ரூபி நண்பன்

சிறுவர்கள் கணினி நிரலாக்கம் (புரோக்ராமிங்)
செய்ய உதவும் பாடநூல்

–ஆசிரியர்கள்–
டக் ரைட்
ஆடம் ஸ்டார்

–தமிழாக்கம்–
பல தமிழ் ஆர்வலர்கள்
எழில் மொழி அறக்கட்டளை

ரூபி நண்பன்

டக் ரைட், ஆடம் ஸ்டார்

தமிழாக்கம் - எழில் மொழி அறக்கட்டளை

மின்னூல் வெளியீடு :

FreeTamilEbooks.com

□□ மின்னூலாக்கம் - அ.சூர்யா -

suriya.alagar97@gmail.com

This Book was produced using LaTeX + Pandoc

பொருளடக்கம்

1. நிரலாக்கம் என்றால் என்ன ? 8

2. எண்கள் 17

3. சரங்கள் (Strings) 28

4. மாறிகள் (Variables) 38

5. ஆனால் இல்லை நிபந்தனை கட்டளை (If & Else) 48

6. மடக்கு வாக்கியம் 63

7. தரவமைப்புகள் (Collections) 79

8. நிரல்பாகங்கள் 100

9. எண்வகை தரவமைப்புகள் 120

10. கோப்புகள் 144

முடிவு 153

11. பின் இணைப்புகள் 155

Appendix 155

இ.1 பயிற்சி கேள்விகளின் விடைகள் 155

பாடம் 2 - எண்கள் 155

பாடம் 3 - சரங்கள் 158

பாடம் 4 - மாறிகள் 162

பாடம் 5 - If and Else 164

12. Numbers have object id’s that don’t change. 166

பாடம் 6 - Loops 167

பாடம் 7 - Collections 168

பாடம் 8 - Methods 175

பாடம் 9 - Enumerables 178

13. இ.2 தமிழ்க் கணிமைச் சொற்கள் 182

14. இ.3 தொடர்புகளுக்கு 186

15. இந்தப் புத்தகம் RubyKin.com என்ற ஆங்கிலப் புத்தகத்தின் தமிழ் மொழியாக்கம் 190

16. வாருங்கள், இந்தப் பயணத்தைத் தொடரலாம்! 192

17. நூல் வடிவமைப்பு 194

1. நிரலாக்கம் என்றால் என்ன ?

நாம் நண்பர்களுடன் உரையாட, பழக, பேச்சுமொழி தேவை. அதைப்போல, கணினியுடன் உரையாட, பழக நமக்கு என்ன தேவை ? என்ற கேள்விக்கான விடையே நிரலாக்கம். ஆணை என்பது பின்பற்றி நடக்க வேண்டிய விதி என்று பொருள். ஆணை என்பதைக் கட்டளை (‘command’) என்றும் சொல்லலாம். நிரல் என்பது கணினி இயங்கவேண்டிய ஆணைகளின் தொகுப்பு, அவ்வளவு தான்.

ஒரு வீட்டை எப்படிக் கட்ட வேண்டும் என்று சொல்லத் தேவைப்படும் மூலவரைபடத்தைப் (blueprint) போன்றது, இவ்வாணைகள். மூலவரைபடம் இன்றிக் கட்டிடத்தொழிலாளிக்கு, கட்டுமானி (கட்டிட வடிவமைப்பாளரின்(architect)யின் தேவையை அறிய இயலாது. நிரல் இல்லாமல், கணினிக்கு என்ன செய்வது

என்ற தெரியாது.

உங்கள் தேவைக்கேற்ப நிரலை உருவாக்கும் செயலே, நிரலாக்கம்.

நிரலாக்க உதவும் மொழியைக், கணினிமொழி எனலாம்.

நிரல்மொழியைக் கற்பது, மனிதர்கள் பேசும் மொழிகளைக் கற்பதைப் போன்றதுதான் . முதலில் அடிப்படைக் கூறுகளில் (அ, ஆ, இ..) இருந்து தொடங்குவோம். அக்கூறுகளைக் கொண்டு சொற்கள், சொற்றொடர்கள் மற்றும் கணினியைப் பணிக்க நாம் இட வேண்டிய கட்டளைகள் வரை படிப்படியாக நாம் எளிதில் கற்க முடியும்.

உங்கள் கணினியிடம் "ஒரு படம் எடு", இல்லை "ஞாயிறு மாலை 5 மணி அளவில் என்னை எழுப்பு" என்று சொன்னால் அதற்குப் புரியுமா ? நேராகச்

சொன்னால் புரியாது. காரணம் கணினியில் அதற்கு ஒரு தனிப்பட்ட மொழியே புரியும் - இதனை *"machine language"* இயந்திர மொழி என்று கூறலாம். சராசரியாக இப்படிக் கணினியிடம் பேசுவதற்கு இயந்திர மொழி என்பது பதில் கணினி நிரலாக்க மொழி மிக முக்கியமானது.

கணினி சற்று செல்ல நாய்க்குட்டி போல நடக்கும்; நல்ல தோழமை, நட்பு எல்லாம் அளிக்கும் ஆனால் அதனை ஒன்று செய்ய வைக்கவேண்டும் என்றால் படிப் படியாகச் சொல்லிக் கொடுக்க வேண்டும். அயல் மொழிகள் - கிரேக்கம், இசுப்பானியம், சீனம், பிரெஞ்சு, ஆங்கிலம் - போல கணினியிலும் நிரலாக்க மொழிகள் பல உள்ளன. இந்த மொழிகளில் உள்ள பொது அம்சங்கள் போலவே கட்டளை *(imperative)*, செயல்நோக்கு *(functional)* மற்றும் பொருள்நோக்கு *(object oriented)* கணினி மொழிகளிலும் பொது அம்சங்கள்

உள்ளன.

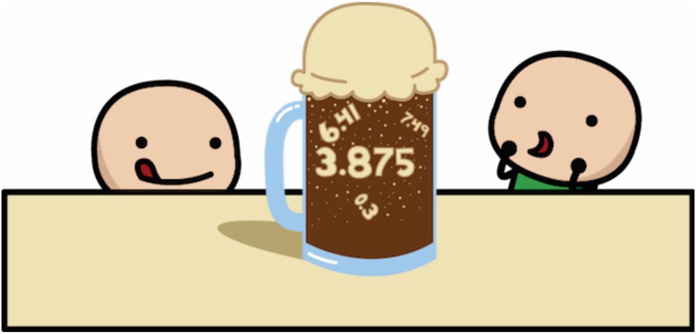
இந்தப் புத்தகம் உங்களுக்கு ரூபி கணினி மொழி கற்றுக்கொள்ள உதவும்; இந்தத் தேர்ச்சி வருங்காலத்தில் மற்ற கணினி மொழிகளைக்கற்கப் பயனளிக்கும்.

இந்தப் புத்தகத்தில் முதல் அத்யாயங்கள் பின்வரும் அத்தியாயங்களை தெளிவு படுத்தும் வகையில் அதிகமான சிக்கல் அளவில் அமைக்கப்பட்டுள்ளது. நாம் இப்போது எளிதான சில சிந்தனைகளுடன் தொடங்குவோம்.

முழுஎண், மெய்யெண், சரம்... போன்ற அடிப்படை வகைகள்

படம் 1.1.: சோடாவில் ஐசுகிரீம் - ஜிகிர்தண்டா போன்ற குளிபானமாக இருக்கும் நிரலாக்கம்

நிரல் எழுதுவதுக்கு முன் நம்ம தரவு மதிப்புகளை



பற்றி அறிந்து கொள்ளவது முக்கியம். ரூபி மொழியில் எண்கள் இரண்டு வகையில் அமையும்; *Integer* என்று சொல்லப்படும் முழு எண்கள் - தசம புள்ளி இல்லாமல் முழுமையான மதிப்பை குறிக்கும் எண்கள் ஒரு வகை. மற்றவை தசம எண்கள் என்ற *floating point numbers* - இவை முழுமை இல்லாத பகுதி எண்களாக மட்டுமே அமையும் எண்கள் - இதனை மெய்யெண் என்றும் சொல்லலாம்.

உதாரணத்திற்கு முழு எண் என்பது ரூபியில் இப்படி எழுதலாம்:

1, 7, 0, 13, 2000

உதாரணத்திற்கு தசம எண் என்பது ரூபியில் இப்படி எழுதலாம்:

1.2, 3.14, 5.12345, 0.35

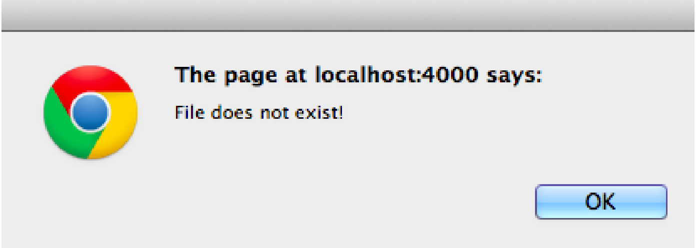
முழு எண் என்பது புள்ளி குறியீடை கொள்ளாமல் உருவெடுக்கும். தசம எண் அல்லது மெய் எண் என்பது புள்ளி குறியீடையும் எண்கள் கொண்டு அமையும். இதனை சும்மா பார்த்தாலே சொல்லிவிடலாம் - ஜுஜுபி.

சரம் - சொற்கள் தரவு

கணினியில் ஒரு கட்டுரை எழுதி, பதிவு செய்யும் பொழுது நாம் செய்யும் வேலை கணினியில்

சரம் உள்ளீடுகளை இடுவது. சரி சரம் என்றால் என்ன? "மேற்கோள் குறியீடுகளுக்கு உள்ளே இட்ட எழுத்துக்கள் எல்லாம் சரம்". கடைசி வாக்கியம் மேற்கோள் குறியீடுக்குள் உள்ளதால் இதுவும் சரம் என்று ஆகும்! நீங்கள் சரங்களை ஏற்கெனவே பார்த்திருப்பீர்கள்!

