විවරණ</u>

විවරණයක් බව අගවන සලකුණ '#' සලකුණයි. මෙම සලකුණ යෙදූ පසු එම පේලිය පරිගණකය විසින් විධානයක් ලෙස නොසලකා හරී. යම් යම් අවස්ථාවල විධානයන් සහ විවරණයන් එකම පෙළක ඇතුල් කිරීමටද හැකියාව ඇත.

උදා :- # Examples. Py මෙය සම්පූර්ණ පේළියම විවරණයක් ලෙස භාවිතා වූ අවස්ථාවකි .

Char-count = len (x) # Compute the length

ඉහත අවස්ථාව විධානයක් හා විවරණයක් එක පේළියේ ඇති අවස්ථාවකි.'#' පසු කොටස විවරණයක් සේ සලකා විධානයක් ලෙස නොසලකා හරී.

බහු පද පේළි විවරණ

එක පෙලකට වඩා විශාල විවිරණ සඳහා මෙය භාවිතා වේ. “”” “”” හෝ ''' ''' සලකුණු යුගලක් තුළ විවරණය අන්තර්ගත කරනු ලබයි.

උදාහරණ

“”” This is a long comment and it extents too many lines.
You can type long comments further”””

හඳුන්වන හෙවත් විචල්‍ය නාම

හඳුන්වනයක් යනු පයිතන් භාෂාවේදී (ක්‍රමලේඛනයේදී) භාවිතයට ගන්නා යම්යම් ශ්‍රිතයන් , නියතයන් වැනි දෑ හඳුනා ගැනීම සඳහා අප (ක්‍රම ලේඛකයා) ලබාදෙන නාමයන් වේ. හඳුන්වන භාවිතයේදී ඒ සඳහා අනුගමනය කල යුතු ක්‍රමවේදයක් ඇත.

1. හඳුන්වනයක් නිර්මාණය කිරීමේදී එහි මුල් අක්ෂරය ලෙස කිසි විටෙක ඉලක්කමක් නොවියයුතු අතර ඉන්පසු ඉලක්කම් භාවිතයට ගැටළුවක් නොමැත.

2. එසේම හඳුන්වනයක් අක්ෂරයකින් හෝ ‘-‘ (underscore) ආරම්භ කල යුතුවේ.

3. මෙම හඳුන්වනයක තිබිය හැකි අක්ෂර වල උපරිම සීමාවක් නොමැත. (අක්ෂර අතරම ඉලක්කමක්ද ඇතුළු විය හැක.)

4. හඳුන්වන (Identifiers) ලෙස මුල් පද (key words) භාවිතා කල නොහැක.

5. හඳුන්වන ඉංග්‍රීසි භාෂාවේ ඇති කැපිටල් සිම්පල් යන අකුරු මත සංවේදී වේ. එනම් name යැයි හඳුන්වනය සහ Name යන හඳුන්වනය, හඳුන්වන දෙකක් ලෙස සලකනු ලැබේ.

6. හඳුන්වන තුළ !,@,#,$.% ආදී විශේෂිත සලකුණු භාවිතා කල නොහැක.

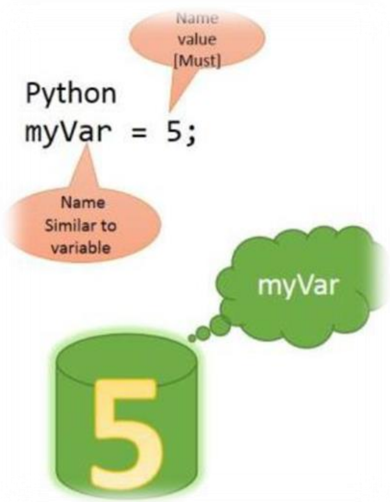
පයිතන් භාෂාව තුළ “හඳුන්වන හෙවත් විචල්‍ය නාම” මගින් විචල්‍යන් නිර්මාණය කිරීම.

පරිගණක ක්‍රමලේකනයන් තුළ සාමාන්‍යයෙන් විචල්‍යන් නිර්මාණය කිරීම කොටස් ත්‍රිත්වයක එකතුවකින් සිදුවේ. පයිතන් භාෂාව මගින් මෙම කාර්ය ඉතා පහසුවෙන් සිදුකල හැකි අතර පයිතන් භාෂාවේ ඇති ගුණාත්මක දියුණුවත් සමග නිර්මාණය කරනු ලබන විචල්‍යන් තුළ ගබඩා වන දත්ත සඳහා එම දත්තයේ ස්වරූපය මත දත්ත වර්ගය ස්වයංව තීරණය කිරීමේ හැකියාව ද

එයට ඇත. පයිතන් භාෂාවේ භාවිතා වන දත්ත ප්‍රරූප පිළිබඳ ඉදිරි පරිච්ඡේදය මගින් වඩාත් පුළුල් ලෙස විස්තර කර ඇත.

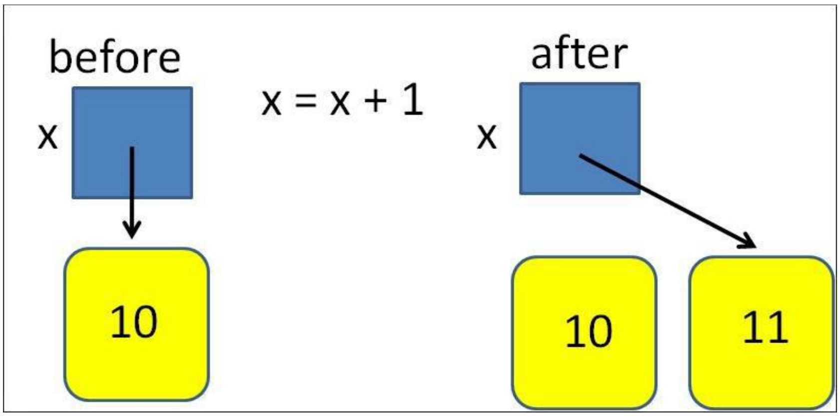
<u>විචල්‍ය නිර්මාණය කිරීමට අදාල උදාහරණ කිහිපයක්.</u>

myVar = 5 - “myVar” විචල්‍ය නාමයෙන් යුතු විචල්‍යකට 5 යන අගය ඇතුලත් වේ. මෙහිදී ‘=’ ලකුණ හෙවත් පැවරුම් ප්‍රකාශය මගින් වැදගත් ක්‍රියාවක් සිදු කරනු ලබයි. එනම් දකුණු පස ඇති අගය වම්පස ඇති විචල්‍ය තුළට පැවරීම සිදු කරනු ලබයි. පහත රූපමය උදාහරණය මගින් එය තවදුරටත් පැහැදිළි කර ඇත. එසේම පැවරුම් ප්‍රකාශනය සමාන කිරීමක් නොව දකුණු පස ඇති අගය වම්පසට පැවරීමක් සිදුකරනා බැවින් ප්‍රකාශනයක් මෙලෙස ඉදිරිපත් කිරීමටද හැකියාව ඇත.



myVar = myVar + 5

එවිට පළමුව දකුණු පස සුළු වී myVar තුළ ඇති අගයට 5 එකතුවී මුළු අගය 5 කින් ඉහල යනු ලබයි. පහත රූපය මගින් එය තවදුරටත් පැහැදිලි වේ.



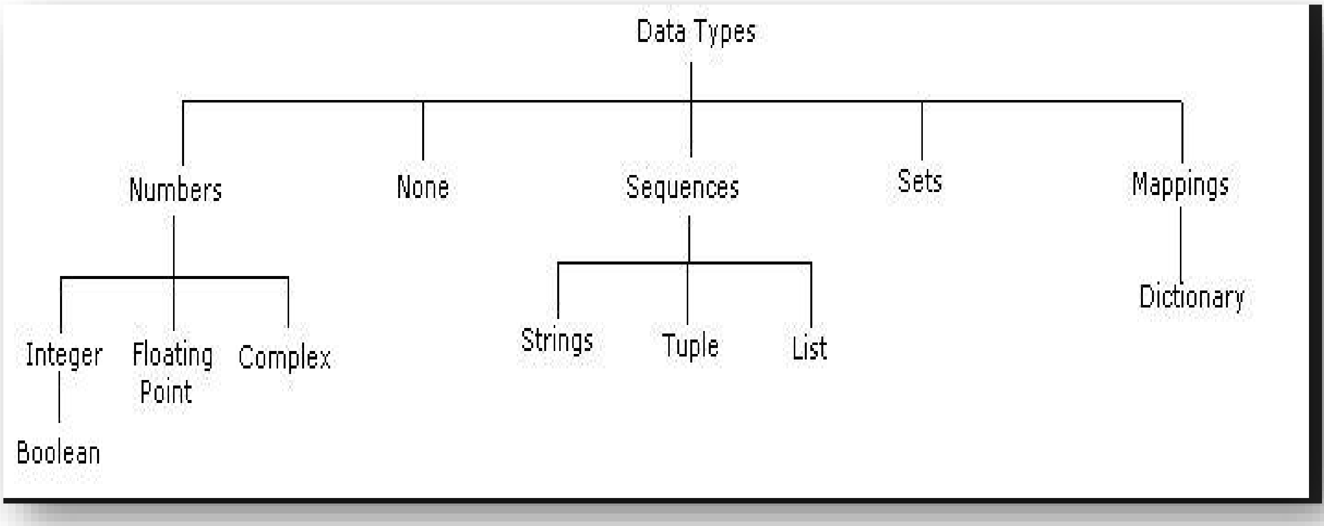
විචල්‍යන් කිහිපයකට එකවර අගයක් ඇතුළු කිරීම, පහත දැක්වෙන පරිධි සිදු කල හැක.

A = b = c = 1

මෙහිදී A, b, c යන විචල්‍යන් සඳහා 1 අගය එක් වර ඇතුලත් වේ.

පයිතන් දත්ත ප්‍රරූප හෙවත් දත්ත වර්ග.

