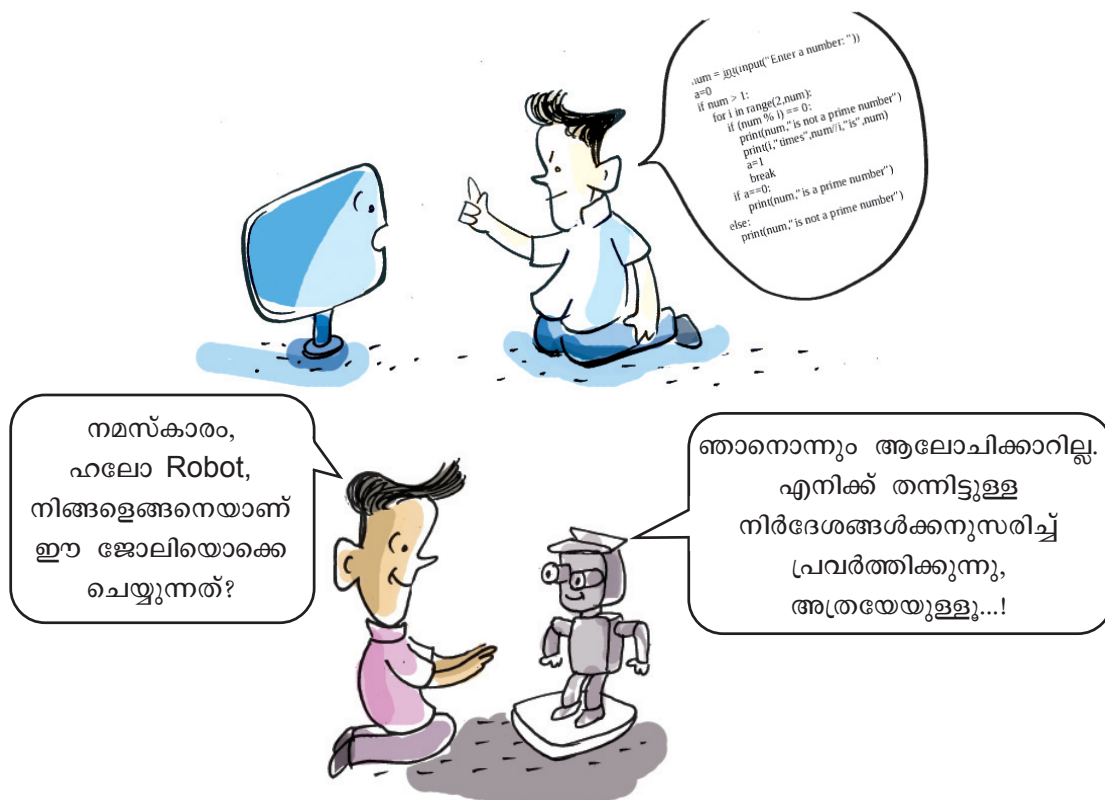


പ്രോഗ്രാമിങ്



റോബോട്ടുകളിൽ മാത്രമല്ല, നിങ്ങൾ പരിചയപ്പെട്ട എല്ലാ സോഫ്റ്റ്‌വെയറുകളുടെയും പ്രവർത്തനത്തിനു പിന്നിലും അവയ്ക്ക് മുൻകൂട്ടി നൽകിയ നിർദ്ദേശങ്ങളാണുള്ളത്.

പ്രോഗ്രാമുകൾ

കമ്പ്യൂട്ടറിനു നൽകുന്ന നിർദ്ദേശങ്ങളുടെ കൂട്ടമാണ് പ്രോഗ്രാമുകൾ എന്നറിയപ്പെടുന്നത്.

നിങ്ങൾ പരിചയപ്പെട്ട ജിമ്പ്, ലിബർടാഫീസ് റൈറ്റർ, കാൽക്ക്, ഇംപ്രസ് തുടങ്ങിയ എല്ലാ സോഫ്റ്റ്‌വെയറുകളും തയ്യാറാക്കിയിരിക്കുന്നത് വിവിധ പ്രോഗ്രാമിങ് ഭാഷകളുപയോഗിച്ചാണ്.

എട്ടാം ക്ലാസിൽ നിങ്ങൾ സ്ക്രാച്ച് സോഫ്റ്റ്‌വെയർ ഉപയോഗിച്ച് കമ്പ്യൂട്ടർ ഗെയിമുകൾ തയ്യാറാക്കിയത് ഓർമ്മയുണ്ടല്ലോ. സ്ക്രാച്ച് സോഫ്റ്റ്‌വെയറിൽ സ്ക്രൈപ്റ്റുകളെ നിയന്ത്രിക്കാനായി വ്യത്യസ്ത ബ്ലോക്കുകൾ ഉപയോഗിച്ചിരുന്നു. ഓരോ ബ്ലോക്ക് ഉൾപ്പെടുത്തുമ്പോഴും സ്ക്രൈപ്റ്റിനെ ചലിപ്പിക്കാൻ പ്രത്യേകമായ ചില നിർദ്ദേശങ്ങളടങ്ങിയ ഒരു പ്രോഗ്രാമാണ് പ്രവർത്തിക്കുന്നത് എന്നു നിങ്ങൾ തിരിച്ചറിഞ്ഞിട്ടുണ്ടോ?

ഇതുപോലെ ഒരു കൂട്ടം നിർദ്ദേശങ്ങൾ (പ്രോഗ്രാമുകൾ) ഓരോ സോഫ്റ്റ്‌വെയറിനു പിന്നിലും പ്രവർത്തിക്കുന്നുണ്ടാകുമല്ലോ.

ഇത്തരം പ്രോഗ്രാമുകൾ തയ്യാറാക്കുന്നത് എങ്ങനെയാണ് മനസ്സിലാക്കാം.

പ്രോഗ്രാമിങ് ഭാഷകൾ

നാം കൊടുക്കുന്ന എല്ലാ നിർദ്ദേശങ്ങളും കമ്പ്യൂട്ടറിന് നേരിട്ട് മനസ്സിലാക്കാൻ സാധിക്കുമോ?

കമ്പ്യൂട്ടറിന് നേരിട്ടു മനസ്സിലാക്കാൻ കഴിയുന്ന ഭാഷ ബൈനറി ഭാഷയാണ്.

0,1 എന്നീ രണ്ടു ചിഹ്നങ്ങൾ മാത്രമുപയോഗിച്ചുകൊണ്ടുള്ള നിർദ്ദേശങ്ങളടങ്ങിയ ഭാഷയാണ് 'ബൈനറിഭാഷ'.

