WHAT'S THIS?

This document is the official user's manual for the programming language – 'Snick'.

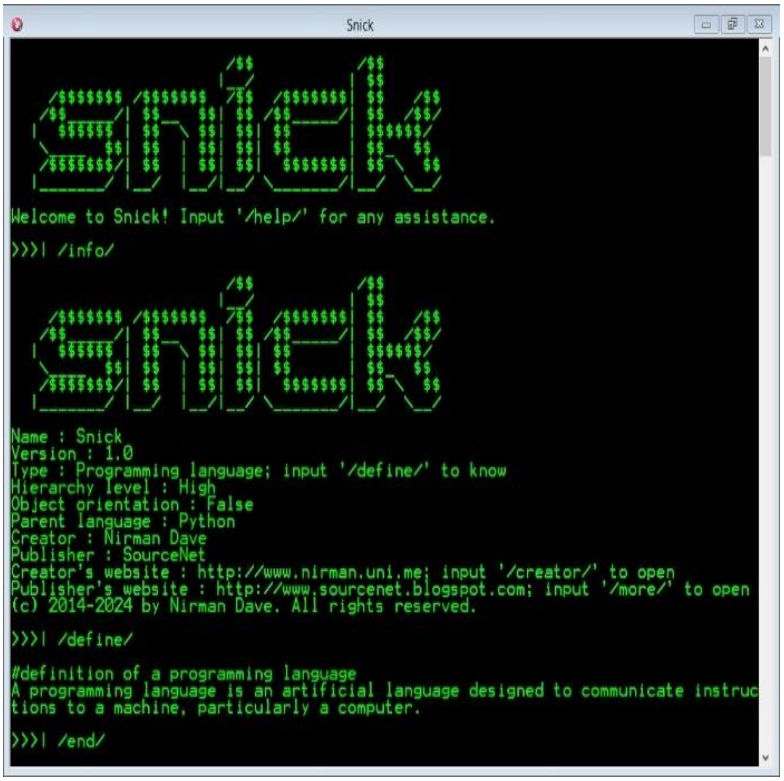
WRITTEN BY

Nirman Dave

A Snick Peek

What is Snick?

Snick is a completely different concept of programming. It comprises of modules which are imported each time a code is to be written. The module can be executed within the main frame. Snick GUI allows the user to both code and run the program within. The program runs at the 'snick' of the written code. Snick is aimed at beginners who have absolutely no expertise in programming and are about to enter the vast field. Hence, develops a base for such novices so that they can learn further coding with ease.



Snick technical details

Name : Snick
Version : 1.0
Type : Programming language
Hierarchy level : High
Object orientation : False
Parent language : Python
Creator : Nirman Dave
Publisher : SourceNet
Creator's website : http://www.nirman.uni.me
Publisher's website : http://www.sourcenet.blogspot.com
Snick official website : http://www.snick.uni.me

About the creator

I am Nirman Dave, 17 years old and just graduated from The Galaxy School in Rajkot, Gujarat. I have pursued the International Baccalaureate Diploma Programme. Innovation, creativity, programming and movies are the things I live for. I strongly believe in helping the society through the use of technology, a unique but a challenging task. Besides being the CEO of SourceNet and co-founder of Antable, I also teach computer programming at The Galaxy School and will soon mentor the 'Introduction to computer science' course at a university in my town. Additionally, drama, basketball, chess, singing, watching movies and following up to Suits, Sherlock-BBC and FRIENDS are also the activities that keep me occupied throughout the day!

User's Manual

This is the official user's manual for Snick. Don't worry, it's not too long! And you may not need to go through the whole document. Snick isn't that hard after all!