පයිතන් භාෂාව මගින් විචල්‍යක් නිර්මාණය කර ගන්නා ආකාරය සහ එයට දත්ත ඇතුළු කිරීම සිදු කරගන්නා අකාරය පිළිබඳ දැන් අප දනිමු. එසේ ඇතුලුවන දත්ත එකිනෙකා අතර විවිධ ක්‍රියාකාරකම් සිදු කරනු ලබයි.උදාහරණයක් ලෙස අංක 1 සහ අංක 2 යන දත්ත එකතු වී අංක 3 යන දත්තය නිර්මාණය කරයි. එසේ නමුත් අංක 1 සහ A යන අක්ෂරය සහිත දත්තයන් දෙක එකතු වුවහොත් පිළිතුර කුමක්ද ? මෙවැනි ගැටළුකාරී අවස්ථා නිරාකරණය කරගැනීමට විවිධ පරිගණක භාෂාවන් සතුව ඒවා භාවිතා කරනු ලබන දත්ත පිළිබඳ මූලික කාණ්ඩ කිරීමක් සහ ඒවා අතර සිදුවන ක්‍රියාකාරකම් පිළිබඳ විග්‍රහයක් ඇත. ඒ අනුව විවිධ දත්ත ඒවායේ ස්භාවය අනුව හඳුනා ගනිමින් අනෙකුත් දත්තයන් සමඟ සිදුකල හැකි අන්තර් ක්‍රියාවන් පිළිබඳ පැහැදිලි නිර්ණය කිරීමක් සිදු කල හැක. එම ව්‍යුහය පරිගණක භාෂාවක දත්ත ප්‍රරූප ලෙස හඳුන්වනු ලැබේ. පයිතන් භාෂාවේ පවතින මූලික දත්ත ප්‍රරූප පිළිබඳ පහත දැක්වේ.



මෙහිදී මූලික වශයෙන් දත්ත කාණ්ඩ පහකට වෙන් කර දක්වා ඇති අතර ඒවා පිළිබඳ වෙන වෙනම ඉදිරිපත් කරන ලද විස්තරයක් පහත දැක්වේ.

Numbers

මෙම දත්ත කාණ්ඩය හෙවත් ප්‍රරූපය තුළ ඉලක්කම් සහිත දත්ත පිළිබඳ හඳුන්වා දෙනු ලැබේ. මෙම වර්ගයේ දත්ත එකිනෙක ක්‍රියාත්මක වි ගණිතමය අගයන් ලබා දෙනු ලැබේ. මෙම දත්ත වර්ගය තුළ ඇති දත්ත තව දුරටත් මූලික කොටස් තුනකට බෙදා ඇත.

Integer

මෙම කාණ්ඩයට අයත් වන්නේ + හෝ - පූර්ණ සංඛ්‍යාවන් හෙවත් නිඛිල සංඛ්‍යාවන් වේ.

උදා- 10 , -105

මෙම කාණ්ඩයට අයත්ව තවත් උප කාණ්ඩයක් ඇතුලත් වේ

Booliyan – මෙයට සත්‍ය / අසත්‍ය හෙවත් 1 සහ 0 යන අවස්ථා දෙක නිරූපනය කිරීමට අවශ්‍ය දත්ත දෙක ඇතුලත් වේ.

Floating point

මෙහිදී + හෝ - දශම සංඛ්‍යා කාණ්ඩ කරනු ලැබේ.

උදා- 15.20 , -21.9

Complex

මෙයට විශාල ප්‍රමාණයේ සංඛ්‍යාවන් ඇතුලත් වේ.

උදා- 45.j , -.6545+0J

Sequences

පරිගණකයේ යතුරු පුවරුවේ ඇති යතු මගින් නිරූපනය වන අක්ෂර සහ සලකුණු මෙම දත්ත කාණ්ඩයට අයත් වේ.මෙම දත්ත කාණ්ඩය තුළ ඇති දත්ත තව දුරටත් උප කාණ්ඩ තුනකට බෙදා දක්වනු ලැබේ.

String

පරිගණක යතුරු පුවරුවේ ඇති අනුලක්‍ෂ දාමයකින් සමන්විත දත්තයක් මෙසේ හඳුන්වනු ලබයි.මේවා පරිගණයට හඳුන්වා දීම සඳහා තනි ‘ ’ හෝ යුගල “” උඩු කොමා තුළ ඇතුලත් කල යුතු වේ.

උදාහරණ - str = 'Hello World!' හෝ str = “Hello World!”

මෙසේ හඳුන්වා දෙනු ලබන අනුලක්‍ෂ දාමයන් බින්දුවෙන් පටන් ගනු ලබන අනුක්‍රමික අංකනයක් යටතේ පෙළගැසෙනු ලැබේ. එසේ පෙළ ගැසෙනු ලබන අනුලක්‍ෂ වෙත වෙන වෙනම පිවිසීමට අදාල විචල්‍ය නාමයත් කොටු වරහන් මගින් [ ] අදාල අනුලක්ෂයේ අනුක්‍රමික අංකයත් යොදා ගනිමින් හැකියාව ඇත. එය පහත දක්වා ඇති උදාහරණය මගින් තව දුරටත් පැහැදිළි වේ.

```
str = 'Hello World'
```

ඉහත උදාහරණයේදී 'Hello World!' යන අනුලක්‍ෂ දාමයකින් සමන්විත දත්තය str නම් විචල්‍ය තුලට ගමන් කරනු ලබයි. හිස් තැන්ද අනුලක්‍ෂ ලෙස සලකනු ලැබේ.

අනුක්‍රමික අංකනය	0	1	2	3	4	5	6	7	8	9	10
අනුලක්‍ෂය	H	e	l	l	o		W	o	r	l	d

```
print (str) # විචල්‍යයේ අනුලක්‍ෂ සම්පූර්ණයෙන් පෙන්වයි.
print (str[0]) # පළමු අනුලක්ෂය පෙන්වයි.(H)
print (str[2:5]) # තෙවන අනුලක්ෂයේ සිට පස්වන අනුලක්ෂය දක්වා පෙන්වයි.(l - o)
print (str[2:]) # තෙවන අනුලක්ෂයේ සිට ඉදිරියට සියල්ල පෙන්වයි.(llo World)
```

```
print (str * 2) # අනුලක්ෂදාමය දෙවරක් පෙන්වයි.
print (str + "TEST") # අනුලක්ෂදාමය අවසානයට TEST වචනය එකතු වේ.
```

List

මෙම දත්ත වර්ගයේ ව්‍යුහයද ඉහත දත්ත වර්ගයට මූලික සමානකමක් දක්වන අතර මෙහි විශේෂත්වය වන්නේ එක් අනුක්‍රමික අංකය පෙළගැස්මක් යටතේ අනුලක්ෂ දාමයකින් යුතු දත්ත හෝ ඉලක්මන් සහිත දත්ත ගණනාවක් එක වර ඇතුලත් කර තැබීමට අති හැකියාවයි. එසේම මෙම දත්ත වෙන්කිරීම කොමා මගින්ද මුළු හඳුන්වාදීම කොටු වරහන් මගින්ද සිදු කරනු ලබයි. එක් එක් අනුක්‍රමික අංකයන් යටතේ ඇති දත්ත ඉහත පරිධිම වෙන් කර ගත හැක.
උදාහරණය- පහත දැක්වෙන උදාහරණයේ list සහ tinylist යන නමින් විචල්‍යන් දෙකක් නිර්මාණය කර එමගින් පහත දක්වා ඇති ක්‍රියාකාරකම් සිදුකර ඇත. (ඉලක්කම් සහිත දත්ත සාමාන්‍ය ලෙසත් අනුලක්ෂ සහිත දත්ත උඩු කොමා යටතේත් දක්වා ඇති බව නිරීක්ෂණය කරන්න)

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']

print (list) # Prints complete list
print (list[0]) # Prints first element of the list
print (list[1:3]) # Prints elements starting from 2nd till 3rd
print (list[2:]) # Prints elements starting from 3rd element
print (tinylist * 2) # Prints list two times
print (list + tinylist) # Prints concatenated lists
```

ප්‍රතිදානයන්

```
['abcd', 786, 2.23, 'john', 70.20000000000003]
abcd
[786, 2.23]
[2.23, 'john', 70.20000000000003]
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.20000000000003, 123, 'john']
```

Tuple

මෙම දත්ත වර්ගය බොහෝ සෙයින් "List" දත්ත වර්ගයට සමාන නමුත් කොමා මගින් එකිනෙක වෙන් කර දක්වන දත්ත සමූහය සාමාන්‍ය වරහන් මගින් වටකර දක්වනු ලබයි. එසේම List දත්ත වර්ගයේ මෙන් අන්තර්ගත දත්ත වෙනස් කිරීමට මෙම දත්ත වර්ගය තුළ හැකියාවක් නොමැත. එම නිසා එය කියවීමට පමණක් හැකි List ලෙසද (read-only lists) හඳුන්වනු ලැබේ. පහත දක්වා ඇති උදාහරණය මගින් එය පිළිබඳ ඔබගේ අවබෝධය තවදුරටත් වැඩිවනු ඇත. එම උදාහරණය tuple සහ tinytuple ලෙස Tuple දත්ත වර්ගයේ විචල්‍ය දෙකක් මගින් සිදුකර ඇත.
උදාහරණය -

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )
tinytuple = (123, 'john')

print (tuple) # Prints complete tuple
print (tuple[0]) # Prints first element of the tuple
print (tuple[1:3]) # Prints elements starting from 2nd till 3rd
print (tuple[2:]) # Prints elements starting from 3rd element
print (tinytuple * 2) # Prints tuple two times
print (tuple + tinytuple) # Prints concatenated tuple
```

ප්‍රතිදානයන්

```
('abcd', 786, 2.23, 'john', 70.20000000000003)
abcd
(786, 2.23)
(2.23, 'john', 70.20000000000003)
(123, 'john', 123, 'john')
('abcd', 786, 2.23, 'john', 70.20000000000003, 123, 'john')
```

SET

මෙය ‘Sequences’ දත්ත කාණ්ඩයට අයත් නොවන නමුත් එහි මූලික ලක්ෂණ සහිත දත්ත කාණ්ඩයකි. මෙහිදීද කොමා මගින් දත්ත වෙන්කරනු ලබන අතර ඒවා සඟල වරහන් මගින් වටකර දක්වනු ලබයි. එසේම මෙහි ඇති තවත් සුවිශේෂිත්වයක් වන්නේ එකම දත්තය අනුපිටපත් වීමට ඉඩ ලබා නොදීමයි. තවද මෙහි **tuple, string** සහ **number** යන දත්ත ප්‍රරූප පමණක් රඳවා ගනු ලබයි.