അടിസ്ഥാനപരമായി, കമ്പ്യൂട്ടർ ഒരു ഇലക്ട്രോണിക് യന്ത്രമാണല്ലോ. ഏതൊരു യന്ത്രത്തിനും വൈദ്യുതിയുടെ സാന്നിധ്യവും അസാന്നിധ്യവും മാത്രമാണ് മനസ്സിലാക്കാൻ സാധിക്കുക. വൈദ്യുതിയുടെ സാന്നിധ്യത്തെ 1 കൊണ്ടും അസാന്നിധ്യത്തെ 0 കൊണ്ടും സൂചിപ്പിക്കാറുണ്ട്. ബൈനറി ഭാഷയിലെഴുതുന്നതിന് സമാനമായി വൈദ്യുതി പശുസുകളുടെ സാന്നിധ്യവും അസാന്നിധ്യവും യന്ത്രത്തിൽ ഉണ്ടാക്കാം. അതു കൊണ്ടുതന്നെ ബൈനറിഭാഷ യന്ത്രഭാഷ എന്നും അറിയപ്പെടുന്നു.

കമ്പ്യൂട്ടറിന് ബൈനറിഭാഷ മാത്രമേ മനസ്സിലാകൂ എങ്കിൽ എല്ലാ നിർദ്ദേശങ്ങളും ഈ ഭാഷയിലാക്കേണ്ടി വരില്ലേ?



അതു ശരിയാ, ഇത് വല്ലാത്ത ബുദ്ധിമുട്ടല്ലേ?



അൽഗോരിതം

കമ്പ്യൂട്ടറിൽ ഒരു പ്രവർത്തനത്തിനുള്ള നിർദ്ദേശങ്ങൾ നൽകുമ്പോൾ അവ ശരിയായ രീതിയിൽ ചെറിയ ചെറിയ പ്രവർത്തനഘട്ടങ്ങളായി ക്രമീകരിക്കേണ്ടി വരും. ഇങ്ങനെ ഒരു പ്രശ്ന നിർധാരണത്തിനു നൽകുന്ന ഘട്ടംഘട്ടമായ പ്രവർത്തന രീതിയാണ് അൽഗോരിതം.

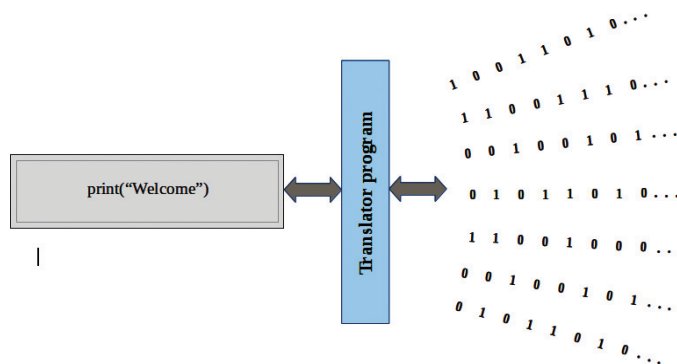
പക്ഷേ, ബൈനറിഭാഷയിൽ നിർദ്ദേശങ്ങൾ തയ്യാറാക്കുക എന്നത് എളുപ്പമുള്ള കാര്യമല്ല. കുറേ പേജുകളുള്ള ഒരു പ്രോഗ്രാമിൽ 0 വും 1 ഉം മാത്രമേയുള്ളൂ എന്നു കരുതുക. പിന്നീട് ഈ പ്രോഗ്രാമൊന്നു തിരുത്തേണ്ടിവന്നാലുള്ള അവസ്ഥ ആലോചിച്ചുനോക്കുക. തല കറങ്ങിപ്പോകും, അല്ലേ! അങ്ങനെയാണ് എളുപ്പം മനസ്സിലാക്കാൻ സാധിക്കുന്ന നിർദ്ദേശങ്ങളടങ്ങിയ പ്രോഗ്രാമിങ് ഭാഷകൾ രൂപംകൊണ്ടത്. അത്തരമൊരു പ്രോഗ്രാമിങ് ഭാഷയാണ് Python. മറ്റുചില പ്രോഗ്രാമിങ് ഭാഷകളാണ് C, C++ , Java എന്നിവ.

പൈത്തൺ

വളരെ ലളിതമായ ഒരു പ്രോഗ്രാമിങ് ഭാഷയാണ് പൈത്തൺ. എളുപ്പത്തിൽ മനസ്സിലാക്കാൻ കഴിയുന്ന സിന്റാക്സ് (പദവിന്യാസ ഘടന) ആണ് പൈത്തണിനുള്ളത്. ജാവ, സി തുടങ്ങിയ പ്രോഗ്രാമിങ് ഭാഷകളിൽ ഉള്ളതിലും വളരെ കുറച്ച് ചിഹ്നങ്ങൾ മാത്രമേ ഇതിൽ ഉപയോഗിക്കുന്നുള്ളൂ. ഗെയ്ഡോ വാൻ റോസ്സത്തിന്റെ (Guido van Rossum) നേതൃത്വത്തിലാണ് പൈത്തൺ രൂപകല്പന ചെയ്തത്. ഓപ്പൺ സോഴ്സ് ലൈസൻസോടു കൂടിയ സോഫ്റ്റ്‌വെയറാണ് ഇത്. ബ്ലേൻഡർ, ഓപ്പൺഷോട്ട് വിഡിയോ എഡിറ്റർ തുടങ്ങിയ പല സോഫ്റ്റ്‌വെയറുകളും പൈത്തൺ ഭാഷയിൽ തയ്യാറാക്കിയിട്ടുണ്ട്.

പക്ഷേ, എങ്ങനെയാണ് ഈ നിർദ്ദേശങ്ങൾ കമ്പ്യൂട്ടർ മനസ്സിലാക്കുക?

ഇത്തരം പ്രോഗ്രാമിങ് ഭാഷകളിലുള്ള നിർദ്ദേശങ്ങൾ ഒരു ട്രാൻസ്‌ലേറ്റർ പ്രോഗ്രാമിന്റെ സഹായത്തോടെയാണ് കമ്പ്യൂട്ടർ മനസ്സിലാക്കുന്നത്. താഴെയുള്ള ചിത്രം നോക്കുക (ചിത്രം 4.1).



ചിത്രം 4.1 ട്രാൻസ്‌ലേറ്റർ പ്രോഗ്രാമിന്റെ ചിത്രീകരണം

എല്ലാ പ്രോഗ്രാമിങ് ഭാഷകൾക്കും അതിലെ നിർദ്ദേശങ്ങൾക്ക് യോജിക്കുന്ന ഒരു ട്രാൻസ്‌ലേറ്റർ പ്രോഗ്രാം ഉണ്ടായിരിക്കും. നിർദ്ദേശങ്ങൾ നൽകിയാൽ അതിനെ യന്ത്രഭാഷയിലേക്ക് മാറ്റിക്കൊടുക്കുന്നത് ഈ പ്രോഗ്രാം ചെയ്തുകൊള്ളും. പക്ഷേ, ഈ ട്രാൻസ്‌ലേറ്റർ പ്രോഗ്രാമിന് യന്ത്രഭാഷയാക്കി മാറ്റാൻ സാധിക്കുന്ന തരത്തിലുള്ള നിർദ്ദേശങ്ങൾ മാത്രമേ കൊടുക്കാവൂ. ഈ നിർദ്ദേശങ്ങളും ട്രാൻസ്‌ലേറ്റർ പ്രോഗ്രാമും ഓരോ പ്രോഗ്രാമിങ് ഭാഷയ്ക്കും വ്യത്യസ്തമായിരിക്കും.