உதாரணத்திற்கு கீழே உள்ள எச்சரிக்கை செய்தியில் வருவது 'சரம்':


The page at localhost:4000 says:

File does not exist!

படம் 1.2: நிங்கள் தேடிய ஆவணம் (கோப்பு) கிடைக்கவில்லை

இந்த மாதிரி வெளியீடு வருவதற்கு யாரோ நிரலாளர் ஒருவர் *“File does not exist!”* என்ற சொற்களை எழுதி மேற்கோள் குறிக்குள் போட்டுச் சரம் என்று எழுதி இருப்பார். அதன்பின் உங்கள் செயலியில் தேவையான நேரத்தில் இந்தச் செயல்பாடு, சரம் உங்களுக்கு காட்ட பட்டது.

சரம் என்பது நிரலாக்கத்தில் ஒரு டிபன் கூடை, அடுக்கு டப்பா போல - தேவையானது எல்லாத்தையும் ஒரு பெரிய மூட்டையா கட்டி வைத்துக்கொள்ளலாம். சரம் என்பது கூடையைக் காட்டும் கயிருக்கு பதில் மேற்கோள் குறிகளை கொண்டது. இந்த மாதிரியான சரங்களில் நம்ம சொற்கள், தொடர்கள், அல்லது முழு நீள கோப்புகளையும் சேமித்து வைத்துக்கொள்ளலாம். இதோ நீங்களே பாருங்களேன்

“என்னையும் சேர்த்துக்கொள்ளுங்கள்!”

“அப்ப_என்னையும்_கூடத்தான்”

“9-இது எண் அல்ல சரம்”

“பெரிய பெரிய பத்தி. அதிலும் புரியாத

குறியீடு இதனை அப்பரம் விளக்குவோம்.”

இந்த அத்தியாயத்தின் முடிவில் உங்களுக்குக் கணினியில் உள்ள அடிப்படையான இரண்டு தரவு வகைகள் - எண்கள், சரம் பற்றி கற்று கொண்டீர்கள் என்று எண்ணுகிறேன். இப்போது இவற்றைப் பற்றி மேலோட்டமாக மட்டுமில்லாமல் சற்று ஆழமாகப் பார்க்கலாம்.

2. எண்கள்

உங்களுக்குக் கூட்டல், கழித்தல், பெருக்கல், வகுத்தல் எல்லாம் தெரியுமா ? சபாஷ், ரூபி மொழிக்கும் அதெல்லாம் தெரியும்!

2 + 2 => 4

9 - 3 => 6

*2 * 3 => 6*

4 / 2 => 2

செயல்பாடு = *Operation* ஏரணம் = *logic*

ரூபி கணித, ஏரணச் செயல்பாடுகளைச் செய்யும் ஆற்றல்பெற்றது. எ.கா.

-+ கூட்டல்

– கழித்தல்

-> பெரியதா?

-< சிறியதா?

4 > 2 => true (மெய்)

7 < 2 => false (பொய்)

3 >= 3 => true (மெய்)

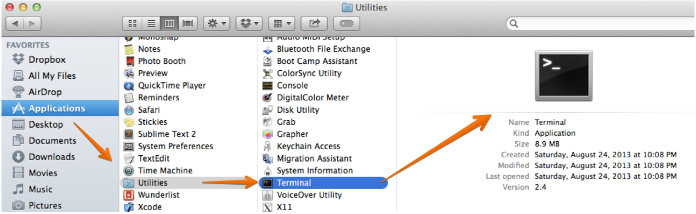
0 <= 1 => true (மெய்)

இக்கட்டளைகளை ரூபி இயக்கியில் நீங்கள் செயல்படுத்த, ரூபியை உங்கள் கணினியில் நிறுவ வேண்டும்.

ரூபியினை நிறுவுதல்

நீங்கள் OS X இணை பயன்படுத்துகிறீர்கள்

என்றால், அதில் ரூபி ஏற்கனவே நிறுவப்பட்டிருக்கும். இப்போது உங்களுடைய *Terminal application* திறக்கவும். இதனைக் கண்டுபிடிக்க திரையின் வலது மேல் மூலையில் உள்ள *magnifying glass*(பூதக் கண்ணாடி) பொத்தானை அழுத்தவும். பின் *Terminal* எனத் தட்டச்சு செய்து முதலாவது விளைவைத் தெரிவு செய்க. (இல்லையெனில் நீங்கள் *Applications - Utilities - Terminal* என்பதன் ஊடாகச் சென்று இரு தடவை *terminal* ல் அழுத்தித் திறக்க முடியும்.)



Name Terminal
Kind Application
Size 8.9 MB
Created Saturday, August 24, 2013 at 10:08 PM
Modified Saturday, August 24, 2013 at 10:08 PM
Last opened Saturday, August 24, 2013 at 10:08 PM
Version 2.4

படம் 2.1: *Terminal* (கட்டளைகளை இடும் முனையம்) என்பதை *Mac-OSX*-ல் திறக்க இதுவழி

நீங்கள் *Linux* பயனராக இருந்தால், ஒரு *shell* இனைத் திறந்து, *irb* எனத் தட்டச்சு செய்து *enter* இனை அழுத்தவும். மேலும் நீங்கள் *Windows* பயனர் என்றால், *Ruby* பகுதியிலுள்ள *Start Menu*இல் உள்ள *fxri* இனைத் திறக்கவும். மூன்றாம் தெரிவாக *Repl.it (http://repl.it/languages/Ruby)* என்பதனைத் தேடிப்பார்க்கவும். இது ஒரு அற்புதமான இலகுவான கருவி. இது தேடல் பொறிகளில் நிரலாக்கம் எழுத உங்களை அனுமதிக்கிறது. இங்கு நிறுவுதல் அவசியம் இல்லை.

அடுத்து *irb* என *terminal* அல்லது *shell* திரையில் தட்டச்சு செய்து *enter* இனை அழுத்தவும்.

IRB என்றால் என்ன?

IRB என்பது *Interactive Ruby Shell* என்பதன் சுருக்கம் ஆகும். *IRB* என்பது உங்கள் கணினியில் உள்ள

ஒரு சிறு இரகசிய கோட்டை போன்றது. நீங்கள் உங்கள் *IRB* சூழ்நிலைக்கவும்) கீழ் தரப்படமாதிரி சில எளிய கணகீடுகளை தட்டச்சு செய்து என்ன மாதிரியான வெளியீடுகளை கணணி தருகின்றது என்பதைப் பார்க்கவும். அல்லது சுயமாயைப் பயன்படுத்தி *Ruby* பற்றி அறிந்து கொள்ளவும் வெவ்வேறான கட்டளைகளைத் தெரிந்து கொள்ளவும் முடியும். *IRB* யைத் திறந்து *(shell window* அல்லது *www.repl.it/languages/Ruby* பாவித்து திறக முயற்சி செய்து பார்க்கவும்.

```
2 + 2

4 < 7

5 > 10

7 / 4
```

Arrows => பற்றி அறிய ஆர்வமாக உள்ளதா? *Ruby* பொறியியலாளர்கள் இதனை எண்ணிம அடைவு ராக்கட் என அழைக்க விரும்புவார்கள். *IRB*னுள் *3 + 2* என தட்டச்சு செய்தாலோ அல்லது வேறு வகையான காரணிகளை உள்ளீடு செய்தாலோ எப்போதும் அது சுட்டியால் => குறிப்பிடப்பட்ட ஒரு மதிப்பை விளைவாக தரும்.

/> சிறு துளி கணிதம்

ரூபி போன்ற நிரல்மொழிகள் பல மடங்கு பெரிய கணித கணக்குகளை இயக்கவல்லது. நமக்குச் சலிக்கும் ஆனால், கணினி சலிக்காமல், சளைக்காமல் வேலை செய்து விடை சொல்லும். இந்தப் பக்கத்தில் அடுத்து, வகுத்தல்மீதம் *(modulo)*, மற்றும் அடுக்குக்குறி *(exponent)* ஆகிய கணிதச் செயல்பாடுகளை ரூபி எவ்வாறு கையாள்கிறது என்று பார்ப்போம்!

அடுக்குக்குறி

அடுக்குக்குறி ஓர் எண் எவ்வளவு முறை பெருக்கப்பட வேண்டும் என்று கூறும். எ.கா. 2 பெருக்கல் 2 என்பது 4. 4 பெருக்கல் 2 என்பது 8. அதாவது 2ˆ3 என்றால் 2 பெருக்கல் 2 பெருக்கல் 2 , என்பதின் விடை 8

அடுக்குக்குறி பற்றி அறிந்தால்தான், ரூபியில் நிரலாக்கம் செய்யலாம் என்றில்லை. ரூபி இரண்டு பெருக்கல்குறியைக் கொண்டு அடுக்குக்குறியைக் குறியிடும்.

```
2 ** 3 => 8

3 ** 2 => 9

10 ** 3 => 1000

‘#’ 10 முறை 10 முறை 10!
```

‘#’ குறி, குறிப்புரை எழுத உதவும்

‘#’ (ரூபி இக்குறி உள்ள வரியைப் பொருட்படுத்தாது.)

வகுத்தல்மீதம்

இயல்பான கணிதச் செயல்பாடுகள் இல்லாமல், கணினியில் வகுத்தல்மீதம் என்ற ஒரு செயல்பாடு உண்டு. அச்செயல்பாட்டை % குறிக்கும்.

இச்செயல்பாடு ஓர் எண்ணை மற்றொரு எண்ணால் வகுக்கும் போது கிடைக்கும் மீதத்தைத் தரும். எ.கா.