උදාහරණය -

basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'} දත්ත අනුපිටපත් සහිතව basket නමැති **set** විචල්‍ය නිර්මාණය කරයි.

print(basket)

ප්‍රතිදානයන්

{'orange', 'banana', 'pear', 'apple'} මෙහි දත්ත අනුපිටපත්ව නොමැත.එසේම පෙර දත්ත ප්‍රරූප තුළ මෙන් basket[**0**] ලෙස අනුක්‍රමික අංකය මගින් දත්ත වෙන කර ගත නොහැක.

Dictionary

මෙම දත්ත ප්‍රරූපය තුළ අතූ විශේෂය වන්නේ දත්ත පෙලගස්වෙනු ලබන අනුක්‍රමය පරිශීලකයාගේ අභිමතය පරිදි නිර්මාණය කරගැනීමට ඇති හැකියාවයි. ඒ අනුව පළමුව අනුක්‍රමයත් කෝලන් (:) සලකුණ මගින් වෙන්කර එයට අයත් දත්තයත් සටහන් කරනු ලබයි. පසුව එම යුගලය කොමා මගින් වෙන්කර නැවත අනෙක් දත්තයට අදාල අනුක්‍රමික අංකය සහ

දත්තය ඇතුළු කරයි. මෙසේ සකසන දත්ත පෙළ සඟල වරහන් මගින් වටකරනු ලබයි. මෙම දත්ත වර්ගය තුළ ඇති දත්ත වෙන් කර දැක්වීම, වෙනස් කිරීම, මකා දැමීම ආදිය පහසුවෙන් සිදුකල හැක.පහත උදාහරණය මගින් ඒවා තාදූරටත් පැහැදිලි වෙනු ඇත.

උදාහරණය - Days නම් විචල්‍ය තුලට සතියේ දින මගින් නම්කරන ලද අනුක්‍රමයක් තුළ දත්ත පෙළගස්වා ඇත.

Days={“Monday” : “Working day”, “Tuesday” :”Working day”, “Wednesday” : “Working day”, “Sunday” : “Holiday”, “Saturday “ : “Holiday”}

print(Days['Monday'])

ප්‍රතිදානය

Working day

<u>None</u>

ඉහත කිසිඳු දත්ත ප්‍රරූපයකට අයත් නොවන දත්ත None දත්ත වර්ගයට යටතට ඇතුල් වේ.

** ඕනෑම විචල්‍යක් තුළ අන්තර්ගත දත්තයේ ප්‍රරූපය පරික්ෂා කිරීමට පයිතන් භාෂාවේ ඇති type (විචල්‍ය නාමය) ශ්‍රිතය භාවිතා කල හැක.

වෙනස් වන සහ වෙනස් නොවන දත්ත ප්‍රරූප (Mutable vs Immutable).

ඉහත හඳුන්වා දෙනු ලැබූ දත්ත ප්‍රරූප ඒවායේ වෙනස් වන සහ නොවන ස්වරූපය මත ප්‍රධාන කොටස් දෙකකට බෙදා දැක්වීමට හැක. සාමාන්‍යයෙන් යම් විචල්‍යක් සඳහා දත්තයක් පැවරූ පසු එය නැවත වෙනස් කල හැකි නම් එය වෙනස් වන දත්ත ප්‍රරූපයක් (Mutable) ලෙසද එය නැවත වෙනස් කල නොහැකි නම් වෙනස් නොවන දත්ත ප්‍රරූප (Immutable) ලෙසද හඳුන්වනු ලැබේ.

වෙනස් වන දත්ත ප්‍රරූප (Mutable).

list, dict, set

වෙනස් නොවන දත්ත ප්‍රරූප (Immutable).

int, float, complex, string, tuple

ඉහත තත්වය තවදුරටත් පැහැදිළි කළහොත්,
පයිතන් භාෂාව වස්තු නැඹුරු භාෂාවකි (Object Oriented). ඒ අනුව එහි විචල්‍ය තුලට පැවරෙන දත්ත, වස්තුන් ලෙස හඳුනා ගැනීම සිදුවේ. මෙසේ හඳුනා ගනු ලබන වස්තුවන් සඳහා පයිතන් භාෂාව විසින් අභ්‍යන්තර හැඳුනුම් අංකයක් ලබා දෙනු ලැබේ. එය පරික්ෂා කිරීම සඳහා id( ) නම් ශ්‍රිතය භාවිතා කල හැක. තවද type() ශ්‍රිතය මගින් එහි දත්ත ප්‍රරූපය හඳුනා ගන්නා ආකාරයද දක්වා ඇත.

පහත උදාහරණයේදී i සහ k විචල්‍යයන්ට **list** දත්ත ප්‍රරූපයට අයත් සමාන දත්තයක් පවරා ඇති අතර id() නම් ශ්‍රිතය යොදා ගනිමින් එහි හැඳුනුම වෙනස් වන (Mutable) ආකාරය පැහැදිළි කර ඇත.

```
>>> i=['s','d','l'] # i නම් විචල්‍යට 2 නම් int දත්ත ප්‍රරූපයේ වස්තුවක් ඇතුළු කර ඇත.
>>> k=['s','d','l']        # k නම් විචල්‍යට 2 නම් int දත්ත ප්‍රරූපයේ වස්තුවක් ඇතුළු කර ඇත.
>>> type(i)                # විචල්‍යට දෙකම එකම දත්ත ප්‍රරූපයේ අයත් වේ.
<class 'list'>
>>> type(k)
<class 'list'>
>>>
>>> id(i)
47376824                   # විචල්‍යට දෙකෙහිම එකම වස්තුව ඇතුලත් වුවද එහි හැඳුනුම වෙනස්ව ඇත.
>>> id(k)
47376224
>>>
```

පහත දැක්වෙන උදාහරණයේදී x සහ y විචල්‍යයන්ට **int** දත්ත ප්‍රරූයේ සමාන දත්තයක් පවරා, id() නම් ශ්‍රිතය යොදා ගනිමින් එහි හැඳුනුම වෙනස් නොවන (Immutable). ආකාරය පැහැදිළි කර ඇත.

>>> x=2 # x නම් විචල්‍යට 2 නම් int දත්ත ප්‍රරූයේ වස්තුවක් ඇතුළු කර ඇත.
>>> y=2 # y නම් විචල්‍යට 2 නම් int දත්ත ප්‍රරූයේ වස්තුවක් ඇතුළු කර ඇත.
>>> type(x) # විචල්‍යට දෙකම එකම දත්ත ප්‍රරූපයට අයත් වේ.
<class 'int'>
>>> type(y)
<class 'int'>
>>>
>>> id(x)
1525900400 # විචල්‍ය දෙකෙහිම එකම වස්තුව ඇතුළත් බැවින් එහි හැඳුනුම වෙනස්ව නැත.
>>> id(y)
1525900400
>>>

<u>පයිතන් භාෂාවේ නියත.</u>

නියතයක් යනු විචල්‍යක් ලෙසම යම් නමක් ලබා දෙමින් (හඳුන්වනයක් හෙවත් විචල්‍ය නාමයක්) පරිගණකයේ ප්‍රධාන මතකයේ වෙන් කරන ලද ඉඩක් තුළ ගබඩා කරන ලද අගයකි. නමුත් නියත හා විචල්‍ය අතර ප්‍රධාන වෙනස්කම වන්නේ විචල්‍ය තුළ ඇති අගයන් නිරන්තරයෙන් වෙනස්වීමත් විචල්‍යන් තුළ අගයන් වෙනසක් නොවී නියතව පැවතීමත් හේතුවෙනි.
අනෙකුත් පරිගණක භාෂාවනට සාපේක්ෂව පයිතන් භාෂාව තුළ නියති නිර්මාණය කිරීමට තරමක විශේෂ කාර්යක් සිදුකල යුතු වේ. එසේ නියතයක් නිර්මාණය කර ගනු ලබන අකාරය පිළිබඳ පියවරාත්මකව පහත දක්වා ඇත.

පළමුව නව පයිතන් ගොණුවක් ආරම්භකර එහි අගයන් සහිත නියත නිර්මාණය කරන්න. සැබවින්ම ඒවා එම ගොණුව තුළ ඇති විචල්‍යන් ලෙස හැඳින්විය හැක. ඒ අනුව "constant.py" නම් නව ගොණුව තුළ පහත දැක්වෙන නියත (විචල්‍ය) නිර්මාණය කර ඇත.

```
A=5
B=10
```

ඉන් පසු අදාල ක්‍රමලේඛනය සහිත ගොණුවට එම අගයන් පහත දක්වා ඇති පරිදි කැඳවනු ලැබේ.

```
import constant
```

"constant.py" ගොණුව අදාල ක්‍රමලේඛනය තුලට කැඳවනු ලබන්නේ import යන මුරපදය මගිනි. පසුව නියතින් අඩංගු ගොණු නාමය (constant) ලබා දී ඇත. ඉන් පසු එම අගයන් **print** විධානය මගින් ප්‍රදර්ශනය කර ඇත.

```
print(constant.A)
print(constant.B)
```

එවිට ප්‍රතිදානය ලෙස 5 සහ 10 අගයන් ලැබෙනු ඇත. මෙහිදී නියත අඩංගු ගොණුවත්, ඒවා කැඳවනු ලබන අනෙකුත් ගොණුව හෝ ගොණුත් එකම ෆෝල්ඩරයක් තුළ සුරැකිය (save) යුතු වේ. මෙසේ දෙවන ගොණුවකට කැඳවනු ලබන අගයන් නොවෙනස්ව ක්‍රියාත්මකවේ. ඒවා වෙනස් කිරීමට නම් නියත නිර්මාණය කර ඇති පළමු ගොණුව තුලට ගොස් වෙනස් කල යුතුවේ.

පයිතන් භාෂාවේ ආදාන හා ප්‍රතිදාන ක්‍රියාත්මක කිරීම.

පයිතන් භාෂාව තුළ නිර්මාණය කරන ලද විචල්‍යයන් තුළට දත්ත ආදානය කිරීමත්, විචල්‍ය තුළ අන්තර්ගත දත්ත ප්‍රතිදානය කිරීමත් සඳහා නිර්මාණය කරන ලද ප්‍රධාන ශ්‍රිතයන් තුනක් පවතී. ඒවා පිළිබඳ වෙන වෙනම පහත දක්වා ඇත.