ഇനി നമുക്ക് പൈത്തൺഭാഷയിൽ ഒരു പ്രോഗ്രാം തയ്യാറാക്കുന്നത് എങ്ങനെയാണെന്ന് പരിചയപ്പെടാം.

ട്രാൻസ്‌ലേറ്റർ പ്രോഗ്രാമുകൾ

പ്രോഗ്രാമിങ് ഭാഷയിലുള്ള നിർദ്ദേശങ്ങളെ ബൈനറി ഭാഷയിലേക്കും തിരിച്ച് ബൈനറി ഭാഷയിലുള്ള നിർദ്ദേശങ്ങളെ പ്രോഗ്രാമിങ് ഭാഷയിലേക്കും മാറ്റുന്നതിന് ട്രാൻസ്‌ലേറ്റർ പ്രോഗ്രാമുകൾ ഉപയോഗിക്കുന്നു.

പ്രവർത്തനം 4.1 - print സ്റ്റേറ്റ്‌മെന്റ്

നിങ്ങളുടെ പേര് പ്രദർശിപ്പിക്കാനുള്ള ഒരു പ്രോഗ്രാം സ്റ്റേറ്റ്‌മെന്റ് പൈത്തൺ ഭാഷയിൽ തയ്യാറാക്കിനോക്കാം.

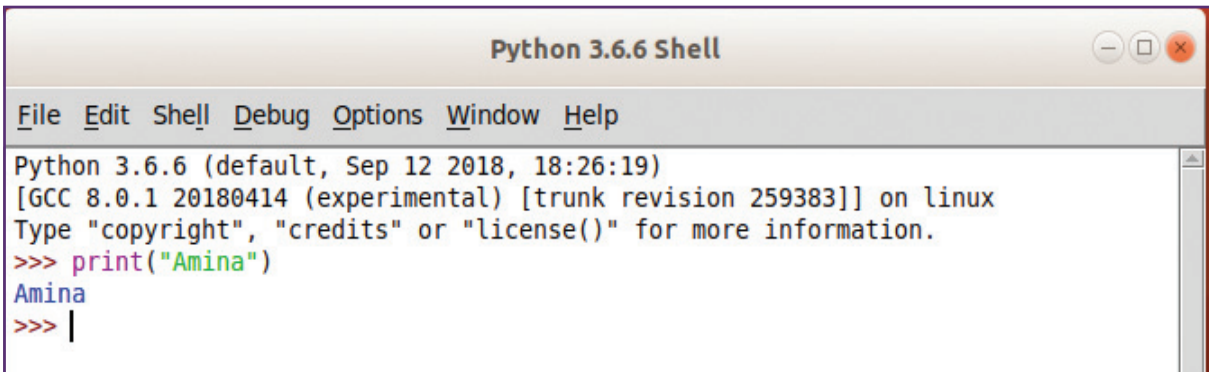
```
print("Amina")
```

ഈ പ്രോഗ്രാമിൽ print എന്നത് പേര് പ്രദർശിപ്പിക്കാനുള്ള പൈത്തൺ നിർദ്ദേശവും ഉദ്ധരണിയിലുള്ളത് പ്രദർശിപ്പിക്കേണ്ട വാക്കുമാണ്.

ഈ പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുന്നതിന് ചുവടെ നൽകിയ

പ്രവർത്തനം ചെയ്തുനോക്കൂ.

- ◆ Programming മെനുവിൽനിന്ന് IDLE3 തുറക്കുക. Python Shell ജാലകം തുറന്നുവരും.
- ◆ ഷെൽ പ്രോംപ്റ്റിൽ `print("Amina")` എന്ന് ടൈപ്പ് ചെയ്ത് എന്റർ കീ അമർത്തുക.
- ◆ ഇതിന്റെ ഒരുപൂട്ട് Python Shell ജാലകത്തിൽത്തന്നെ ലഭിക്കുന്നില്ലേ? (ചിത്രം 4.2)



```

Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
Python 3.6.6 (default, Sep 12 2018, 18:26:19)
[GCC 8.0.1 20180414 (experimental) [trunk revision 259383]] on linux
Type "copyright", "credits" or "license()" for more information.
>>> print("Amina")
Amina
>>> |
  
```

ചിത്രം 4.2 പൈത്തൺ ഷെൽ ജാലകം

ഇവിടെ Amina എന്നത് ഒരു ഇംഗ്ലീഷ് വാക്കാണല്ലോ. ഇത് പ്രോഗ്രാമിന്റെ പദാവലിയിൽ ഉൾപ്പെടുന്നുമില്ല. ഇത്തരം വാക്കുകളെ സ്ട്രിങ്ങുകൾ എന്നു വിളിക്കുന്നു. സ്ട്രിങ്ങുകൾ പ്രദർശിപ്പിക്കാൻ `print` സ്റ്റേറ്റ്‌മെന്റിന്റെ കൂടെ അവയെ ഉദ്ധരണിയിൽ നൽകണം എന്നു മനസ്സിലായല്ലോ.

IDE (Integrated Development Environment)

പൈത്തൺ പ്രോഗ്രാം കോഡുകൾ എഴുതാൻ ടെക്സ്റ്റ് എഡിറ്ററുകൾ ഉപയോഗിക്കാം. ഇങ്ങനെ എഴുതി സേവ് ചെയ്ത പ്രോഗ്രാം ഒരു ടെർമിനൽ ഉപയോഗിച്ച് പ്രവർത്തിപ്പിക്കാനും സാധിക്കും. ഉദാഹരണമായി, പൈത്തൺ പ്രോഗ്രാം അടങ്ങിയ ഒരു ഫയൽ `pgm1.py` എന്ന പേരിൽ നിങ്ങളുടെ ഫോൾഡറിൽ സേവ് ചെയ്തിട്ടുണ്ട് എന്നു കരുതുക. ഈ പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കാൻ ഫോൾഡറിൽ നിന്ന് ടെർമിനൽ തുറന്ന് അതിൽ `python3 pgm1.py` എന്ന് ടൈപ്പ് ചെയ്ത് എന്റർ ചെയ്താൽ മതിയാകും.