Common notification symbols

Symbol	*Meaning*
#abc	# is for notifying about actions
/	/ is for breaking or exiting a module
{non}	{non} symbolizes no error in the code, the code ran successfully
{error000 – error name : suggestion}	This is how an error is reported

Common commands

Command	*Meaning*
Import abc	Imports module named 'abc'
/info/	Reports information
/help/	Provides assistance
/end/	Ends the program
/more/	Opens SourceNet website
/creator/	Opens creator's website
/snick/	Opens snick website
>>>| abc	Gives command abc, if the command is not recognised the string is printed
time	Shows current time
date	Shows current date
dimedate.detail	Shows calendar details

Programming math

Importing mathadd

Importing mathadd allows you to add two numbers.

```
>>>| import mathadd
#mathadd imported
enter whole integer or break off : 2
enter another whole integer : 3
5
{non}
```

To break out from a mathadd module input '/'

```
>>>| import mathadd
#mathadd imported
enter whole integer or break off : /

>>>|
```

Importing mathsub

Importing mathsub allows you to subtract two numbers.

```
>>>| import mathsub
#mathsub imported
enter whole integer or break off : 5
enter another whole integer : 2
3
{non}
```

To break out from a mathsub module input '/'

```
>>>| import mathsub
#mathsub imported
enter whole integer or break off : /

>>>|
```

Importing mathmulti

Importing mathmulti allows you to multiply two numbers.

```
>>>| import mathmulti
#mathmulti imported
enter whole integer or break off : 2
enter another whole integer : 3
6
{non}
```

To break out from a mathmulti module input '/'

```
>>>| import mathmulti
#mathmulti imported
enter whole integer or break off : /

>>>|
```

Importing mathdiv

Importing mathdiv allows you to add divide numbers.

```
>>>| import mathdiv
#mathadd imported
enter whole integer or break off : 100
enter another whole integer : 4
25
{non}
```

To break out from a mathdiv module input '/'

```
>>>| import mathdiv
#mathdiv imported
enter whole integer or break off : /

>>>|
```

Importing mathsqrt

Importing mathsqrt allows you to find the square root of a number.

```
>>>| import mathsqrt
#mathsqrt imported
enter whole integer : 9
3.0
{non}
```

To break out from a mathsqrt module input '/'

```
>>>| import mathsqrt
#mathsqrt imported
enter whole integer : /

>>>|
```

Importing mathpow

Importing mathpow allows you to raise a number to a power.

```
>>>| import mathpow
#mathpow imported
enter whole integer : 9
enter a power : 3
729
{non}
```

To break out from a mathpow module input '/'

```
>>>| import mathpow
#mathpow imported
enter whole integer : /

>>>|
```

Importing mathran

Importing mathran allows you to random numbers.

```
>>>| import mathran
#mathran imported
>>>{
```

Creating a new random number

A new random number can only be created once the mathran module is imported. Once the module is imported the command prompt changes from >>>| to >>>{
After this enter the code to generate a new random integer.

```
>>>| import mathran
#mathran imported
>>>{ int.random

87
{non}
int or break :
```

This will generate a random number in the range 0 through 100. If you wish to specify a range then, break the int.random function by using '/' and then type down int.random.range this will allow you to specify a range.

```
>>>| import mathran
#mathran imported
>>>{ int.random

87
{non}
int or break :/

>>>{ int.random.range

start value : 1
end value : 6
#value appended
{non}

>>>{ int.random

3
{non}
int or break :
```

Enter key will repeat the process until broken.

List actions

Creating a list: item addition method

A list can be created in two ways. One by adding strings of texts and another using the whole text itself.

Start with importing the list module. Once that is done, create a new list and name it. Then start adding actions. Put '/' to finalize the list.

```
>>>| import list
#list imported
```

```
>>>} list.new
list.name} friends

action or add: ross
action or add: joey
action or add: chandler
action or add: monica
action or add: racheal
action or add: phoebe
action or add: gunther
action or add: jenis
action or add: /

>>>}
```

Once the list is completed, you can enter list.show to view the list in a matrix form. All list actions can only take place when you are in the list module or the command prompt says >>>} instead of the default >>>|

```
>>>| import list
#list imported
>>>} list.new
list.name} friends

action or add: ross
action or add: joey
action or add: chandler
action or add: monica
action or add: racheal
action or add: phoebe
action or add: gunther
action or add: jenis
action or add: /

>>>} list.show

list name : friends
['ross', 'joey', 'chandler', 'monica', 'racheal', 'phoebe', 'gunther', 'jenis']
{non}

>>>}
```

Creating a list: single string method

A list can also be created using a single string method. Where you just have to input a world and Snick will separate it into letters for you and create a list.

```
>>>| import list
#list imported
>>>} list.new.word
list.name} element

word : titanium
#list created
['t', 'i', 't', 'a', 'n', 'i', 'u', 'm']
{non}

>>>}
```

Editing a list

Any list that you create can be edited using the following functions.

