## අ.පො.ස (උසස් පෙළ)

## තොරතුරු හා සන්නිවේදන තාක්ෂණය

ගැටලු විසදීම සදහා පරිගණක භාවිතය

# පයිතන් ක්‍රමලේඛ භාෂාව

B . P ද සිල්වා

අධ්‍යයන පොදු සහතික පත්‍ර (උසස් පෙළ )
තොරතුරු හා සන්නිවේදන තාක්ෂණය

ගැටලු විසඳීම සඳහා පරිගණක භාවිතය

# පයිතන් ක්‍රමලේඛ භාෂාව

B . P ද සිල්වා.

ප්‍රථම මුද්‍රණය
2014

පිට කවරය සැකසුම: B.P ද සිල්වා

**©සියළු හිමිකම් ඇවිරිණි**
කර්තෘගෙන් ලිඛිත අවසරයක් නොලබා මෙම පොත සම්පූර්ණයෙන්ම හෝ කොටසක් වශයෙන් හෝ පිටපත් කිරීම, නැවත මුද්‍රණය කිරීම හා බෙදාහැරීම සපුරා තහනම් වේ.

# පෙර වදන

2011 වසරේ සිට නිර්දේෂ වූ අධ්‍යයන පොදු සහතික පත්‍ර (උසස් පෙළ ) විභාගය සඳහා තොරතුරු හා සන්නිවේදන තාක්ෂණ විෂය සිසුනට තවමත් අලුත් විෂයකි. මන්දයත් මීට පෙර පැවැත්වුණු අ.පො.ස උ.පෙළ විභාගය සඳහා සිසුන්ගේ ප්‍රතිඵල වල සාධනය අවම අගයක පවතී. මාගේ දැක්ම අනුව, මේ සඳහා හේතුවක් නම්, සිසුන්ගේ පරිශීලනය සඳහා විෂයානුබද්ධව සිංහල මාධ්‍යයෙන් ලියවී ඇති පොත්පත් ප්‍රමාණය අල්පයක් වීමයි.

මෙම හේතුව මත පදනම්ව ලියවී ඇති මෙම ග්‍රන්ථය සිසුන්ගේ දැනුම වර්ධනය කිරීමෙහිලා පිටුවහලක් වෙතැයි මාගේ විශ්වාසයයි. අ.පො.ස උ.පෙළ විභාගය සඳහා සිසුනට ලැබෙන ප්‍රශ්න පත්‍රයේ වඩාත්ම නියෝජනයක් දක්වන, විෂය නිදේෂයේ 7 වන ඒකකය අලලා මෙම ග්‍රන්ථය ලිය වී ඇත.
උ.පෙළ විෂය නිදේෂයට අදාල 7 වන ඒකකය සඳහා සම්මත වන ක්‍රමලේඛ භාෂාව පයිතන් (Python) වේ. මෙම ක්‍රමලේඛ භාෂාවට අදාල දැනුම් කොටස් සරලව, ග්‍රන්ථ පරිශීලකයාට තේරුම් ගැනීමට හැකි වන ලෙස, සාම්ප්‍රදායික ග්‍රන්ථ ලිවීමේ බස වෙනුවට කථන බස, මෙම ග්‍රන්ථය ලිවීම සඳහා භාවිත කර ඇත. මන්දයත් ග්‍රන්ථ පරිශීලකයාට වඩාත් සමීපව, මෙම දැනුම් කොටස් වටහා දීම පිණිසය.

තවද මෙම ග්‍රන්ථය ලිවීම සඳහා පාදක වී ඇත්තේ පරිගණක විද්‍යාව සම්බන්ධව හසල වූ දැනුමක් ඇති ඇමෙරිකානු ජාතික බී.රොබට්ස් මහතාගේ පයිතන් ක්‍රමලේඛ ලිවීම සඳහා වූ පාඩම් මාලාවකිනි. මේ හේතුවෙන් ඔබට, පයිතන් ක්‍රමලේඛ භාෂාව පිළිබඳ මූලික වූ සිද්ධාන්ත සියුම් ලෙස ඉගෙනුමට මහඟු පිටුවහල් වන්නේය.

අවසාන වශයෙන්,මෙම ග්‍රන්ථය සඳහා වදන් සැකසීමට මා හට සහය වූ ජ්ල්. නිමලසූරිය මහතාට මාගේ විශේෂ ස්තූතිය හිමිවේ.

**B.P ද සිල්වා.**

ගණිත අංශය,
ධර්මරාජ විද්‍යාලය,
මහනුවර.
2014.02.22

# පටුන


01. Installing Python (පයිතන් ස්ථාපිත කිරීම) ....................................................................1

02. Numbers and Maths (සරල ගණිත කර්ම) ....................................................................3

03. Variables ( විචල්‍යයන් ) ....................................................................5

04. Modules and Functions( මොඩියුල සහ ශ්‍රිත) ....................................................................7

05. How to save your programmes (ක්‍රමලේඛ සුරකීම) ....................................................................9

06. Strings (වදන්) ....................................................................14

07. More on String ( වදන් තවදුරටත් ) ....................................................................16

08. RAW Input (වදන් ප්‍රදානය ) ....................................................................18

09. Sequences and lists ( අනුක්‍රම සහ ලැයිස්තු ) ....................................................................19

10. Slicing ( කොටස් කිරීම ) ....................................................................21

11 . Editing Sequences ( අනුක්‍රමයන් සැදීම ) ....................................................................23

12. More List Functions (ලැයිස්තු ආශ්‍රිත ශ්‍රිත තවදුරටත්) ....................................................................25

13. Slicing Lists (ලැයිස්තු කොටස් කිරීම) ....................................................................27

14. Intro to Methods (රීතීන් සඳහා ප්‍රවිෂ්ඨය ) ....................................................................29

15. More Methods ( රීතීන් තවදුරටත් ) ....................................................................31

16. Sort and tuples (පෙළගැස්වීම සහ ටපල්ස් ) ....................................................................32

17. Stings and Stuff ( වදන් ආශ්‍රිත මෙවලම්) ....................................................................34

18. String Methods ( වදන් ආශ්‍රිත රීති තවදුරටත් ) ....................................................................36

19. Dictionary (වදන් කෝෂ) ....................................................................37

20. If Statements (if ප්‍රකාශන) ....................................................................40

21 Else and elif (Else සහ elif අනුකේතන) ....................................................................42

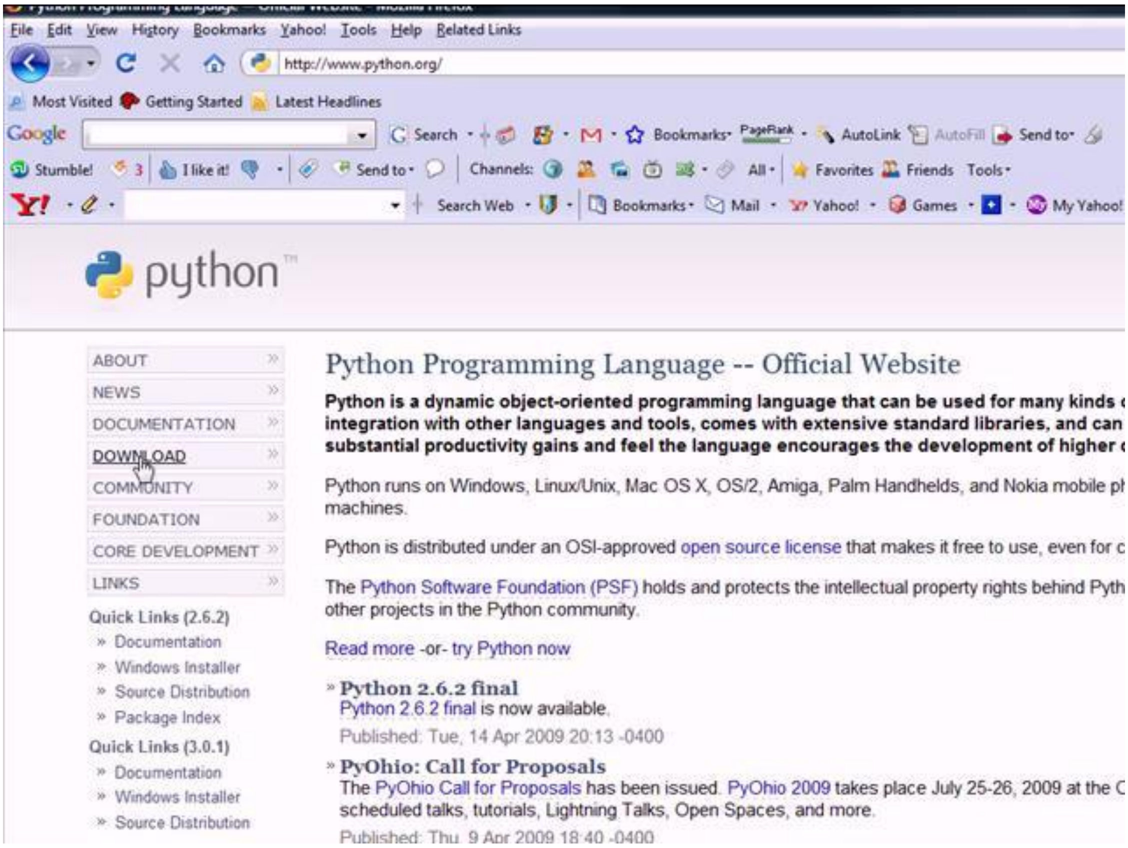
22. Nesting Statements (අනු-ප්‍රකාශන) ....................................................................44

# 1. Installing Python (පයිතන් ස්ථාපිත කිරීම)

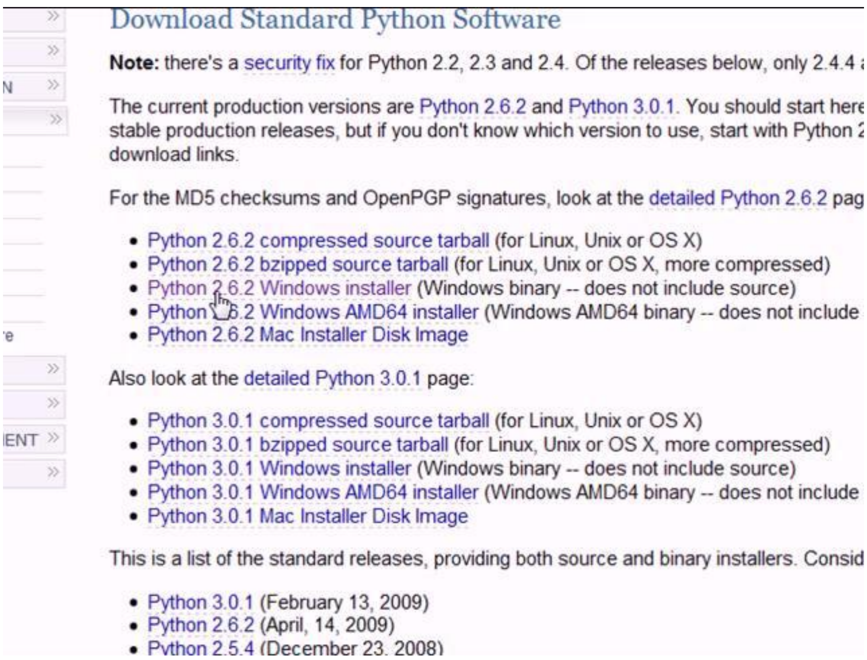
අපි ඉස්සෙල්ලාම බලමු, Python Download කරගන්න හැටි. මේ Programme එකෙන් තමයි, Python ක්‍රමලේඛ (Python codes ) ලියන්නේ.

ඉස්සෙල්ලාම ඔයාගේ Web browser එකට ගිහින් , www.python.org කියල type කරන්න.

Page එක load උනාට පස්සේ page එකේ වම් පැත්තේ tabs ටිකක් තියෙනවා. එතන තියෙන DOWNLOAD කියන tab එක click කරන්න.




එතන තියෙන python 2.6.2 windows install එක click කරන්න. මොකද මම ඔයාට python 2.6.2 version එකෙනුයි programme ලියන හැටි උගන්වන්නේ. python 3.0 වගේ අලුත් versions තිබුණත් python 2.6 වගේත් version එකකින් programme ලියන්න දන්නවනම් අලුතෙන් එන ඕනම version එකකින් programme ලිවිමේ හැකියාව අපිට ලැබෙනවා. එක් හරියට මේ වගේ, “පරණ ජාතියක කාර් එකක් එලවන්න දන්නවනම් , අලුත් ජාතියේ කාර් එකක් එලවන එක මහලොකු දෙයක් නෙවෙයි.”



හරි දැන් web page එකෙන් දෙන file එක download කරගෙන install කර ගන්න.

ඊට පස්සේ ඔයාගේ පරිගණකයේ Start menu එකට ගිහින් All programmes වලට යන්න.

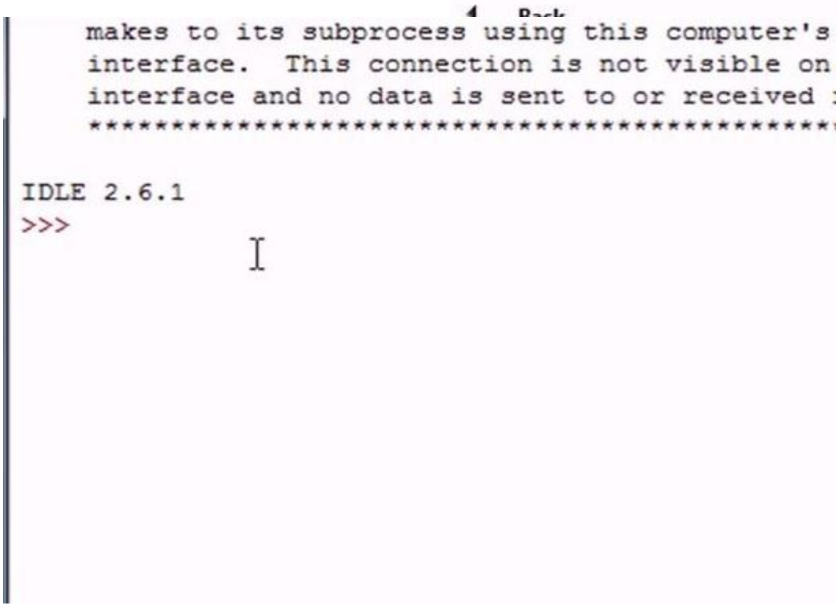
ඊට පස්සේ ඔයාට දකින්න ලැබෙයි , Python 2.6 කියල folder එකක් . ඒ යටතේ ගොඩක් දේවල් තියෙනවා නේද ?

ඉස්සෙල්ලාම යන්න , IDLE ( Python GUI ) එකට . ඒක තමයි Python වල අතුරු මුහුණත (Graphical User Interface)

හරි දැන් IDLE එක Open වුණා නේද ? මම දැන් කියල දෙන්නම්, IDLE එක පාවිච්චි කරන විදිහ.

ඉස්සෙල්ලාම අපි test කරල බලමු IDLE එක වැඩද කියල.

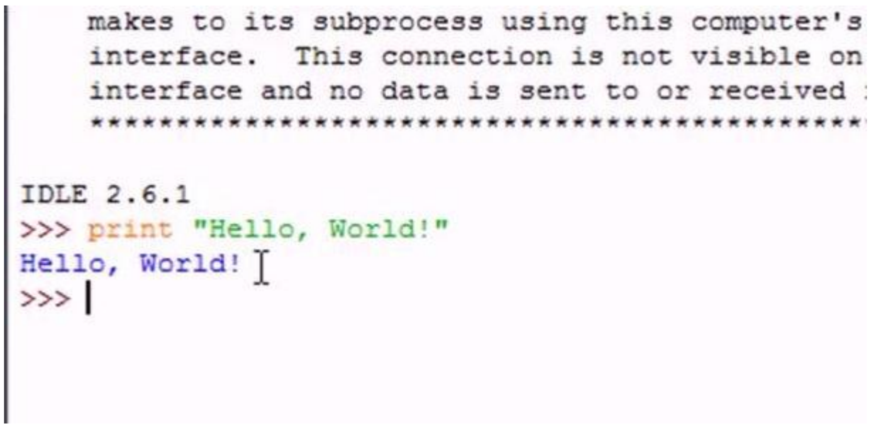
print කියල ලියල , උඩුකොමා ඇතුලේ " Hello World ! " කියල ලියමු.




print " Hello World ! "

Hello World !

අපිට Python වලින් output එකක් ලැබුනා.




Python කියන්නේ ඇත්තටම ගොඩක් ආකර්ෂණීය programme language එකක්. අපිට මේකෙන් ගොඩක් වැඩගන්න පුළුවන්,. ඒ වගේම නූතනයේ බහුලව පාවිච්චි වෙන programme language එකක් තමයි Python කියන්නේ.

# 2. Numbers and Maths (සරල ගණිත කර්ම)

මේ පාඩමේදී මම කතා කරන්න යන්නේ Python වල තියෙන සරල ගණිත කර්ම ගැන.

Python IDLE එක ඇත්තටම calculator එක විදිහට පාවිච්චි කරන්න පුලුවන්.

2+2=4

6-3=3

18/3=6

18/7=2

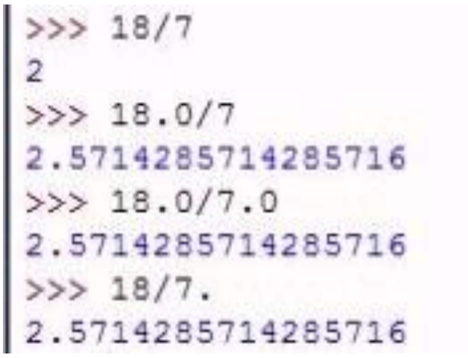
හරියටම බෙදුනේ නෑ නේද ?

නිඛිලයකින් නිඛිලයක් බෙදුවහම Python වලින් Output එක එන්නේ නිඛිලයකින්ම තමයි. ඒකයි 18/ 7 = 2 එන්න හේතුව.

ඒ හින්දා හරියටම 18 / 7 ගන්න නම් දශම තිතත් යොදන්න වෙනවා. එතකොට Python වලිනුත් දෙන්නේ දශමය සමඟ Output එකක්.

දශමය තියෙන සංඛ්‍යාවලට Computer Languages වල සාමාන්‍යයෙන් කියන්නේ Floats කියලයි.

හරි දැන් පිළිතුරට දශමය එන්න හදනවනම් 18 / 7 කියන එකේ කොහෙට හරි දශම තිතත් තියන්න වෙනවා. අපි ඒ ගැන බලමු.