എന്നാൽ പ്രോഗ്രാം കോഡുകൾ എഴുതാനും പ്രവർത്തിപ്പിക്കാനും സഹായിക്കുന്ന വിവിധ സംയോജിത സോഫ്റ്റ്‌വെയറുകളുണ്ട്. ഇവയെ IDE (Integrated Development Environment) എന്നു പറയുന്നു. IDLE എന്നത് ലളിതമായ ഒരു IDE സോഫ്റ്റ്‌വെയറാണ്. പൈത്തണിന്റെ പൈത്തൺ 2, പൈത്തൺ 3 എന്നീ പതിപ്പുകളിൽ പദവിന്യാസ ഘടനയിൽ ചെറിയ വ്യത്യാസങ്ങളുണ്ട്. ഈ പാഠഭാഗത്ത് പൈത്തൺ 3 ആണ് പ്രതിപാദിക്കുന്നത്. IDLE3 പതിപ്പ് ഇതിനായി ഉപയോഗിക്കാം.

സ്ട്രിങ്ങുകൾ

വാക്കുകളുടെ അല്ലെങ്കിൽ അക്ഷരങ്ങളുടെയും അക്ഷരങ്ങളുടെയും ചിഹ്നങ്ങളുടെയും കൂട്ടത്തെ സ്ട്രിങ് എന്നു വിളിക്കാം. ഉദാഹരണത്തിൽ നൽകുന്നത് എന്തുതന്നെയായാലും (നമ്പറുകൾ ആണെങ്കിലും) അത് സ്ട്രിങ്ങായി പരിഗണിക്കപ്പെടും.

പ്രവർത്തനം 4.2 - പൈത്തൺ ഷെല്ലിൽ പ്രോഗ്രാം പ്രവർത്തന പരിശീലനം

ചുവടെ കൊടുത്ത സ്റ്റേറ്റ്‌മെന്റുകൾ ഓരോന്നായി പൈത്തൺ ഷെല്ലിൽ ടൈപ്പ് ചെയ്ത് ഔട്ട്പുട്ട് നിരീക്ഷിച്ച് പട്ടിക പൂർത്തിയാക്കുക.

പ്രോഗ്രാം സ്റ്റേറ്റ്‌മെന്റ്	ഔട്ട്പുട്ട്
print ("Welcome")	Welcome
print ("123")	
print (123)	
print (8+9)	
print ("8"+"9")	

കൂടുതൽ പ്രോഗ്രാമുകൾ പരിചയപ്പെടാം

പ്രവർത്തനം 4.3 - പരപ്പളവ് കാണാം

സ്കൂളിലെ കളിസ്ഥലത്തിന്റെ പരപ്പളവ് കാണണം എന്നിരിക്കട്ടെ. കളിസ്ഥലത്തിന്റെ നീളവും വീതിയും കണ്ടെത്തിയല്ലോ. ഇതിന്റെ പരപ്പളവ് കാണാൻ എന്താണ് ചെയ്യേണ്ടത്?

കമ്പ്യൂട്ടറിന് ഇതിനുള്ള നിർദ്ദേശങ്ങൾ നൽകിയാലോ?

നീളം 80 മീറ്ററും വീതി 60 മീറ്ററും ആണെന്നിരിക്കട്ടെ,

പരപ്പളവ് കാണേണ്ട?

l (length) എന്ന ചരം നീളമായും b (breadth) എന്ന ചരം വീതിയായും പരിഗണിച്ചാൽ,

$l = 80$ എന്നും $b = 60$ എന്നും നൽകാം. പരപ്പളവിനെ A എന്ന് ചരം കൊണ്ട് സൂചിപ്പിക്കുകയാണെങ്കിൽ,

$A = l * b$ ആയിരിക്കും, അല്ലേ? ഇനി A പ്രദർശിപ്പിക്കാം.

ഇത് പൈത്തൺ പ്രോഗ്രാമായി എഴുതുമ്പോൾ എങ്ങനെയായിരിക്കും?

$l = 80$

length = 80

$b = 60$

breadth = 60

ചരങ്ങൾ

വിവരങ്ങൾ ശേഖരിച്ചു വയ്ക്കാൻ ചരങ്ങൾ ഉപയോഗിക്കാം. അക്ഷരങ്ങളോ വാക്കുകളോ ചരങ്ങളായി സൂചിപ്പിക്കാം. നമ്പറുകൾ, സ്ട്രിങ്ങുകൾ തുടങ്ങിയവ ചരങ്ങൾക്ക് വിലകളായി സ്വീകരിക്കാം.

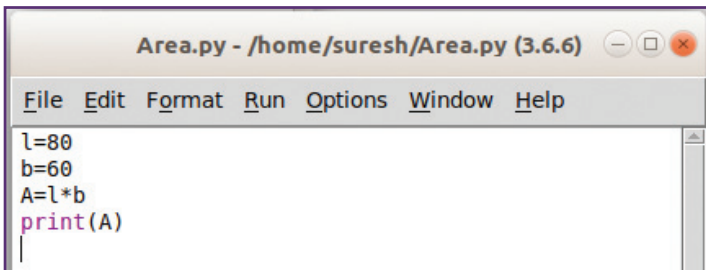

```
A = l*b      # area = length x breadth
print(A)     # display value of A
```

ഈ പ്രോഗ്രാമിലുള്ള സ്റ്റേറ്റ്‌മെന്റുകൾ ഓരോന്നായി Python Shell ൽ പ്രവർത്തിപ്പിച്ചുനോക്കൂ.

ഇത് മുഴുവനും ഒന്നിച്ചു പ്രവർത്തിപ്പിക്കാൻ ബുദ്ധിമുട്ടല്ലേ?

ഒരു പുതിയ ഫയലുണ്ടാക്കി ഈ പ്രോഗ്രാം തയ്യാറാക്കി പ്രവർത്തിപ്പിച്ചാലോ?

Python Shell ജാലകത്തിൽനിന്നു പുതിയ ഫയൽ തുറക്കുക (File → New File). തുറന്നുവരുന്ന ജാലകത്തിൽ (ചിത്രം 4.3) പ്രോഗ്രാം ടൈപ്പ് ചെയ്ത് നിങ്ങളുടെ ഫോൾഡറിൽ സേവ് ചെയ്യുക. സേവ് ചെയ്യുമ്പോൾ യോജിച്ച ഫയൽ നാമം നൽകുക.



ചിത്രം 4.3 പൈത്തൺ പ്രോഗ്രാം ടൈപ്പ് ചെയ്യാനുള്ള ജാലകം

തയ്യാറാക്കിയ പൈത്തൺ കോഡുകൾ പ്രവർത്തിപ്പിച്ചു നോക്കണ്ടേ?