Function	*Action*
list.multi	Multiplies the list data an input number of times
list.test	Tests for a particular item in the list
list.putin	Add extra items to the list
list.clear	Delete all items in the list
list.show	Displays the list
list .slash	Cuts the list into a specific value
list.min	Displays the item with the minimum value in the list
list.max	Displays the item with the maximum value in the list
list.len	Displays the length of the list
list.del	Deletes one particular item in the list

Numbering the items in the list

The list you created is numbered in a particular fashion by the computer. Hence, each item is given a value. So, if you wish to select an item in the list you should be able to define its value. Below is the numbering shown for the friends list created above.

```
  0      1      2          3        4        5        6        7
['ross', 'joey', 'chandler', 'monica', 'racheal', 'phoebe', 'gunther', 'jenis']
```

Therefore, the list consists of 8 items, numbered from 0 through 7.

Using the delete function with the numbering pattern

Here is how you can use the delete function keeping in mind the numbering pattern shown above.

```
>>>} list.show

list name : friends
['ross', 'joey', 'chandler', 'monica', 'racheal', 'phoebe', 'gunther', 'jenis']
{non}

>>>} list.del

position: 0
#item deleted
['joey', 'chandler', 'monica', 'racheal', 'phoebe', 'gunther', 'jenis']
{non}

>>>}
```

Learn box actions

Creating a new learn box

A learn box is like a list where every item has a value. The value can be a string or an integer. Import learn, once imported create a new learn function using learn.new, after which define the name of the learn box. This will create a learn box.

```
>>>| import learn
#learn imported
>>>] learn.new
learn.name} friends

#learn has been created
name: friends
{'default word' : 'default meaning'}
{non}

>>>]
```

Once you are in the learn module displaying >>>] instead of >>>| you can code the learn functions. One such function is learn.putin which allows you to add items in the learn box. Putting a '/' will exit the add function and take you to the main module.

```
>>>| import learn
#learn imported
>>>] learn.new
learn.name} friends

#learn has been created
name: friends
{'default item' : 'default value'}
{non}

>>>] learn.putin

enter an item : ross
enter its value : david

#item has been added
{non}

enter an item : chandler
enter its value : matthew

#item has been added
{non}

enter an item : joey
enter its value : matt

#item has been added
{non}

enter an item : /

>>>]
```

All the items have been added to the learn box. Input learn.show to view the box.

```
>>>] learn.show

name : friends
{'default item' : 'default value', 'ross' : 'david', 'chandler' : 'matthew', 'joey' : 'matt'}
{non}
```

Any list that you create can be edited using the following functions.

Function	*Action*
learn.clear	Deletes all items in the box
learn.del	Deleted particular items in a box
learn.find	Finds the value of an item in a box

Else-if-elseif actions

Programming if actions

An else-if-elseif module allows you to define a situation and all possible outcomes. Then Snick allows you to access the situation in order to test the outcomes.

```
>>>| import if
#if imported
If text >>>) lol
else if [press enter to pass] >>>) LOL
else if [press enter to pass] >>>) lOl
else if [press enter to pass] >>>) LoL
else if [press enter to pass] >>>)
then print >>>( Ha Ha Ha
else print >>>( Ho Ho Ho
{non}

run value : lol

Ha Ha Ha
{non}

run value : roflmao

Ho Ho Ho
{non}

run value : /

if text >>>) /

>>>|
```

Programming askif actions

Askif actions are similar to if actions but allow you to ask the user a question. To which multiple choices are given.

```
>>>| import askif
#askif imported
ask text >>>) What is 2+2?
if text >>>) 4
else if [press enter to pass] >>>) four
else if [press enter to pass] >>>) Four
else if [press enter to pass] >>>) FOUR
else if [press enter to pass] >>>) FoUr
then print >>>( Correct
rlse print >>>( Incorrect
{non}

What is 2+2? : 4
```

```
Correct
{non}

What is 2+2? : 22

Incorrect
{non}

What is 2+2? : /

ask text >>>)/

>>>|
```