18.0/7=2.5714285714285716

හරි නේද උත්තරේ ආවා නේද ?

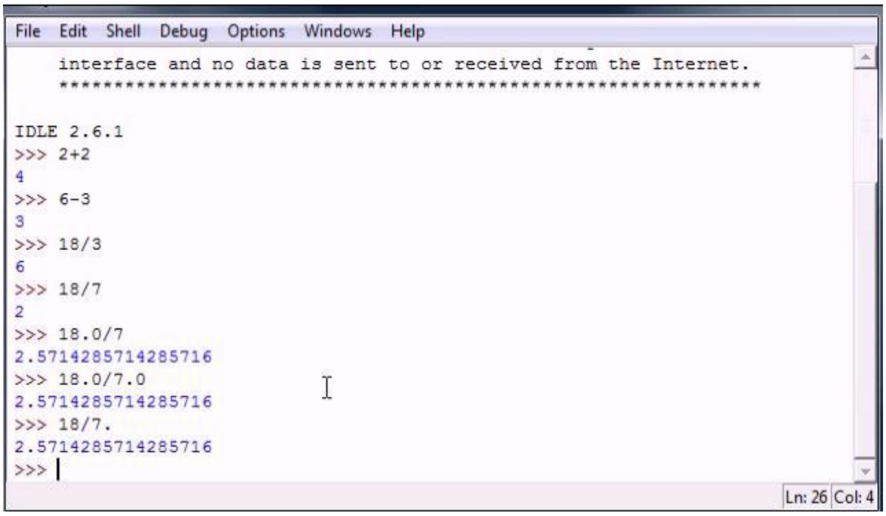
ඒ වගේම මෙහෙම කරන්නත් පුලුවන්

18.0/7.0=2.5714285714285716

18/7. =2.5714285714285716

මේකෙදි අවශ්‍ය වන්නේ ගණිතකරණයේ කොහෙට හරි දශමයක් තියෙන එක විතරයි.

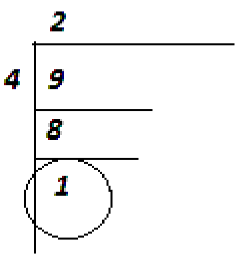
මේකෙදි තියෙන වැදගත්ම දේ තමයි Python වල **නිඛිල සංඛ්‍යා** (Integer) හෝ **දශමය තියෙන සංඛ්‍යා**(Float) අතර ගණිතකර්ම කරද්දී Integer එක Float වලට පරිවර්තනය වෙලා Output එක අවසානයේදී එන්නේ Float වලින්. මේක Python වල තියෙන විශේෂ ලක්ෂණයක්.



# % - Mod (මොඩ්)

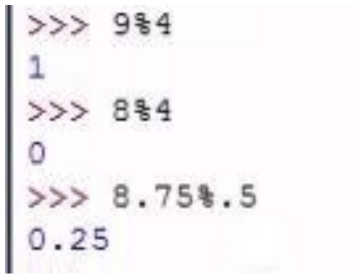
මේ ගණිත කර්මය ඉතාමත් වැදගත් එකක්. Python වල මෙම ගණිත කර්මය තියෙනවා. මේකෙදි කරන දේ මම පෙන්නන්නම්.

අපි, 9 / 4 බෙදමු.




මෙන්න ඉතුරු 1 ක් ආවා නේද ? අපි ඒකට **ශේෂය (Reminder)** කියල කියනවා.

මෙන්න මේ ශේෂය තමයි **% - Mod** වලින් අපිට Output එක විදිහට දෙන්නේ.




8 % 4 මේකෙ ශේෂය 0 නේද ?

ඔව් 0 යි.

එතකොට 8.75 % 5

0.25 හරි නේද ?

සාරාංශය ගත්තොත් % එකෙන් අපිට දෙන්නේ යම් සංඛ්‍යාවක් තවත් සංඛ්‍යාවකින් බෙදුවහම ඉතිරිවන අගයයි. (**ශේෂය**)

දැන් අපි බලමු ගුණ කිරීම ගැන

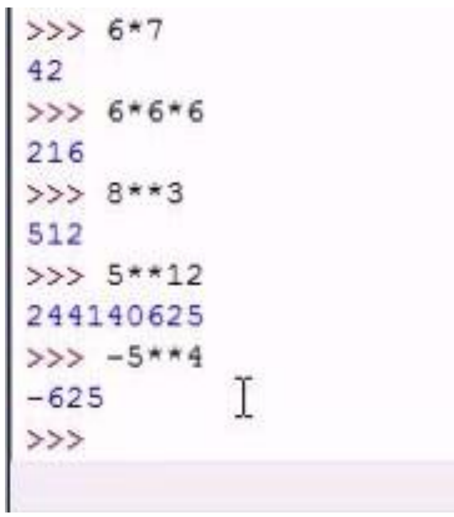
6 * 7

42

අපිට 6 ඒවා 3 ක් ඕන

එහෙනම් , 6*6*6

216




ඒත් ඊටත් වැඩිය ලෙහෙසි ක්‍රමයක් තියෙනවා . ඒක කරන්නේ මෙහෙමයි.

| | | |
|---|---|---|
| 8**3 | 5**3 | -5**3 |
| 512 | 24414062 | -625 |

හරි අපි ඉගෙන ගත්ත Python වලින් සරල ගණිතකර්ම කරන හැටි

# 3. Variables ( විචල්‍යයන් )

Python වල හැටියට **Variable** එකක් කියන්නේ, ඕනම දෙයක්( string , numbers ,.. ආදි ) **තාවකාලිකව තැන්පත් කිරීමක්** .

කොහොමද අපි Variable එකක් හදන්නේ , ඉස්සෙල්ලාම Variable එකේ නම ලියන්න. ඊට පස්සේ එයට කුමකට හෝ සමාන කරන්න.

**X = 18**

හරි මේකෙදි X කියන නම 18ට සමාන කරල තියෙනවා.

දැන් Enter එක ඔබන්න.

හරි දැන් **තාවකාලිකව පරිගණකයේ මතකයට** X = 18ට සමාන වෙනවා. දැන් 18 වෙනුවට අපිට X කියන නම භාවිතා කරන්න පුලුවන්.

X + 18 = 33 මම කියපු දේ හරි නේද ?

X**3= 5832

Y = 54 ට සමාන කරමු.

හරි දැන් අපිට Variables 2ක් තියෙනවා . මේ දෙකම පාවිච්චි කලොත්

X + Y = 72

```
>>> x = 18
>>> x + 15
33
>>> x ** 3
5832
>>> y = 54
>>> x + y
72
```

මේවා ඉතාමත් වැදගත්, අපි හිතමු, නිතරම භාවිතා වෙන දෙයක්, ඒක සාමාන්‍යයන් ලොකු අංකයක් කියලා ( 878 වාගේ එකක් ) මේක නිතරම type කරන්න යාම බොරුවට කාලේ නාස්ති කිරීමක්. ඒ නිසා තනි අගයකට ( Z = 878 වැනි ) සමාන කරගත්තහම ඒක ලේසියෙන් නැවත නැවත භාවිතා කරන්න පුලුවන්. Variable වල තියෙන වැදගත්කම ඔයාලට ඉස්සරහට බලා ගන්න පුලුවන්.

අපි නිකන් හිතමු, අපි හදන programme එකකට User ගෙන් Input එකක් ගන්න වෙනවා. ඒක කරන්නේ කොහොමද ? ඇත්තටම ඒක කරන්න මේ Variables පාවිච්චි කරල තමයි. අපි ඒ ගැන බලමු.

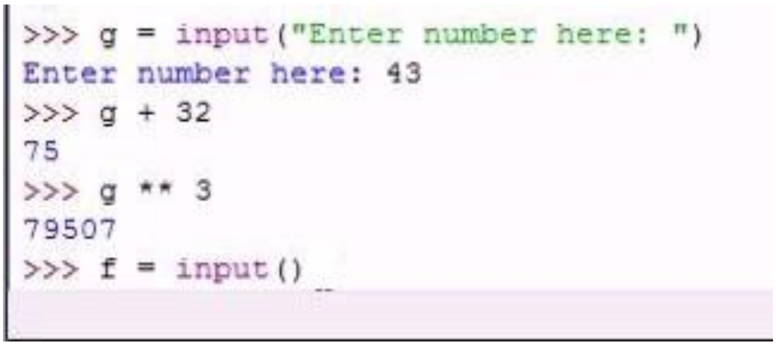
මේකෙදි Input කියල Function එකක් භාවිතා වෙනවා

g = input (“ ”) දෙපැත්තට ( ) **වරහන් දාලා** ( “...... “ ) **උඩු කොමා** යටතේ **User** ට **දකින්න ඕන Message** එක අපිට ලියන්න පුලුවන්.

g = input (“Enter number here: ”)

දැන් Enter එක ඔබමු.

දැන් User දකින Message එක අපිට පේනවා.




Enter number here:

අපි දැන් 43 කියලා type කරමු.

දැන් g = 43 ට සමාන වෙලයි තියෙන්නේ.

අපි දැන් බලමු

g + 32 = 75

g**3 =79507

සාරාංශය බැලුවොත්, Variable එකක් අපිට හදන්න පුලුවන්, දැන්. ඒ වගේම User ගෙන් ලැබෙන input එකක් Variable එකකට සමාන කරගන්න පුලුවන්.

( F = input ( ) )

# 4. Modules and Functions( මොඩියුල සහ ශ්‍රිත)

සාමාන්‍යයෙන් **ක්‍රමලේක භාෂා (Programme Languages)** වලට අනුව **ස්ථාපිත ශ්‍රිත (Building Function)** කියන්නේ **ක්‍රමලේක භාෂාව ලිවීමට ගන්නා මෘදුකාංගය** තුලම පිහිටුවනු ලැබූ ක්‍රමලේක වලට. Python වලටත් මේ දෙය පොදුයි.

5**4= 625

හරි, මේ වැඩේම ලෙහෙසියෙන් කරන්න, Python වල Function එකක් තියෙනවා.

අපි කොහොමද Function එකක් පාවිච්චි කරන්නේ.

ඉස්සෙල්ලාම Function එකේ නම ලියන්න ඕනේ.

ඊට පස්සේ **Parameters (පරාමිතීන්)** නම් කරන්න ඕන.

පරාමිතීන් කියන්නේ කුමන පරාසයද, මේ අගයන් විචලනය විය යුත්තේ යන්නයි.

හරි අපි ඒ ගැන බලමු.

කිසියම් සංඛ්‍යාවක බලය ගන්න Function එකක් Python වල තියෙනවා.

ඒකට කියන්නේ **Pow ( )**

```
>>> 5**4
625
>>> pow(5,4)
625
>>> abs(-18)
18
>>> abs(5)
5
```

වරහන් ඇතුලේ දාන්නේ ඉස්සෙල්ලාම, අදාල සංඛ්‍යාව.
ඊට පස්සේ **එම සංඛ්‍යාව කුමන බලයෙන් වැඩිකල යුතුද** කියලා.
ඒ කියන්නේ බලය තමයි දෙවනියට දාන්නේ.

Pow ( 5 , 4 ) = 625

**abs ( )**

මේකෙන් කරන්නේ යම් සංඛ්‍යාවක මාපාංකය ලබා දීමයි.
ඒ කියන්නේ input එක සෘණ සංඛ්‍යාවක්( - ) දුන්නොත් Output එක එන්නේ ධන( + ) වලින්.

abs ( -18 ) = 18

abs ( 5 ) = 5

Python අතුරු මුහුණතට (Interface) තියෙන්නේ Function යම් ප්‍රමාණයක් පමණයි. එම නිසා අපිට තවත් Function අවශ්‍ය නම් **import ( ආනයනය )** කරන්න වෙනවා.

## Modules(මොඩියුල)

Modules වල තමයි මේ Functions තැන්පත් කරල තියෙන්නේ. Module එකක් කියන්නේ,
**ශ්‍රිත ගොනුවක් (A set of Functions)**.

හරි ඒ කතාව අමතක කරල අපි දැන් **floor ( )** කියන Function එක සලකමු. මේකෙදි කරන්නේ කිසියම් දශමය සංඛ්‍යාවක් මෙහි වරහන් ඇතුලට දාපුවහම දශමය සංඛ්‍යා ටික අයින් කරල දෙන එකයි.

උදා- 15.00111345 කියල දුන්නොත්

15.0 ලෙස අපිට ප්‍රතිදානය ලැබෙනවා

floor ( 18.7 ) කියල දීල බලමු. ඒක ආවේ නෑ නේද ? ඒකට හේතුව තමයි මීට අදාල Module එක අපි import කරපු නැති එකයි.

හරි ඒක import කරන්නේ මෙහෙමයි.

**import** math

math කියන module එකේ, ගණිතකර්ම වලට අදාල Functions list එකක්ම තියෙනවා.

හරි දැන් Enter කරන්න.

නමුත් දැනුත් floor ( ) කියල type කරන්න බෑ.

ඒත් ඒ Function එක පාවිච්චි කරන්නේ මේ විදිහටයි.

math.floor ( 18.7 )
18.0

math.sqrt ( 81)
9.0

```
>>> floor(18.7)

Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    floor(18.7)
NameError: name 'floor' is not defined
>>> import math
>>> math.floor(18.7)
18.0
>>> math.sqrt(81)
9.0
```

ඉතින් import Function වල මුලට module name එක ලියන්න වෙනවා.

ඒක ලියන්නේ මෙන්න මේ විදිහටයි.

<u>module.function</u>

එහෙම කරදරයි නම් ,
අපිට මේවා මොකේකට හරි සමාන කරගන්න පුලුවන්.

මෙන්න මේ වගේ.

bucky = math.sqrt

දැන් bucky කියන නමට math.sqrt සමාන කරල තියෙන්නේ. ඒක නිසා දැන් math.sqrt වෙනුවට buckey පාවිච්චි කරන්න පුලුවන්.

bucky (9)
3.0

තවත් ඒ විදිහට කරල බලමු.

bucky = math.floor
bucky (19.8 )
19.0

```
>>> import math
>>> math.floor(18.7)
18.0
>>> math.sqrt(81)
9.0
>>> bucky = math.sqrt
>>> bucky(9)
3.0
>>> bucky=math.floor
>>> bucky(19:8)
19.0
```

හරි නේද ? ඉතින් මේක ගොඩක් ප්‍රයෝජනවත්.

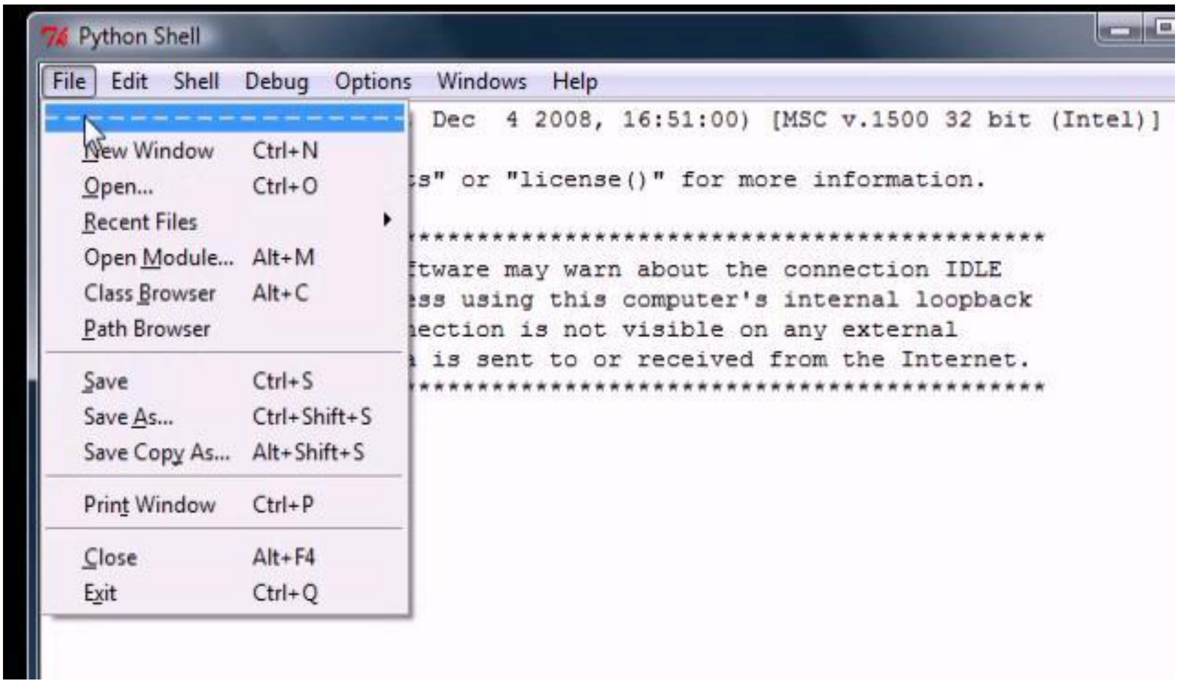
# 5. How to save your programmes (ක්‍රමලේඛ සුරකීම)

programme එකක් හදල save කරන හැටි අපි බලමු.

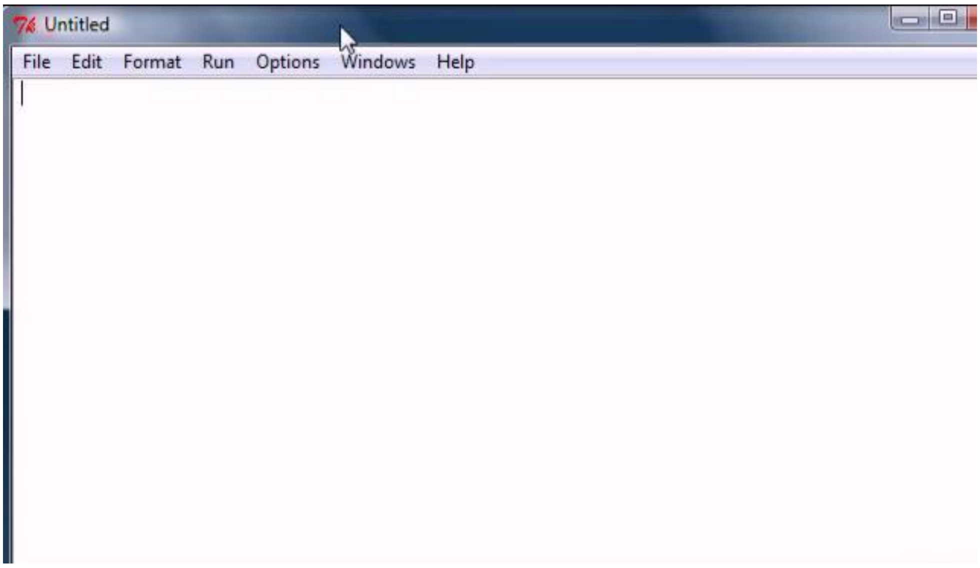
අපි තාමත් ඉන්නේ IDLE එකේ , මේක තමයි Python වල Interface එක. IDLE එකෙත් පුලුවන් type කරන දේවල් වල Output එක එවෙලේම බලාගන්න.