Run മെനുവിൽ Run Module സെലക്ട് ചെയ്ത് ഇതു പ്രവർത്തിപ്പിക്കാം. ഇതിന്റെ ഔട്ട്പുട്ട് Python Shell ജാലകത്തിലാണു ലഭിക്കുന്നത്. വീണ്ടും ഈ പ്രോഗ്രാം പ്രവർത്തിപ്പിച്ചുനോക്കൂ. എന്താണ് ഉത്തരം ലഭിക്കുന്നത്? ഈ പ്രോഗ്രാം എത്ര പ്രാവശ്യം പ്രവർത്തിപ്പിച്ചാലും ഒരേ ഉത്തരം മാത്രമേ ലഭിക്കുന്നുള്ളൂ, അല്ലേ? എന്താണു കാരണം?



വിവരണം (Comment)

പൈത്തൺ പ്രോഗ്രാമിൽ ഓരോ സ്റ്റേറ്റ്‌മെന്റിന്റെയും വിവരണം (Comment) # ചിഹ്നത്തിനു ശേഷം ചേർക്കാവുന്നതാണ്. # ചിഹ്നത്തിന് ശേഷം ആ വരിയിൽ ചേർത്ത വിവരണങ്ങൾ പ്രോഗ്രാം പ്രവർത്തിക്കുമ്പോൾ പരിഗണിക്കില്ല.

ഇതുപോലുള്ള വിവരണങ്ങൾ (Comments) നിങ്ങൾ തയ്യാറാക്കുന്ന എല്ലാ പ്രോഗ്രാമുകളിലും ഉൾപ്പെടുത്തുമല്ലോ.

ഫയൽ എക്സ്റ്റൻഷൻ

പൈത്തൺ ഫയലുകളുടെ എക്സ്റ്റൻഷൻ .py ആണ്. IDLE സോഫ്റ്റ്‌വെയറിൽ പൈത്തൺ ഫയലുകൾ സേവ് ചെയ്യുമ്പോൾ .py എക്സ്റ്റൻഷനോടുകൂടിയാണ് സേവ് ആകുന്നത്.



ഇൻ്റർപ്രിട്ടറും കമ്പയിലറും

പ്രോഗ്രാമിങ് ഭാഷയിലുള്ള നിർദ്ദേശങ്ങളെ യന്ത്രഭാഷയിലേക്ക് മാറ്റാൻ പ്രധാനമായും ഇൻ്റർപ്രിട്ടർ, കമ്പയിലർ എന്നിങ്ങനെ രണ്ടുതരത്തിലുള്ള ട്രാൻസ്ലേറ്റർ പ്രോഗ്രാമുകൾ ഉപയോഗിക്കാറുണ്ട്. ഇൻ്റർപ്രിട്ടർ പ്രോഗ്രാമിലെ ഓരോ സ്റ്റേറ്റ്മെന്റും പ്രത്യേകമായി യന്ത്രഭാഷയിലേക്കു മാറ്റുന്നു. എന്നാൽ കമ്പയിലർ പ്രോഗ്രാമ മുഴുവൻ ഒരുമിച്ച് യന്ത്രഭാഷയിലേക്കു മാറ്റുകയാണ് ചെയ്യുന്നത്.

വ്യത്യസ്തമായ അളവുകൾ ഉപയോഗിച്ച് പരസ്പരം കണ്ടുപിടിക്കാനുള്ള ഒരു പ്രോഗ്രാമായി ഇതു മാറ്റണമെങ്കിൽ എന്തൊക്കെ വ്യത്യാസങ്ങൾ വരുത്തണം?

- ◆ നീളം (l), വീതി (b) എന്നിവയുടെ വില പ്രോഗ്രാം പ്രവർത്തിക്കുന്ന സമയത്ത് നൽകാൻ കഴിയണം.

ഇതിനായി ഉപയോഗിക്കാവുന്ന പൈത്തൺ നിർദ്ദേശമാണ് `eval(input())`.

- ◆ ഈ പ്രോഗ്രാം പ്രവർത്തിപ്പിച്ചുനോക്കൂ.

```
l=eval(input())
```

```
b=eval(input())
```

```
A=l*b
```

```
print(A)
```

പ്രോഗ്രാം പ്രവർത്തിക്കുന്ന സമയത്ത് l, b എന്നിവയ്ക്ക് വ്യത്യസ്ത അളവുകൾ നൽകി എൻ്റർ ചെയ്തുനോക്കുക. അളവുകൾ മാറുന്നതിനനുസരിച്ച് വ്യത്യസ്ത പരസ്പരം ലഭിക്കുന്നില്ലേ?

എന്നാൽ ഇവിടെ ഈ പ്രോഗ്രാം പ്രവർത്തിക്കുമ്പോൾ ഏതൊക്കെ അളവുകളാണ് നൽകേണ്ടതെന്നും (input) എന്താണ് ഉത്തരം ലഭിക്കേണ്ടതെന്നും (output) പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുന്ന ആളിന് സൂചനകളൊന്നുമില്ല.

പ്രോഗ്രാം എഴുതുമ്പോൾ input സ്റ്റേറ്റ്മെന്റിനൊപ്പവും print സ്റ്റേറ്റ്മെന്റിനൊപ്പവും ഇതിനുള്ള സൂചന നൽകാനാവും. മേൽപ്പറഞ്ഞപോലെ പ്രോഗ്രാം മാറ്റി എഴുതിയിരിക്കുന്നത് നോക്കുക.

```
l=eval(input("Enter length of the rectangle:"))
```

```
b=eval(input("Enter breadth of the rectangle:"))
```

```
A=l*b
```

```
print("Area of the rectangle=",A)
```

ഈ പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുമ്പോൾ, മുൻപ് എഴുതിയിരുന്ന തിരിനിന്ന് എന്തെല്ലാം മാറ്റങ്ങളാണ് നിങ്ങൾക്കു നിരീക്ഷിക്കാൻ കഴിയുന്നത്?

പ്രവർത്തനം 4.4 - സ്ക്രിപ്റ്റ് സംയോജനം നടത്താം

പ്രോഗ്രാം പ്രവർത്തിക്കുന്ന സമയത്ത് നിങ്ങളുടെ സ്കൂളിന്റെ പേരു നൽകിയാൽ നിങ്ങൾ ആ സ്കൂളിലെ വിദ്യാർത്ഥിയാണ് എന്ന് മറുപടി ലഭിക്കുന്ന ഒരു പ്രോഗ്രാം തയ്യാറാക്കി നോക്കാം. പ്രോഗ്രാം പ്രവർത്തിക്കുന്ന സമയത്ത് നൽകേണ്ട വില സ്ക്രിപ്റ്റ് ആണെങ്കിൽ `eval(input())` ന്റെ സ്ഥാനത്ത് `input()` എന്നു നൽകിയാൽ മതി.

```
s=input("Enter your School's name:")
```

```
print("You are a student of",s)
```

പ്രവർത്തനം 4.5 - ഗണിതക്രിയകളുടെ ഉപയോഗം

ഒരു ബഹുഭുജത്തിന്റെ വശങ്ങളുടെ എണ്ണം നൽകിയാൽ ബഹുഭുജത്തിന്റെ കോണുകളുടെ അളവുകളുടെ തുക ലഭിക്കുന്നതിനുള്ള പ്രോഗ്രാം തയ്യാറാക്കുക.