Creating a 'Hello, World!' application

A 'Hello, World!' application can be created using the else-if-elseif actions. By importing if, we can define the print to hello world string.

```
Import if
#if imported
if text >>>) greet
else if [press enter to pass] >>>) Greet
else if [press enter to pass] >>>) GREET
else if [press enter to pass] >>>) GrEeT
else if [press enter to pass] >>>) gReEt
then print >>>( Hello, World!
else print >>>( Go home human, you're drunk!
{non}

run value : Greet

Hello, World!
{non}

run value : /

if text >>>) /

>>>|
```

Programming the web work action

Programming the web work action using the weberks module

The web work action consists of two main commands. One is to browse the web and another is to crack the website. The code below shows you how to browse the web using the weberks module.

```
>>>| import weberks
#weberks imported

load action : http://www.sourcenet.blogspot.com
#loaded
{non}

load action : /

>>>|
```

Make sure that the weberks action you type contains 'http://'.

Hacking websites using the webhack module

The webhack module allows you to retrieve the source codes of any websites hence access any hidden or visible data in the website. All you have to do is import the module create a new webhack tool and input the website's universal resource locator.

```
>>>| import webhack
>>>) webhack.new

input an existing webpage link including 'http://'

>>>( http:.//www.snick.uni.me

#the website is being hacked…

  <html><head><title>Snick!</title><meta name="keywords" content=""><meta
  name="description" content=""><meta name="revisit-after" content=""><meta name="robots"
  content="INDEX, FOLLOW"></head><frameset rows="100%,*" frameborder="NO" border="0"
  framespacing="0"><frame name="main"
  src="http://www.snick.uni.me/"></frameset><noframes><body bgcolor="#FFFFFF"
  text="#000000"><a href="http://www.snick.uni.me /">Click here to continue to
  http://www.snick.uni.me /</a></body></noframes></html>

#website has been hacked
{non}

>>>( /
>>>) /
>>>|
```

Creating mathematical sequences

Programming sequences through the seq module

A sequence is a list of numbers. The order in which the numbers are listed is important. The seq module can program two widely known sequences. The arithmetic sequence and the Fibonacci sequence.

```
>>>| import seq
#seq imported
>>>:
```

Creating an arithmetic sequence

An arithmetic sequence is a sequence of numbers such that the difference between the consecutive terms is constant. This sequence is mathematically represented by the following formula:

$$a_n = a_1 + (n-1)d$$

Where,

$$n = n^{th}\ term$$
$$a_1 = first\ term$$
$$d = difference$$

With that in mind, once the seq module is imported, command the formation of the sequence using the seqa.generate function. This will prompt you to enter a start value, an end value and a difference.

```
>>>| import seq
#seq imported
>>>: seqa.generate

define min : 1
define max : 10
define difference : 2

1
3
5
7
9
{non}

define min : /

>>>: /

>>>|
```

Creating a Fibonacci sequence

The Fibonacci sequence is a list of numbers in which each number (Fibonacci number) is the sum of the two preceding numbers. The sequence is listed below:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 \ldots$$

This sequence can be mathematically represented using the formula:

$$F_n = F_{n-1} + F_{n-2}$$

Where,

$$F = Fibonacci\ number$$
$$n = n^{th}\ term$$

With that in mind, once the seq module is imported, command the formation of the sequence using the seqf.generate function. This will prompt you to enter the number of digits you wish to see in the sequence. Once the value is input the sequence is generated.

```
>>>| import seq
#seq imported
>>>: seqf.generate

number of terms : 10

1
1
2
3
5
8
13
21
34
55
89
144
{non}

number of terms : /

>>>: /

>>>|
```

Programming your own module

It is very simple to program your own module and add it to Snick so that the whole world can access it with ease. In order to program your own module, write down a series of events in the python syntax. To upload, log on to: http://www.snick.uni.me and go to the contribute page or send it to sourcenet@outlook.com.

For further queries log on to http://www.snick.uni.me and visit the FAQs page