ආදානය සඳහා

මේ සඳහා input( ) ශ්‍රිතය භාවිතා කරනු ලැබේ. පහත දැක්වෙන ප්‍රයෝගික පරීක්ෂණයේ දී input( ) ශ්‍රිතය භාවිතා කර ඇති අතර ආදානය කල යුතු දත්තය පිළිබඳ පණිවිඩයක් වරහන් තුළ ('Enter the value:') **ලෙස ඇතුලත් කර ඇත. ශ්‍රිතය ක්‍රියාත්මකව x විචල්‍ය තුලට ඇතුලත් කරනු කරන ලද දත්තය ඉලක්කමක් (10) වුවද එහි ප්‍රරූපය පරික්ෂා කිරීමේදී එය** string **දත්ත ප්‍රරූපයට අයත් බව පෙනී යයි. එම නිසා ආදානය කරනු ලබන දත්තය int හෝ float ලෙස පවත්වා ගැනීම සඳහා සමස්ත අදානයම int() හෝ float() ශ්‍රිතයන් මගින් ඇතුලත් කල යුතු වේ.**

```
>>> x=input('Enter the value:')
Enter the value:10
>>> x
'10'
>>> type(x)
<class 'str'>

>>> x=int(input('Enter value:'))
Enter value:10
>>> x
10
>>> type(x)
<class 'int'>

>>> x=float(input('Enter value:'))
Enter value:10
>>> x
10.0
>>> type(x)
<class 'float'>
>>>
```

ප්‍රතිදානය සඳහා

විචල්‍යයන් තුළ ඇති දත්ත පිටතට ලබා ගැනීම හෙවත් ප්‍රතිදානය කිරීම සඳහා ප්‍රධාන ශ්‍රිත දෙකක් භාවිතා කරනු ලබයි. ඒවා නම් print() සහ return() **වන අතර** return( ) **ශ්‍රිතය භාවිතා කරනු ලබන්නේ ක්‍රමලේඛනය තුළ නිර්මාණය කරනු ලබන වෙනත් ශ්‍රිතයන්ගේ විචල්‍ය තුළ ඇති දත්ත ප්‍රරූපයේ වෙනසකින් තොරව පිටතට ලබා ගැනීම සඳහා වේ. (ශ්‍රිත කොටසේදී වැඩි දුර පැහැදිලි කරනු ලැබේ.) එම නිසා සාමාන්‍ය අවස්ථා වලදී බහුලව භාවිතා කරනු ලබන්නේ print() ශ්‍රිතය වන අතර වරහන් තුළ අගය ප්‍රතිදානය කල යුතු විචල්‍ය ඇතුලත් කරනු ලබයි.**

```
>>> print(x)
10.0
>>>
```

ප්‍රතිදානය යම් පණිවිඩයක් නම් එය **print()** ශ්‍රිතයේ වරහන් තුළ තනි උඩු කොමා හෝ ද්විත්ව උඩු කොමා යටතේ දක්වනු ලබයි.

```
>>> print('Hello Python')
Hello Python
>>> print("Hello Python")
Hello Python
>>>
```

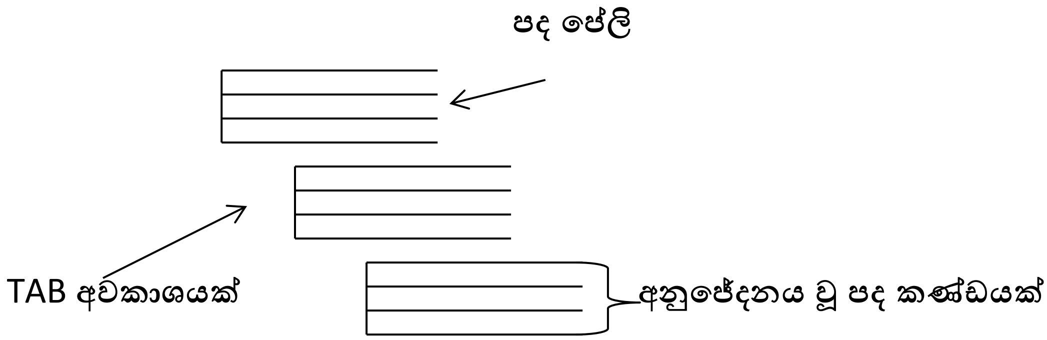
ප්‍රතිදානය සමඟ යම් පණිවිඩය ඇතුලත් කිරීමේදී පණිවිඩ කොටස උඩු කොමා යටතේ සහ විචල්‍ය නාමය එක පෙලට ලබා දෙන අතර එම කොටස් කොමාවක් මගින් වෙන් කරනු ලබයි.

```
>>> print('Value is :',x)
Value is : 10.0
>>>
```

<u>ක්‍රමලේඛන ව්‍යුහය</u>

එක් එක් ක්‍රම ලේඛන (Programming Languages) ක්‍රියාත්මක වීමේදී ඒවායේ විධානයන් (පද පේළින්) පරිගණකයකට හඳුනාගනිමේ පහසුව සහ ක්‍රියාත්මක කිරීමේ පහසුව මත කාණ්ඩ ගත කිරීම සිදුකරයි.පයිතන් භාෂාවේදී මෙය සිදුකරනුයේ පරිගණක යතුරු පුවරුවේ ඇති (key board) TAB යතුර මගින් පද පේළියට ඉදිරියෙන් ඇතිකරනු ලබන TAB අවකාශ මගිනි. මෙයට අනුජේදනය (Indentation) ලෙස හඳුන්වයි. මෙයට "Space" යතුර භාවිතා කල හොත් කාරක රීති දෝෂ ඇතිවිය හැක. TAB යතුර මගින් ඇතිවන අවකාශය මත පරිගණකය, ක්‍රම ලේඛනයේ ඇති අනුජේදන වූ පද පේළි කට්ටල පැහැදිලි ලෙස වෙනවෙනම කාණ්ඩ ගත කර හඳුනා ගනී.

උදා :-



<u>මෙහෙයවන (Operators)</u>

පයිතන් භාෂාව තුළ අන්තර්ගත මෙහෙයවනයන් ඒවායේ ක්‍රියාත්මක වන ආකාරයන් මත මූලික කාණ්ඩ හතකට බෙදා දැක්විය හැක. එම කාණ්ඩ ප්‍රයෝගික උදාහරණද සමගින් පහත දක්වා ඇත.

<u>ගණිතමය මෙහෙයවන (Python Arithmetic Operators)</u>

විචලයන් තුල ඇති අගයන් ගණිතමය ක්‍රියාකාරකම් සඳහා භාවිතා කිරීමට මෙම මෙහෙයවන යොදා ගනී.

Operator	Meaning	Example
+	Addition	4 + 7 ⟶ 11
-	Subtraction	12 - 5 ⟶ 7
*	Multiplication	6 * 6 ⟶ 36
/	Division	30 / 5 ⟶ 6
%	Modulus	10 % 4 ⟶ 2
//	Quotient	18 // 5 ⟶ 3
**	Exponent	3 ** 5 ⟶ 243

ප්‍රයෝගික උදාහරණ -

```
#!/usr/bin/python3
a = 21
b = 10
c = 0
c = a + b
print ("Line 1 - Value of c is ", c)
c = a - b
print ("Line 2 - Value of c is ", c )
c = a * b
print ("Line 3 - Value of c is ", c)
c = a / b
print ("Line 4 - Value of c is ", c )
c = a % b
print ("Line 5 - Value of c is ", c)
a = 2
b = 3
c = a**b
print ("Line 6 - Value of c is ", c)
a = 10
b = 5
c = a//b
print ("Line 7 - Value of c is ", c)
```

ප්‍රතිදානයන්

```
Line 1 - Value of c is 31
Line 2 - Value of c is 11
Line 3 - Value of c is 210
Line 4 - Value of c is 2.1
Line 5 - Value of c is 1
Line 6 - Value of c is 8
Line 7 - Value of c is 2
```

<u>සංසන්දනාත්මක මෙහෙයවන (Python Comparison Operators)</u>

විචලයන් තුළඇති අගයන් සංසන්දනය කරමින් විවිධ කොන්දේසි නිර්මාණය කිරීම් ආදී කටයුතු සඳහා මෙම මෙහෙයවන භාවිතා කරනු ලබයි.

Operator	Meaning
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

ප්‍රයෝගික උදාහරණ - (උදාහරණය තේරුම් ගැනීමට හැකිතාක් උත්සාහ කරන්න.ඉදිරියේදී if – else පිළිබඳ හඳුන්වා දෙනු ලැබේ.)

```
#!/usr/bin/python3
a = 21
b = 10
if ( a == b ):
print ("Line 1 - a is equal to b")
else:
print ("Line 1 - a is not equal to b")
if ( a != b ):
print ("Line 2 - a is not equal to b")
```

```
else:
print ("Line 2 - a is equal to b")
if ( a < b ):
print ("Line 3 - a is less than b" )
else:
print ("Line 3 - a is not less than b")
if ( a > b ):
print ("Line 4 - a is greater than b")
else:
print ("Line 4 - a is not greater than b")
a,b=b,a #values of a and b swapped. a becomes 10, b becomes 21
if ( a <= b ):
print ("Line 5 - a is either less than or equal to b")
else:
print ("Line 5 - a is neither less than nor equal to b")
if ( b >= a ):
print ("Line 6 - b is either greater than or equal to b")
else:
print ("Line 6 - b is neither greater than nor equal to b")
```

ප්‍රතිදානයන්

```
Line 1 - a is not equal to b
Line 2 - a is not equal to b
Line 3 - a is not less than b
Line 4 - a is greater than b
Line 5 - a is either less than or equal to b
Line 6 - b is either greater than or equal to b
```

පැවරුම් මෙහෙයවන (Python Assignment Operators)

එක් විචල්‍යක් තුළ ඇති අගයක් තවත් විචල්‍යක් තුලට පැවරීම සඳහා මෙම විචල්‍ය භාවිතා කරනු ලබයි. මෙහිදී සෑම විටම මෙහෙයවනයට දකුණු පස ඇති විචල්‍යයේ හෝ විචල්‍යන්ගේ දත්ත තනි අගයකට සකස් වී වම්පස ඇති විචල්‍ය තුලට ඇතුලත් වේ.