ඒත් මෙහි තියෙන ගැටලුව තමයි , මේකේ programme එකක් ලිව්වත් Python වලින් exit වෙනකොට අපි ලියපු සේරම නැති වෙනවා. ආයෙ අපිට ඒක ලබා ගන්න බෑ.

ඒකට විසඳුමක් තියෙනවා අපි ඒක ගැන කතා කරමු.

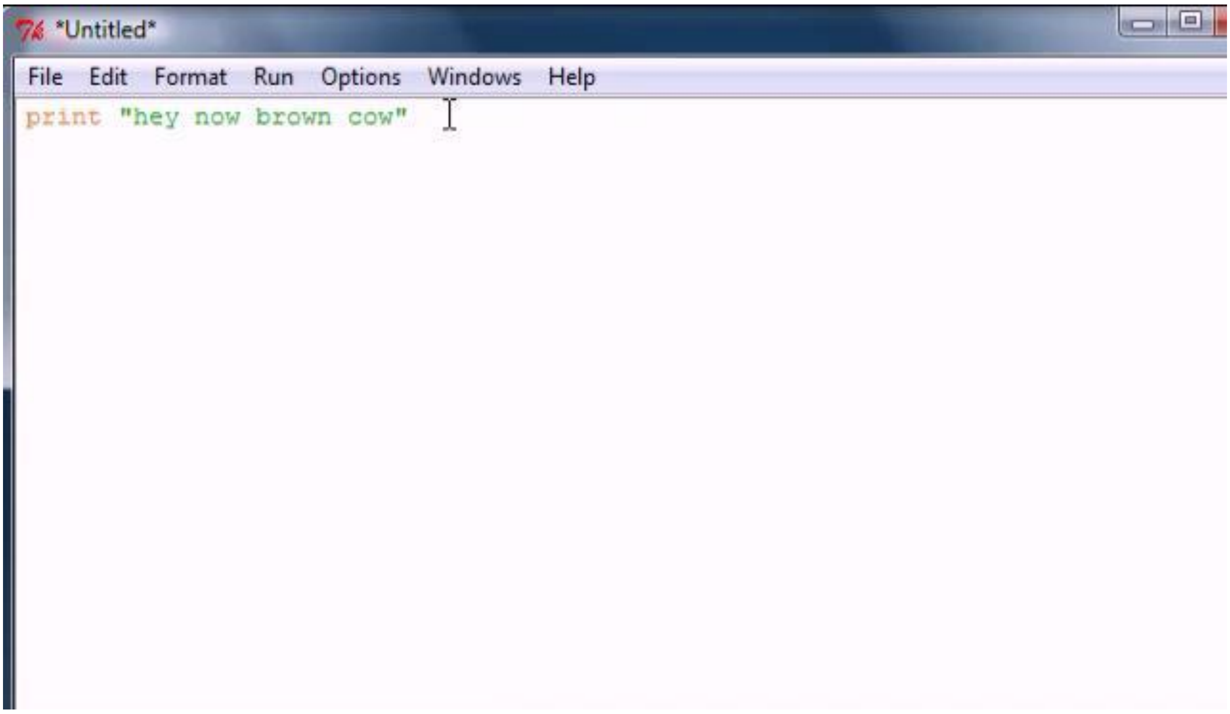



IDLE එකේ ඉන්නවනම් යන්න පුලුවන්. File --- New Window හරි දැන් අපිට ලැබෙන්නේ අලුත් කවුලුවක් , හරි මෙතනින් අපිට අවසර ලැබෙනවා programme එකක් save කරන්න. ඒ වගේම ඒ programme එක run කරන්න.

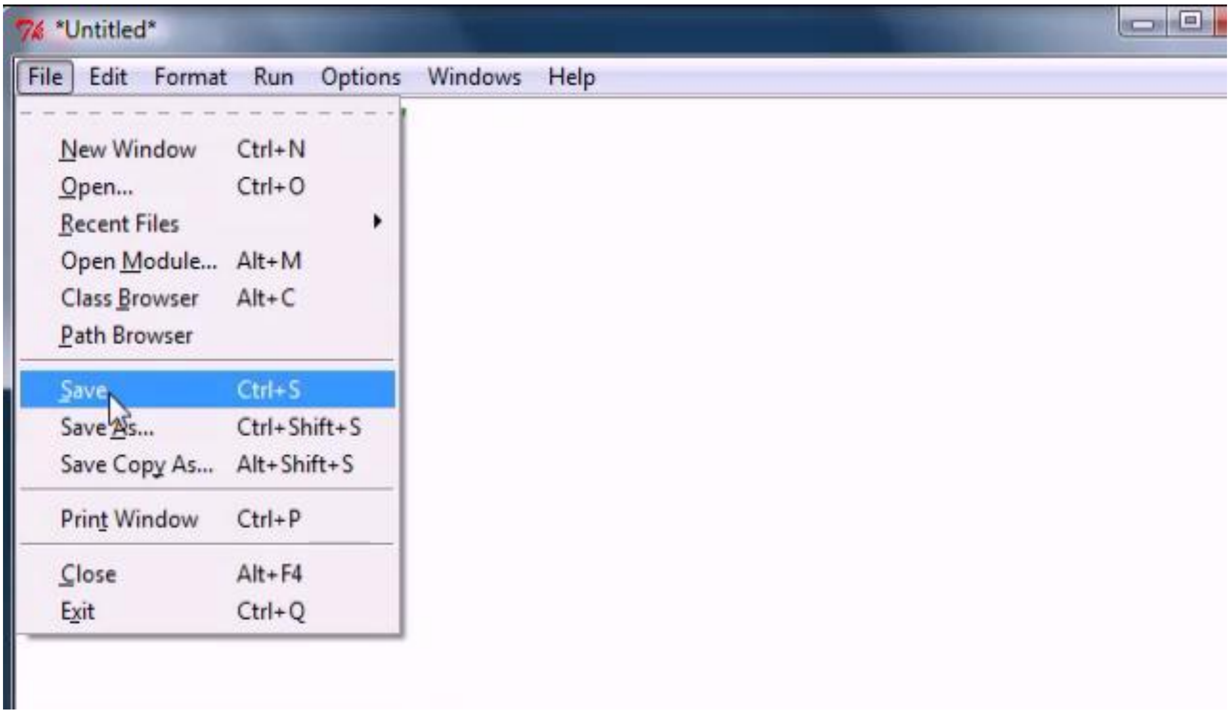


අපි මේ window එකේ දෙයක් ලියමු .

print ”hey now brown cow”




අපි මේක දැන් save කරල බලමු

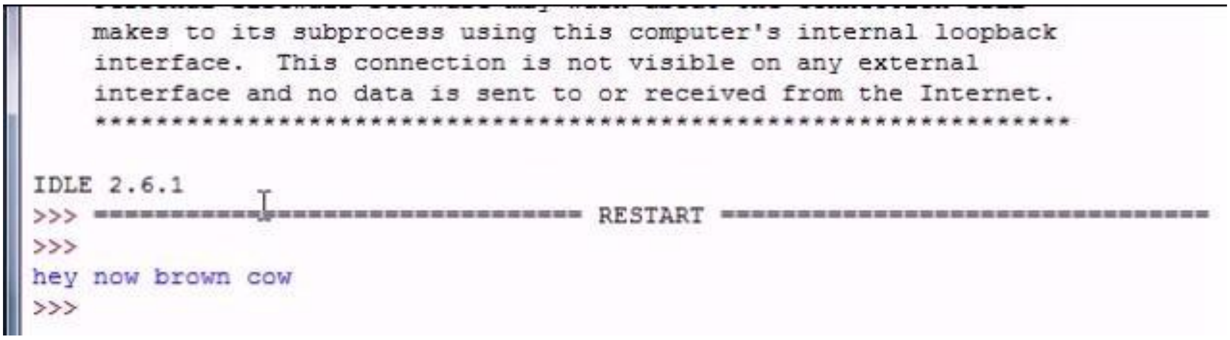



දැන් table එකක් එනවා. මෙතන ඒකට ඔයාල කැමති දෙයක් දෙන්න පුළුවන්. ඒත් අමතක කරන්න එපා file name එක අගට .py කියල යොදන්න .py කියල save කරපුවහම ඒක python file එකක් විදිහට save වෙනවා.

මේ window එකේ programme එකක් ලියල නිකම්ම run කරන්න බෑ. අනිවාර්යයෙන්ම save කරන්න ඕන.

දැන් අපි මේක run කරන හැටි බලමු . window එකේ menu එකට යන්න . ඊට පස්සේ run --- run module හෝ ලේසියෙන්ම keyboard එකේ F5 ඔබලත් programme එක run කරන්න පුළුවන්.

දැන් පේනවා නේද ? IDLE එක run වුනා




දැන් අපි පොඩි programme එකක් ලියමු.

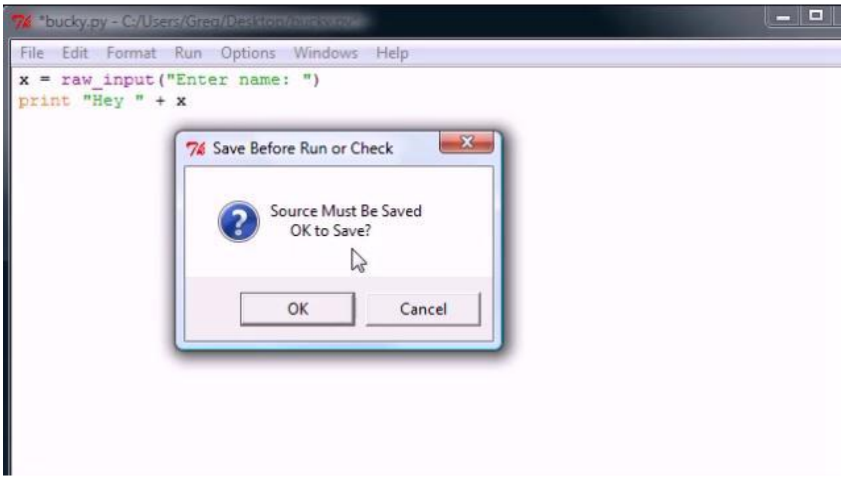
x = raw_input (“Enter name:”)

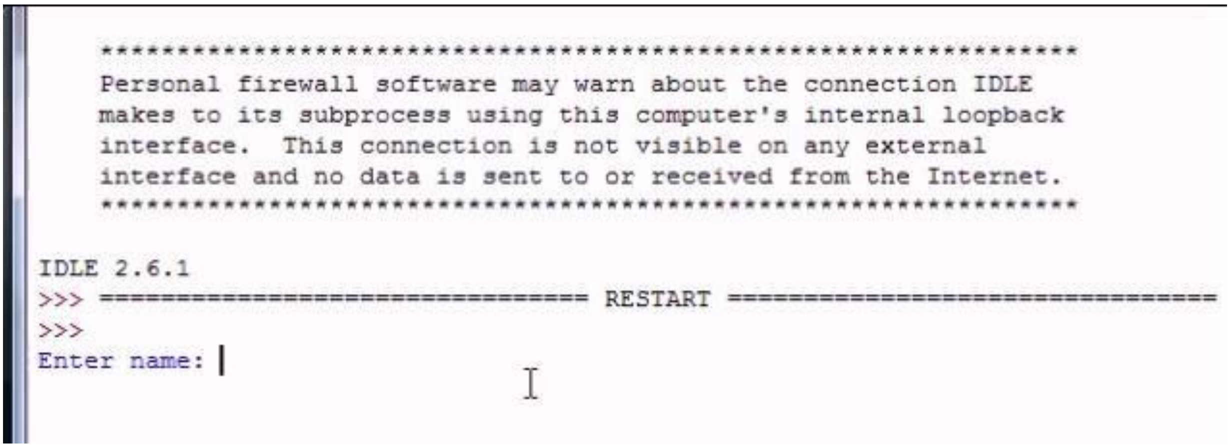
print “Hey”+x

මේකෙදී නිකම්ම input නැතුව raw_input කියල දාපුවහම x ට සමාන වෙන්නේ සංකේතයක් විදිහටයි.
(As a string) (මේ ගැන වැඩි දුරටත් පසුව සාකච්ඡා කරයි)

අපි දැන් මේක නිකම්ම run කරල බලමු. F5 ඔබන්න.

මම කිව්ව දේ හරි නේද ? Save කරන්නේ නැතුව Programme එක run කරන්න දෙන්නේ නෑ. දැන් ok කරන්න.

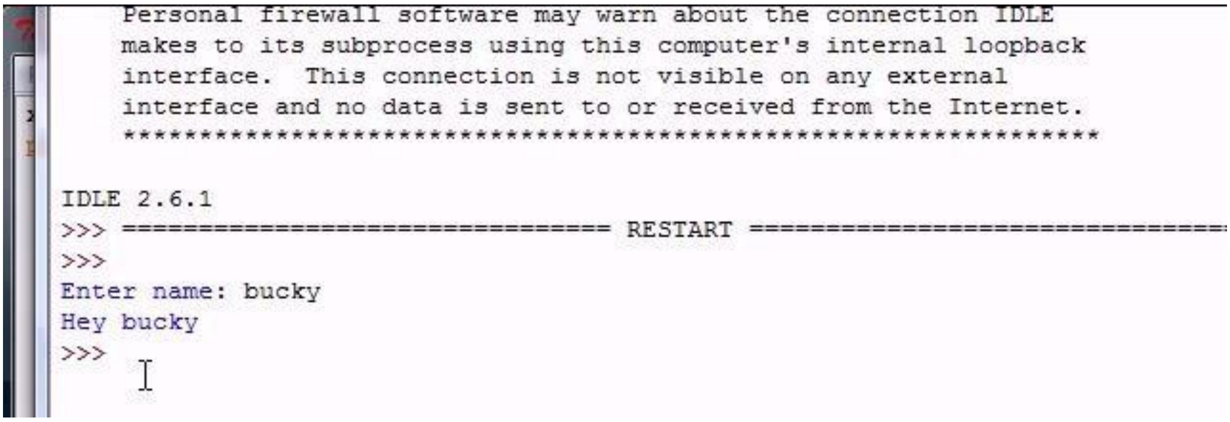





Enter name :

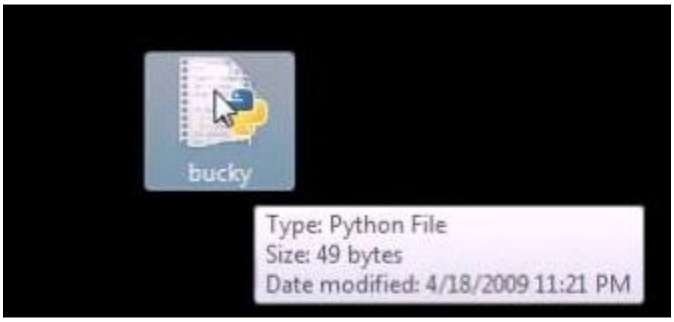
දැන් මේ විදිහට. එනවා . දැන් නමක් දාලා බලමු මේකට.

Enter name = bukey දැන් Enter ඔබන්න.




මේක ඇත්තටම පොඩි Programme එකක්.

ඒක නෙමෙයි දැන් Programme එක save කරපු තැනට ගියොත්,




මේක අපිට වෙන Programmes run කරන විදිහටම run කරන්න පුළුවන්. ( ඒ කියන්නේ Ms word, Adobe Photoshop Run කරනවා වගේම)

දැන් අපේ Programme එක Double click කරල run කරමු.

Programme එක run උනා නේද ?

හරි අපි නමක් දාලා බලමු

ඇත්තටම මොකද වුනේ ?

එක පාරටම Programme එක Exit වුනා නේද ?

ඇයි එහෙම කියල මම කියල දෙන්නම්.මොකක්ද මේ Programme එකෙන් කරන්නේ.

x = raw-input (“Enter name:”)

මේක run කරනවා

ඊට පස්සේ

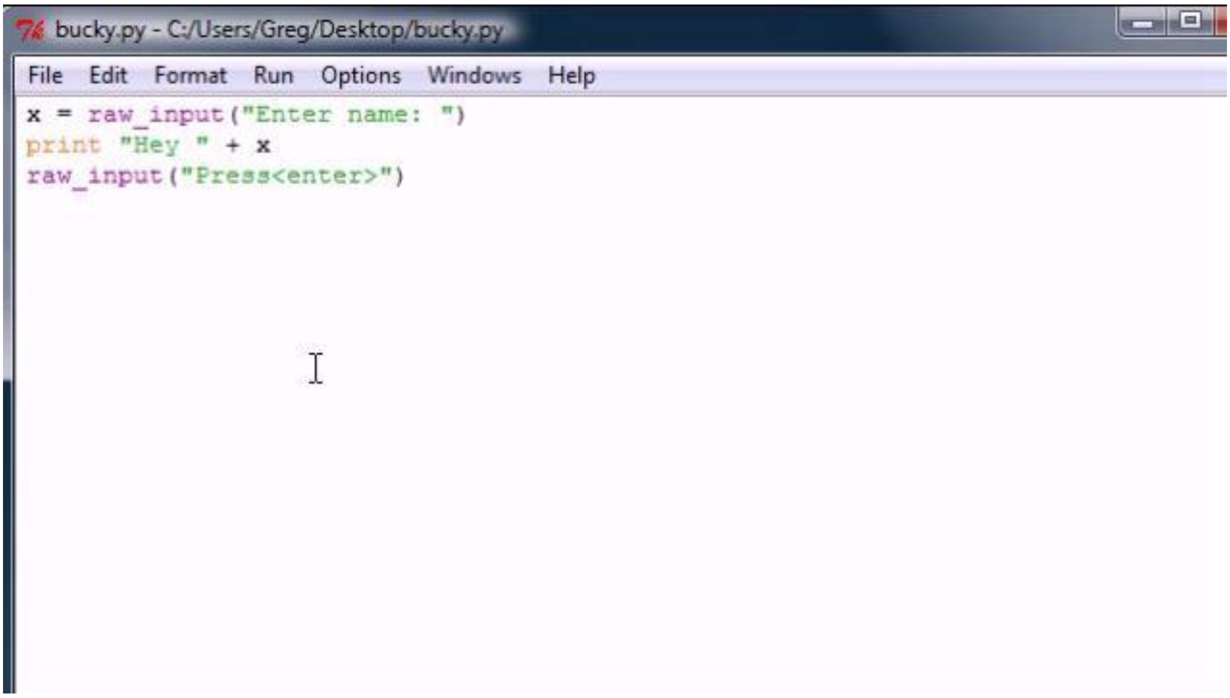
Print “ Hey ” + x

කියලත් run කරනවා.