ബഹുഭുജത്തിന്റെ പേര് `a` എന്ന ചരത്തിലും വശങ്ങളുടെ എണ്ണം `n` എന്ന ചരത്തിലും സ്വീകരിക്കുന്നു എന്നിരിക്കട്ടെ.

കോണുകളുടെ തുക, $s=(n-2)*180$ ആണല്ലോ. ഇത് കണ്ടുപിടിക്കണം, തുടർന്ന് ആ വില പ്രദർശിപ്പിക്കണം.

```
a=input("Enter the name of polygon:")
```

```
n=eval(input("Enter number of sides:"))
```

```
s=(n-2)*180
```

```
print("Sum of angles of ",a," is ",s)
```

പൈത്തൺ പ്രോഗ്രാമിൽ സ്ക്രിപ്റ്റുകളുടെ ഉപയോഗവും ഗണിതക്രിയകളുടെ ഉപയോഗവും പരിശീലിച്ചല്ലോ. ഒരു വില പരിശോധിച്ച് വ്യത്യസ്ത തീരുമാനങ്ങൾ എടുക്കേണ്ട പല സന്ദർഭങ്ങളും വന്നുചേരാറുണ്ട്. ഇങ്ങനെയുള്ള സന്ദർഭങ്ങൾ കൈകാര്യം ചെയ്യാൻ സാധിക്കുന്ന ഒരു പ്രവർത്തനം ചെയ്തുനോക്കാം.

പ്രവർത്തനം 4.6 - സ്കോർ പരിശോധന

സ്കൂൾതല കിസ് മത്സരത്തിന് നിങ്ങൾക്കു ലഭിച്ച സ്കോർ നൽകുമ്പോൾ, സ്കോർ പരിശോധിച്ച് ജില്ലാതല മത്സരത്തിലേക്കു തിരഞ്ഞെടുക്കപ്പെട്ടിട്ടുണ്ടോ എന്ന് അറിയിക്കുന്ന ഒരു പ്രോഗ്രാം തയ്യാറാക്കുക (ജില്ലാതല മത്സരത്തിലേക്ക് 80 ൽ കൂടുതൽ സ്കോർ ലഭിച്ചവരെ മാത്രമേ പങ്കെടുപ്പിക്കുകയുള്ളൂ എന്നു കരുതുക).

ഇവിടെ എന്തൊക്കെ നിർദ്ദേശങ്ങൾ നൽകണം?

സ്ക്രിപ്റ്റ് സംയോജനം

`print` സ്റ്റേറ്റ്‌മെന്റ് ഉപയോഗിച്ച് ഒന്നിലധികം സ്ക്രിപ്റ്റുകൾ ഒന്നിച്ച് പ്രദർശിപ്പിക്കുമ്പോഴും സ്ക്രിപ്റ്റുകളും ചരങ്ങളുടെ വിലയും ഒന്നിച്ച് പ്രദർശിപ്പിക്കുമ്പോഴും ഉദ്ധരണിയിലുള്ള സ്ക്രിപ്റ്റുകളും ചരങ്ങളും കോമയിട്ട് വേർതിരിക്കേണ്ടതാണ്.



കണ്ടീഷണൽ സ്റ്റേറ്റ്മെന്റ്. if...else

ഒരു നിബന്ധന പാലിക്കുന്നുണ്ടോ ഇല്ലയോ എന്നു പരിശോധിച്ച്, പാലിക്കുന്നുണ്ടെങ്കിൽ എന്തു ചെയ്യണം എന്നും ഇല്ലെങ്കിൽ എന്തു ചെയ്യണമെന്നും നിർദ്ദേശിക്കുന്നതിന് if...else എന്ന കണ്ടീഷണൽ സ്റ്റേറ്റ്മെന്റ് ഉപയോഗിക്കാം. if, else എന്നിവയ്ക്ക് ശേഷം ഉപയോഗിക്കുന്ന സ്റ്റേറ്റ്മെന്റുകൾക്കുള്ള indent ശ്രദ്ധിക്കുക.

```
*score.py - /home/suresh/score.py (3.6.6)*
File Edit Format Run Options Window Help
a=eval(input("Enter your score: "))
if a>80:
    print("Congratulations, You are selected")
else:
    print("Sorry, You are not selected")
```

ചിത്രം 4.4 if...else സ്റ്റേറ്റ്മെന്റ് ഉപയോഗിച്ച പ്രോഗ്രാം

- ◆ പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുമ്പോൾ നൽകുന്ന സ്കോർ ഒരു ചരത്തിൽ സ്വീകരിക്കണം (ചരം a ആണെന്ന് കരുതുക). ഇതിന് eval(input()) സ്റ്റേറ്റ്മെന്റ് ഉപയോഗിക്കാം.
- ◆ കിട്ടിയ സ്കോർ 80 ൽ കൂടുതൽ ആണോ എന്നു പരിശോധിക്കണം. ഒരു നിബന്ധന പാലിക്കുന്നുണ്ടോ എന്നു പരിശോധിക്കാൻ പ്രോഗ്രാമുകളിൽ കണ്ടീഷണൽ സ്റ്റേറ്റ്മെന്റ് ഉപയോഗിക്കാം. ഇവിടെ if എന്ന കണ്ടീഷണൽ സ്റ്റേറ്റ്മെന്റ് ഉപയോഗിച്ച് a>80 ആണോ എന്നു പരിശോധിക്കാം.
- ◆ ഈ നിബന്ധന ശരിയാകുമ്പോൾ Congratulations, You are Selected എന്നു പ്രദർശിപ്പിക്കണം.

പ്രോഗ്രാം എങ്ങനെയായിരിക്കും?

```
a=eval(input("Enter your score:"))
```

```
if a>80:
```

```
    print ("Congratulations, You are Selected")
```

ഇവിടെ പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുമ്പോൾ എൺപതോ അതിൽ കുറവോ ആയ സ്കോർ നൽകിയാലോ?

ഒന്നും ഔട്ട്പുട്ട് ആയി പ്രദർശിപ്പിക്കുന്നില്ല, അല്ലേ?

അതായത് പരിശോധിക്കുന്ന നിബന്ധന ശരിയല്ലെങ്കിൽ മറുപടി ലഭിക്കുന്നില്ല. അതുകൊണ്ട് ഇതിനെ if...else സ്റ്റേറ്റ്മെന്റ് ഉപയോഗിച്ച് വിപുലപ്പെടുത്താം.