-9 என்ற எண் மூன்றால் சரியாக வகுபடும், அதாவது மீதம் 0. அதனால் 9 % 3 என்பது 0.

-9 என்ற எண்ணை இரண்டால் வகுத்தால் அதாவது மீதம் 1. அதனால் 9 % 2 என்பது 1.

பின்வரும் எடுத்துக்காட்டுகளை முயலுங்கள்

8 % 2 => 0

9 % 2 => 1

9 % 5 => 4

வகுத்தல்மீதம் மற்றும் அடுக்குக்குறி சற்று சிக்கலாக இருந்தால் கவலைபட வேண்டாம். நிரலாக்கம் செய்ய இவை கட்டாயமா தேவையில்லை.

தற்போதைக்கு, கணினி உங்களுக்காகக் கணித செயல்பாடுகளைச் செய்யும் என்ற அடிப்படையைப் புரிந்தாலே, சிறப்பு.

பயிற்சி

கீழ்க் கண்ட பயிற்சி படங்களை நீங்கள் தாளில் அல்லது நினைவில் தீர்வு செய்யுங்கள். பின்பு Ruby shell

(IRB)-இல் உள்ளிட்டு தீர்வு செய்து பாருங்கள்.

1. *2 + 3 + 5*
2. *10 - 3*
3. *9 / 3*
4. *4 * 2*
5. *4 ** 2*

அடுத்த கட்டத்துப் பாடங்களைக் கீழே முயற்சி செய்யுங்கள்.

6. இதன் *11 % 5* விடை என்ன *?*
7. இதன் *14 % 3* விடை என்ன *?*
8. ஒரு மாயாவி அவன் இரு கையில் இரண்டு எண்களைக் கொண்டவன் - ஒரு எண்

ஒற்றைப்படை இரண்டாவது இரட்டைப் படை எண். இவன் உங்களுக்கு எந்தக் கையில் எந்த எண் உள்ளது என்று சொல்லமாட்டான் - *modulo* *'%'* செயலுருபு பயன்படுத்தி இதனை உங்களால் கண்டெடுக்க முடியுமா*?*

உதவிக் குறிப்பு: இரட்டித்த எண் என்றால் '% 2' வழி வகுத்தால் மீதம் பூஜ்யம்.

3. சரங்கள் (Strings)

வாழ்த்துக்கள் குழந்தைகளே!

எண், சரம் எனும் நிரலாக்கத்தின் அடிப்படைத் தொகுதிகளில் (blocks) பாதிக்குமேல் தெரிந்து கொண்டீர்கள்

சரம் என்றால் என்ன ?

சரங்கள் என்பவை மேற்கோள் குறிகளுக்கு(“) இடையே இருக்கும் தகவல் (சில சொற்கள்) என்பதை நினைவில் கொள்ளுங்கள். எ.கா.”தமிழ், ஒரு சிறந்த மொழி”

ரூபியில், நம் நண்பர்களாக ஆகிவிட்ட சரங்களின் மீது கணிதச் செயல்பாடுகளையும் செய்யலாம். உதாரணத்திற்கு, சரத்தை இவ்வாறு பெருக்கலாம்:

*“கபடி” * 3*

=> “கபடி கபடி கபடி”

மேல் குறிப்பிட்டதில், “கபடி” மும்முறை பெருக்கப்படுகிறது . ரூபி 3 என்பதைச் செயல்படுத்தும் போது, அது உண்மையில் அந்தச் சரத்தை மும்முறை ஒன்றாக இவ்வாறு சேர்க்கிறது:

“கபடி” + “கபடி” + “கபடி”

=> “கபடி கபடி கபடி”

*3 + 3 + 3 = 9 என்றும், 3 * 3 = 9 என்றும் இரு வகையில் எழுதினாலும் ஒரே விடை தான் வரும் அல்லவா? அது போலத்தான் சரங்களுக்கும். இவ்வாறு கணினி, சரங்களை ஒன்றாக இணைப்பதை, ஒன்றிணைத்தல் (அ) ஒன்றிணைப்பு எனலாம்.*

“நாய் மற்றும்” + “பூனை”

=> “நாய் மற்றும் பூனை”

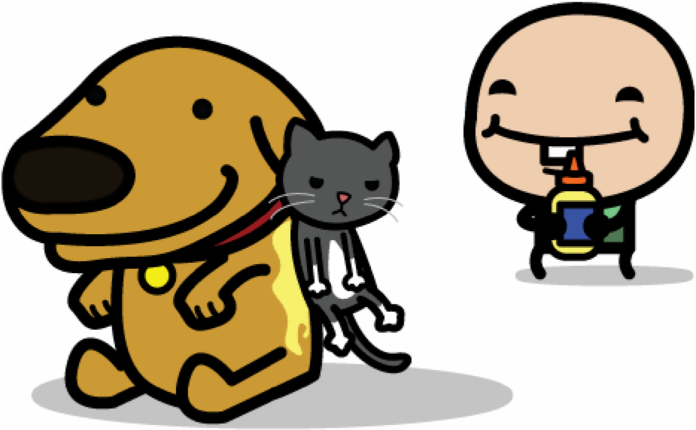
மேல் குறிப்பிட்டுள்ள உதாரணத்தில் இரு சரங்கள் உள்ளன; முதலாவது “நாய் மற்றும்”, இரண்டாவது “பூனை”. அவ்விரு சரங்களையும் + என்ற மந்திரக்கட்டளை இட்டால் அவற்றை இணைக்க முடியும்.

ஒன்றிணைப்பு என்பதற்கு ஆங்கிலத்தில் இணையான சொல் தான், concatenation.

முயல்வோமா?

மீண்டும் முனையத்தை (terminal) திறந்து (அல்லது http://repl.it/languages/Ruby) சில சரங்களைக் கூட்டி, பெருக்கிப் பாருங்கள்.

கூட்டி, பெருக்கிப் பார்த்தாச்சா ? சரி, இப்போது ஓர் எண்ணையும், ஒரு சரத்தையும் கூட்டிப் பார்க்கலாமா,



படம் 3.1: செல்லப் பிராணிகள் உங்கள் வீட்டில் யாவை ?

அது வேலை செய்யாதே ஏன் தெரியுமா?

முந்தைய பாடம் ஒன்றில் நாம் பார்த்த கோமாளி சரம் “9” நினைவில் இருக்கிறதா? அதை ரூபி எண் 9 எனக் கொள்ளாது. மாறாக, மேற்கோள் குறிகளுக்குள் இருப்பதால் அதைச் சரம் “9” என்றே எடுத்துக் கொள்ளும்.

ரூபியைப் பொறுத்தமட்டில், மேற்கோள் குறிகளுக்குள் இருக்கும் அனைத்தும் சரங்கள் மட்டுமே; சொல்லோ, எண்ணோ அல்ல. எனவே, ஒரு சரத்தை ஓர் எண்ணோடு கூட்ட முற்பட்டால், ரூபி அது பிழை என்று நமக்குப் பதிலளிக்கும்:

```
“9” + 9

=> TypeError: can’t convert Fixnum into String
```

(இதில் String என்பது சரத்தைக் குறிக்கும்.)

மேல் குறிப்பிட்டுள்ள உதாரணத்தில், *"9" + 9* என்பதற்கான விடை *18* என நாம் கருத வாய்ப்பு உண்டு; ஆனால் ரூபி அவ்வாறு கருதாது. ரூபி அதைச் "சரம்" + எண் எனத் தான் கருதும். நாம் ஒருபோதும் சரங்களையும், எண்களையும் ஒன்றாகக் கூட்ட முடியாது. ஏனெனில், அவை இரு வேறு தரவு வகைகள்(*data type*) ஆகும்.

ரூபியில் இதுபோன்ற சமன்பாடுகளை நாம் இயக்கவே முடியாது என்பது பொருள் அன்று. அதன் பொருள் என்னவென்றால், ரூபி நாம் சொல்வதைப் புரிந்து கொள்ளும் விதமாக நாம் சில தந்திரங்களைச் செய்ய வேண்டும் என்பதாகும். அதற்கெனச் சில சிறப்புச் செயற்கூறுகள் *(methods)* அல்லது செயல்கள் *(actions)* ரூபி மொழியில் உள்ளன.

அவ்வாறான சில செயல்களைப் பின்னர் பார்க்கலாம். ஆனால், தற்போது, *to integer* அல்லது

*to_i*என்பதயைும் பயன்படுத்தி இந்தச் சிக்கலைத் தீர்க்கலாம்.

இந்தச் செயற்கூறு சரத்தை எண்ணாக மாற்றிவிடும் (குறிப்பாக முழு எண்ணாக). பிறகு, அவ்விரு எண்களைக் கொண்டு ரூபியால் கணிதச் சமன்பாடுகளை இயக்க முடியும்:

‘

```
"9".to_i + 9

        => 18
```

முதல் பாடத்தில் நாம் படித்த நினைவில் உள்ளதா? இதை உங்கள் சரத்தில் சேர்த்துக் கொண்டால், சேர்த்த இடத்தில் ஒரு புது வரியை உள்ளிட வேண்டும் என ரூபி புரிந்து கொள்ளும். அப்படியானால், பின்வரும் சரம்:

```
print "முதல் வரி. மற்றொரு வரி. மற்றுமொரு வரி."
```

இவ்வாறாகப் பதிக்கப்படும்:

முதல் வரி.

மற்றொரு வரி.

மற்றுமொரு வரி.

இவ்வழியைப் பயன்படுத்தி, ரூபியால், ஒரேயொரு சரத்தில் ஒரு வாக்கியத்தையோ, ஒரு முழுக் கோப்பையோ *(file)* கூடச் சேமித்துவிட முடியும். தற்போது உங்களுக்குச் சரங்கள் பற்றி ஓரளவு தெரிந்துவிட்டபடியால், பின்வரும் எடுத்துக் காட்டுகளை முயற்சி செய்து பாருங்கள்.