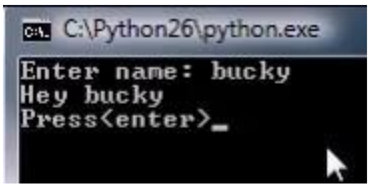
ඊට පස්සේ මොනවත් නෑ නේද ? ඉතින් Programme එක එයාගේ වැඩේ කරල ඉවරයි. Programme එක දැන් කියනවා “මගේ වැඩේ කරල ඉවරයි, මම යනවා” ඇත්තටම ඒක තමයි මෙතනදි වෙන්නේ.

ඉතින් අපි මොකද කරන්නේ මේ ප්‍රශ්නෙට. ඒක හරිම ලේසි දෙයක්.

raw-input (press <Enter>) කියල type කරමු.




දැන් අපි Programme එක run කරමු.




හරි ප්‍රශ්නෙට විසඳුමක ලැබුණා නේද ?

අන්තිමට සාරාංශයක් විදිහට ගත්තොත් IDLE එක ගොඩක් හොඳයි. එහෙත් අපිට Programme එකක් Save කරන්න නම් File එකක් හදන්න වෙනවා. ( New window ) එකට ගිහින් මතකයි නේද ඒක.

# 6. Strings (වදන්)

දැන් අපි ආයෙත් IDLE එකට යනවා.

IDLE එකේ type කරමු.

“ Hey now” දැන් Enter එක ඔබන්න.

‘ Hey now ‘ මේක String එකක්. මෙකද සංකේත කිහිපයකින් (characters) මේක හැදිල තියෙන්නේ.

‘ Hey now ‘
 මෙහිදී ද්විත්ව උඩුකොමා (Double codes) , නැතුව උඩු කොමා (Single codes)  පාවිච්චි කළත් එකම දේ තමයි වෙන්නේ.

```
IDLE 2.6.1
>>> "hey now"
'hey now'
>>> 'hey now'
'hey now'
>>> he's a jerk
```

වෙනසක් ඇත්තෙම නෑ.

ඒත් සමහර අවස්ථා වලදී තියෙන්න පුළුවන්, ප්‍රශ්න එන තැන්.

මම ඒ ගැන පෙන්වන්නම්.

he’s a jerk ’

‘he’ s a jerk

Python වලින්, String එකක් විදිහට යම් දෙයක් හඳුනාගන්නේ, උඩුකොමාව පටන් ගන්න තැන හා අවසන් වෙන තැනයි.

ඉතින් ද්විත්ව උඩුකොමා දැම්මොත් ඒ ප්‍රශ්න විසදෙනවා.

```
IDLE 2.6.1
>>> "hey now"
'hey now'
>>> 'hey now'
'hey now'
>>> "he's a jerk"
"he's a jerk"
>>>
```

ඒත් තවත් විදිහක් තියෙනවා . මේක ඉතාමත් වැදගත්.

‘ he\’s  a  jerk ’

```
>>> 'he\'s a jerk'
"he's a jerk"
```

පිටු පසට ඇල ඉර **[ back slash ( \ ) ]** යොදන්න ඕන, අපට මඟ හරින්න අවශ්‍ය කොමාව ඉදිරියෙන්.

Python එකෙන් බලනවා ( \ ) ට පස්සේ තියෙන සලකුණු දිහා ඊට පස්සේ හිතනවා , “හරි, මේ කොමාවේ ඇත්තම තේරුම නෙවෙයි මේ තියෙන්නේ. ඒක නිකම්ම print  කරන්න අවශ්‍ය දෙයක් විතරයි” කියල.

ඉතින් ‘ he\’s  a  jerk ’ ලියල Enter එබුවොත් output එක බලාගන්න පුළුවන් අපිට.

මේ වැඩේ අත්‍යාවශ්‍ය වන තැනක් බලමු දැන්. අපේ string එකට **ද්විත්ව උඩුකොමා** එකතු කරන්න අවශ්‍ය උනොත් අපි ඒක කරන්නේ කොහොමද ?

buckey said “ hey now “ to me

“ buckey said , “ hey now “ to me “

```
>>> "bucky said,"hey now" to me"
```

ඉතින් දැන් Python වලින් උඩුකොමා තියෙන තැන් බලනවා.

“ buckey said , “ hey now “ to me “

හරි පටන් ගද්දිම ද්විත්ව උඩුකොමා තියෙනවා.
ඊට පස්සේ said, “ කියන තැනින් string එක අවසන් වෙනවා. මොකද උඩුකොමාතෙ තියෙන්නේ. ඊට පස්සේ hey now “ to me “ වලින් තවත් එකක් පටන්ගෙන me “ කියන තැනින් ඉවරවෙනවා.
ඒත් අපි බලාපොරොත්තු උනේ ඒ දේ නෙමෙයි.
ඉතින් අපි මොකද කරන්නේ මෙහෙම ආපුවහම. වැඩේ හරිම ලේසියි.

```
>>> "bucky said,\"hey now\" to me"
'bucky said,"hey now" to me'
>>>
```

“ buckey said ,\ “ hey now\ “ to me “

පිටුපස ඇල කොමාවට පසුව තියෙන සලකුණ, Python වලින් නොසලකන නිසා, අපට අවශ්‍ය Output එක ලැබෙන්න ඕන දැන්.

දැන් strings එකතු කරන එක ගැන අපි බලමු.

```
>>> a, b
('bucky', 'roberts')
>>> a = "bucky "
>>> a + b
'bucky roberts'
```

a = “ buckey “
b= “ roberts “

a+b = ‘ buckeyroberts ’
a+b කියපුවහම Python වලින් කරන්නේ a හි තියෙන සලකුණු ටික ( characters ), b හි තියෙන සලකුණු ටිකට, එකතු කරල output කරන එකයි.

ඒත් අපිට අවශ්‍ය ( buckey roberts ) කියල මැදට හිස්තැනක් සහිතව මේ string දෙක එකතු කරන්න නම්, අපි මොකද කරන්නේ ?

a b කියල type කරමු. ඒක හරියන්නේ නෑ නේද ?

```
>>> a b
SyntaxError: invalid syntax
```

එහෙනම් a,b කියල type කරමු.
( ‘ buckey , ‘roberts’ )

```
>>> a, b
('bucky', 'roberts')
```

ඒත් අපි බලාපොරොත්තු වුන output එක නෙවෙයි මේ.

හරි කියන්නම්, ඇත්තටම මොකක්ද වෙන්න ඕන කියල.

a = “ buckey ” ( හිස්තැනක් සහිතව ඒක හදන්න )

දැන් බලමු

```
>>> a = "bucky "
>>> a + b
'bucky roberts'
```

a +b
‘ buckey roberts ‘

වැඩේ සාර්ථකයි නේද ? මේකෙන් ඉගෙන ගන්න දෙයක් තියෙනවා. මොකක්ද දන්නවද a +b කියපුවහම Python වලින් එකතු කරනවා a හා b string දෙකට අදාල සලකුණු ( characters ). හරි, අපි a = “ buckey “ කියල අන්තිමට හිස්තැනක් තිබ්බ. ඒකෙන් අපිට තේරෙන්නේ, එක හිස්තැනක් (space) Python වලින් character එකක් විදිහට සලකනවා කියලයි. මම හිතනවා, මම කියපු දේ ඔයාට තේරුණා කියලා.

අපි මේ ඉගෙන ගත්තේ string වල මූලික අඩිතාලම . ඉතින් python වලින් ඉස්සරහට යද්දී මේ අපිට strings ගොඩාක් වැදගත් වෙනවා .

# 7. More on String ( වදන් තවදුරටත් )

String වලින් කරන්න පුළුවන් තවත් දේවල් ටිකක්, කියල දෙන්නයි මේ යන්නේ.

num = 18

අපි, num = 18 විදිහට Variable එකක් හදමු.

```
>>> num = 18
>>> num + 16
34
```

num + 16

34

මේ විදිහට අපේ Variable එක පාවිච්චි කරල ලේසියෙන්ම අපිට එකතු කරන්න පුළුවන්.

print “ BUckey id “ + num කියල දෙමු.

මට අවශ්‍ය , BUckey 18 කියල print වෙන්නයි.

```
>>> print "BUcky id" + num

Traceback (most recent call last):
  File "<pyshell#42>", line 1, in <module>
    print "BUcky id" + num
TypeError: cannot concatenate 'str' and 'int' objects
>>> |
```

ඒත් වැඩේ හරියන්නේ නෑ වගේ. මොකද, string හා numbers එකතු කරන්න බැරි නිසා.

අපි ඒකට විසඳුමක් හොයමු.

num = str ( 18 ) කියල දෙමු.

මේකෙදි කරන්නේ, 18 කියල තියෙන්නේ, ඇත්තටම number එකක් නෙවෙයි, ඒක string එකක් කියල Python වලට තේරුම් කරන එකයි.

ඇත්තටම කීවොත් 18 , string එකක් විදිහට පරිවර්ථනය වෙනවා . මේ **str ( ) ,** function එක ඇතුලට දාපුවම

දැන් බලමු.

print “ buckey is “ + num

buckey is 18

```
>>> num = str(18)
>>> print "bucky is" + num
bucky is18
```

හරි මේ විදියටයි, අපි numbers, string වලට පරිවර්ථනය කර ගන්නේ. මේක පාවිච්චි කරන්න පුළුවන් user ගෙන් input එකක් ගන්නකොට, user ගේ input එක string එකක්ම වෙලා එන විදිහට.

අපි variable එකක් හදමු දැන්.

num 2 = 32

print “ my mom is “ + num2

අපි දන්නවා මෙහෙම කරන්න බෑ කියලා.

ඒත් str පාවිච්චි කරන්නේ නැතිව බැරිද මේ වැඩේ කරන්න . ඇත්තටම පුළුවන් ඒ වැඩේ කරන්නේ මෙහෙමයි.

ඔයාගේ key board එකේ black text කියන key එක දන්නවද ? දන්නැත්තම් දැනගන්න. ( ඒක තමයි Tab key එකට හරියටම උඩින් තියෙන key එක. පෙනුමෙන් ඇත්තටම උඩු කොමාවක් වගේ තමයි.)

දැන් අහන්න , print “ my mom is “ + ` num 2 `  කියන එකේ ` num 2 `  දෙපසට මේ ` ( black text ) දෙකක් දාන්න. දැන් Enter කරල බලන්න.

my mom is 32

```
>>> num2 = 32
>>> print "my mom is " + `num2`
my mom is 32
```

ඒ විදිහටයි num එකකුයි string එකකුයි ලේසියෙන්ම එකතු කරන්නේ.

තව දෙයක් තියෙනවා .
**repr ( )** , මෙන්න මේ function එක , මේක හාත්පසින්ම **str ( )** , function එක වගේමයි.

ඉතින් **str ( )** වෙනුවට අපිට ඕනෙනම් **repr ( )** , function එක හාවිතා කරන්න පුළුවන්.

```
>>> repr()
```

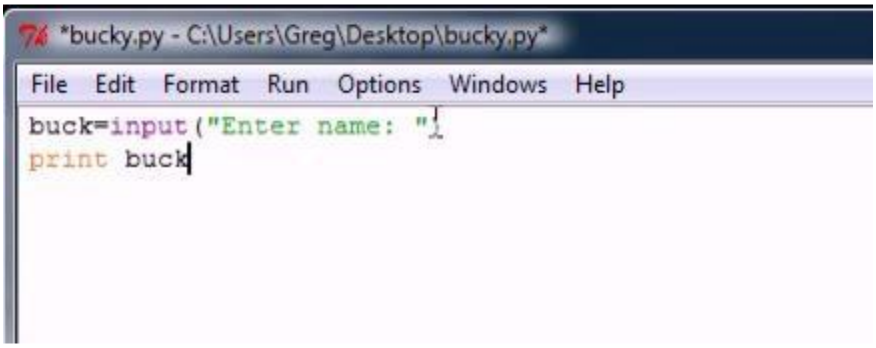
# 8. RAW Input (වදන් ප්‍රදානය )

**Raw_input( )** හා නිකම්ම **input ( )** යන function වල වෙනස මම කලින් කතා කලා.

ඒත් ඒ ගැන තවත් හොඳින් මම කියල දෙන්නම්.

මම දැන් ඉන්නේ Python, window එකේ මෙහෙම type කරමු.

buck= input("Enter name: ")
print buck

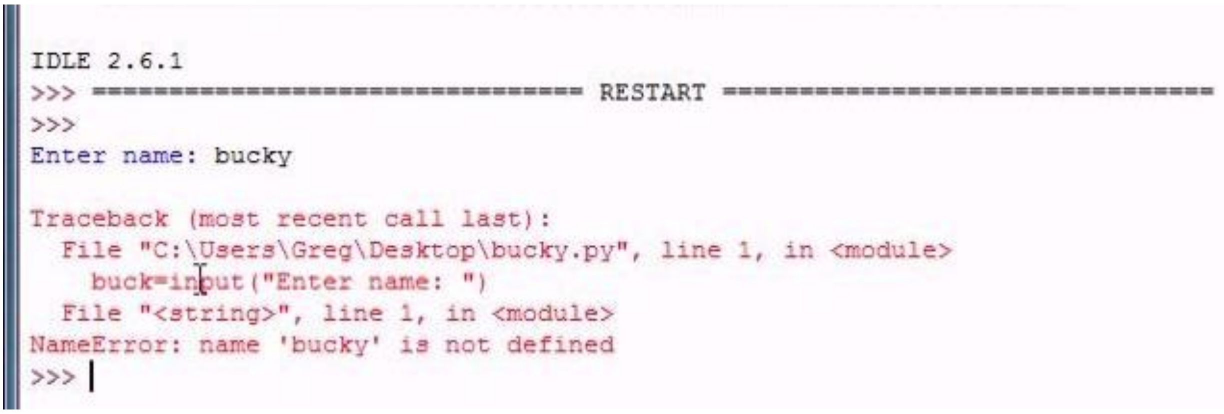


```
buck=input("Enter name: ")
print buck
```

දැන් මේක run කරල බලමු.

අහන නමට bukey අපි කියල දෙමු.

Enter name : bucky



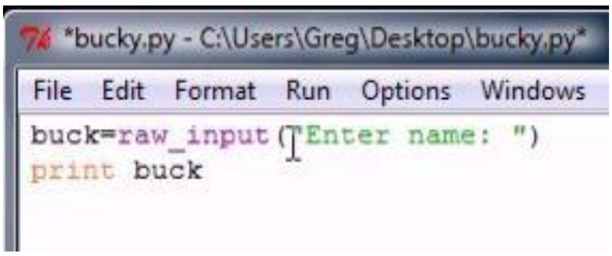
```
IDLE 2.6.1
>>> ================================ RESTART ================================
>>>
Enter name: bucky

Traceback (most recent call last):
  File "C:\Users\Greg\Desktop\bucky.py", line 1, in <module>
    buck=input("Enter name: ")
  File "<string>", line 1, in <module>
NameError: name 'bucky' is not defined
>>> |
```

මොකක්ද ප්‍රශ්නේ ? මේකයි , අපි වචනයක්, input එකට දුන්නා . ඒත් ඒක string විදිහට Python වලට දුන්න නැති නිසා, Python වලින් කෙලින්ම Print කරන්න දෙන්නේ නෑ, ඒක.

ඒකට විසඳුම තමයි , **raw_input ( )**

buck = raw_input ("Enter name:")

print buck



```
buck=raw_input("Enter name: ")
print buck
```

මේකෙදි වෙන්නේ user ගෙ input එක string එකක් විදිහට පරිවර්ථනය වෙන එකයි. දැන් run කරල බැලුවොත්.... දැන් වැඩේ හරි .

```
IDLE 2.6.1
>>> ================================ RESTART ================================
>>>
Enter name: bucky
bucky
>>> |
```

# 9. Sequences and lists ( අනුක්‍රම සහ ලැයිස්තු )

දැන් මම list එකක් හදල පෙන්වන්නම්. ඉස්සෙල්ලාම අපි හදන list එකේ නම ලියන්න ඕනේ.

family = [ 'mom' , 'dad' , 'bro' , 'sis' , 'dog' ]

ඊට පස්සේ කොටු වරහන් යොදලා ඒකේ ඇතුලේ list එකට අදාල අවයව ටික ලියන්න ඕනේ.

( mom , dad , bro , ..... කියන ඒවා නිසා අනිවාර්යයෙන්ම උඩුකොමා යොදන්න )

දැන් Enter එක ඔබපුවහම computer එකේ memory එකේ කොහෙහරි save වෙනවා මේ list එක. ඒක ප්‍රශ්නයක් නෙවෙයි.

හරි , මම මේ list එක ගැන පෙන්වන්නම්.

family = [ 'mom' , 'dad' , 'bro' , 'sis' , 'dog' ]

0 1 2 3 4

මේකේ තියෙන අවයව ගත්තහම **හැම අවයවයකටම** අදාල **( index number )** අංකයක් තියෙනවා. ඉතින් computer එකෙන්, පළවෙනි අවයවය ' 0 ' කියලයි නම් කරන්නේ. ඊට පස්සේ ඒවා 1 , 2 , 3 , ...... වන ලෙස පිළිවෙලට සලකුණු වෙනවා.

දැන් අපි බැලුවොත් ,

family [ 3 ]
' sis '

හොඳට මතක තියාගන්න මුල් අවයවය 0 කියලයි හඳුන්වන්නේ. ඒ නිසයි family [ 3 ] කියල type කරපුවහම 'sis 'කියල ආවේ.