Operator	Example	Equals To
=	a = 10	a = 10
+=	a += 10	a = a+10
-=	a -= 10	a = a-10
*=	a *= 10	a = a*10
/=	a /= 10	a = a / 10
%=	a %= 10	a = a % 10
//=	a //= 10	a = a // 10

ප්‍රයෝගික උදාහරණ -

```
#!/usr/bin/python3
a = 21
b = 10
c = 0
c = a + b
print ("Line 1 - Value of c is ", c)
c += a
print ("Line 2 - Value of c is ", c )
c *= a
print ("Line 3 - Value of c is ", c )
35
c /= a
print ("Line 4 - Value of c is ", c )
c = 2
c %= a
print ("Line 5 - Value of c is ", c)
c **= a
print ("Line 6 - Value of c is ", c)
c //= a
print ("Line 7 - Value of c is ", c)
```

ප්‍රතිදානයන්

```
Line 1 - Value of c is 31
Line 2 - Value of c is 52
Line 3 - Value of c is 1092
Line 4 - Value of c is 52.0
Line 5 - Value of c is 2
```

```
Line 6 - Value of c is 2097152
Line 7 - Value of c is 99864
```

<u>තර්කානුකූල මෙහෙයවන (Python Logical Operators)</u>

මෙම මෙහෙයවනයන්ට දෙපස ඇති විචල්‍ය තුළ ඇති දත්ත තර්කානුකූලව සලකා බලා සත්‍ය / අසත්‍ය තත්වය ප්‍රතිදානය කරනු ලබයි.

- not

x	not x
False	True
True	False

- and

x	y	x and y
False	False	False
False	True	False
True	False	False
True	True	True

- or

x	y	x or y
False	False	False
False	True	True
True	False	True
True	True	True

ප්‍රයෝගික උදාහරණ -

```
Python Shell
File  Edit  Shell  Debug  Options  Windows  Help
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
 (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> x = 12
>>> y = 15
>>> (x>10 and y>10) # Test whether both x and y are greater than
 10
True
>>> (x>14 or y>14) # Test either x>14 or y>14
True
>>> not(x>10 and y>10) # Here both x and y greater than 10, i.e.
 (x>10 and y>10) will return True and not operator makes it Fals
e
False
>>> |
Ln: 11 Col: 4
```

බිටුමය මෙහෙයවන. (Python Bitwise Operators)

මෙම මෙහෙයවන මගින් ක්‍රියාත්මක වන විචල්‍ය අගයන් ද්විමය බවට පරිවර්තනය වී අදාල ප්‍රතිදානයන් සකස් වේ.

Operator	Name
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise 1's complement
<<	Left-Shift
>>	Right-Shift

ප්‍රයෝගික උදාහරණ -

```
#!/usr/bin/python3
a = 60 # 60 = 0011 1100
b = 13 # 13 = 0000 1101
print ('a=',a,':',bin(a),'b=',b,':',bin(b))
c = 0
c = a & b; # 12 = 0000 1100
print ("result of AND is ", c,':',bin(c))
c = a | b; # 61 = 0011 1101
print ("result of OR is ", c,':',bin(c))
c = a ^ b; # 49 = 0011 0001
print ("result of EXOR is ", c,':',bin(c))
c = ~a; # -61 = 1100 0011
print ("result of COMPLEMENT is ", c,':',bin(c))
c = a << 2; # 240 = 1111 0000
print ("result of LEFT SHIFT is ", c,':',bin(c))
c = a >> 2; # 15 = 0000 1111
print ("result of RIGHT SHIFT is ", c,':',bin(c))
```

ප්‍රතිදානයන්

```
60 : 0b111100 b= 13 : 0b1101
result of AND is 12 : 0b1100
result of OR is 61 : 0b111101
result of EXOR is 49 : 0b110001
result of COMPLEMENT is -61 : -0b111101
result of LEFT SHIFT is 240 : 0b11110000
result of RIGHT SHIFT is 15 : 0b111
```

සාමාජිකත්ව මෙහෙයවන (Python Membership Operators)(උ.පෙ විෂය නිර්දේශය තුළ අන්තර්ගත නොවේ. අමතර දැනුම සඳහා ඉදිරිපත් කර ඇත)

Sequence දත්ත කාණ්ඩයට අයත් අනුලක්ෂ දාම, ලැයිස්තු, ටපල වැනි දත්ත ප්‍රරූප තුළ අඩංගු දත්තයක් පරික්ෂා කිරීමට මෙම මෙහෙයවන යොදා ගනී.

Operator	Description
in	It returns true if value/variable is found in the sequence and false otherwise
not in	It returns true if value/variable is not found in the sequence and false otherwise

ප්‍රයෝගික උදාහරණ -

```
Python 2.7.14 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:19:30) [MSC v.1500 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = "python operators"
>>> b = {1:'x', 2:'y'}
>>> print("p" in a)
True
>>> print("python" not in a)
False
>>> print(1 in b)
True
>>> print('y' in b)
False
```

හැඳුනුම් මෙහෙයවන (Python Identity Operators) (උ.පෙ විෂය නිර්දේශය තුළ අන්තර්ගත නොවේ. අමතර දැනුම සඳහා ඉදිරිපත් කර ඇත)

විචල්‍යන් තුල අන්තර්ගත දත්තයන් සඳහා ඇති පරිගණක මතකයේ ස්ථානීය හැදුනුම පරික්ෂා කිරීමට මෙම මෙහෙයවන භාවිතා කරනු ලබයි.

Operator	Description
is	It returns true if two variables point the same object and false otherwise
is not	It returns false if two variables point the same object and true otherwise

ප්‍රයෝගික උදාහරණ -

```
Python 3.6.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
>>> x = 2
>>> y = 2
>>> z = 3
>>> print (x is y) #check if both point same memory location
True
>>> print (y is z)
False
>>> print (x is not y) #check if both point separate memory location
False
>>> print (y is not z)
True
Ln: 75  Col: 4
```

<u>මෙහෙයවන ප්‍රමුඛතාවය (Precedence Rules)</u>

පරිගණකය විසින් ක්‍රමලේඛනයක් කියවනු ලබන්නේ වමේ සිට දකුණටත්, ඉහල සිට පහළටත්ය. එම කියවීමේදී හමුවන මෙහෙයවනයන් පහත දැක්වෙන මට්ටම් අනුපිළිවෙලට ප්‍රමුකතාවය ලබා දෙමින් ක්‍රියාත්මක කරවනු ලබයි. එකම මට්ටමේ පිහිටි මෙහෙයවනයන්, ක්‍රමලේකනය දකුණේ සිට වමට කියවා යන විට හමුවන අනුපිළිවෙළින් ක්‍රියාත්මක කරවනු ලැබේ.

Python Operator Precedence

Precedence	Operator Sign	Operator Name
Highest	**	Exponentiation
	+x, -x, ~x	Unary positive, unary negative, bitwise negation
	*, /, //, %	Multiplication, division, floor, division, modulus
	+, -	Addition, subtraction
	<<, >>	Left-shift, right-shift
	&	Bitwise AND
	^	Bitwise XOR
	|	Bitwise OR
	==, !=, <, <=, >, >=, is, is not	Comparison, identity
	not	Boolean NOT
	and	Boolean AND
Lowest	or	Boolean OR

පයිතන් භාෂාව තුළ ගැලීම් පාලන ව්‍යුහ පිහිටවීම

ඇල්ගොරිතම කොටසේදී හඳුනා ගනු ලැබූ ගැලීම් පාලන ව්‍යුහ ආකාර තුනෙන් අනුක්‍රමය සඳහා විශේෂිත ක්‍රමවේදයක් අවශ්‍ය නොවන අතර තේරීම සහ පුනර්කරණය යන ව්‍යුහ ගොඩනැංවීම සඳහා විශේෂිත ක්‍රමවේදයක් අනුගමනය කලයුතු වේ. එහිදී යම් යම් තත්වයෙන් මත ක්‍රමලේඛනයේ ගලායාම පාලනය වන අතර, මෙම ව්‍යුහයන් දෙකෙහිම තත්වයන් හෙවත් පාලනය විය යුතු කොන්දේසි හඳුන්වා දෙනු ලබන පද පේලි : (කෝලන්) සලකුණක් මගින් අවසන් කරනු ලබන අතර එම කොන්දේසිය/කොන්දේසි සත්‍ය වේනම් සිදු කල යුතු කොටස් අනුජේදනය කරමින් ඇතුලත් කරනු ලබයි.

<u>තේරීම ව්‍යුහය සඳහා</u>

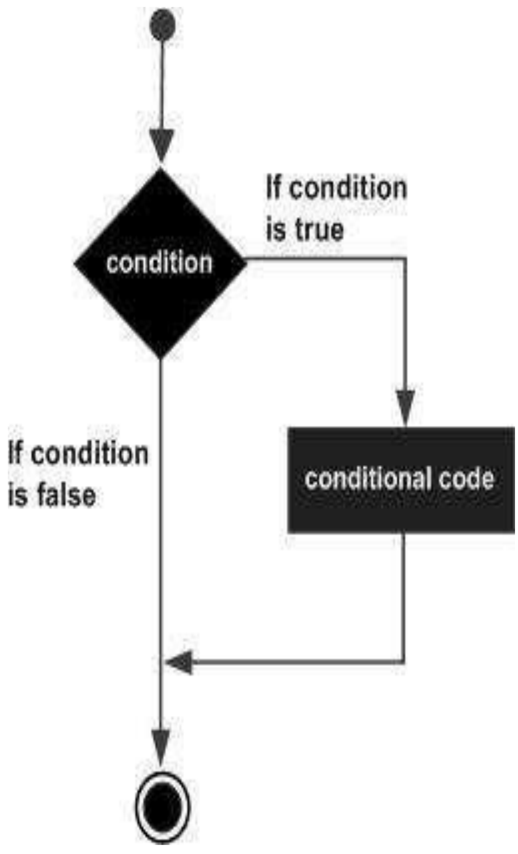
තේරීම ව්‍යුහය තුළ සරළව වෙන් කොට හඳුනා ගත හැකි ආකාර තුනකි (ඇල්ගොරිතම කොටසේ විස්තර කර ඇති පරිදි). ඒවා ප්‍රයෝගික ක්‍රියාකාරකම් මගින් වෙන් වෙන් ලෙස පැහැදිලි කර ඇත.