பயிற்சி

கீழ் உள்ள நிரல் துண்டுகளின் வெளியீடு என்ன ? (இவற்றை IRB Ruby இல் இயக்கி பார்க்கவும்)

1. *puts “இந்த நிரல் துண்டின்” + ” வெளியீடு ” + “என்ன?”*

2. *“இந்தச் சரம் கழித்தல்” - “இந்தச் சரம்”*

3. *“1234.55”.to_i*

4. *“1234.55”.to_f*

5. *“Not a number”.to_i*

6. *puts “1* `\n` *2* `\n`*3* `\n`*”*

7. *இந்த மாதிரியான செயல்பாட்டு முறை to_i (to integer) எண் வடிவில் மாற்ற முடியாத சரங்களுக்கு பூஜ்யம் வெளியீடு தருவது, ஏன், எங்கு உதவியாக இருக்கும் ?*

8. *நீளம் என்கிற ‘length’ நிரல்பாகம் சார்பு என்ன செய்யும் என்று எண்ணுகிறீர்கள் ? “Count”. length*

9. *‘split’* செயல்பாடு என்ன செய்யும்? *“Count”.split(“”)*

10. *‘slice’* செயல்பாடு என்ன செய்யும்? *“Count”.slice(2)*

சரம் சார்ந்த மாற்ற செயல்பாடுகள் எவை ? *methods* என்ற செயல்பாட்டைச் சரம் வகுப்பில் இயக்குங்கள்! *String.methods* என்று தடச்ச்சு செய்யுங்கள் *IRB:*

IRB$ String.methods # some of the built-in Ruby methods for String

*[:try_convert, :allocate, :new, :superclass, :freeze, :===, :==, :<=>, :<, :<=, :>, :>=, :to_s, :included_modules, :include?, :name, :ancestors, :instance_methods, :public_instance_methods, :protected_instance_methods, :private_instance_methods, :constants, :const_get, :const_*

இவற்றோடு இன்னும் பல………

4. மாறிகள் (Variables)

எளிய நோக்கில், மாறிகள், தகவலைச் சேமிக்கும் கலன்கள் / பாத்திரங்கள் (containers) என்று சொல்லலாம்.

நீங்களும் மாறிகளை உருவாக்கத் தேவையான பொருட்கள்,

1. கீழ்வரிசை ஆங்கில எழுத்துக்கள்
2. சமக்குறி
3. மதிப்பு

பெரும்பாலும் மதிப்பு, ஓர் எண்ணாகவோ, சரமாகவோ, சற்று சிக்கலான குறியாகவும் இருக்கலாம். நீங்க கணக்குப் பாடத்தில் பார்த்திருப்பீங்க.

$x = 12$

x –   மதிப்பு என்ன?

=> 12

சமக்குறிக்கு இடப்புறமாக எதை எழுதினாலும், அது ஒரு மாறியின் பெயரை வரையறுக்கும். சமக்குறி வலப்புறமாக இருக்கும் மதிப்பை மாறிக்கு அளிக்கும். மாறிகளை வரையறுக்கும் சில எடுத்துக்காட்டுக்கள்.

myVar = "சரத்தின் மதிப்பு"

a_long_var_name = 42

myCat = "முறுக்கு மீசை"

தேக்குதல் / சேமித்தல் / சேர்த்தல்

1. தண்ணீரை, ஏரி *(lake)* குளம் *(pond)*, கண்மாய்ப் போன்ற நீர்நிலைகளில் தேக்குவதன் *(store)* மூலம், வேளாண்மைக்குப் பயன்படுத்தலாம்.

2. *பணத்தைச் சேமித்து வைக்கும் பழக்கம் நல்லது.*

மாறியும் தேக்கமும்

நினைவகத்திற்கும் (memory) மாறிக்கும் என்ன தொடர்பு? என்று புரிந்துகொள்வது சற்றுக் கடினமாகத் தெரியலாம், எடுத்தவுடனே புரியவில்லை என்றாலும் நிரல்களை எழுத எழுத புரியும்.

எதையாவது தேக்க (அ) சேமிக்க நமக்கு இடம் தேவை, அப்படித்தான் கணினிக்கும் **மதிப்பு***களை இடம் தேவை. கணினி, மதிப்புகளைச் சேமிக்கத் தேவைப்படும் இடங்களை நினைவகத்தில் உருவாக்கும்.*

அவ்விடங்கள் வீணாக்காமல் இருக்க, கணினி தகவல்களின் போலிநகல்களைத் தேக்காது. இடம் என்று ஒன்று இருந்தால், அதற்கு முகவரி இருக்கும். இது நினைவகத்தில் இருக்கும் இடத்திற்கும் பொருந்தும்.

x = 12

மேலுள்ள எ.கா வில் *12* என்ற மதிப்பைத் தேக்குகிற இடத்திற்கு முகவரி *ABC12* என்று வைத்துக்கொள்வோம். அப்போ இந்த *x* என்னதான் செய்கிறது? இது ஒரு நல்லகேள்வி ?

பொதுவாக, மாறிகள் மதிப்புகளைத் தேக்காது, அவை தேங்குகிற இடத்தைச் சுட்டும்.

மதிப்பை தேக்குகிற முகவரி, மாறி தேக்கும்.

பின்வரும் எ.கா *(X, Y, Z)* ஆகிய மூன்று மாறிகளும் அதே இடத்தைப் பயன்படுத்தும், ஏன் தெரியுமா ? அவை அனைத்தும் *12* என்ற மதிப்பை சேமிப்பதால்.

x = 12 => நினைவக முகவரி: *ABC12*

y = x => நினைவக முகவரி: *ABC12*

z = 12 => நினைவக முகவரி: ABC12

இப்போது உங்களுக்குப் புரிந்து இருக்கும், மாறிகளை எப்படித்தான் மதிப்பை தேக்குகிறது என்று. கீழுள்ள கட்டளைகளின் இறுதியில் *x* என்ற மாறி சுட்டும் மதிப்பு என்ன? கண்டுபிடிங்க பார்க்கலாம்.

y = 5

'#' y நினைவக-முகவரி AB1யைச் சுட்டும்,

'#' AB1 என்ற நினைவக-முகவரியில், 5 என்ற மதிப்பு தேக்கப்பட்டு / சேமிக்கப்பட்டு இருக்கிறது.

x =' y

'#' இப்போது x நினைவக-முகவரி AB1யைச் சுட்டும்

y = 7

'#' y நினைவக-முகவரி CD1யைச் சுட்டும்,

‘#’ CD1-யில், 7 என்ற மதிப்பு தேக்கப்பட்டு / சேமிக்கப்பட்டு இருக்கிறது

x யின் மதிப்பு என்ன?

X, Y இரண்டும் சமம் இல்லை என்பதைப் புரிந்துகொள்ளுதல் முக்கியம்.

X = Y என்ற கட்டளை இயக்கப்படும் போது, Y தான் சுட்டும் AB1 என்ற நினைவக முகவரியைத்தான் Xக்கு அளிக்கும், அதில் இருக்கும் மதிப்பை அல்ல.

பின்னர் Y எதனைச் சுட்டினாலும், X தான் சுட்டும் AB1யை மறக்காது.

Y = 7 என்ற கட்டளை இயக்கப்படும் போது, நாம் கணினிக்கு புதிய நினைவக முகவரியை உருவாக்கி, அம்முகவரியை Y சுட்ட செய்யக் கூறுகிறோம். இதனால் X சுட்டும் முகவரி மாறாது, அதனால் 5

என்ற மதிப்பையே பெரும்.

object_id என்ற ரூபியின் உள்ளார்ந்த செயற்கூறு, பொருளின் அடையாளயெண்ணைத் தரும். இவ்வெண் நாம் கிட்டத்தட்ட தனித்துவம் பெற்ற நினைவகமுகவரியைப் போன்றது.

object_id செயற்கூற்றை வைத்துக்கொண்டு சில எடுத்துக்காட்டுகளைப் பார்க்கலாம் வாங்க,

*y = "***வணக்கம்***" => y.object_id => 7021*

x = y => x.object_id => 7021

இதுவரை, "வணக்கம்" என்ற ஒரு மதிப்பைத் தேக்க ஒரேயொரு நினைவக இடம் மட்டுமே ஒதுக்கப்பட்டுள்ளது, *X* மற்றும் *Y* அவ்விடத்தைப் பகிர்ந்துகொள்ளும்.

*y = "***வாழ்க***" => y.object_id => 8333* *x* மதிப்பு என்ன? => *x.object_id* => 7021

*Y*க்கு "வாழ்க" என்ற புதிய மதிப்பை அளிக்க, நாம் புதிய நினைவக இடத்தை ஒதுக்க வேண்டும். தற்போது *X* மதிப்பு "வாழ்க" என்று இருக்காது, அது இன்னமும் "வணக்கம்" என்றே இருக்கும்.

"வணக்கம்" என்ற மதிப்பை அளிக்கும்போது எந்த நினைவகஇடத்தைக் கொடுத்தோமோ அதையேதான் *X* இன்னமும் சுட்டும்.

நாம் மாறிகளைப், புதுத் தகவல்களை மாற்றும் போதோ , *Ruby* தற்போது ஒதுக்கியுள்ள இடங்களில் நாம் மாற்றும் மதிப்பு இல்லை என்றால் மட்டுமே புது இடங்களை ஒதுக்கும்.

சற்று குழப்பமாக இருப்பது மாதிரி இருக்கும், கவலைப்பட வேண்டாம்!