നിബന്ധന ശരിയാകുമ്പോൾ "Congratulations, You are Selected" എന്നും ശരിയല്ലെങ്കിൽ "Sorry, You are not Selected" എന്നും പ്രദർശിപ്പിക്കണം.

```
a=eval(input("Enter your score:"))
```

```
if a>80:
```

```
    print ("Congratulations, You are Selected")
```

```
else:
```

```
    print("Sorry, You are not Selected")
```

വ്യത്യസ്ത ചരങ്ങളുടെ വിലയായി വ്യത്യസ്ത വിലകൾ നൽകുന്ന രീതി പരിചയപ്പെടുവല്ലോ. ഇനി ഒരുകൂട്ടം വിലകളെ സൂചിപ്പിക്കാനുള്ള പൈത്തൺ സ്റ്റേറ്റ്മെന്റ് പരിചയപ്പെടാം.



range

പൈത്തൺ ഭാഷയിൽ ഒരു കൂട്ടം വിലകളെ സൂചിപ്പിക്കാൻ റേഞ്ച് (range()) ഉപയോഗിക്കുന്നു.

range(10) എന്നത് 10 ൽ കുറവായ 10 സംഖ്യകളെ സൂചിപ്പിക്കുന്നു. ഇവിടെ തുടക്കസംഖ്യ 0 ആയും വർധന 1 ആയും പരിഗണിക്കുന്നു. അതായത് 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

range(1,10) എന്നത് 10 ൽ കുറവായ 1 മുതലുള്ള സംഖ്യകളെ സൂചിപ്പിക്കുന്നു. ഇവിടെ വർധന 1 ആയിരിക്കും. അതായത് 1, 2, 3, 4, 5, 6, 7, 8, 9.

range(1,20,2) എന്നത് 20 ൽ കുറവായ 1 മുതലുള്ള ഒറ്റ സംഖ്യകളെ സൂചിപ്പിക്കുന്നു. ഇവിടെ വർധന 2 ആയിരിക്കും. അതായത് 1, 3, 5, 7, 9, 11, 13, 15, 17, 19.

പ്രവർത്തനം 4.7 - range നിർദ്ദേശങ്ങൾ ഉപയോഗിച്ച് സൂചിപ്പിക്കുന്ന സംഖ്യകൾ കണ്ടെത്തുക.

താഴെ കൊടുത്തിരിക്കുന്ന range നിർദ്ദേശങ്ങൾ സൂചിപ്പിക്കുന്ന സംഖ്യകൾ ഒന്നെഴുതിനോക്കൂ.

നിർദ്ദേശം	സൂചിപ്പിക്കുന്ന സംഖ്യകൾ
range (3, 100, 5)	3, 8, 13, 18, 23, 28.....83, 88, 93, 98
range (0, 50, 10)	
range (50, 0, -10)	
range (2, 20)	
range (15)	

ഈ നിർദ്ദേശങ്ങൾ പൈത്തൺ ഷെല്ലിൽ ടൈപ്പ് ചെയ്ത് പ്രവർത്തിപ്പിച്ച് അവ സൂചിപ്പിക്കുന്ന സംഖ്യകൾ കണ്ടെത്താം.

പൈത്തൺ ഷെല്ലിൽ പ്രവർത്തിപ്പിക്കുമ്പോൾ range() എന്നതിനു പകരം list(range()) എന്നു സൂചിപ്പിക്കണം.

നിങ്ങൾ എഴുതിയ ഉത്തരം ശരിയാണോ എന്നു പരിശോധിക്കുമല്ലോ?

ഒരു കൂട്ടം സംഖ്യകളെ സൂചിപ്പിക്കുന്ന range() സ്റ്റേറ്റ്‌മെന്റ് പരിചയപ്പെട്ടല്ലോ. ഒന്നോ അതിലധികമോ പ്രവർത്തനങ്ങൾ ആവർത്തിച്ചു വരേണ്ട സന്ദർഭങ്ങളിൽ range() എങ്ങനെ ഉപയോഗിക്കുന്നു എന്നു പരിചയപ്പെടാം.

പ്രവർത്തനം 4.8 - ആവർത്തിക്കാം പ്രവർത്തനങ്ങൾ

ആവർത്തിച്ച് ചെയ്യേണ്ട നിർദ്ദേശങ്ങൾ കൊടുക്കാൻ for ലൂപ്പുകൾ

പ്രോഗ്രാമുകളിൽ ഒന്നോ ഒരു കൂട്ടമോ സ്റ്റേറ്റ്‌മെന്റുകൾ ആവർത്തിക്കേണ്ടി വരുമ്പോൾ അവയെ ഒരു ലൂപ്പ് (Loop) ൽ ഉൾപ്പെടുത്താം. പൈത്തണിൽ ഉപയോഗിക്കുന്ന ഒരു ലൂപ്പ് സ്റ്റേറ്റ്‌മെന്റ് ആണ് **for** ലൂപ്പ്.

താഴെ കൊടുത്തിരിക്കുന്ന പ്രോഗ്രാം നിരീക്ഷിക്കൂ.

```
for i in range(1,11):
```

```
    print(i)
```

1 മുതൽ 10 വരെയുള്ള എണ്ണൽ സംഖ്യകൾ പ്രദർശിപ്പിക്കാനുള്ള ഒരു പ്രോഗ്രാമാണിത്.

for ലൂപ്പിൽ i എന്ന ചരത്തിന് 1,2,3,4,5,6,7,8,9,10 എന്നീ വിലകൾ ഓരോന്നും സ്വീകരിക്കുമ്പോഴും print(i) എന്ന സ്റ്റേറ്റ്‌മെന്റ് പ്രവർത്തിക്കുന്നു.

അതായത് ഇവിടെ 10 പ്രാവശ്യം i യുടെ വ്യത്യസ്ത വിലകൾ പ്രദർശിപ്പിക്കുന്നു.

ആവർത്തിക്കുന്ന സ്റ്റേറ്റ്‌മെന്റുകൾ for ലൂപ്പിനകത്ത് ഉപയോഗിക്കുമ്പോഴുള്ള ഇന്റന്റ് ശ്രദ്ധിച്ചല്ലോ.