අපි හිතමු ගොඩක් දිග list එකක් තියෙනවා. අපිට අවශ්‍යයි මේකේ අවයවයක් output කරගන්න. එතකොට ගොඩක් කරදරයිනේ. අවයව මුල ඉඳල ඒකේ index number එක ගණන් කර කර, හොයන්න. ඉතින් ඒකට ලේසි ක්‍රමයක් තියෙනවා.

ඒක තමයි. අන්තිම ඉඳල මුලට ගණන් කිරීම.

family = [ 'mom' , 'dad' , 'bro' , 'sis' , 'dog' ]

-5 -4 -3 -2 -1

```
IDLE 2.6.1
>>> family = ['mom','dad','bro','sis','dog']
>>> family[3]
'sis'
>>> family[-2]
'sis'
```

ඉතින් අන්තිම අවයවය - 1 කියලයි අපි ගන්නේ. ඉස්සෙල්ල නම් 0 න් පටන් ගත්තේ ඇයි මේකෙන් -0 න් පටන් ගන්න බැරිද ? කොහොමටවත් බෑ. -0 කීවත්, 0 කීවත් බින්දුව බින්දුවම තමයි.

ඉතින් පිළිවලට -1 , -2 , -3 , ....... කියල අපිට අන්තිමේ ඉඳල මුලට අවයව ටික නම් කරන්න පුළුවන්.

උදාහරණයක් ගත්තොත්

family [ -2 ]

' sis ' කියල අපිට output එක එනවා

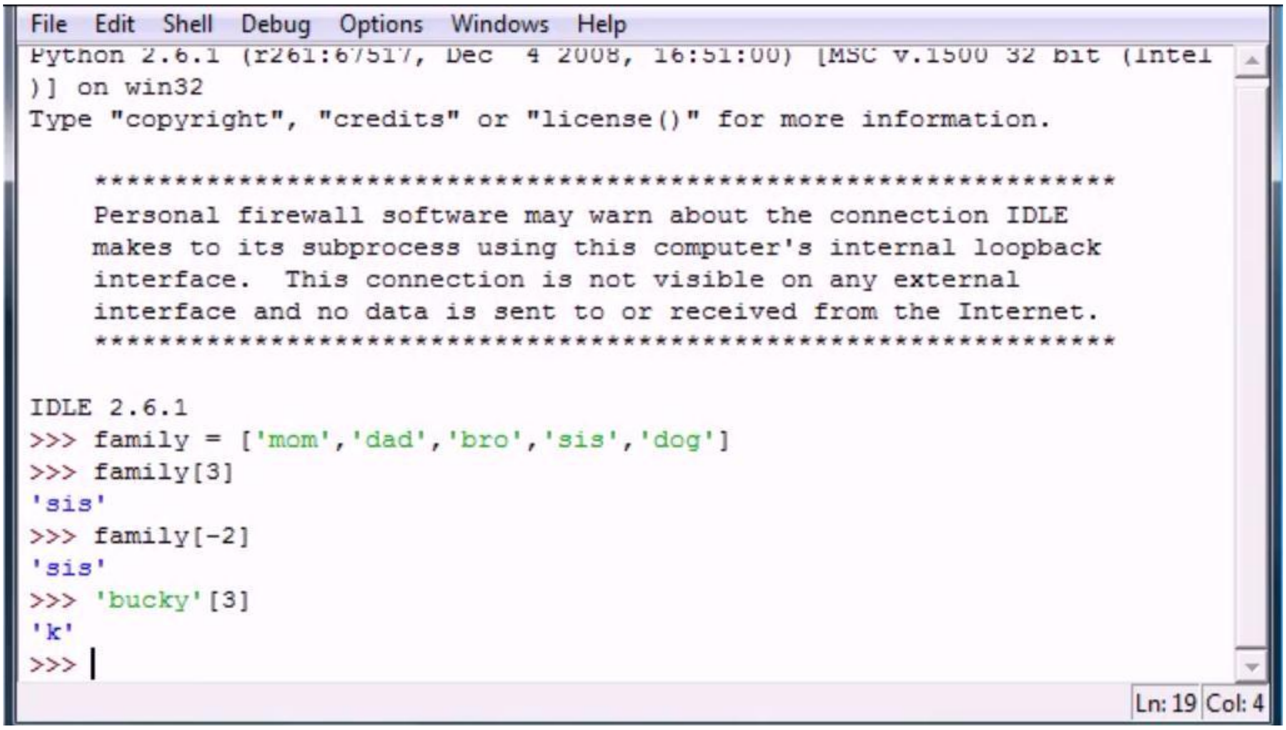
සාරාංශයක් විදිහට ගත්තොත්,
list එකක අවයව ක්‍රම  දෙකකට නම් කරන්න පුළුවන්.

1. මුල සිට අගට (0 , 1 , 2 , 3 , ....... )
2. අග සිට මුලට ( ........ , -3 , -2 , -1 )

තව දෙයක් තියෙනවා list විතරක් නෙමෙයි string එකක තියෙන අගයන් උනත් මෙහෙම නම් කරන්න පුළුවන්.

ඒක කරන්නෙ මෙහෙමයි,

‘ bucky ‘ [ 3 ]
‘  k ‘



```
File  Edit  Shell  Debug  Options  Windows  Help
Python 2.6.1 (r261:67517, Dec  4 2008, 16:51:00) [MSC v.1500 32 bit (Intel
)] on win32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 2.6.1
>>> family = ['mom','dad','bro','sis','dog']
>>> family[3]
'sis'
>>> family[-2]
'sis'
>>> 'bucky'[3]
'k'
>>> |
Ln: 19 Col: 4
```

# 10. Slicing ( කොටස් කිරීම )

මොකක්ද Slicing කියන්නේ ? list හෝ වෙනයම් දෙයක එහි අවයව කොටසක් වෙන් කරල ගන්න එකට තමයි Slicing කියන්නේ.

example = [ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 ] මෙහෙම list එකක් ගැන හිතමු.

අපිට මේ list එකෙන් කොටසක් කඩල ගන්න ඕන නම් කරන්න තියෙන්නෙ මේකයි.

**example[4:8]**
[ 4 , 5 , 6, 7 ]

```
>>>
>>> example=[0,1,2,3,4,5,6,7,8,9]
>>> example[4:8]
[4, 5, 6, 7]
```

ඉස්සෙල්ලාම list එකේ නම දාන්න ඕන.

ඊට පස්සේ කොටු වරහන් ඇතුලේ **මුලින්ම list එකේ කඩල ගන්න අවශ්‍ය අවයවයේ index එක** දාන්න ඕන. ඊට පස්සේ **කඩල අවසන් කරන්න ඕන අවයවයට පසු අවයවයේ index එක** දාන්න ඕන ඒ ගැන හොදට මතක තියාගන්න.

තව එකක් බලමු.

example [ 4:9]
[4,5,6,7,8]

example [ 4 :10 ]
[4,5,6,7,8,9]

```
>>> example[4:9]
[4, 5, 6, 7, 8]
>>> example[4:10]
[4, 5, 6, 7, 8, 9]
```

Slicing කරන එක ක්‍රමයක්, තමයි මේ.

තවත් ක්‍රමයක් තියෙනවා.
අපිට පිටුපසට ගණන් කරන්නත් පුළුවන්. ගොඩක් දිග listඑකක් නම් තියෙන්නේ , මේ ක්‍රමය ගොඩක් ලේසියි.

example [-5 : -1 ]
[ 5,6,7,8 ]

example = [ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 ]
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1

මට දැන් අවශ්‍යයි, 5 ඉදන් ඉතුරු අවයව සියල්ලම ගන්න.

example [ -5 : 0 ]
[ ]

ඇයි මේ විදිහට ආවේ. ?
මම කලින් පාඩමේ කියල දුන්න විදිහට - 0 කියල එකක් නෑ.
0 කියන්නේ 0 වම තමයි. ඒ නිසයි මෙහෙම වුනේ.

ඒත් මට අවශ්‍ය, listඑකේ තියෙන 9 ත්, එන විදිහට කඩල ගන්න.

```
>>> example[-5:-1]
[5, 6, 7, 8]
>>> example[-5:0]
[]
>>> example[-5:]
[5, 6, 7, 8, 9]
```

example [ -5 : ]

මේ විදිහට දෙවනුව හිස්තැනක් තියන්න . ඒකේ තේරුම තමයි. -5 ඉදල තියෙන ඔක්කොම අවයව ටික එලියට ගන්න කියන එකයි.
[ 5,6,7,8,9 ]

example [  : 7 ]
[ 0,1,2,3,4,5,6 ]

```
>>> example[:7]
[0, 1, 2, 3, 4, 5, 6]
>>>
```

මේකත් ඒ වගේමයි. index එක 7 ට ඉස්සරහ තියෙන index සියල්ලටම අදාල අවයවයන් එළියට ගන්න කියලයි මේකෙන් කියන්නේ.

හරි මෙහෙමත් බලමු.

example [ : ]
කිසිම index num එකක් නැවත හිස්වම දුන්නොත් මොකද වෙන්නේ ?

[ 0,1,2,3,4,5,6,7,8,9 ]

```
>>> example[:]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

මුළු list එකම ආවා නේද ?

මුළු list එකම අපිට එකවැර ගන්න ඕනෙනම් , මේ විදිහටත් ලේසියෙන්ම ගන්න පුළුවන්.

තව දෙයක් තියෙනවා . අපි දන්නේ පරාමිතීන් ( parameters ) දෙකයිනේ.
ඒත් තවත් තෙවන පරාමිතයක් (3rd parameter) තියෙනවා.

තෙවන පරාමිතියෙන් කියන්නේ අපි යම් පරාසයක අවයවයන් එකකින් ගනිද්දි , කිසියම් රටාවකට ( පරතරයකින් ) අවයවයන් කඩල ගන්න අවශ්‍ය ඉලක්කමයි.

example [ 1 : 8 : 2 ]
[ 1,3,5,7 ]
දැන් කියපු දේ තේරෙනවා නේද ?

```
>>> example=[0,1,2,3,4,5,6,7,8,9]
```

example [ 10 : 0 : -2 ]
[ 9 , 7 , 5 , 3 , 1 ]
මේ විදිහට පිටුපසට සංඛ්‍යා මගහැර ගන්නත් පුළුවන්.
මේක ගොඩක් වටිනා දෙයක්.

```
>>> example[1:8:2]
[1, 3, 5, 7]
>>> example[10:0:-2]
[9, 7, 5, 3, 1]
```

ඉස්සෙල්ලා කරපු විදිහට කරන්න කම්මැලියි වාගේ නම් ඒකම කරන්න ලේසි විදිහක් තියෙනවා

Example [  : : - 2 ]

[ 9 , 7 , 5 , 3 , 1 ]

```
>>> example[::-2]
[9, 7, 5, 3, 1]
```

මුළු list එකම පිටුපසට -2 ක ගානේ, සංඛ්‍යා මග හැරල output වෙන එකයි, මේකෙදි ඇත්තටම වෙන්නේ. මේවා තමයි slicing වල මූලිකම දේවල් ඉතින් මේ විදිහට list එකක තියෙන ඕනෙම දත්තයක් එළියට ගන්න පුළුවන් ලේසියෙන්ම.

# 11 . Editing Sequences ( අනුක්‍රමයන් සැදීම )

Sequence කියන්නේ ඇත්තටම list , strings , tables වාගේ දේ වලට තමයි. ඉතින් Sequence වලින් කරන්න පුළුවන් පොඩි වැඩ ටිකක් ගැන බලමු.

[7, 4, 5 ] + [ 4, 6 ,5 ]
ඇත්තටම මේ Sequence 2ක්. ඉතින් එකම ජාතියේ Sequences එකතු කරන්න , ( + ) ලකුණු පාවිච්චි කරනවා. enter එක එබුවොත් අපිට output එකක් එනවා.

```
IDLE 2.6.1
>>> [7,4,5]+[4,6,5]
[7, 4, 5, 4, 6, 5]
>>>
```

[ 7, 4, 5, 4, 6, 5 ]
මේකෙදි වුණේ Sequences 2ක් එකතු වෙලා තනි Sequence එකක්, ඒ කියන්නේ තනි list එකක් හැදුන එකයි.

තව එකක් බලමු.

‘ bucky ’ + ‘ roberts ‘

‘ buckyroberts ‘

```
>>> 'bucky'+'roberts'
'buckyroberts'
>>>
```

දැන් අපි, list එකකුයි string එකකුයි එකතු කරමු.

[ 4, 5, 6 ] + ‘ bucky ‘

```
>>> [4,5,6]+'bucky'

Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    [4,5,6]+'bucky'
TypeError: can only concatenate list (not "str") to list
>>>
```

අපිට මේ දෙක එකතු කරන්න දෙන්නේ නෑ නේද ?

ඇයි දන්නවද එහෙම වෙන්නේ.
මේකෙදි මෙහෙම එකතු කරන්න පුළුවන් වෙන්නේ එකම ජාතියේ Sequences විතරයි. ඒක හොඳට මතක තියාගන්න.

අපි දැන් බලමු, Sequence ගුණ කිරීම ගැන.

‘ bucky ‘ * 10

bucky ලා 10 ක් ආවා නේද ?

```
>>> 'bucky'*10
'buckybuckybuckybuckybuckybuckybuckybuckybuckybucky'
>>>
```

ඉතින් මේකෙදි වුනේ bucky කියන එක 10 වතාවක් නැවත නැවත පිටපත් විමයි.

21 * 10
210

ඇත්තටම මේ ගුණ කිරීම නම් හොදට වුනා. ඒත් මම බලාපොරොත්තු වුනේ ඒක නෙවෙයි. මට ඕන වුනේ ඉස්සෙල්ලා වාගේ bucky කියන එක 10 වතාවක් පිටපත් වුන විදිහටම 21 ත් 10 වතාවක් පිටපත් කරන්නයි.

ඇත්තටම එහෙම කරන්නෙ කොහොමද ?

[ 21 ] * 10

වැඩේ හරි නේද ?

```
>>> 21*10
210
>>> [21]*10
[21, 21, 21, 21, 21, 21, 21, 21, 21, 21]
```

ඉතින් මේ ක්‍රමේ ගොඩක් වටිනවා කියල ඔයාලට තේරෙනවා ඇති.

හරි මම දැන් අලුත් දෙයක් කියල දෙන්නයි යන්නේ.

name = ‘ roast beef ‘
මෙහෙම අපි type කරමු.

‘ z ’ in name
False

මේකේ **in** කියන එක Python වල **build in keyword** එකක්. ‘ z ‘ in name වලින් ඇත්තටම කරන්නේ z අකුර ,name එක තුල තියෙනවාද කියල පරීක්ෂා කරල දෙන එකයි.මේ සඳහා විශේෂ වූ in කියන build in keyword එක පාවිච්චි වෙනවා.

```
>>> name='roastbeef'
>>> 'z' in name
False
>>> 'r' in name
True
```

‘ r ‘ in name
True
මෙහෙම වෙන්න හේතුව name = ‘ roast beef ‘ එක තුල  r අකුර තියෙන නිසයි.

අපි පොඩි list එකක් හදමු.
family = [ ‘ mom ’ , ‘ dad ‘ , ‘ bro ‘ ]

```
>>> family=['mom','dad','bro']
```

‘ sis ‘ in family
False

‘mom’ in family
True

```
>>> 'sis' in family
False
>>> 'mom' in family
True
>>>
```

### *අවයවය (element)* **in** *අනුක්‍රමය (Sequence)*

ඔයාලට ඒ කරුණ තවත් තහවුරු වෙන්න ඇති දැන්.මේ විදිහට අපිට ඕනම අනුක්‍රමයක(Sequence) අවයව(elements) පරීක්ෂා කරල බලන්න පුළුවන්.

# 12. More List Functions (ලැයිස්තු ආශ්‍රිත ශ්‍රිත තවදුරටත්)

අපි list එකක් හදමු.

numbers = [8, 1, 4, 17, 28, 165, 7 ]

```
>>> numbers=[8,1,4,17,28,165,7]
```

අපිට දැන් ඕන , මේ list එකේ අවයවයන් කීයක් විතර තියෙනවද කියල ගණන් කරල බලන්න.

නිකමට ගණන් කරල බලමු.
7ක් තියෙනවා නේද ?

මේ list එකේනම් අවයවයන් ටික ගණන් කරන්න එච්චරම අමාරු නෑ. ඒත්.. ලොකු, දිග list එකක් තිබුණොත් එහෙම, අපිට තරු පේනවනේ.

ඉතින් වැඩිය දගලන්නේ නැතිව අවයවයන් කීයක් තියෙනවද කියල ගන්න ලේසි ක්‍රමයක් තියෙනවා. ඉස්සෙල්ලාම මේකයි කරන්නේ.

**len** කියන function එක type කරල වරහන් ඇතුලේ list එකේ නම දාන්න ඕනේ.

len (numbers)
7
හරි අවයව ගාණ, ආවා නේද ?

```
>>> len(numbers)
7
```

හරි, මේ වගේ සරල function ටිකක් ගැන මම දැන් කියන්නම්,

```
>>> numbers=[8,1,4,17,28,165,7]
```

max (numbers)
165
මේකෙදි ලැබෙන්නේ list එකේ තියෙන වැඩිම අගයක් සහිත අවයවයයි.

min (numbers)
1
මේකෙදි වෙන්නේ, list එකේ තියෙන අඩුම අගයක් සහිත අවයවය, අපිට දෙන එකයි.

```
>>> max(numbers)
165
>>> min(numbers)
1
```

තවත් function එකක් ගැන කතා කලොත් ,
**list ( )**
මේකෙදි කරන්නේ string එකක අවයවයන්, list එකකට වෙන වෙනම පරිවර්ථනය කරල දීමයි.

list( ' bucky ' )
[ 'b' , ' u ' , ' c ' , ' k ' , ' y ' ]

```
>>> list('bucky')
['b', 'u', 'c', 'k', 'y']
```

numbers කියල type කරමු.
එතකොට අපේ list එක අපිට ලැබෙනවා.

[ 8 , 1 , 4 , 17 , 28 , 165 , 7 ]

```
>>> numbers
[8, 1, 4, 17, 28, 165, 7]
```

අපිට මේකේ තියෙන අවයවයන් වෙනස් කරගන්න ඕන වෙනවා.

```
>>> numbers
[8, 1, 4, 17, 28, 165, 7]
```

අපි ඒක කොහොමද කරන්නේ.

numbers[ 3 ] = 77

මේකේ තේරුම තමයි ,
index එක 3 වන අවයවය, වෙනුවට 77 ආදේශ කරන්න කියන එකයි.

ඉතින් දැන් Enter එක ඔබලා බලමු.

මොනවත් වුනේ නෑ නේද ? ඇත්තටම දෙයක් වුනා කිවොත් හරි ,
දැන් index එක 3 වෙන තැනට 77 ආදේශ වෙලයි තියෙන්නේ.