நிரலாக்க செய்ய நினைவகஇடம், ஒதுக்கீடு, முகவரி பற்றி முழுவதும் தெரிந்து வேண்டும் என்ற தேவை இல்லை.

மற்ற முக்கிய நிரலாக்கக் கருத்துருக்களைக் கற்கும் போது, உங்களுக்கு மேலும் புரியும்.

தற்போதைக்குப் புரிந்து கொள்ள வேண்டியது

= குறிக்கு பொருள் என்ன?

= சமக்குறி, வலப்புறம் இருக்கும் மதிப்பை இடப்புறம் இருக்கும் பெயருக்குத் தரும்

number12 = 12

(variable name) (அளிக்கும்) எண் 12

number12 = 12

பயிற்சி

1. *‘54 / 3’* மதிப்பை *‘x’* என்ற மாறிலியில் சேமித்தால் *‘x’*-இன் மதிப்பு என்ன?

2. கேள்வி *1*-இல் உள்ள *‘x’* மாறிலியின் மதிப்பை *‘y’* என்பதற்கு அளித்து ஒரு புது *‘y’* மாறிலி உருவாக்குக. பின்பு *‘x’* என்பதை *3*-ஆல் வகுத்து தனக்கே ஈடு செய்க. தற்போது (இவற்றை எல்லாம் செய்த பின்) *‘x’*, *‘y’* இவற்றின் மதிப்பு என்ன ?

3. சில கணித கணக்குகளை செய்து பார்க்கலாம்; *‘x’*-ஐ *12* என்ற மதிப்பிற்கு இடுக. அதன்பின் *‘x’ 3*-ஆல் வெகு - இதன்பின் *‘x’*-இன் மதிப்பு என்ன?

4. தற்போது *‘x’*-இன் மதிப்பு என்ன ?

5. ஆனால் இல்லை நிபந்தனை கட்டளை

(If & Else)

ஒருநாள் நீங்கள் படுக்கையில் இருந்து சனிக்கிழமை எழுந்திருக்கிறீங்க:

“அறை சுத்தம் ஆனால் நீ விளையாட செல்லலாம், இல்லை வீட்டில் உள்ளேயே தான் இருக்க வேண்டும்” - அம்மா

என்ற ஒரு தாளில் எழுத்தி நிபந்தனையாக அம்மா அறை கதவில் ஒட்டினாள்.

இதில் இரு சொறக்கள் ஆனால் , இல்லை வாக்கியங்கள் ரூபியிடம் உங்களுக்கு என்ன திசையில் நிரல் இயக்கம் தொடரவேண்டும் என்று சொல்லியும், ரூபியின் இயக்கத்தையும் கண்காணிக்கும்.

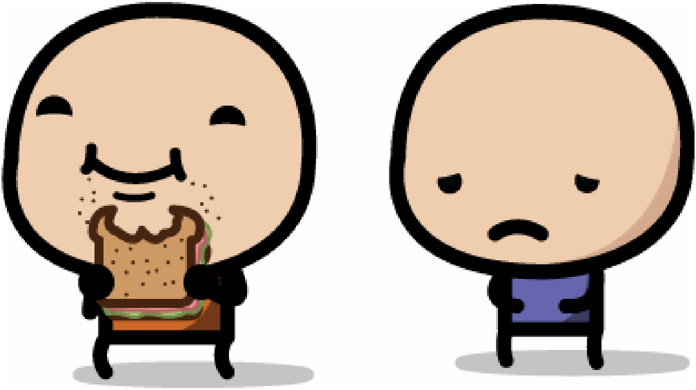
இதுவரை, ரூபியில் எப்படி எண்களைப் பயன்படுத்திக் கணிதம் செய்வது, சொற்களையும் வாக்கியங்களையும் சரங்களில் சேமிப்பது என்றும், இவ்விரண்டையும் மாறியில் கணினியின் நினைவில் சேமிப்பது என்றும் படித்து புரிந்துகொண்டோம். இதன் வழி நாம் கணினியில் தகவல்களை எண்களாகவும், சரம் என்றும் குறியீடு செய்ய கற்றுக்கொண்டோம் - ஆனால் இதனை வைத்து எப்படிக் கணினியில் இயங்கும் திசையை நிர்ணயிப்பது என்று நமக்கு இயலவில்லை.

இயங்கும் திசையை நிர்ணயிக்க ஒரு வழி நிபந்தனைகளுடன் நிரல்கள் எழுதுவது. நிபந்தனைகளுடன் இயங்கும் நிரல் என்றால் என்ன? இதனைப் புரிந்து கொள்ள இந்த நிலையை யோசியுங்கள்; உங்கள் பெற்றோர் உங்களைத் தடையில்லாமல், அளவில்லாமல் அன்பு செலுத்துகிறார்கள் அல்லவா - அதாவது ஐம்பது

சதவிகிதம் இல்லை முழு நூறு சதவிகிதம் கேள்விகள் இல்லாமல் அன்பு செலுத்துகிறார்கள் - இதனுடைய எதிர்மறையானது நிபந்தனை உடன் இருக்கும் அன்பு - “செல்லம் நீ கணக்கு பரிட்சையில் 100/100 வாங்கினால் மட்டும் உனக்கு இந்த ஆண்டு பிறந்தநாள் விழா”. இந்த மட்டற்ற அன்பு, கேள்விகள் இல்லாத தூய தாயின் அன்பு கணினியிடம் கிடையாது - ரூபி இயங்குவாரத்திற்கு நிபந்தனைகளுடன் ரூபியை சமாளிக்க வேண்டும்.

சரி, நிபந்தனைகளை எப்படிப் பயன்படுத்துவது என்று பார்ப்போம். நிபந்தனைகளுக்கு வெளி மாறி நிலையை ஒட்டித் தீர்வு அமையும். உதாரணம், இங்கு ஏதோ ஒன்று நடந்தது/மாறியது, ஆனால் ஒரு குறிப்பிட்ட காரியம் செய்யவும், இல்லை, மற்றொரு காரியம் செய்யவும். உதாரணத்திற்கு: உங்களுக்குப் பசித்தால், “இட்லியைச் சாப்பிடுங்க” , இல்லையா,

“இட்லியைச் சாப்பிடாதீங்க”!



படம் 5.1: ஆகா சாண்டுவிச் போச்சே! வருத்தம் வேண்டாம் - மேலும் ரூபியில் பயிலுங்கள்

ரூபியில் நிபந்தனை வாக்கியம் கீழ்க் கண்டது போலவே அமையும்:

‘#’ x மாறியை 5 என்ற மதிப்பிற்கு சமப் படுத்தவும்

x = 5

‘#’ நிபந்தனையைத் தொடங்க *‘if’, ‘else’, ‘end’* குறிச் சொல்லைப் பயன்படுத்தவும்

```
if x <= 10
puts x
else
puts “எண் 10-க்குக் குறைந்ததாக இருக்க வேண்டும்”
end
```

நிபந்தனை கட்டளை என்றால், கணினியில் ரூபி இயக்கி *“if”* என்ற குறிசொல்லுக்கு அடுத்து வரும் துணுக்கு மெய் *(true)* என்கிற மதிப்புடையதா என்று பரிசோதிக்கும். இந்தப் பகுதி “நிரல் துண்டு” என்று அழைக்கப்படும். மேலே உள்ள எடுத்துக்காட்டில் நமக்கு “நிரல் துண்டு” இங்கே *X <= 10.*

தற்போது நமது நிரல் துண்டு உண்மை (மெய், true) என்று ஆகியிருந்தால் (அதாவது 'x' மாறி 10-இன் கீழ் இருந்தால்), கணினி நிபந்தனை வாக்கியத்தின் மெய் அளவில் உள்ள அடுத்த கட்டளைக்குச் சென்றிருக்கும் - அதாவது மாறியின் மதிப்பான 5-ஐ திரையில் "puts" வழி அச்சிடும். ஆனால் மாறி 'x' 11 என மதிப்பு கொண்டிருந்தால் நிபந்தனை "நிரல் துண்டு" பொய் எனத் தீர்மானிக்கப்பட்டு நிபந்தனை வாக்கியத்தின் பொய் பகுதிக்குச் சென்று "எண் 10-க்குக் குறைந்ததாக இருக்க வேண்டும்" என்ற செய்தியைத் திரையில் இடும். இதுவே நிபந்தனை வாக்கியத்தின் செயல்பாடு.

உங்களது ரூபி மொழி நிரலில் பல "நிரல் துண்டு" நிபந்தனைகளைக் கோத்து வரிசையாக நிரல் எழுதலாம் - இதுவும் முதல் "நிரல் துண்டு" உண்மை என தீர்மானிக்கப் படும் வரை வரிசையில் பரிசோதிக்கப்பட்டு வரும்; கடைசியாக எது முதலில்

தீர்மானிக்கப் படுகிறதோ அத்துடன் இயக்கம் முடியும் - பின்வரும் நிபந்தனை வாக்கியத்தின் அமைப்புகள் இயங்கமாட்டாது.

இரும மதிப்பு வகை (Boolean) என்றால் என்ன?