<u>if ව්‍යුහය</u>

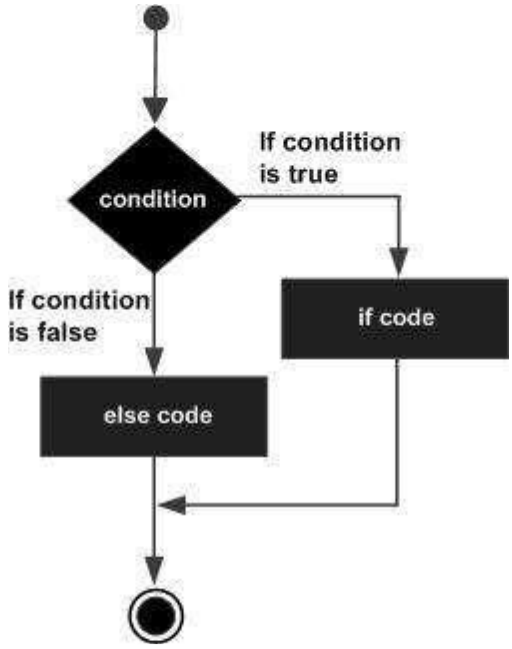
මෙහිදී පරික්ෂා කරනු ලබන යම් කොන්දේසියක සත්‍ය / අසත්‍ය තත්වය මත ක්‍රමලේකයනයේ ගලා යාම සිදුවේ. ඒ අනුව කොන්දිසිය සත්‍ය නම් අමතර පියවර කිහිපයකුත් අසත්‍ය නම් සාමාන්‍ය පරිදිත් ක්‍රමලේඛනයේ ගලායාම පාලනය කරනු ලබයි.

```
x=int(input('Enter the value:'))
if x>20:
   print('GOOD')
```

ඉහත උදාහරණයේදී x විචල්‍ය තුලට ලබා දෙන අගය 20 වැඩි නම් පමණක් GOODයන පණිවිඩය දිස් වේ.



if – else ව්‍යුහය.



මෙය ඉහත තත්වයේ දිගුවක් ලෙස දැක්විය හැක. මෙහිදී කොන්දේසිය සත්‍ය නම් if තුළ කොටසත් අසත්‍ය නම් else තුළ කොටසත් ක්‍රියාත්මක වන පරිදි පැහැදිලි කොටස් දෙකකට බෙදී යනු ලැබේ.(else පදය අවසානයද කෝලන් සලකුණක් මගින් සටහන් වේ.)

Enter the value:**10**

NOT GOOD

>>>

Enter the value:**50**

GOOD

>>>

if – elif ව්‍යුහය.

ඉහත අවස්ථා දෙකෙහිම එක් කොන්දේසියක් මත ක්‍රමලේඛනයේ ගලායාම තීරණය කරනු ලැබිණි. නමුත් මෙම ව්‍යුහය තුලදී ක්‍රමලේඛනයේ ගලායාම කොන්දේසි ගණනාවක් අතරින් තෝරාගනු ලබයි. පළමු කොන්දිසිය if මගින් ආරම්භවන අතර ඉන් පසු elif මගින් ඕනෑම කොන්දේසි ප්‍රමාණයක් මෙම ව්‍යුහයට ඇතුළත් කල හැක. එම කොන්දේසි කිසිවක් සත්‍ය නොවේනම් else මගින් මෙම ව්‍යුහයේ අවසානය දැක්වේ.

Enter the value:**60**

BEST

>>>

Enter the value:**40**

GOOD

>>>

Enter the value:**45**

VERY GOOD

>>>

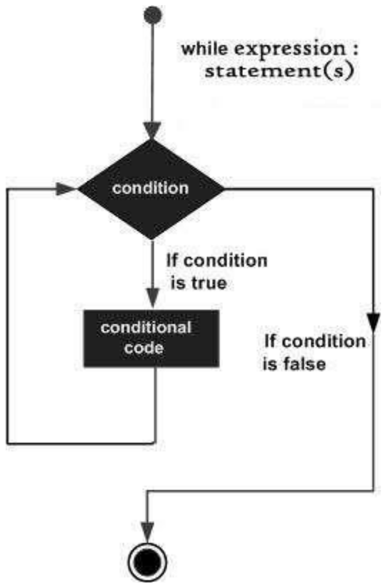
Enter the value:**10**

NOT GOOD

>>>

පුනර්කරණය ව්‍යුහය සඳහා

පයිතන් භාෂාවේ මෙම ව්‍යුහ පිහිටවීම සඳහා **while** සහ **for** භාවිතා කරනු ලබයි. එම තත්වයන් පිළිබඳ පහත වෙන වෙනම විස්තර කර ඇත.



while ව්‍යුහය

මෙය මගින් යම් කොන්දේසියක් ලබාදී එය සත්‍යව පවතින තුරු ඒ යටතේ ලබාදී ඇති එකම ක්‍රියාවලිය නැවත නැවත ක්‍රියාත්මක වීම සිදුකරනු ලබයි.

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

ඉහත දක්වා ඇති ක්‍රමලේඛනයේදී **n** විචල්‍ය **0** වෙන් ආරම්භ වන අතර **while** මගින් **n** හි අගය **10** කුඩාව පවතින තුරු එහි අගය ප්‍රදර්ශනය කිරීම සහ එහි අගය එකකින් වැඩි කිරීම සිදු කරනු ලබයි.

for ව්‍යුහය

මෙම ව්‍යුහය ඉතා සරළ නමුත් වඩා කාර්යක්ෂම ක්‍රියාවලියක නිරත වේ. මෙය **for** යන පදයෙන් ආරම්භව ඊට පසුව ලබා දෙන විචල්‍ය සඳහා **in** යන පදයෙන් (**for** සහ **in** යනු පයිතන් භාෂාවේ මුරපද වේ) සම්බන්ධ කොටසින් නිකුත් කරනු ලබන අගයන් මත තම පුනර්කරණ ක්‍රියාවලිය පවත්වාගෙන යනු ලබයි. පහත දක්වා ඇති ප්‍රයෝගික නිදර්ශන පරිශීලනය කිරීමෙන් ඒ පිළිබඳ තව දුරටත් අවබෝධ වනු ඇත.

```
>>> for x in [1,2,3]:
    print(x)
```

ප්‍රතිදානය

1

2

3

ඉහත ක්‍රියාකරකම තුළ **in** මගින් ලයිතුවක් (**list**) සම්බන්ධ කර ඇති අතර **for** පුනර්කරණය තුලදී ලයිතුවේ ඇති කොටස් අනුපිළිවෙළින් **x** විචල්‍යට ප්‍රවිෂ්ඨ වී සෑම පුනර්කරණ වාරයක් තුලදීම **print()** ශ්‍රිතය මගින් **x** හි අගය පිටතට නිරූපනය කරනු ලබයි. ලයිස්තුවේ සියලු කොටස් අවසානයේ පුනර්කරණ ක්‍රියාවලිය නතර වනු ලැබේ. Number දත්ත ප්‍රරූපය හැර සියලු දත්ත ප්‍රරූප සඳහා මෙලෙස **for** ව්‍යුහය ක්‍රියාත්මක වන අතර Dictionary දත්ත ප්‍රරූපයේ දී පමණක් එහි අනුක්‍රමික අංක විචල්‍ය තුලට ලබාදේ.

```
>>> for x in {'a':1,'b':2,'c':3}:
```

```
    print(x)
```

ප්‍රතිදානය

```
a
b
c
>>>
```

for ව්‍යුහය Number දත්ත ප්‍රරූපය සමඟ ක්‍රියාත්මක වීමේ දී **range( )** ශ්‍රිතය් සමඟ එකාබද්ධව ක්‍රියාත්මක වේ. **range( )** ශ්‍රිතය ක්‍රියාත්මක වන ප්‍රධාන ආකාර කිහිපයක් පහත දක්වා ඇත. **(**මෙහිදී ආරම්භය **0** හෝ ලබා දී ඇති ස්ථානයකින් සිදුවන අතර අවසානය ලබා දී ඇති ස්ථානයට පෙර සටහන් වේ.**)**

ක්‍රියාකාරකම **01**

```
>>> for x in range(5):                    # 0 සිට 4 දක්වා පරාසක් තුළ පුනර්කරණය වේ.
    print(x)
```

ප්‍රතිදානය

```
0
1
2
3
4
>>>
```

ක්‍රියාකාරකම **02**

```
for x in range(1,8):    # 1 සිට 7 දක්වා පරාසක් තුළ පුනර්කරණය වේ.
    print(x)
```

ප්‍රතිදානය

```
1
2
3
4
5
6
7
```

>>>

ක්‍රියාකාරකම 03

```
>>> for x in range(1,10,2):   # 1 සිට 10 දක්වා 2 පරතයක් පවත්වා ගනිමින්  තුළ පුනර්කරණය වේ.
    print(x)
```

ප්‍රතිදානය

1

3

5

7

9

>>>

ක්‍රියාකාරකම 04

```
>>> for x in range(10,1,-1):# 10 සිට 1 දක්වා -1 පරතරයක් පවත්වා ගනිමින් අවරෝහණ ලෙස පුනර්කරණය වේ.
    print(x)
```

ප්‍රතිදානය

10

9

8

7

6

5

4

3

2

<u>ශ්‍රිතයන්.(Functions)</u>

පයිතන් භාෂාව තුළ ඇති ශ්‍රිතයන් මූලික ආකාර දෙකකට වෙන් කළ හැක. එනම් භාෂාව නිර්මාණයේදී එහි නිර්මාතෘ විසින් නිර්මාණය කර ඇති ශ්‍රිතයන් හෙවත් නිර්මිත ශ්‍රිත සහ විවිධ අවශ්‍යතා මත පරිශීලකයන් විසින් ක්‍රමලේඛන තුළ නිර්මාණය කරනු ලබන ශ්‍රිතයන් ලෙස වේ. නිර්මිත ශ්‍රිත අදාල අවශ්‍යතාවනට අනුව කැඳවීම පහසුවෙන් කල හැක.

උදා- print ( ) , input ( )

එසේම ඕනෑම ශ්‍රිතයක් ක්‍රියාත්මක වීම මගින් ප්‍රතිදානයක් පිටතට ලබා දිය යුතු වේ.