ആദ്യ പ്രവർത്തനത്തിൽ (പ്രവർത്തനം 4.1) print("Amina") എന്ന സ്റ്റേറ്റ്‌മെന്റിലൂടെ നിങ്ങളുടെ പേര് പ്രദർശിപ്പിക്കാനുള്ള നിർദ്ദേശം പരിചയപ്പെട്ടല്ലോ. ഈ പേര് 20 പ്രാവശ്യം പ്രദർശിപ്പിക്കണമെങ്കിൽ എന്താക്കെ നിർദ്ദേശങ്ങളാണ് കൂടുതൽ നൽകേണ്ടിവരുക?

print("Amina") എന്ന സ്റ്റേറ്റ്‌മെന്റ് 20 പ്രാവശ്യം ആവർത്തിക്കേണ്ടതുകൊണ്ട് അത് ഒരു ലൂപ്പിൽ ഉൾപ്പെടുത്താം. അതുകൊണ്ട് പ്രോഗ്രാം ഇങ്ങനെ എഴുതാം:

```
for i in range(20):
```

```
    print("Amina")
```

ഇവിടെ range(20) എന്നത് [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19] എന്നീ 20 വിലകളെ സൂചിപ്പിക്കുന്നു. i എന്ന ചരം ഇവയിൽ ഓരോ വില സ്വീകരിക്കുമ്പോഴും, print("Amina") എന്ന സ്റ്റേറ്റ്‌മെന്റ് പ്രവർത്തിക്കുന്നു. അതായത് i, പൂജ്യം എന്ന വില സ്വീകരിക്കുമ്പോൾ Amina എന്നു പ്രിന്റ് ചെയ്യുന്നു. തുടർന്ന് അടുത്ത വില (i=1) സ്വീകരിക്കുമ്പോഴും Amina എന്നു പ്രിന്റ് ചെയ്യുന്നു. ഇങ്ങനെ i യുടെ വില ലിസ്റ്റിലെ എന്തുതന്നെയൊരു Amina എന്നുതന്നെയൊന്നല്ലോ പ്രിന്റ് ചെയ്യുക. അതുകൊണ്ട് ഈ വാക്ക് ആകെ 20 തവണ പ്രിന്റ് ചെയ്യപ്പെടുന്നു.

പ്രവർത്തനം 4.9 - സംഖ്യകൾ പ്രദർശിപ്പിക്കുക

2 മുതൽ 100 വരെയുള്ള ഇരട്ടസംഖ്യകളെ പ്രദർശിപ്പിക്കാനുള്ള ഒരു പ്രോഗ്രാം തയ്യാറാക്കണമെന്നിരിക്കട്ടെ. ഈ സംഖ്യകളെ സൂചിപ്പിക്കാൻ range (2,101,2) ഉപയോഗിക്കാം. k എന്ന ചരത്തിന് ഈ വ്യത്യസ്ത വിലകൾ നൽകി പ്രദർശിപ്പിക്കാം.

```
for k in range(2,101,2):
```

```
    print(k)
```

while ലൂപ്പ്

പൈത്തണിൽ ഉപയോഗിക്കുന്ന മറ്റൊരു ലൂപ്പ് സ്റ്റേറ്റ്‌മെന്റാണ് while ലൂപ്പ്. for ലൂപ്പിനു പകരം while ലൂപ്പ് ഉപയോഗിക്കുമ്പോൾ ചരത്തിന്റെ തുടക്കവിലയെ സൂചിപ്പിക്കുന്നതും വർധനവിനെ സൂചിപ്പിക്കുന്നതുമായ സ്റ്റേറ്റ്‌മെന്റുകൾ പ്രത്യേകമായി നൽകേണ്ടിവരും. for ലൂപ്പിലുള്ള ഒരു പ്രോഗ്രാമിനെ while ലൂപ്പിലേക്കു മാറ്റി എഴുതിയിരിക്കുന്നതു കാണുക (പട്ടിക 4.1). എന്താക്കെ വ്യത്യാസങ്ങളാണ് കാണാൻ സാധിക്കുന്നത്?

for ലൂപ്പ്	while ലൂപ്പ്
<pre>for k in range (2, 101, 2): print (k)</pre>	<pre>k = 2 while k<101: print (k) k = k + 2</pre>
<p>ഓരോ പ്രാവശ്യവും k യുടെ വില 2 വർദ്ധിച്ച് പുതിയ വിലയായി മാറുന്നതിന് $k = k + 2$ എന്നത് ഉപയോഗിക്കുന്നു.</p>	

പട്ടിക 4.1 for ലൂപ്പിനു പകരം while ലൂപ്പ് ഉപയോഗിച്ച പ്രോഗ്രാം



വിലയിരുത്താം

- ചുവടെ കൊടുത്തിരിക്കുന്ന പ്രോഗ്രാമിന്റെ ഔട്ട്പുട്ട് (a യുടെ വില) എന്തായിരിക്കും?

```
a=2
a=a+3
print(a)
```

a. 5 b. 6 c. 2 d. 3
- 1 മുതൽ 20 വരെയുള്ള എണ്ണൽസംഖ്യകളെ സൂചിപ്പിക്കാൻ പൈത്തണിൽ ഉപയോഗിക്കുന്നത് ചുവടെ കൊടുത്തിരിക്കുന്നവയിൽ ഏതാണ്?

a. range(20) b. range(1,20) c. range(1,21) d. range(1,21,2)
- for i in range(1,5):

```
print("Welcome")
```

ഈ പ്രോഗ്രാമിന്റെ ഔട്ട്പുട്ടിൽ Welcome എന്ന് എത്ര പ്രാവശ്യം പ്രദർശിപ്പിക്കും?

a. 5 b. 4 c. 2 d. 1
- a="3"
b="2"

```
print(a+b)
```

ഈ പ്രോഗ്രാമിന്റെ ഔട്ട്പുട്ട് എന്തായിരിക്കും?

a. 5 b. 6 c. 23 d. 32

5. അനുവിന്, 1 മുതൽ 25 വരെയുള്ള എണ്ണൽസംഖ്യകളുടെ തുക പ്രദർശിപ്പിക്കാനുള്ള ഒരു പ്രോഗ്രാം തയ്യാറാക്കണം. അനു തയ്യാറാക്കിയ പ്രോഗ്രാം താഴെ കൊടുത്തിരിക്കുന്നു. ഇതിൽ തെറ്റുകൾ കടന്നുകൂടിയിട്ടുണ്ട്. ഇതൊന്ന് ശരിയാക്കിക്കൊടുക്കാമോ?

```
s=0
```

```
for i in range(25):
```

```
    s=s+i
```

```
print(s)
```



തുടർപ്രവർത്തനങ്ങൾ

- ◆ 200 ൽ കുറവായ 7 ന്റെ ഗുണിതങ്ങളായ സംഖ്യകൾ പ്രദർശിപ്പിക്കാനുള്ള പൈത്തൺ പ്രോഗ്രാം തയ്യാറാക്കുക.
- ◆ 2 മുതൽ 50 വരെയുള്ള ഇരട്ടസംഖ്യകളുടെ തുക കണ്ടുപിടിക്കാനുള്ള പൈത്തൺ പ്രോഗ്രാം തയ്യാറാക്കുക.
- ◆ ഒരു സംഖ്യ ഇൻപുട്ട് ആയി സ്വീകരിച്ച് ആ സംഖ്യയുടെ 20 വരെയുള്ള ഗുണനപ്പട്ടിക പ്രദർശിപ്പിക്കുന്ന ഒരു പൈത്തൺ പ്രോഗ്രാം തയ്യാറാക്കുക.