கடந்த பகுதியில் ஒரு மாறியின் மதிப்பை மெய் *true* (மெய்) அல்லது *false* (பொய்) என்று தீர்வு காண்பதும் அதனை கொண்டு நிபந்தனை வாக்கியம் எழுதுவதையும் பற்றி கண்டோம். இப்படி மெய் அல்லது பொய் என்று இரண்டே மதிப்புகளை கொண்ட மாறி 'இரும மதிப்பு' (Boolean) வகையைச் சேர்ந்தது [Boolean என்று பெயர் வர காரணம் கணித மேதை ஜார்ஜு பூள் அவரது கண்டுபிடிப்பைப் போற்றும் வகையில் இதை Boolean என அழைக்கிறோம்] என்று சொல்லலாம். இது போன்ற இரும மதிப்புகளைக் கொண்டு நாம் நிபந்தனைகளைக் கீழே உள்ளபடி எழுதலாம்:

```
if false

puts “பொய்” #பதிவிடாது

elsif true

puts “மெய்” #பதிவிடும்

else

puts “பதிவிடாது” #பதிவிடாது!

end
```

மேலே உள்ள நிரலைப் படிப்படியாக மனத்தில் இயக்கி பார்க்கலாம் - இந்த வழிமுறை பின்னாட்களில் *“debugging”* என்று சொல்லும் “வழு நீக்குதல்”-க்கு உதவியளிக்கும். *Nil* பூஜியம் மற்றும் (பொய்) *False*மதிப்புகள் மட்டுமே ரூபி மொழியில் ‘பொய்’ என்ற தர்க மதிப்பு பெறும் - இவைகள் மட்டுமே பொய்

என்று குறிக்க நாம் பயன்படுத்தலாம். *Nil* என்பது மண்ணாங்கட்டி ஒண்ணுமில்லாததைக் குறிக்கும் - அதாவது அங்கு ஒண்ணுமே கிடையாது - பூஜியம் - சுழி!

மேலே உள்ள உதாரண நிரலில் நிபந்தனை *"நிரல் துண்டு"* மதிப்பு *(பொய் என்ற உள்ளது)* இங்கு மெய் என்று மதிப்பிடுமா என்று பரிசோதிக்கும். அது அப்படி செய்ய, "பொய்" என்று மதிப்பு எப்போதுமே மெய் என்பதற்கு ஈடாகாது என்பதனால், கணினி அடுத்த நிபந்தனை வாக்கியத்தின் பகுதிக்கு செல்லும் - இங்கோ துண்டு *"true"* என்பதன் மதிப்பு மெய் ஆகவும் அமைய, அது "மெய்" என்று திரையில் அச்சிடும் *"puts"* செயல்பாட்டை இயக்குகிறது.

கடைசி *"else"* கட்டளை இயக்காமலே நிரல் முடிவடையும் - இது ஏன் என்றால் ரூபி ஒரு நிபந்தனையில் முதல் மெய்யான .

ரூபி நிபந்தனைகளை மதிப்பிடப் பயன்படுத்தும் தொடர்புறு செயற்குறிகள் *(relational operators):*

< # குறைவு

கூடுதல்

<= # சமம் அல்லது குறைவு

= # கூடுதல் அல்லது சமம்

== # சமம்

!= # சமம் இல்லை

மேல் கண்ட ரூபி செயலுருபுகள் கணித குறியீடுகளாகவே இருக்கும். இரு உறுபடிகள் சமமா என்று கணக்கிட "==" இந்தக் குறியீட்டைப் பயன்படுத்துவோம் - இது நீங்கள் வகுப்பில் படிக்கும் கணிதத்தைப் போல இல்லாமல் சற்று

மாறுபட்டு அமைந்தது. ரூபியில் "=" சமம் என்று குறியீடு ஒரு மதிப்பை ஒரு குறிப்பிட்ட மாறிக்கு அளிக்கும் வகை அமைந்தது. இரு உறுபடிகளும் சமம் இல்லாமல் இருப்பதை கண்டறிய "!=" என்ற குறியீடை பயன்படுத்தலாம். ரூபியில் "இல்லை" என்ற சொல் *not* என்பதையும் "தர்க எதிர்மறை" *(logical not)* என்பதற்கு பயன்படுத்தலாம்.

நிரலக்த்தில் "பொருள்" என்றால் என்ன?

நிரலகத்தில் "பொருள்" என்றால் என்ன என்பதை அறிந்து கொள்ள நிஜ வாழ்க்கையில் "பொருள்" என்றால் என்ன என்பதையும் பார்த்தால் புரிந்து கொள்ள உதவியாக இருக்கும். உங்கள் விளையாட்டுப் பொருட்கள், செல்ல நாய்க்குட்டி, பந்து, கிரிக்கெட் பேட் – இவை எல்லாமே நிஜ வாழ்க்கையில் பொருட்கள் தான். இந்த மாதிரியான நிஜ வாழ்க்கைப் பொருட்களின் தன்மைகளை கணினியில் குறியீடு செய்ய நம்ம

பயன்படுத்தப் போவது நிரலாக்க “பொருள்” (*Object* என்று இதனை ஆங்கிலத்தில் கூறுவர்; இதுவே *“object oriented programming”* என்ற பொருள் நோக்கு நிரலாக்கத்திற்கு அடிப்படையாக அமைந்தது.

உதாரணத்திற்கு உங்கள் தெருவில் வாழ்பவர்களின் செல்ல நாய்க்குட்டிகளை தினமும் நடக்க நீங்கள் கூட்டிச் செல்வீர்கள் என்று எடுத்து கொள்வோம். இவற்றின் பெயர்கள் : ராமு, சோமு, முத்து.

ஒருநாள் நீங்கள் இந்த ரூபி புத்தகத்தைப் படித்துவிட்டு, “சரி நாய்க்குட்டிகளின் பெயரை நம்ம கணினியில் விடுவோமா?” என்று சிந்தித்தால் எப்படி இதனைச் சாதிப்பது ? ஏற்கெனவே படித்த “அணி” என்ற தரவமைப்பில் “சரம்” என்று மூன்று பெயர்கள் இடலாம் :

[“ராமு”, “சோமு”, “முத்து”]

மேல் உள்ள நிரல் துண்டில் மூன்று சரங்களும் பொருள் என்று ரூபிக்குள் தோற்றமளிக்கும்; அடுத்து இந்த "அணி" என்பதும் நான்காவது பொருள் என்றும் தோன்றும். ஆம் - ரூபியில் எங்கும் பொருள் உள்ளது (அணி), எதிலும் பொருள் உள்ளது (சரம்).

ரூபியில் எல்லாம் பொருள் மயம்

இது முக்கிய பாடம் - இதுவரை புத்தகத்தில் நீங்கள் படித்த ரூபியின் அம்சங்கள் அனைத்தும் பொருள் என்று அமையும் - இவை எண்கள், சரம், மாறிகள், நிபந்தனைகள், இரும தர்க்க வகைகள், ஏன் நிபந்தனை வாக்கியங்கள் கூட ...எல்லாமே "பொருள்" தான்!

அடுத்த அத்தியாயத்தில் மடக்கு (loop) வாக்கியங்கள் பற்றிக் கற்போம் (அதுவும் "பொருள்" தான்!). நீங்கள் இருக்கைப் பிடிப்பைப் (seat belt)

பிடித்துக்கொள்ள வேண்டாம் - இது இராட்டினம் போன்ற வயிற்றைக் கலக்கும் நடுவனத்தில் மடிப்புகள் அல்ல - வெறும் மடக்குக் கட்டளை தான்.

பயிற்சி

கணினி (ஒப்பிடுதல் செயற்குறி தொடரில்) இதை எப்படி மதிப்பிடும் ? IRB-இல் இயக்கம் முன் உங்கள் மனதில் இவற்றை செய்து பாருங்கள். நினைவிருக்கட்டும், விடைகள் இரும 'பூலின்' மதிப்பில் இருக்கும்.

1. *3 > 5 உண்மையா?*
2. *3 < 5 உண்மையா? பற்றி உள்ள மற்ற கேள்விகள்*
3. *5 == 5 உண்மையா?*
4. *10 >= 10 உண்மையா?*

5. *10 <= 12* உண்மையா?

6. *10 != 10* உண்மையா ?

7. *10.object_id == 10.object_id*

சரம் பற்றி உள்ள மற்ற கேள்விகள் முயற்சி செய்யுங்கள் ?

8. *"dog" == "cat"* உண்மையா?

9. *"cat" == "cat"* உண்மையா?

10. *"cat.object_id" == "cat.object_id"* உண்மையா ?

6. மடக்கு வாக்கியம்

அடிப்படையில் கணினியைப் பலமுறை ஒரு குறிப்பிட்ட சில கட்டளைகளைச் செய்ய முனைவது மடக்கு வாக்கியம். கணினி மனிதர்களைப் போல் இன்பம், வலி, பசி போன்ற உணர்வுகளுக்கு ஆளாகாததால் இந்த இயந்திரம் சளைக்காமல் வேலைகளைச் *(கணக்குகளை)* செய்யும் - அதற்கு "போர் *(boredom)*" அலுப்படையாமல் வேலைகளைச் செய்யும் திறன் கொண்டது.

ஒரு மடக்கு வாக்கியம் *(loop* "லூப்" என்று ஆங்கிலத்தில் சொல்வார்கள்) என்பது ஒரே செயலை (அல்லது சில குறிப்பிட்ட செயல்களை) பலமுறை திரும்பத் திரும்பச் செய்யும் பொருள் கொண்டது; கணினி ஒரு மின் இயந்திரம் என்பதால் கடகட என்று நொடிக்கு பல மில்லியன் *"flops" (floating-point operations* *per second)* என்ற வேகத்தில் நீங்கள் இட்ட மடக்கு வாக்கியம் சூழ்ந்த கட்டளைகளைக் கண்சிமிட்டும் நேரத்தில் இயக்கி முடிக்கும். கீழ் வரும் நிரல் துண்டைப் படித்து அதில் உள்ள *“while”* மடக்கு வாக்கியத்தை ஆராய்ந்து பார்க்கலாம் வாருங்கள்.

```
x = 0
while x < 5
puts x
x = x + 1
end
puts “மடக்கு கட்டளை முடிந்தது”
```

மடக்கு வாக்கியத்தை ரூபி மொழியில் இந்த எடுத்துக்காட்டு விளக்கியது. *while* என்ற குறிச்சொல்

அடுத்து வரும் மாறி *'true'* (உண்மை) என்றவரை மட்டுமே இந்த மடக்கு வாக்கியம் தொடர்ந்து இயங்கும் - இங்கு அது x என்ற மாறியின் மதிப்பு 5 ஆகும் வரை இயங்கும். அதன் பின் மடக்கு கட்டளைகளில் இருந்து வெளியேறி "மடக்கு கட்டளை முடிந்தது" என்ற செய்தியை அச்சிட்டு முடியும்.