<u>අවශ්‍යතා මත ශ්‍රිතයන් නිර්මාණය කර ගැනීම.</u>

සාමාන්‍ය ආකාරයෙන් ශ්‍රිතයක් නිර්මාණය කර ගැනීමත් එම ශ්‍රිතය භාවිතා කිරීම හෙවත් කැඳවීමත් සිදු කරන අන්දම පහත ක්‍රියාකාර කම මගින් දැක්වේ.

```
>>> def display():
    print("Hello Function")



>>> display()
Hello Function
>>>
```

ඕනෑම ශ්‍රිතයක් නිර්මාණය කර ගැනීම 'def' මුරපදය මගින් සිදුකරනු ලබන අතර ඉන් පසුව එම ශ්‍රිතයට අදාල නම ලබා දෙනු ලැබේ. එම ශ්‍රිත නාමයන් ලබා දීමේදී හඳුන්වන හෙවත් විචල්‍ය නාම ලබාදීමේදී අනුගමනය කරනු ලබන නීතීන්ට අනුවම එය සිදු කරනු ලබයි. ඒ අනුව ඉහත උදාහරණයේ දැක්වෙන ශ්‍රිතයේ නම 'display' වේ. ශ්‍රිතයේ නම ලබාදීමෙන් පසු වරහන් යුගලක් ඇතුළත් කරන අතර ඒවා තුළ ශ්‍රිතය ක්‍රියාත්මක වීමට අදාළ පරාමිතීන් ඇතුලත් කරනු ලබයි (පරාමිතීන් සමඟ ශ්‍රිතයන් නිර්මාණය කරනා ආකාරය ඉදිරියේදී විස්තර කරනු ඇත.). ඉහත උදාහරණයේ වරහන් තුළ එවැනි පරාමිතික අගයන් ලබා දී නොමැත. වරහන් වලින් පසු ශ්‍රිතය හඳුන්වා දෙනු ලබන පද පෙළ අවසානය දැක්වීමට කෝලන් සලකුණ ලබාදේ. ඉන් පසු ශ්‍රිතය තුළ ක්‍රියාත්මන විය යුතු විධානයන් ඇතුලත් කරනු ලබන අතර එම කොටස අනුජේදනයක් මගින් වෙන් කොට දක්වයි. ඉහත උදාහරණය මගින් ලබා දී ඇති ශ්‍රිතය ක්‍රියාත්මක වී 'Hello Function' යන පණිවිඩය පිටතට ලබා දේ.

<u>පරාමිතීන් රහිත ශ්‍රිත.</u>

ශ්‍රිතයක කැඳවීම ශ්‍රිත නාමය සහ වරන් තුළ එම ශ්‍රිතයේ පරාමිතීන් ලබා දෙමින් සිදු කරනු ලබයි. නමුත් ඉහත උදාහරණයේ දැක්වෙන ශ්‍රිතය පරාමිතීන් රහිත ශ්‍රිතයක් වන අතර එය display() කැඳවා ඇත. එම කැඳවීමට අනුව ශ්‍රිතය ක්‍රියාත්මක වි 'Hello Function' පණිවිඩය පිටතට ලබාදේ. (පරාමිතීන් සහිත ශ්‍රිතයන් කැඳවීම කරනා ආකාරය ඉදිරියේදී විස්තර කරනු ඇත.). මෙසේ විවිධ අවශ්‍යතා සඳහා නිර්මාණය කරනු ලබන ශ්‍රිතයන්ගේ මූලික ස්රූපයන් මත පහත දැක්වෙන ලෙස ඒවා කාණ්ඩගත කල හැක.

අනිවාර්ය පරාමිතික ශ්‍රිත (Required Arguments).

මෙවැනි ශ්‍රිතයන් නිර්මාණය කිරීමේදී වරහන් තුළ පරාමිතීන් ලබාදෙන අතර, එම ශ්‍රිතයන් කැඳවීමේදී අනිවාර්යෙන්ම පරාමිතික අගයන් සහිතව කැඳවීම සිදුකල යුතුවේ. එසේ නොවුනහොත් එය ක්‍රමලේඛණ ව්‍යාකරණ දෝෂයක් ලෙස හඳුනාගනු ලබයි. එසේම ශ්‍රිතය කැඳවන අවස්ථාවේදී පරාමිතික අගයන් ලබා දෙනු ලබන ස්ථානීය අනුපිලිවෙලට ඒවා ශ්‍රිතය තුලට ඇතුලත්වේ. එය පහත උදාහරණ මගින් පැහැදිලි කොට ඇත.

```
>>> def cal(a,b):
    z=a+b
    print(z)
>>> cal(16,5)
21
>>>
```

එතා දක්වා ඇති උදාහරණයේ cal නම් ශ්‍රිතය a,b නම් ශ්‍රිතයන් දෙක සමඟ නිර්මාණය කර ඇත. එම ශ්‍රිතය කැඳවීමේදී (cal(16,5)) වරහන් තුළ ලබාදී ඇති අගයන් වන 16 , a සඳහාත් 5, b සඳහාත් ස්ථානීයව ආදේශ වේ. මෙම අගයන් ලබා නොදුනහොත් එය ක්‍රමලේඛණ ව්‍යාකරණ දෝෂයක් ලෙස හඳුනාගනු ලබයි.

මූරපද පරාමිතික ශ්‍රිත (Keyword Arguments).

ශ්‍රිතය කැඳවන අවස්ථාවේ මෙහි ක්‍රියාකාරිත්වය සිදු වේ. ශ්‍රිතය කැඳවීමේදී අදාල පරාමිතික අහයන් ස්ථානීය අනුරූපීව ආදේශ වන ආකාරය ඉහත පැහැදිළි කිරීම මගින් දක්නට ලැබිණ (අනිවාර්ය පරාමිතික ශ්‍රිත (Required Arguments)). නමුත් ශ්‍රිතය කැඳවන අවස්ථාවේ ස්ථානීය අනුපිළිවෙල නොතකමින් ශ්‍රිතය සහ ඒවාට අදාළ අගයන් ලබා දෙමින් ක්‍රියාත්මක කිරීම මෙහිදී සිදුවේ.

```
>>> def call(x,y):
    w=x*y
    print(w)
```

ස්ථානීය අනුපිළිවෙල නොතකමින් ශ්‍රිතය කැඳවා ඇත. ඒ අනුව ශ්‍රිත නාමය ශ්‍රිතය කැඳවන අවස්ථාවේ මුල් පදයක ස්වරූපයෙන් ක්‍රියාත්මක වි ඇති බව පෙනේ.

```
>>> call(y=5,x=2)
10
>>>
```

මූලික පරාමිතික ශ්‍රිත (Default Arguments).

ඇතැම් අවශ්‍යතාවන්ට අනුව ශ්‍රිතය නිර්මාණය කරන අවස්ථාවේම ශ්‍රිතයට අදාලව මූලික අගයන් ලබා දෙමින් ශ්‍රිතය ගොඩනැංවීම සිදුකරනු ලබයි. පහත උදාහරණයේ දැක්වෙන ශ්‍රිතයේ x=5,y=9 සහ z=10 ලෙස පරාමිතීන් තුනක් ඒවාට මූලික අගයන් ලබා දෙමින් නිර්මාණය කර ඇත. තවදුරටත් එම ශ්‍රිතය විවිධ පරාමිතික අවස්ථාවන්ට අනුව හැසිරෙන ආකාරය පහත දක්වා ඇත.

```
>>> def cal (x=5,y=9,z=10):
    w=x+y+z
    print(w)
```

පරාමිතක අගයන් නැතිව කැඳවා ඇත.

```
>>> cal()
24
```

එක් පරාමිතක අගයක් සහිතව කැඳවා ඇත. එම අගය x පරාමිතිය සඳහා ස්ථානියව ආදේශ වී ඇති අතර අනෙක් පරාමිතීන් මුල් අවස්ථාවේ ලබා දෙන ලද අගයන් සමඟ ක්‍රියාත්මක වී ඇත.

```
>>> cal(6)
25
```

z=15 පරාමිතිය පමණක් නව අගයක් සහිතව කැඳවා ඇත. අනෙක් පරාමිතීන් මුල් අවස්ථාවේ ලබා දෙන ලද අගයන් සමඟ ක්‍රියාත්මක වී ඇත.

```
>>> cal(z=15)
29
>>>
```

ශ්‍රිතයන් තුලින් ප්‍රතිදානයන් ලබා ගැනීම.

ඕනෑම ශ්‍රිතයක් ක්‍රියාත්මක වීම මගින් ප්‍රතිදානයක් පිටතට ලබා දිය යුතු බව ශ්‍රිත පිළිබඳ හඳුන්වා දීමේදීම ප්‍රකාශ කරන ලදී. ඒ අනුව ශ්‍රිතයක් තුලින් ප්‍රතිදානයන් ලබා ගන්නා ප්‍රධාන ආකාර දෙක පිළිබඳ පහත දක්වා ඇත.

print() නිර්මිත ශ්‍රිතය භාවිතයෙන්.

print() නම් පයිතන් භාෂාවේ නිර්මාණය කොට ඇති ශ්‍රිතය භාවිතයෙන් අගයන් පිටතට ලබා ගන්නා ආකාරය ඉහත උදාහරණ තුලින් ඔබට දක්නට ලැබිණි. නමුත් මෙසේ **print()** ශ්‍රිතය මගින් පිටතට ලබාදෙනු ලබන විචල්‍ය සහ එම අගයන් වෙනත් ක්‍රියාකාරකම් සඳහා යොදා ගත නොහැක. එසේ වන්නේ එම විචල්‍ය අගයන් පිටතට පැමිණෙන්නේ **none** දත්ත ප්‍රරූපය තුලින් බැවිණි. එම

නිසා **print()** ශ්‍රිතය මගින් ශ්‍රිතයක ප්‍රතිදානයන් නිරූපනය කිරීමක් පමණක් සිදුකරනු ලබන බව ප්‍රකාශ කල හැක.

```
>>> def output(c,d):
    r=c/d
    print(r)
```

ඉහත ශ්‍රිතය කැඳවන අවස්ථාවේ **type()** නිර්මිත ශ්‍රිතය යොදාගෙන දත්ත ප්‍රරූපය පරික්ෂා කොට ඇත. එය 'NoneType' දත්ත ප්‍රරූපයට අයත් බව පැහැදිලි වේ.

```
>>> type(output(10,2))
5.0
<class 'NoneType'>
>>>
```

<u>return () නිර්මිත ශ්‍රිතය භාවිතයෙන්.</u>

ඉහත ශ්‍රිතයේම return () ශ්‍රිතය මගින් පිටතට ලබා ගෙන දත්ත පරික්ෂා කළ හොත් එය නිවැරදි දත්ත ප්‍රරූපයෙන්ම ප්‍රතිදානය බව පැහැදිලි වේ (**float** මගින්).

```
def output(c,d):
    r=c/d
    return(r)
>>> type(output(10,2))
<class 'float'>
```

<u>ශ්‍රිතයන් තුළ විචල්‍ය</u>

ශ්‍රිතයක් තුළ විචල්‍ය සහ ඊට පිටතින් ඇති විචල්‍ය අතර ක්‍රියාකාරිත්වය අනුව ඒවා ප්‍රධාන කාණ්ඩ දෙකකට වෙන් කළ හැක. එනම් ස්ථානීය විචල්‍ය සහ ගෝලීය විචල්‍ය ලෙසටය.