```
>>> numbers[3]=77
>>> numbers
[8, 1, 4, 77, 28, 165, 7]
```

ඒක පරීක්ෂා කරල බලමු අපි.

අපි numbers කියල type කරල list එක බලමු.
[ 8, 1, 4, 77, 28, 165, 7 ]

හරි නේද ? index එක 3 වෙන තැනට 77 වැටිල.

අවයවයන් මකන්නත් පුළුවන් මේ විදිහටම.
ඉස්සෙල්ලාම ***del* keyword** එක type කරල list එක සමග අවයවයේ index එක දෙන්න.

මෙන්න මේ විදිහට

**del** numbers (3)

හරි දැන් මැකිලයි තියෙන්නේ.

```
>>> del numbers[3]
>>> numbers
[8, 1, 4, 28, 165, 7]
>>>
```

හරියටම තහවුරු කරගන්න list එක අරන්ම බලමු.

numbers
[ 8, 1, 4, 28, 165, 7 ]

77 මැකිල නේද ?

හරි, අපි ඉගෙන ගත්තා ,
අලුත් function ටිකක්, ඒවාගෙම අලුත් වැඩ කෑලි ටිකක්. මේ තාක් මම කියල දුන්න ඒවායෙන්,
හොද programme එකක් ඔයාටම හදන්න පුළුවන් කියල මට හිතෙනවා .
ඔයාට නිතරම තියෙන සරල ගැටලුවක්, මේවායෙන් ලේසියෙන්ම විසඳගන්න පුළුවන්.

Python වලින් එක පාරටම ගැටලුව ගැන හිතාගන්න අමාරුයි වගේ නම් ඉස්සෙල්ලාම,
Algorithm එකක් හදන්න .
ඊට පස්සේ flow chart එකත් හදා ගන්න.
ඊට පස්සේ pseudo code එකකුත් ඕනෙනම් ලියාගෙන, python language එකෙන් programme එක ලියන්න පටන් ගන්න.

**Algorithm => Flow chart => Pseudo code => Programme language**

# 13. Slicing Lists (ලැයිස්තු කොටස් කිරීම)

Slicing ගැන ඔයාල දැනටමත් ගොඩක් දේවල් දන්නවනේ. Slicing ආශ්‍රිත තවත් අලුත් දේවල් ටිකක් ඔයාට කියල දෙන්නයි, මම මේ යන්නේ.

වචනයක තියෙන අකුරු list එකක අවයව කරන්නේ මෙහෙමයි.

example = list (‘ easyhoss ‘)

මෙහෙම type කරල Enter එක ඔබපුවහම list එකක් හැදෙනවා.

හරි දැන් අපේ list එක පරීක්ෂා කරල බලමු.

example
[‘e’ , ‘a’ , ‘s’ , ‘y’ , ‘h’, ‘o’ , ‘s’ , ‘s’ ]

```
>>> example=list('easyhoss')
>>> example
['e', 'a', 's', 'y', 'h', 'o', 's', 's']
```

හරි අපි දැන් මේ list එකට අලුත් දේවල් ටිකක් එකතු කරමු.
example [ 4 :   ] = list ( ‘ baby ‘ )

මේකෙන් ඇත්තටම කියන්නේ,
index එක 4 වන අවයවයත් ඇතුළුව ඉන් එහාට තියෙන අවයව ටික මකල ඒ වෙනුවට ‘ baby ’ යන වචනය අදාල අවයව ටිකට ආදේශ කරන්න කියලයි.

ඒක පරීක්ෂා කරල බලමු.
example
[‘e’ , ‘a’ , ‘s’ , ‘y’ , ‘b’, ‘a’ , ‘b’ , ‘y’ ]

```
>>> example[4:]=list('baby')
>>> example
['e', 'a', 's', 'y', 'b', 'a', 'b', 'y']
```

තවත් විදිහකට ඒ ගැන බලමු.
example [ 4 :  ] = list ( ‘race cars’)

```
>>> example[4:]=list('racecars')
>>> example
['e', 'a', 's', 'y', 'r', 'a', 'c', 'e', 'c', 'a', 'r', 's']
>>>
```

හරි, ඒ ගැන ඔයාට තේරෙන්න ඇති.

තව ගොඩක් වැදගත් දෙයක් තියෙනවා, කියල දෙන්න.
මේකයි, දැන් ඔය list එකක, අවයව 2 ක් මැදට අපිට අලුත් අවයවයක් දාගන්න පුළුවන්ද?

ඒ ගැන දැන් බලමු.
අපි අලුත් list එකක් හදලම වැඩේ පටන්ගමු.

example = [ 7 , 8 , 9 ]
හරි, list එක හැදුනනෙ දැන්.

```
>>> example=[7,8,9]
>>> example
[7, 8, 9]
```

example = [ 7 , 8 , 9 ]
0 1 2

example[1:1] = [ 3 , 3 , 3 ]

මේකෙන් වෙන දේ මම කියන්නම්, ඔයාලට මතක ඇතිනේ, අවයව වල index නම් කරන හැටි ගැන (0,1,2,3,......). ඉතින් මේකෙන් කියන්නේ, index number = 1වෙන අවයවය ලඟට යන්න, ඊට පස්සේ ආයෙත්, index number = 1වෙන තැනට යන්න, හරියටම එතනට, [ 3 , 3 , 3 ] පතබාන්න කියලයි.
ඇත්තටම,ඕකෙන් කියන්නේ 7,8 අතරට (එක 0 හා 1 වන අවයව අතරට), [ 3 , 3 , 3 ] දාන්න කියලයි.

දැන් list එක check කරල බලමු, output එක ගැන.
[ 7 , 3 , 3 , 3, 8 , 9 ]
0 1 2 3 4 5

```
>>> example[1:1]=[3,3,3]
>>> example
[7, 3, 3, 3, 8, 9]
```

මම කිව්ව දේ හරි නේද?

තව පොඩි දෙයක්, මම ඔයාට කියල දෙන්නම්.
example[1:5] = [ ]
මෙහෙම කළොත් මොකද වෙන්නේ?

ඔයාට මතක ඇති,
example[1:5]
මේකෙදි, දෙවනුව සඳහන් කරන, index එක නැතුව, ඊට පෙර අවයව ටික ගන්න එක ගැන.

list එක check කරලම බලමු.
[ 7 , 9 ]
මම හිතනව ඒක ඔයාට තේරෙන්න ඇති.

```
>>> example[1:5]=[]
>>> example
[7, 9]
```

ඉතින් මේ විදිහට list එකක තියෙන අවයව මකන්න අපිට පුළුවන්.

# 14. Intro to Methods (රීතීන් සඳහා ප්‍රවිෂ්ඨය )

Python වල හැටියට methods කියන්නේ, එක්තරා විදිහක building function වලටම තමයි. ඉතින් මේවා වලින් වැඩගන්න හැටියි, මම දැන් කියල දෙන්න යන්නෙ.

ඉස්සෙල්ලාම පොඩි list එකක් හදමු.

face = [ 21 , 18 , 30 ]

```
>>> face=[21,18,30]
>>> face
[21, 18, 30]
```

මේ list එකේ අගට අලුත් අගයක් එකතු කරන්න, ගොඩක්ම ලේසි ක්‍රමයක් තියෙනව. ඒකට පාවිච්චි වෙන්නෙ append කියන method එකයි.

ඒක මේ විදිහට පාවිච්චි කරන්න පුළුවන්.
ඉස්සෙල්ලාම අපේ list එකේ (object) නම ලියන්න ඕන. ඊට පස්සෙ තිතක්(dot) තියල, අදාල method එක පාවිච්චි කරන්න පුළුවන්.

**face.append(45)**
face
[ 21 , 18 , 30 , 45 ]
අපට ඕන දේ, ලේසියෙන්ම උනා නේද?

```
>>> face.append(45)
>>> face
[21, 18, 30, 45]
```

ඉතින් මේ methods පාවිච්චි කරද්දි සාමාන්‍ය වරහන්, අනිවාර්යෙන්ම පාවිච්චි කරන්න. මතකනේ face.append එක අගට 45 යොදන්න අපි පාවිච්චි කලේ, සාමාන්‍ය වරහන්. ඉතින් ඒ දේ ගැන හොඳට මතක තියාගන්න.

තවත් ගොඩක් වැදගත් වෙන method එකක් ගැනයි, මම ඊළඟට කියන්න යන්නේ.
ඒකට පොඩි list එකක් හදමු.

apples = [ 'i' ,'love' ,'apples' ,'apples' ,'now']

list එක check කරලත් බලමු.
apples
[ 'i' ,'love' ,'apples' ,'apples' ,'now']

```
>>> apples=['i','love','apples','apples','now']
>>> apples
['i', 'love', 'apples', 'apples', 'now']
```

හරි, බලමු දැන්,
**apples.count('apples')**
2

```
>>> apples.count('apples')
2
```

ඇයි 2 ආවේ? ඒක වෙන්නෙ මෙහෙමයි. count එකෙන් කරන්නේ, අපි දෙන වචනෙ, මේ list එකේ කී පාරක් තියෙනවද කියල, හොයල දෙන එකයි. ඉතින් උඩ තියෙන විධානය ක්‍රියාත්මක කරපු ගමන්, මෙයා list එකේ මුල ඉඳන් බලනව, අපි දුන්න වචනයට ගැලපෙන වචනයක් තියෙනවද කියල. ඔහොම ගණන් කරගෙන යනකොට මෙයාට අහුවෙනව apples කියල දෙවතාවක සඳහන් වෙලා තියෙනව කියල. ඉතින් එයා output එක, 2 කියල අපිට පෙන්වනව.

අපි දැන් list එකේ නැති වචනයක් දාල බලමු.
**apples.count('bow')**
0
උත්තරේ හරි නේද?

```
>>> apples.count('bow')
0
```

තවත් ලස්සන method එකක් ගැන, මම ඔයාට කියල දෙන්නම්.

```
>>> one = [ 1 , 2 , 3]
>>> two = [ 4 , 5 , 6 ]
>>>
>>> one
[1, 2, 3]
>>> two
[4, 5, 6]
```

one = [ 1 , 2 , 3]            two = [ 4 , 5 , 6 ]
මම list දෙකක් හදාගත්ත.

**one.extend(two)**
මේකෙන් ඇත්තටම කරන්නෙ විහිදුවීමක්. මේකෙදි වෙන්නෙ one කියන list එකේ අගට two කියන list එක එකතු කිරීමයි. මෙහිදී one කියන list එක විහිදුවීමක් වෙනව. මේකෙදි ප්‍රධාන වෙන්නේ one කියන list එකයි.

අපි දැන් one, list එක check කරල බලමු.
one
[ 1 , 2 , 3 , 4 , 5 , 6 ]

```
>>> one.extend(two)
>>> one
[1, 2, 3, 4, 5, 6]
```

සාරාංශයක් විදිහට ගත්තොත්,
**object.method(argument)**
වස්තුව.ක්‍රමය(තර්කය)

මේ විදිහට සාධාරණ ව්‍යුහයකින් පෙන්නන්න පුළුවන්, methods පාවිච්චි කරන හැටි.

ඉදිරියටත් තවත් අලුත් methods ටිකක් කියල දෙන්නම්. ඉතින් මේ methods වලින් ලේසියෙන්ම, ගොඩක් කාර්යක්ෂමව programmes හදාගන්න අපට උදව් වෙනව.

# 15. More Methods ( රීතීන් තවදුරටත් )

දැන් මම කියල දෙන්න යන්නේ index කියන method එක ගැන.

say = [ ‘hey’ , ‘how’ , ‘brown’ , ‘cow’ ]

හරි අපි listඑකකුත් හැදුවා.

```
>>> say=['hey','now','brown','cow']
>>> say
['hey', 'now', 'brown', 'cow']
```

දැන් මෙහෙම type කරමු.
**say.index(‘brown’)**
2

```
>>> say.index('brown')
2
```

මේකෙදි වෙන දේ ඔයාට දැනටමත් තේරෙනව ඇති.
අදාල අවයවයේ නම දීපුහම, index එක ලේසියෙන්ම අපිට හොයල දෙනව.

**say.insert( 2, ’hoss’ )**
මේකෙදි කරන්නේ, අපි දෙන index එකට, අපි දෙන අවයවය, දාන එකයි. එතකොට ඉස්සෙල්ලා ඒ index එකේ තිබුණ අවයවයට මොකද වෙන්නෙ?

ඒ ගැන list එක check කරලම බලමු.
say
[ ‘hey’ , ‘now’ , ‘hoss’ , ‘brown’ , ‘cow’ ]

```
>>> say.insert(2, 'hoss')
>>> say
['hey', 'now', 'hoss', 'brown', 'cow']
```

තේරෙන්න ඇතිනේ, එතනින් එහා තියෙන අවයව ඔක්කොම එකකට එකක් පිටුපසට යනව.

දැන් අපි බලමු, list එකක අවයව ඉවත් කරන්න තියෙන, methods 2ක් ගැන.

**say.pop(1)**
‘now’
say
[ ‘hey’ , ‘hoss’ , ‘brown’ , ‘cow’ ]

```
>>> say.pop(1)
'now'
>>> say
['hey', 'hoss', 'brown', 'cow']
```

මේකෙදි කරන්නේ, අපට ඕන index එක දීපුහම, ඒ index එකට අදාල අවයවය output කරන ගමන් list එකෙන් ඒ අවයවය ඉවත් කිරීමක්.

**say.remove( ‘brown’ )**
මේක ටිකක් වෙනස්, මේකෙදි අපට ඉවත් කරන්න අදාල අවයවය සඳහන් කරපුහම, ඒ අවයවය list එකෙන් ඉවත් කරනව. බැරි වෙලාවත් අපි සඳහන් කරන අවයවය කිහිප වතාවක, list එකේ තිබුණොත් මොකක් වෙයිද? එතකොට වෙන්නේ මුලින්ම හම්බ වෙන අවයවය ඉවත් කරන එකයි.

```
>>> say.remove('brown')
>>> say
['hey', 'hoss', 'cow']
```

**say.reverse()**
මේ method එකෙන් කරන්නේ අපේ මුළු list එකම කනපිට හරවල දෙන එකයි. ඒ කියන්නේ අග ඉඳල මුලට අවයව ටික පෙළගැස්වීමයි.

```
>>> say.reverse()
>>> say
['cow', 'hoss', 'hey']
```

# 16. Sort and tuples (පෙළගැස්වීම සහ ටපල්ස් )

ඉස්සෙල්ලාම බලමු පෙළගැස්වීම ගැන.

අපි දැන්, සංඛ්‍යා දාපු පොඩි list එකක් හදමු.

new = [ 32 , 54 , 22 , 7 , 98 , 1 ]

new
[ 32 , 54 , 22 , 7 , 98 , 1 ]

```
>>> new=[32,54,22,7,98,1]
>>> new
[32, 54, 22, 7, 98, 1]
```

හරි, දැන් list එකක් හදාගත්තනෙ.

දැන් sort කියන method එක පාවිච්චි කරන්න පුළුවන්. මේකෙදි sort() කියන එකේ, වරහන් ඇතුළට මොනවත් දාන්න එපා.

හරි, දැන් sort කරපුහම list එක ආරෝහන පිළිවෙළට සැකසෙනව.

```
>>> new.sort()
>>> new
[1, 7, 22, 32, 54, 98]
```

new
[ 1 , 7 , 22 , 32 , 54 , 98 ] හරි නේද?

ඒවාගෙම අපිට පුළුවන්, string වුත් පෙළගස්වන්න. ඒක කරන්නෙනම් මෙහෙමයි. ඉස්සල්ල ක්‍රමේට වඩා ටිකක් වෙනස්.

sorted() කියන එකේ වරහන් ඇතුළට, අපිට අවශ්‍ය string එක ලියන්න ඕන.

ඉන්න මම කරල පෙන්වන්නම්,

sorted( 'Easyhoss' )
[ 'E' , 'a' , 'h' , 'o' , 's' , 's' , 'y']

```
>>> sorted('Easyhoss')
['E', 'a', 'h', 'o', 's', 's', 's', 'y']
```

දැන් මේක අකාරාදී පිළිවෙළටයි (Alphabetical Order) සැකසිලයි තියෙන්නේ. Python වල හැටියට අකාරාදී පිළිවෙළට සකසන්න, ඉස්සෙල්ලාම ගන්නෙ Capital අකුරු, ඒ නිසයි ' E ' ඉස්සරහටම ඇවිත් තියෙන්නෙ, ඊට පස්සේ පිළිවෙළට Simple අකුරු පෙළගැස්සිල තියෙනව, ඔයාට ජේනව ඇති.

මම දැන් methods වලින් ටිකක් අයින් වෙලා, tuples ගැන කියල දෙන්නම් ඔයාට.

ඉස්සෙල්ලාම tuple එකක් පෙන්නලාම ඉන්නම්කො.

නිකම්ම, 41,42,32,54 කියල, IDLE එකේ Enter කරොත්, අපිට tuple එකක් ලැබෙනව.

tuple වලින් ලොකු වැඩකැලි දාන්න බෑ, list වලදි වගේ.