மேல் கொடுக்கப்பட்ட மடக்கு வாக்கியம் ஒரு நிபந்தனை ($x < 5$) உண்மையாக இருக்கும் வரை அதனுள் உள்ள எல்லாக் கட்டளைகளையும் இயக்கும்.உங்கள் பெற்றோர் நீங்கள் சாலையைத் தாண்டும் வரை இரண்டு பக்கமும் கவனம் செலுத்தச் சொல்வார்கள் இல்லையா ? அதே போலத் தான் இந்த வரை எனும் *while* கட்டளை செயல்படும். அதாவது உங்கள் பெற்றோர்கள் உங்களையே நிரல்படுத்துகிறது போல என்றும் சொல்லலாம் - அதாவது சாலையில் இரண்டு பக்கமும் கார்கள் இல்லாத வரை மட்டுமே

நீங்கள் சாலையைக் கடக்கலாம்.

ரூபி கட்டளை *puts* திரையில் அதனைச்சார்ந்த மதிப்பை அச்சிடும்; கொஞ்சம் நுட்பமாக சொல்லவேண்டுமானால் திரையில் அச்சிட *puts* என்ற கட்டளைக்கு ஒத்து வரும் மதிப்பை எடுத்துக்கொள்ளும் - அடுத்து இந்த மதிப்பு திரையில் அச்சாகும்!

கீழ் கண்ட மாதிரி கட்டளைகள் *(pseudocode)* என்பது *while* மடக்கு வாக்கியம் எப்படிச் செயலாற்றும் என்று விளக்கும் வண்ணம் கொடுக்கப்பட்டது

```
    '#' pseudo code
1) x is 0

  2)  Is 0 less than 5? True. puts x. x = 0 + 1. x is 1.

  3)  Is 1 less than 5? True. puts x. x = 1 + 1. x is 2.

  4)  Is 2 less than 5? True. puts x. x = 2 + 1. x is 3.
```

5) *Is 3 less than 5? True. puts x. x = 3 + 1. x is 4.*

6) *Is 4 less than 5? True. puts x. x = 4 + 1. x is 5.*

7) *Is 5 less than 5? False. Loop ends.*

8) *"while மடக்கு வாக்கியம் முடிந்தது." is printed to the screen.*

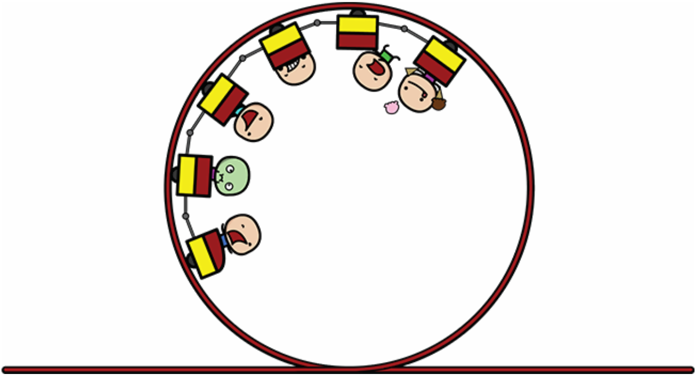
இந்த உதாரணத்தில் மாறி, *x* பூஜ்ஜியம் என்று தொடக்கத்தில் அதாவது, மடக்கு வாக்கியத்தின் முன்பு, மதிப்பு பெறுகிறது. மடக்கு வாக்கியம் தொடங்கும் போது, கணினி இந்த மாறியின் மதிப்பு ஐந்திற்குக் குறைவாக இருக்கிறதா என்று பார்க்கும். பூஜ்ஜியம் என்பது ஐந்திற்குக் குறைவானது அதனால் திரையில் *puts* இன் ஒத்த சரம் வெளியிடப்படும். மடக்கு வாக்கியத்தில் அடுத்த கட்டளை *x* என்பதன் மதிப்பை ஒன்றால் கூட்டக்கூடியது. இதுபோலவே *x* என்பதன் மதிப்பு *5* ஐ எட்டும் வரை தொடர்ந்து இயங்கும் -

ஐந்தை எட்டியபின் மடக்கு வாக்கியம் முடிவடைந்து அடுத்த கட்டளைக்கு இயக்கம் செல்லும் - இங்கு இந்தக் கட்டளை _puts

இதன் இயக்கம் வெளிப்பாடு இதுபோல் அமையும்:

```
0

1

2

3

4

while மடக்கு வாக்கியம் முடிந்தது.
```

While மடக்கு வாக்கியங்களை வைத்துக் கொண்டு வெறும் எண்ணுவது மட்டுமே இதுவரை



படம் 6.1: மடக்கு வாக்கியங்கள் ராட்டினம் சுற்றுவது போல - குறிப்பிட்ட அளவு சுற்றிய பின் எப்படி ராட்டினம் சுற்றி நிற்கின்றதோ, அதேபோல இந்த வாக்கியம் இயங்குகிறது என்று தோராயமாக சொல்லலாம்.

செய்துள்ளோம். அடுத்து இதே *while* வாக்கியத்தை கொண்டு வேறென்ன ஒரு முடிவிலா இயக்கத்தை வடிவமைக்கலாம் என்று பாருங்கள். இந்த எடுத்துக்காட்டில் சரியான சாரம் உள்ளீடு செய்யும் வரை இந்த *while* மடக்கு வாக்கியம் இயங்கி கொண்டே இருக்கும்.

```
answer = “” #காலி சரம் ஒன்றை உருவாக்குங்கள்

while answer != “Ruby”

puts “மன்னிக்கவும் தவரான விடை.” unless answer == “”

puts “மிக சிறந்த கணினி மொழி எது?”

answer = gets.chomp

end
```

puts "அடி சக்கை!"

இங்கு சில புதிய ரூபி சார்புகளைப் பயன்படுத்தினோம்; இவை : *gets* மற்றும் *chomp.* இவை இந்தப் புத்தகம் வாசிக்கும் அளவில் உங்களுக்கு விளக்க படும்.

நிரல் விளக்கம் மேல் உள்ள நிரல் துண்டு "மிக சிறந்த கணினி மொழி எது ?" என்ற கேள்விக்கு *"Ruby"* என்று ஆங்கிலத்தில் கொடுக்கும் வரை இயங்கிக்கொண்டே இருக்கும். உங்கள் உள்ளீடை *gets.chomp*என்ற சார்பு பெற்று கொண்டு *answer* என்ற மாறியில் சேமிக்கும்; இதனையே *while* மடக்கு வாக்கியம் முடியும் வரை இயக்கிக்கொள்ளும்.

chomp என்ற சார்பு கிடைத்த சாரத்தில் கடைசியில் உள்ள வரி முடிவு எழுத்தான `'\n'` என்பதை அழித்துப் பயன்படுத்தும் வண்ணம் உங்களுக்கு அந்தச் சரத்தை

பின் கொடுக்கும்.

உங்கள் நிரலை சரிவர பயன்படுத்தும் பயனர் ,“*Ruby*” என்று கொடுத்ததும் இந்த மடக்கு வாக்கியத்தில் இருந்து வெளியேறும் அதன் பின் “அடி சக்கை” என்று கொஞ்சம் எகத்தாளமாகவே நிரல்போல் திரையில் வெளியிட்டு முடிக்கும்.

அடுத்து இந்த எடுத்துக்காட்டில் *for loop* என்று சொல்லக்கூடிய ஒவ்வொன்றாக *for* குறிச்சொல்லை கொண்ட மடக்கு வாக்கியத்தை பார்க்கலாம்.

```
for number in 1..5 do

puts “தற்போதைய மதிப்பு #{number}”

end
```

இந்த *for* மடக்கு வாக்கியம் *while* என்ற மடக்கு வாக்கியம் போல இல்லாமல் நிபந்தனை

உண்மையாகாமல் இருந்தாலும் தொடங்கும். மேல் உள்ள நிரல் துண்டு 1-இல் இருந்து 5-வரை (மொத்தம் 5 முறை) இந்த *puts* நிரல் கட்டளையை இயக்கும்.

இங்குச் சிறப்பாக நீங்கள் கவனிக்க வேண்டியது, மாறி *‘number’* என்பது - இது மதிப்பு வாக்கியம் இயங்கும் பொழுது இதன் மதிப்பு 1, 2, 3, 4, 5 என மறி இக்கொண்டே போகும். இப்போது *for* மடக்கு வாக்கியத்தை இரண்டில் இருந்து எட்டு வரை, அதாவது ஏழு முறை [2,3,4,5,6,7,8] என இயக்க 2..8 என்று ரூபி மொழியில் கொடுக்கலாம். இப்போது இதையே ஒன்றில் இருந்து இருபத்தைந்து வரை எப்படி எழுதுவீர்கள் ? முயன்று பாருங்களேன்.

puts என்பது எப்படி ஒவ்வொரு முறையும் தனது மதிப்பை மாற்றி கொள்கிறது? இதற்க்கு காரணம் *number* என்பது இடமாற்றம் அடைகிறது (substitution) - ஓவொரு முறை மடக்கு வாக்கியம் இயங்கும்

சமயம் *number* இதன் மதிப்பு மாறுகிறது, பின்பு இதன் இடம் மாற்றத்தால் *puts* இன் மதிப்பும் மாறி திரையில் அச்சாகிறது.

```
தற்போதைய மதிப்பு 1

தற்போதைய மதிப்பு 2

தற்போதைய மதிப்பு 3

தற்போதைய மதிப்பு 4

தற்போதைய மதிப்பு 5
```

உங்களுக்கு ரூபியின் மடக்கு வாக்கியங்களின் பலம் புலப்படும் என்று நினைக்கிறோம். இங்கு கொடுக்கப்பட்ட ரூபி மடக்கு வாக்கியம் *while* மற்றும் *for* தவிர மற்றவையும் உள்ளன அனால் இதை கொண்டு மட்டுமே நாம் தொடக்கத்தில் இயங்கலாம். மடக்கு வாக்கியங்களை தரவமைப்புகளுடன் *(datastructures)*

இணைந்து நிரல்படுத்தினால் இவற்றின் அசுர பலம் காணலாம்!