ස්ථානීය විචල්‍ය

මේවා ශ්‍රිතයක් තුළ නිර්මාණය වේ. මෙම විචල්‍ය බලාත්මක වන්නේ එම ශ්‍රිතය තුළ පමණකි. ශ්‍රිතයෙන් පිටතට එහි අගය ක්‍රියාත්මක නොවේ.

```
>>> def sum(a,b):
    s=a+b
```

```
    print(s)
>>> sum(5,8)
13
>>> print(s)
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    print(s)
NameError: name 's' is not defined
>>>
```

ඉහත උදාහරණයේ ශ්‍රිතය තුළ නිර්මාණය කර ඇති විචල්‍ය අතරින් s විචල්‍යහි අගය හෙවත් එකතුව, විචල්‍ය කැඳවීමෙන් පිටතට ලබා දෙනු ලැබේ. නමුත් ඉන් පසු සාමාන්‍ය පරිදි print() මගින් එය පරික්ෂා කල විට එවැනි ශ්‍රිතයක් නොමැති බවට ප්‍රකාශ වන දෝෂයක් දිස් වේ. ඒ අනුව s විචල්‍ය ශ්‍රිතය සමඟ පමණක් ක්‍රියාත්මක වේ.

ගෝලීය විචල්‍ය

විචල්‍යකින් පිටත සාමාන්‍ය ක්‍රමලේඛනය තුළ නිර්මාණය වන මෙම විචල්‍ය ශ්‍රිතයන් තුළත් විචල්‍ය තුළත් ක්‍රියාත්මක වේ. පහත උදාහරණය මගින් එම තත්වය තව දුරටත් පැහැදිලි වනු ඇත.

```
>>> s=35
```

පළමුව s නම් විචල්‍යක් සාමාන්‍ය ක්‍රමලේඛනය තුළ ගෝලීය විචල්‍යක් ලෙස නිර්මාණය කර ඇත.

```
def sum(a,b):
    p=a+b+s
    print(p)
```

sum() ශ්‍රිතය s විචල්‍ය ද භාවිතා කර ගනිමින් p නම් විචල්‍ය අගය ප්‍රතිදානය කරනු ලබයි. ඒ අනුව ශ්‍රිතය කැඳවීම පහත දක්වා ඇත.

```
>>> sum(6,9)
50
>>>
```

ශ්‍රිතය කැඳවීමේදී පරාමිතික අගයන් ලෙස a සහ b අගයන් පමණක් ලබා දෙන අතර ගෝලීය විචල්‍ය වන s හි අගය ශ්‍රිතය තුළ ද වෙනසකින් තොරව ක්‍රියාත්මකවී අගය ප්‍රතිදානය කරයි.

ඉහත දක්වා ඇති අවස්ථා දෙක සඳහා අගයන් පිටතට ගැනීමට print() වෙනුවට return() ශ්‍රිතය යොදා ගනු ලබුවද ප්‍රතිදානයේ වෙනසක් ඇති නොවේ.

උසස් පෙළ විෂය නිර්දේශයට අනුව ඉහත සඳහන් කොටස් පිළිබඳ පයිතන් ක්‍රමලේඛන භාෂාව තුලින් අධ්‍යනය කල යුතුවේ. තවද පයිතන් භාෂාව මගින් පහත දැක්වෙන ක්‍රියාකාරකම් පිළිබඳව දැනුමද තිබිය යුතු බව විෂය නිර්දේශය මගින් නියම කර ඇත.

ගොණු සමඟ මෙහෙයුම් සිදු කිරීම.

මෙහිදී ගොනුවක් තුළ ඇති දත්ත කියවීම, ලිවිම සහ වෙනස් කිරීම ආදී ක්‍රියාකාරකම් පිළිබඳ විමසා බලනු ලැබේ. මෙම ක්‍රියාකාරකම් සඳහා පයිතන් භාෂාව තුළ නිර්මාණය කර ඇති ශ්‍රිතයන් භාවිතා කරනු ලබන අතර ඒවා පිළිබඳ හඳුන්වා දීමක් පහත දක්වා ඇත.

ගොනු සමඟ කටයුතු ආරම්භ කිරීමට පෙර ඒවා විවෘත කර ගත යුතුවේ. ඒ සඳහා **open ()** නිර්මිත ශ්‍රිතය භාවිතා කරන අතර එක් එක් ක්‍රියාකරකම් සඳහා ගැලපෙන ක්‍රමය **(mode)** භාවිතයෙන් ගොණු විවෘත්ත කිරීම සිදුකල යුතු වේ.

r - ගොනුව තුළ ඇති දත්ත කියවීමට පමණක් විවෘත කිරීම සිදු කරයි.(reading)

w - ගොනුව තුළ දත්ත ඉවත්ව අලුත් දත්ත ඇතුළු කරයි. .(wrting)

a - ගොනුව තුළ ඇති දත්ත අවසානයට අලුතින් දත්ත ඇතුළු කරයි. .(appending)

ඉහත ක්‍රමයන්ට අනුව ගොණු විවෘත කිරීමේ ශ්‍රිතය ක්‍රියාත්මක කරන ආකාරය පහත දක්වා ඇත.

ඉහත අකුරු යොදා ගනිමින් දෙවන පරාමිතිය ලෙස ක්‍රමය(mode) නිරූපණය කල හැක. පරාමිතීන් දෙක කොමාවන් භාවිතයෙන් වෙන් වෙන් වශයෙන් දැක්විය යුතුවේ.

Open (“ගොනු නාමය”, “ක්‍රමය”)

මෙසේ ගොඩනගනු ලබන ශ්‍රිතය මගින් ප්‍රතිදානය කරනු ලබන දත්ත විචල්‍යකට ඇතුලත් කරගනු ලබයි. එවිට ශ්‍රිතයෙන් යම් යම් ක්‍රියාවන්ට දායක කරගැනීමේදී එම එම විචල්‍යයේ නම (හඳුන්වන නාමය) පමණක් භාවිත කල හැක.

Name = open (‘file name’ , ‘mode’)

මෙම **open ()** ශ්‍රිතය අවසානයේ එම ක්‍රියාවලිය අවසාන බව ඇඟවීමට close ( ) නම් ශ්‍රිතය භාවිතා කිරීම සිදුකරනු ලබයි.

පහත ක්‍රියාකාරකම් සිදුකර ඇත්තේ **read** නම txt ගොණුවක් භාවිතා කරමින් වන බැවින් ඔබද පහත ක්‍රියාකාරකම් අත්හදා බැලීමට පෙර එලෙස ගොණුවක් නිර්මාණය කර ගන්න. එම ගොණුවත් අදාල පයිතන් ක්‍රමලේඛනයත් එකම ෆෝල්ඩරයක් තුළ සුරැකීමට **(save)** මතක තබා ගන්න.

<u>read නම් ගොණුව විවර කර එහි දත්ත කියවීම.</u>

add = open("read.txt","r") # add විචල්‍යට **read** ගොනුවේ දත්ත කියවීමට හැකි ලෙස ඇතුළු කරයි.

for line in add: # add විචල්‍යයේ අගයන් **line** විචල්‍ය තුලට **for** ව්‍යුහය මගින් ඇතුලත් කරයි.

    print(line.rstrip())   # **for** පුනර්කරණ තුළ **line** විචල්‍යයේ අගයන් ප්‍රතිදානයකරයි.

add.close()   # ක්‍රියාවලිය අවසාන කරයි.

rstrip() - ශ්‍රිතය පිළිබඳ සහ තවත් ශ්‍රිතයන් කිහිපයක් පිළිබඳ පහත විස්තර කර ඇත.

<u>read නම් ගොණුව තුළ දත්ත ඉවත් කර අලුත් දත්ත ඇතුළු කිරීම.</u>

fopen = open("read.txt", "w")   # fopen විචල්‍යට **read** ගොනුවේ දත්ත ලිවීමට හැකි ලෙස ඇතුළු කරයි.

fopen.write("new data , new data ")   # write **()** ශ්‍රිතය මගින් වරහන් තුළ දත්ත ගොනුවට ඇතුළු වේ.

fopen.close()   # ක්‍රියාවලිය අවසාන කරයි.

<u>read නම් ගොනුවේ අවසානයට දත්ත එකතු කිරීම.</u>

fh = open("read.txt","a")   # fopen විචල්‍යට **read** ගොනුවේ දත්ත එකතු කල හැකි ලෙස ඇතුළු කරයි.

fh.write("add the end")   # write **()** ශ්‍රිතය මගින් වරහන් තුළ දත්ත ගොනුවේ අවසානයට ඇතුළු වේ.

fh.close()   # ක්‍රියාවලිය අවසාන කරයි.

<u>rstrip()/lstrip()/strip() / split(" ")ශ්‍රිතයන්හි හැසිරීම.</u>

rstrip()/lstrip()/strip() මේවා දත්තයන් පිටතට නිරූපනය කිරීමේදී ඇති හිස් අවකාශයන් පාලනය කරනු ලබයි._split(" ") මගින් යම් සලකුණක් ලබා දෙමින් එම ස්ථානයෙන් දත්ත පෙළ වෙන් කල හැක.

```
Python 3.2.3 (default, Apr 11 2012, 07:12:16) [MSC v.1500 64 bit (AMD64)] on
win32

Type "copyright", "credits" or "license()" for more information.

>>> a= " python "

>>> a

'  python  ' # සාමාන්‍ය ලෙස.

>>> a.rstrip()

'  python'  # දකුණු පස හිස් අවකාශය ඉවත්ව ඇත.

>>> a.lstrip()

'python  ' # වම් පස හිස් අවකාශය ඉවත්ව ඇත.

>>> a.strip()

'python'#  දෙපසම හිස් අවකාශය ඉවත්ව ඇත.

>>>

>>> a= "a,b,c".split(",")           # a යන විචල්‍යට ලබාදෙන a,b,c දත්ත පෙළ “,”
                                     සලකුණ ඇති තැනින් වෙන වී  “list” දත්ත
>>> print (a)                        වර්ගයක් ලෙස  ආදේශ වේ.

['a', 'b', 'c']

>>> type(a)

<class 'list'>
```