```
>>> 41,42,32,54
(41, 42, 32, 54)
```

ඇත්තටම කිව්වොත් මේකෙ තියන අවයව ආය වෙනස් කරන්න, අලුත් කරන්න, අයින් කරන්න මොනවත්ම කරගන්න දෙන්නෙ නෑ.
කරන්න පුළුවන් එකම දෙයයි.
ඒකෙ තියෙන අවයවයන් එළියට output කරගැනීම පමණයි.

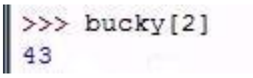
ඉන්න ඒක පෙන්නනම්,

bucky = ( 32 , 32 , 43 , 54 )

```
>>> bucky=(32,32,43,54)
>>> bucky
(32, 32, 43, 54)
```

bucky
( 32 , 32 , 43 , 54 )

හරි මේක tuple එකක්.
දැන් මේකෙ අගයන් එළියට අරන් පෙන්වන්නම්.
bucky[2]
43




ඔයාට දැන් තේරෙනව ඇති, tuple ගැන.

හොදට මතක තියාගන්න, tuple වලින් කරන්න පුළුවන්, අගයන් එළියට ගැනීම පමණයි.

ඉතින් අන්තිමට සාරාංශයක් විදිහට ගත්තොත් පෙළගැස්වීම(sort), අපට ගොඩක් වටින දෙයක් programmes ලියන කොට.

ඒවාගෙම අපි දන්නව, list එකක අවයව වාගෙම, sting එකක අවයවද ලේසියෙන්ම පෙළගැස්වීමට හැකි බව.

# 17. Stings and Stuff ( වදන් ආශ්‍රිත මෙවලම්)

Stings වලට අදාල අලුත් ආකර්ෂණීය දේවල් ටිකක් මම දැන් කියල දෙන්නම්,

මේක දිහා පොඩ්ඩක් බලන්න,

bucky= “ Hey there bucky , hows your head ”

```
>>> bucky="Hey there bucky, hows your head"
```

මේකෙ තියන bucky, head කියන වචන, අපිට එක එක වෙලාවට ක්ෂණිකව වෙනස් කරගන්න ඕන වගේ කියල හිතන්නකො.
ඒත් ඇත්තට අපි කොහොමද ඒක කරන්නෙ?

bucky = “ Hey there %s , hows your %s ”

```
>>> bucky="Hey there %s, hows your %s"
```

%s මොකද්ද මේ? ඇත්තටම මේක මාරම ලස්සන දෙයක්, ඒකෙ s වලින් කියවෙන්නෙ String කියලයි. % කියවෙන්නේ මොන විදිහෙ data එකක්ද? යන්නයි.

ඇත්තටම කිව්වොත් %s කියන්නේ String සහිත විචල්‍යයක් (String variable) , ඒ කියන්නේ වෙනස් කරන්න පුළුවන් හෝ වෙනස් වෙන දෙයක්.

ඉතින් bucky , String එකෙයි, ඕකෙයි ඇති සම්බන්ධය මොකක්ද?

bucky = “ Hey there %s , hows your %s ”

අපිට දැන් ඕන පොඩි අගයන්(වචන) 2ක්. මම දැන් ඒ 2කත් ලෑස්ති කරනවා.
varb = ( ‘betty’ , ‘foot’ )
හරි, අපට දැන් තියෙනව, String එකක්, අගයන්(වචන) 2ක් සහිත.

දැන් මම පෙන්වන්නම්,
bucky හා varb කියන දෙකේ ලස්සන සම්බන්ධතාවයක්.
ඒක කරන්නේ මෙහෙමයි.

```
>>> bucky="Hey there %s, hows your %s"
>>> varb=('betty', 'foot')
```

print bucky%varb
මේකෙ තේරුම තමයි, varb කියන එකෙන්, අගයන් අරගන bucky කියන එකේ, % සහිත තැන් වලට, එම අගයන් ආදේෂ කරන්න කියලයි.

දැන් Enter එක ඔබල බැලුවොත් ඔයාට ඒක තේරෙයි.
Hey there betty, hows your foot

```
>>> print bucky % varb
Hey there betty, hows your foot
```

ඉතින් මේවාගෙම,

varc = ( ‘tuna’ , ’fin’ )

print bucky%varc
Hey there tuna, hows your fin

```
>>> varc=('tuna', 'fin')
>>> print bucky % varc
Hey there tuna, hows your fin
```

ඉතින් මේ ක්‍රමේ පාවිච්චි කරල, අපිට පුළුවන් අමුතු විදිහෙ programmes ලියන්න.
උදාහරණයක් විදිහට, එකේක පුද්ගලයින් අමතන්න, අපට වෙනස් විදිහෙ වචන set කරල, ඒ ගොල්ලන්ව පුදුම කරන්න පුළුවන්.

දැන් මම කියල දෙන්නම්, find කියන method එක ගැන.

example = ” Hey now Bessie nice chops ”

```
>>> example="Hey now bessie nice chops"
>>> example
'Hey now bessie nice chops'
```

අපි හිතමු, අපිට ඕන වෙනව මේකෙ තියෙන වචනයක් හරියටම තියෙන ස්ථානය හොයාගන්න.

ඒක කරන්නෙ මෙහෙමයි.
example.find(‘bessie’)
8

```
>>> example.find('bessie')
8
```

අපිට 8 කියල ආවනේද? ඒකෙ ඇත්තම තේරුම මොකක්ද?

ඒකෙන් කියන්නෙ මේකයි. මේ String එකේ කීවෙනි තැනද (කුමන index එකද)
bessie කියන වචනෙ ( ‘b’ අකුර සඳහන් තැන) පටන් ගන්නෙ කියලයි.

‘Hey now bessie nice chops’
0123456789………………………..

හොඳට මතක තියාගන්න, **හිස්තැනුත්** අනිවාර්යයෙන්ම මේ ගණනය කිරීමට ගන්නව.

තව එකක් බලමු.

example.find(‘chops’)
20

```
>>> example.find('chops')
20
```

මේකට අනුව අපිට හිතාගන්න පුළුවන් ‘ c ’ කියන අකුරට අදාල index එක 20 යි කියල.

# 18. String Methods ( වදන් ආශ්‍රිත රීති තවදුරටත් )

String වලට සම්බන්ධ තවත් ලස්සන method එකක් ගැන මම කියල දෙන්නම්. ඒක අපි හඳුන්වන්නෙ join කියල.

Sequence = [ ‘hey’ , ‘there’ , ‘bessie’ , ‘hoss’ ]

Sequence
[ ‘hey’ , ‘there’ , ‘bessie’ , ‘hoss’ ]
හරි, දැන් String එකක් හැදුව මම.

```
>>> sequence=['hey','there','bessie','hoss']
>>> sequence
['hey', 'there', 'bessie', 'hoss']
```

තවත් පොඩි String එකකුත් හදමු.
glue = ‘hoss’

මේ දෙක අපිට එකට set කරන්න ඕන. ඒක කරන්නෙ මෙහෙමයි. ඉස්සෙල්ලාම list එකේ, හැම අවයවයකම මැදට දාන්න ඕන වචනයට අදාල නම ලියන්න ඕන. ඊට පස්සේ තිතක් තියල join කියල ලියන්න. ඊටත් පස්සේ වරහන් තුළ අර අදාල Sequence එකේ නම ලියන්න.

**glue.join(sequence)**
‘heyhosstherehossbessiehosshosshoss’

```
>>> glue='hoss'
>>> glue.join(sequence)
'heyhosstherehossbessiehosshoss'
```

වැදගත් method එකක් ගැන දැන් මම කියල දෙන්නම්.
ඉස්සෙල්ලාම String එකක් හදමු.

randstr = “I wish i Had No sausage”
randstr
“I wish i Had No sausage”

```
>>> randstr="I wish i Had NO sausage"
>>> randstr
'I wish i Had NO sausage'
```

මට දැන් අවශ්‍යයි, මේකෙ තියෙන හැම අකුරම Simple අකුරු වලට පරිවර්තනය කරන්න.ඒක හරිම ලේසියි.
**randstr.lower()** දැන් Enter එක ඔබල බලමු.
‘i wish i had no sausage’

```
>>> randstr.lower()
'i wish i had no sausage'
```

ඔයාට දැන් තේරෙනව ඇති මේ method එකේ තියෙන වැදගත්කම.

තවද දෙයක් මම කියල දෙන්නම්,

අලුත් String එකක් හදමු.
truth = ‘ I love old women ’

```
>>> truth="I love old women"
>>> truth
'I love old women'
```

මම කෙලින්ම method එක ලියල ඉන්නම්.

**truth.replace(‘women’ , ’men’)**

මේකෙදි ඇත්තටම වෙන්නේ women කියන වචනය වෙනුවට men කියන වචනය ආදේශ වීමයි.
අපි ඒක check කරල බලමු.
වැඩේ හරි නේද? ඉතින් මේ විදිහට, යම් වචනයක් වෙනුවට වෙනත් වචනයක් ලේසියෙන්ම ආදේශ කරගන්න පුළුවන්.

```
>>> truth.replace('women','men')
'I love old men'
```

# 19. Dictionary (වදන් කෝෂ)

මම දැන් කියල දෙන්න යන්නේ python වල තියෙන Dictionary ගැන. මොකක්ද Dictionary එකක් කියන්නේ?

ඇත්තටම මේක අපි දන්න Dictionary එකක් වගේම තමයි. Dictionary එකෙන්, අපි වචනයක් හොයනව කියමුකො, ඒ වචනෙ තියෙන තැනත් හම්බ වුනා. ඉතින් දැන් ඒ වචනෙට අදාල තේරුමක් හෝ විස්තරයක් අපිට දැකගන්න පුළුවන්.

ඉතින් python වල තියෙන Dictionary ත් ඒ වාගෙම තමයි. අපි හොයන වචනෙ වාගෙ දේට **යතුර(key)** කියල කියනව. ඒ වචනයට අදාල විස්තරය, **අගය(value)** කියල අපි කියනව.

මතක තියාගන්න දෙයක් තියෙනව, Dictionary වලට පාවිච්චි වෙන්නේ, **වක්‍ර වූ වරහන්**( "**{.....}**" ). පටන් ගද්දිත්, අවසන් කරද්දිත්, මේ වක්‍ර වරහන් අපි පාවිච්චි කරනව.

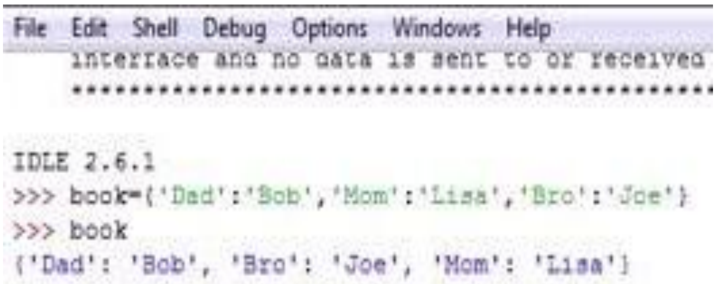
මම එකක් හදලම පෙන්වන්නම්.

book={ 'Dad': 'Bob', 'Bro': 'Joe', 'Mom': 'Lisa'}

මේකෙ තියෙන්නේ, පවුලක සාමාජිකයන්ට අදාල පොඩි Dictionary එකක්. මේකෙ තියෙන යතුරු(keys) තමයි, Dad ,Bro, Mom. ඒවාට අදාල අගයන්(values) වෙන්නේ, Bob, Joe, Lisa .

දැන් Enter එක එබුහම, අපේ Dictionary එක Run වෙනව.

book
{ 'Dad': 'Bob', 'Bro': 'Joe', 'Mom': 'Lisa'}




දැන් අපිට පුළුවන්, අපේ සාමාන්‍ය Dictionary වල තියෙන වැඩේ කරන්න. ඒ කියන්නේ, වචනෙකට අදාල තේරුම හොයාගන්න.

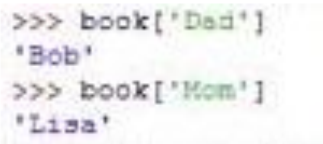
ඒක කරන්නෙ මෙහෙමයි.

ඉස්සෙල්ලාම Dictionary එකේ නම, ඊට පස්සේ කොටු වරහන් ඇතුලෙ අපට අවශ්‍ය යතුර(key) සඳහන් කරන්න ඕන.

ඉන්න මම පෙන්වන්නම්,

book['Dad']
'Bob'

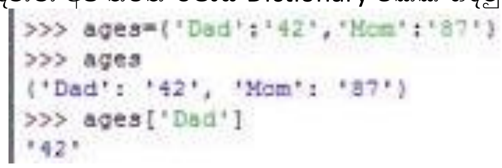
book['Mom']
'Lisa'




ඉතින් තවත් දෙයක් තියෙනව, මේ යතුරු(keys) වලට අදාල අගයන්, string වෙන්නම ඕනෙ නෑ. අපට numbers උනත් දාගන්න පුළුවන්. ඒ ගැන කියල දෙන්න අපි තවත් පොඩි Dictionary එකක් හදමු.

ages = { 'Dad': '42', 'Mom': '87'}

ages['Dad']
42



දැන් දැක්කනෙ, keys වලට අදාල අගයන්, string වෙන්නම අවශ්‍ය නෑ.

දැන් මම කියල දෙන්නම්, Dictionary වලට අදාල methods ටිකක්.

මම ඒක ලියලම පෙන්වන්නම්.

මේක ලියන්න methods ලියන සාමාන්‍ය ආකෘතියටම තමයි[ Sequence.Method(argument) ].

book.clear()

මේකෙදි මේකයි වෙන්නේ, අපි සදහන් කරල තියෙන Dictionary එකේ සියලුම අගයන් ඉවත් කරල, Dictionary එක හිස් කරල දාන එකයි.

Enter එක එබුවොත් දැන් ඒක බලාගන්න පුළුවන්.

{}

වැඩේ හරි නේද?

```
>>> book.clear()
>>> book
{}
>>>
```

තව Method එකක් බලමු.

tuna = ages.copy()

මේකෙදි , අපට අදාල Dictionary එකක්,
වෙනම Dictionary එකක් විදිහට, කොපියක් ගන්න පුළුවන්.

අපි නම් කරපු Dictionary එක, tuna කියන අලුත් Dictionary එකකට කොපි වෙල කියල දැන් ඔයාට පේනව ඇති.

```
>>> tuna=ages.copy()
>>> tuna
{'Dad': '42', 'Mom': '87'}
```

දැන් මම කියල දෙන්න යන්නේ, ගොඩක්ම වැදගත් Method එකක් ගැන.

මේ Method එකෙන් පුළුවන්, අපට අවශ්‍ය Dictionary එකක් ඇතුළෙ අපි සදහන් කරන key එක තියෙනවද නැද්ද කියල හොයාගන්න.

ඒක කරන්නේ මෙහෙමයි,

tuna.has_key('Mom')
True

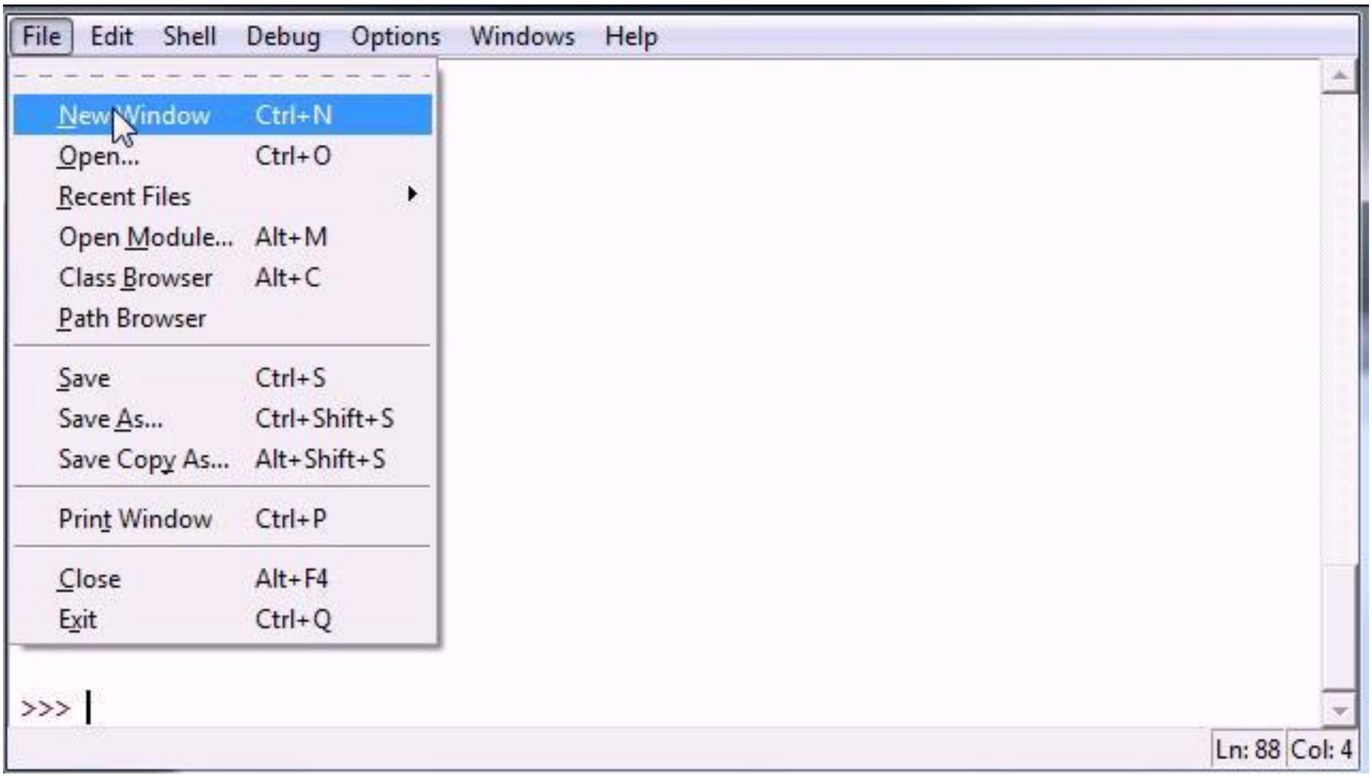
tuna.has_key('Apples')
False

```
>>> tuna.has_key('Mom')
True
>>> tuna.has_key('Apples')
False
>>>
```

ඔයාට ඒ ගැන, දැන් තේරෙනව ඇති.

අදාල key එක, Dictionary එකේ තියෙනවනම්, True කියල output වෙනව, නැත්නම් False කියල output වෙනව.
හරි, අපි Dictionary ගැන ගොඩක් දේවල් කතා කරා. ඒ වගේම Programmes හදද්දි මේ Dictionary අපට ගොඩක් වැදගත් වෙනව.

ඊළඟ පාඩමට..........................................................................................................


File Edit Shell Debug Options Windows Help
New Window Ctrl+N
Open... Ctrl+O
Recent Files
Open Module... Alt+M
Class Browser Alt+C
Path Browser
Save Ctrl+S
Save As... Ctrl+Shift+S
Save Copy As... Alt+Shift+S
Print Window Ctrl+P
Close Alt+F4
Exit Ctrl+Q
>>>
Ln: 88 Col: 4

# 20. If Statements (if ප්‍රකාශන)

අපි මේතාක් දුරට python වල, මූලික දේවල් ගොඩක් ඉගෙනගත්ත. මෙතනින් එහාට මම කියල දෙන්නම්, ක්‍රමලේඛ හදන විදිහ ගැන.

හරි අපි වැඩේට බහිමු.

IDLE එකෙන් දැනට සමු අරගෙන, මම දැන් යනව new window එකකට. New window එකේ programme එකක් ලියල, ඒක Run කරන්න නම්, අනිවාර්යෙන් ඒක Save කරන්න ඕන.

හරි, මම දැන් sting එකක් හදනව.

tuna="fish"

If ප්‍රකාශන ගැන, දැන් මම ඔයාට කියල දෙනව. මේකෙන් ගොඩක් දේවල් ලේසියෙන්ම, අපිට කරන්න පුළුවන්.

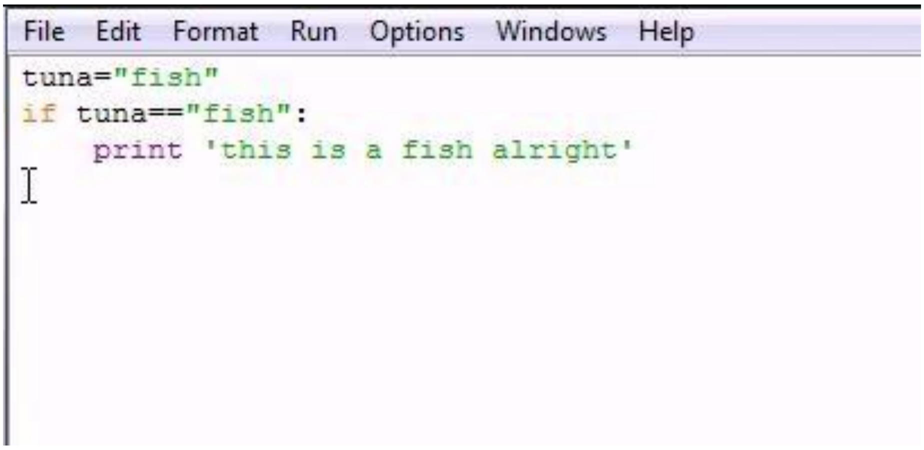
if tuna=="fish":මේකෙදි මම සමාන ලකුණු දෙකක් යොදාගෙන තියෙනව.
මෙතනදි වෙලා තියෙන්නෙ **පරීක්ෂාවක්(testing) .**
යම් අගයක් පරීක්ෂා කිරීමට අපි **සමාන ලකුණු දෙකක්** යොදනව. ඒ වාගෙම අපේ ප්‍රකාශන අවසානයට **දෙතිත**(colon":") යොදනව, හොඳට මතක තියාගන්න(ගොඩක් වැදගත්).

මේ මුළු ප්‍රකාශනේ තේරුම තමයි, ''tuna, fish ට සමාන උනොත්'' කියන එකයි. ඉතින් මේ ප්‍රකාශනය යටතේ, එහෙම උනොත්, අරක කරන්න, මේක කරන්න, කියල අපිට **විවිධ වූ විධාන** දෙන්න පුළුවන්.

If tuna == ''fish'':
    Print 'this is a fish alright'

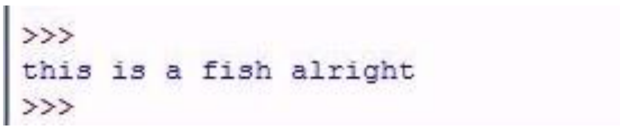
දැන් Enter එක ඔබපුහම, මේ ප්‍රකාශනය යටතේ අපිට අවශ්‍ය කරන දේවල් ලියන්න පුළුවන්.

හොඳට මතක තියාගන්න, දෙතිත දාල Enter එක එබුව ගමන්, එම ප්‍රකාශනය යටතේ තමයි, ඉන්පසු ලියන දේවල් ලියවෙන්නෙ.(indent- පේළිවල සාමාන්‍ය ඉමෙන් ඇතුළට වන සේ ලියවේ.)




දැන් මම මේකෙ කියල තියෙන්නේ tuna, fish ට සමාන වන්නේ නම්, 'this is a fish alright' කියල print කරන්න කියලයි.

දැන් F5 එක ඔබල, මේ programme එක, run කරල බලමු.
'this is a fish alright'



මේකෙ වුන දේ ගැන මම තව පාරක් කියන්නම්.

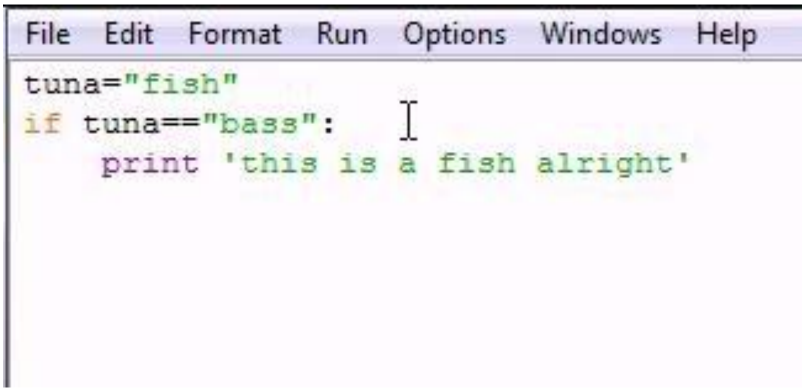
අපට තියෙනව variable එකක් tuna=”fish” කියල.

දැන් අපි පරීක්ෂාවක් කරනව.

tuna කියන එක fish ට සමාන වන්නෙ නම්, ‘this is a fish alright’ කියල print කරන්න, එත් ඒකට සමාන නැත්නම්, මොනවත් කරන්න එපා.

අපේ programme එක මම දැන් පොඩ්ඩක් වෙනස් කරනව.

```
tuna=”fish”
if tuna == “bass”:
        print ‘this is a fish alright’
```



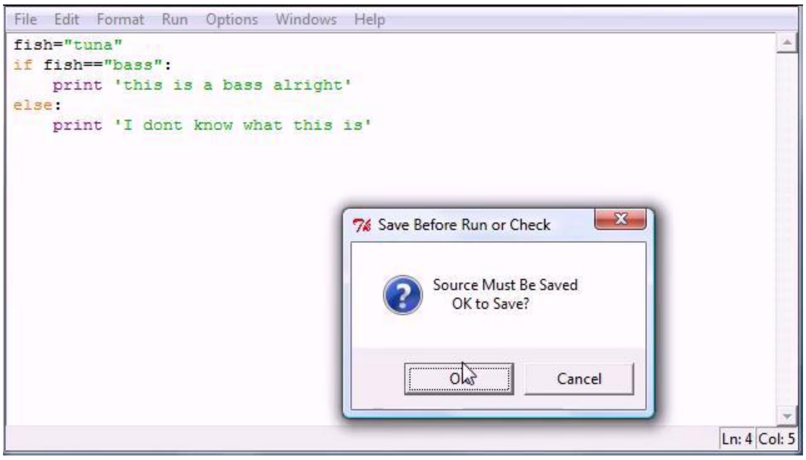

දැන් මොකද වෙන්නෙ කියල බලන්න, මේක run කරමු. ඇත්තටම මොනවත් උනේ නෑ.

හරි, if ප්‍රකාශන වල මූලික දේ මේක තමයි,
අපේ පරීක්ෂාව හරි නම්, අපිට අවශ්‍ය දේවල් ටිකක් කරන්න කියල, විධාන දෙන්න පුළුවන්, ඊට පස්සේ ඒ විධාන run වෙනවා. ඒත් පරීක්ෂාව වැරදි නම්, මොනවත් නොකර ඉන්නව.
ඒත් මේක if ප්‍රකාශන ලියන එක විදිහක් විතරයි.

# 21. Else and elif (Else සහ elif අනුකේතන)

දැන් මම කියල දෙන්නම්, පරීක්ෂාව වැරදි නම්, අපට අවශ්‍ය විධාන හදාගන්න හැටි. ඒක කරන්නේ else කියන අනුකේතයෙන් ඒ ගැන දැන් බලමු.

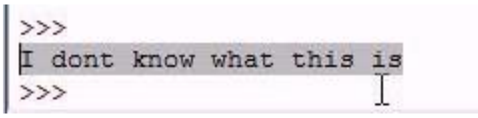
අපේ කලින් උදාහරණයම
මම ඉස්සරහට අරගෙන යනව.




tuna="fish"
if tuna == "bass":
print 'this is a bass alright'
else:
print 'I dont know what this is'

හරි, දැන් මේකයි වෙන්නෙ, ඉස්සෙල්ලාම පළමු ප්‍රකාශනේ සත්‍ය නම්, එයට අදාල විධාන ටික ක්‍රියාත්මක කරනව, ඒත් වැරදි නම් ඔව්, එතකොට අපිට දැන් තවත් තේරීමක් තියෙනව. වැරදි නම්, else යටතේ අපිට කරන්න අවශ්‍ය දේල type කරලයි තියෙන්නෙ.

දැන් run කරල බලමු.
'I dont know what this is'
වැඩේ හරි නේද?

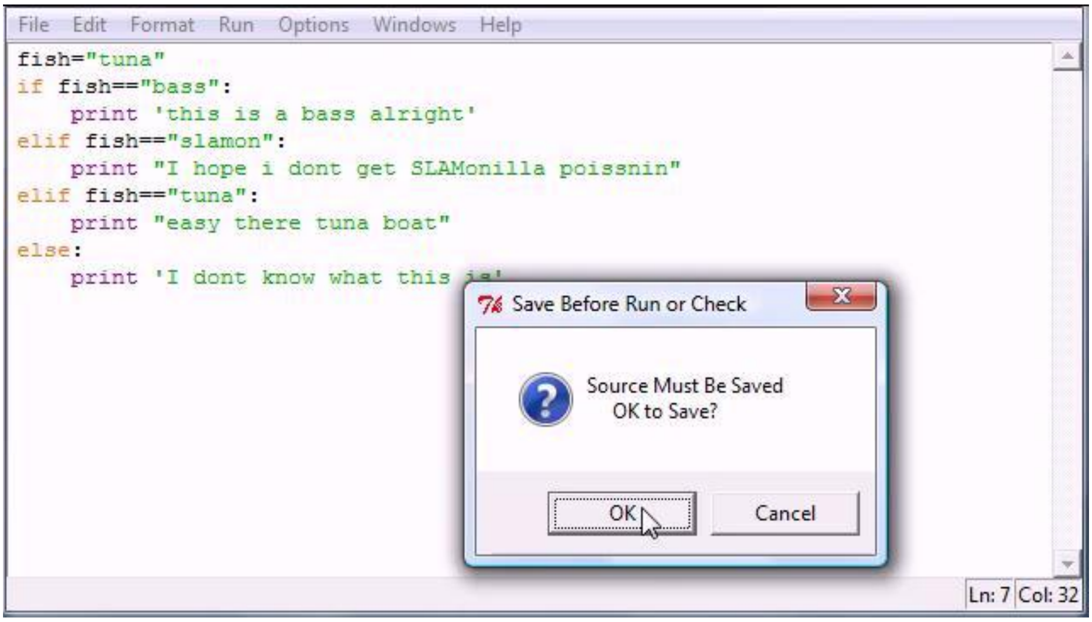



අපි, තවත් මෙතනින් ඉස්සරහට යමු. අපි හිතමු, අපිට ගොඩක් පාරවල් සටහන් කරන්න ඕනෙ කියල. පළවෙනියට දෙයක් පරීක්ෂා කරල හරි ගියේ නැත්නම්, තවත් එකක්, එහෙමත් නැත්නම්, තව එකක්, එහෙමත් වුනේ නැත්නම්, තවත් දෙයක්, ඒ වගේ ගොඩක් පාරවල් සටහන් කරන්න අවශ්‍ය නම්, අපි පාවිච්චි කරනව **elif** කියන අනුකේතනය.

මේක හැම වෙලාවෙම දාන්නේ, if ප්‍රකාශනයයි ,else ප්‍රකාශනයයි අතරට වෙන්නයි. elif ඒවා ඕන තරම් අපිට දාන්න පුළුවන්, අපට අවශ්‍යතාවයන් ගොඩක් තියෙනව නම්.

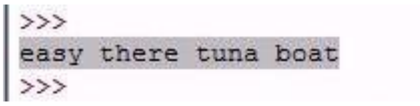
හරි,ඒ ගැන බලමු දැන්.

tuna="fish"
if tuna == "bass":
print 'this is a bass alright'
elif fish == "slamon":
print "I hope I don't get SLAMonilla poissnin"
elif fish == "tuna":
print "easy there tuna boat"
else:
print 'I dont know what this is'



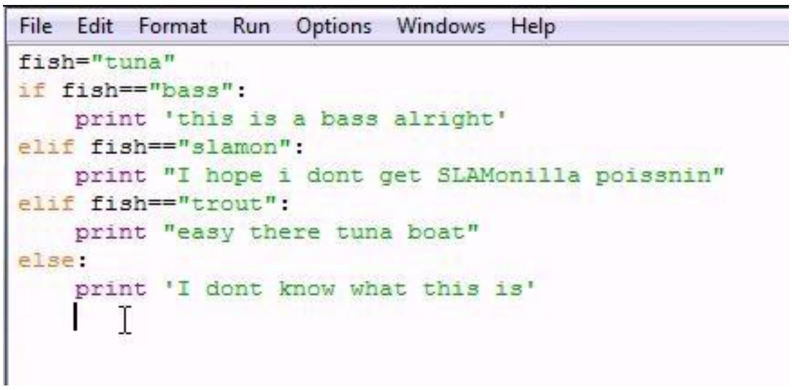

ඉතින්, දැන් මේකෙ වෙන දේ, මම විස්තර කරන්නම්. අපිට තියෙනව tuna=”fish” කියල variable එකක්.
ඉස්සෙල්ලාම if tuna == “bass”: කියන ප්‍රකාශනයට යනව ඒක වැරදි.
එහෙනම් ඊලඟ ප්‍රකාශනයට යනව, elif fish == “slamon”: ඒකත් වැරදි නේද? එහෙනම් අනිත් එකට පනිනව.
elif fish == “tuna”:
ආ..... ඒක නම් හරි! ඉතින් දැන්, ඒක යටතේ තියෙන දේවල් ටික run වෙයි.

Programme එක run කරල බලමු, මම කියපු දේද වෙන්නෙ කියල.
”easy there tuna boat”
හරි, අපි හිතපු දේ වුනා නේද?

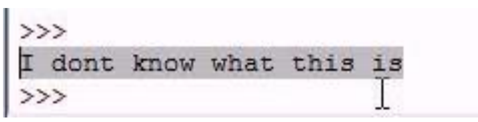



අපි තවත් පොඩි වෙනසක් කරමු.

මේ Programme එකේ tuna කියන එක වෙනුවට, trout කියල දාමු.
elif fish == “trout”




දැන් run කරල බලමු.
’I dont know what this is’




මේ වතාවෙදි වුනේ මේකයි, ප්‍රකාශන එකිනෙක පරීක්ෂා කර කර බැලුව. මොකක්වත් හරි ගියෙ නෑ. ඒ නිසා අන්තිමට else එකට යනව, ඒකෙ තියෙන දේ run කරනව.
අන්තිමට,අපිට තේරෙනව, විසඳුමක් නැතිම වුනාම, programme එක, else යටතේ තියෙන දේ කරල අයින් වෙනව.

# 22. Nesting Statements (අනු-ප්‍රකාශන)

If ප්‍රකාශන ගැන, මම තවත් දෙයක් කියල දෙන්නම්.

ඉස්සෙල්ලාම පොඩි variables ටිකක් හදාගමු

thing = “animal”
animal = “cat”

හරි, If ප්‍රකාශන යටතේ අපිට තවත් if ප්‍රකාශන යොදන්න පුළුවන්. මම කරලම පෙන්වන්නම්.

```
File  Edit  Format  Run  Options  Windows  Help
thing = "animal"
animal = "cat"

if thing == "animal":
    if animal =="cat":
        print 'this is a cat'
    else:
        print 'i dont know what this animal is'

else:
    print 'i dont know what this thing is'
```

දැන් Run කරල බලමු.
this is a cat

```
>>>
this is a cat
```

මෙතනදී වුනේ මේකයි,
thing කියන එක animal වලට සමානයි. ඒ නිසා, ඒ යටතේ ඇති විධාන run වෙනව.
එතකොට හම්බවෙනව, if animal == ‘cat ’කියල එකක්.
අපි හදපු දෙවන variable එක දිහා බලපුහම තේරෙනව, මේ කියන දේ හරි කියල. ඒ නිසා එම ප්‍රකාශනයේ ඇති විධාන run වෙනව. ඒකයි output එක විදිහට this is cat කියල ආවේ.

අපි දැන් thing = “house” කියල වෙනස් කරල, programme එක run කරල බලමු.
I don’t know what this is

```
>>>
i dont know what this thing is
>>>
```

```
File  Edit  Format  Run  Options  Windows  Help
thing = "house"
animal = "cat"

if thing == "animal":
    if animal =="cat":
        print 'this is a cat'
    else:
        print 'i dont know what this animal is'

else:
    print 'i dont know what this thing is'
```

දැන් ඔයාට තේරෙනව,
if thing == “animal”: කියන ප්‍රකාශනය සත්‍ය නොවේ නම්, ඒ යටතේ ඊට පහළින් තියෙන ප්‍රකාශන එතනින් එහාට වැඩ නොකරන බව.

.

ඉතින් මේ විදිහට python ගැන ලියවුනු මාගේ පළමු වන ග්‍රන්ථය අවසන් වෙනව. ඉවරයි කිව්වට මෙතනින් මේ ග්‍රන්ථය ඉවර වෙන්නෙ නෑ .
මෙම ග්‍රන්ථයේ දෙවන කොටසින්, මම ඔයාට python වල තියෙන, මෙහි සඳහන් සිද්ධාන්ත වලට වඩා තරමක් ගැඹුරු වූ සිද්ධාන්ත ගැන කියල දෙනවා.
ඉතින් ඒ දේවල් ඉගෙනගන්න කලින් මෙම ග්‍රන්ථයේ සඳහන්, python වල මූලික දේවල් ගැන නැවත නැවත පුහුණුවෙන්න.
මෙම ග්‍රන්ථයේ සඳහන් ක්‍රමශිල්ප පිළිබඳව යම්තාක් දැනුමක් ඔබ ලබාගත්තේ නම්, python පිළිබඳව පසුගිය අ.පො.ස(උ.පෙළ) ප්‍රශ්ණ පත්‍ර වල සඳහන් ප්‍රශ්න වලින් 60% වත් ඔබට පහසුවෙන් නිවරදි පිළිතුරු සෙවීමට උදව් වනු ඇත.

එසේනම් මෙහි දෙවන ග්‍රන්ථයෙන් අප යළි හමුවෙමු.
ඔබට ජය!