உங்களுக்கு ஒரு பொம்மை அலமாரி, அல்லது ஒரு சொப்பு சாமான் திரட்டு பரப்பி இருந்தால் இவற்றை எண்ணுவது, சரிபார்ப்பது, வரிசைப்படுத்துவது என்பதை எல்லாமே ரூபியிடம் ஒரு அதிவேக கணினியிடம் கொடுத்துவிடலாம். நீங்கள் இதை செய்யாமல் ஒரு செவ்வியல் புத்தகத்தையோ அல்லது தமிழிசை, கர்நாடக இசையை இரசிக்க நேரம் கழிக்கலாம், இல்லாட்டி ஒரு தலைவர் படத்தைக் கூட மறுபடி பார்க்கலாம். ரூபியின் மடக்கு வாக்கியங்கள் பலம் உங்களுக்கு எப்போதும் உண்டு! இந்தத் தொகுதியை முடிப்பதற்கு

பயிற்சி

நாம் கடற்கரையில் நமது மாயாவிக்காகக்

காத்திருக்கிறோம் என்று எண்ணிக்கொள்வோம்; சூரியன் அஸ்தமனம் ஆகும் பொன் மாலை நேரம். நாம் ஒவ்வொரும் மணி நேரம் கழியும் நேரத்திலும் மாயாவி வந்துவிட்டாரா என்று பார்க்கிறோம். இதனை *Ruby* மொழியில் இப்படி எழுதலாம் - அதாவது *'while'* என்ற மடக்கு கட்டளையில் வழியாக. ஒரு *7*-மணி மலையில் சூரியன் அஸ்தமிக்கும் என்று கொள்ளுங்கள்; நாம் *5* மணி மாலை அளவில் கடற்கரையில் வந்தோம் என்று கொள்ளுங்கள்.

```
சூரியன்_மறைவது = 7

நேரம்_இப்போது = 5
```

மேல் உள்ள நிரலில் 'சூரியன்_மறைவது' என்று ஒரு மாறியில் *7* என்று கொடுத்தோம்; 'நேரம்_இப்போது' என்ற மாறியில் *5* என்று கொடுத்தோம்.

```
while நேரம்_இப்போது <= சூரியன்_மறைவது do
```

```
puts “இன்னும் மாயாவிக்கு காத்திருக்கிறோம்.”

puts “இப்போது நேரம் #{நேரம்_இப்போது} மணி மாலை”

நேரம்_இப்போது = நேரம்_இப்போது + 1

end

puts “மாயாவி வந்தார்!”
```

இந்த ‘while’ மடக்கு கட்டளையின் வெளிப்பாடு கீழே:

```
இன்னும் மாயாவிக்கு காத்திருக்கிறோம்.

இப்போது நேரம் 5 மணி மாலை

இன்னும் மாயாவிக்கு காத்திருக்கிறோம்.

இப்போது நேரம் 6 மணி மாலை
```

இன்னும் மாயாவிக்கு காத்திருக்கிறோம்.

இப்போது நேரம் 7 மணி மாலை

மாயாவி வந்தார்!

1. முடிவில் 'இப்போது_நேரம்' என்பதன் மதிப்பு என்ன ?

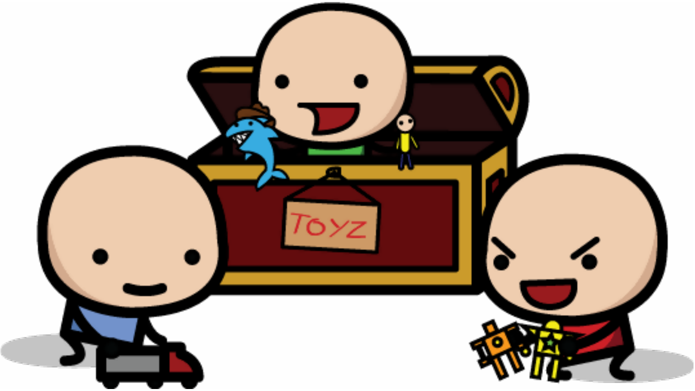
7. தரவமைப்புகள் (Collections)

தரவு மதிப்பு கொண்ட உறுபடிகளை கொள்ளும் ஒரு பாத்திரமே "தரவமைப்புகள்" எனப்படும். இது போன்ற தரவமைப்புகளில் இரண்டு வகையாக நமது உறுபடிகளை சேமிக்கலாம். முதலாவது "அணி" எனப்படும் *(array)* தரவமைப்பு, இரண்டாவது "எண் குறி அடைவு" எனப்படும் *(hashtable)*. தரவமைப்பு என்பது ஒரு விளையாட்டு சாமான் போட்டியைப் போலவே - ஒன்னும் குழப்பமான விஷயம் இல்லை.

முதலில் விளையாட்டு சாமான் அணி எனும் தரவமைப்புடன் இந்தப் பாடத்தைத் தொடங்கலாம்.

1. அணிகள் *(Arrays)*

அணி, அல்லது அணிகள், என்பது வரிசைப்படுத்தப்பட்ட, இடம் வழி அணுகும் வகை



படம் 7.1: விளையாட்டு பெட்டியும் அதனுள் வைக்கப்படும் சாமான்களை போல அமைந்தது அணிகள்

தரவமைப்புகள்; இது போன்ற அணிகளில் எந்த வகை தரவுகளையும் சேர்க்கலாம். இவைகளை வரிசைப்படுத்தலாம் என்றல் இதுவே பொருள்: பள்ளியில் காலை கூட்டத்தின் பின்பு வயது அல்லது பெயர் அகரமுதலி வரிசையில் நின்று வகுப்பறையில் நுழைவீர்கள். அணிகளும் அது போலவே. உங்கள் பள்ளி வகுப்பு மாணவர்களை வயது, பெயர், இடை, உயரம் என்ற அம்சங்களால் வரிசைப்படுத்தலாம். அதாவது இந்த நான்கு அம்சங்களில் ஏதேனும் ஒரு வகை இதனை கொண்டு வரிசைப்படுத்தலாம். இதனை நிரல் என்று எழுதினால், அது 'அணிகள்' என்பதால் குறிக்கப்படும்.

வரிசையில நாலு பிள்ளைகள் இருங்காங்க என்று கொள்ளவோம்: குமரன், முகிலன், செல்வி, மட்டும் யாழினி. நம்ம இந்தக் குழந்தைகளின் பெயரை எழுதியபடியே இவற்றை இடம் சூட்டும் படி எண்

இடலாம். இப்போது குமரன் 1. முகிலன் 2. செல்வி 3. யாழினி 4. இந்த நாலு பிள்ளைகளும் வரிசையில் நிக்க இதையே அணியில் குறியீட்டால் இப்படி எழுதலாம்

[“குமரன்”, “செல்வி”, “முகிலன்”, “யாழினி”]

அணிகள் என்பதை குறியீடே ‘[’ தொடக்கம் மற்றும் ’]’ முடிவ சதுர அடைப்பு குறிகளை பயன்படுத்தலாம். இந்த இரு சதுர அடைப்பு குறிகளுக்குள் நமது சிறுவர்களின் பெயர்களை சரங்களாக எழுதுங்கள்.

சரம் என்பது உள்ளீடு வரிசை படுத்தப்பட்ட படி கணினி நினைவகத்தில் இருக்கும். கணினியில் அணிகளை பிரித்து எடுக்க இவற்றின் இடம் சூட்டும் எண் வரிசை தேவைப்படும் - தினசரி வாழ்வில் ஒன்றில் இருந்து எண்ணுவது போல் இல்லாமல் கணினி இடம் சூட்டும் வகை பூஜ்யத்தில் இருந்து தொடங்கும்.

அதாவது நமது மாணவர்களின் பெயர்களை இடம் *0*-இல் குமரன் *1*-இல் செல்வி *2*-இல் முகிலன் *3*-இல் யாழினி

இடம் சூட்டும் எண் என்பது ஒரு தொலைதூரத்தை அளப்பதற்கு சமமான அளவு. குமரன் என்ற சரம் மதிப்பு இடம் *0*-இல் உள்ளது. தொடர்ந்து செல்வி ஒன்றிலும், முகிலன் இரண்டிலும் கடைசியாக யாழினி மூன்றிலும் இடம் சூட்டி அணியில் சேமிக்கப்பட்டது.

கீழே சிறுவர்களின் பெயர்களை அணியி இட்டபடி அதனை சம்பந்த பட்ட இடம் சூட்டு எண்களை மற்றோரு அணியில் சேமித்தால் அது

kids_array_index என்று தோன்றும்.

kids_array = ["குமரன்", "செல்வி", "முகிலன்", "யாழினி"]

‘#’ kids_array index => [0,1,2,3]

வரிசைப்படுத்திய அணியில் முதல் இலக்கில் உள்ள மதிப்பை எடுக்க, அதாவது இடம் சூட்டி எண் பூஜ்யம், இதில் உள்ள மதிப்பை பெற இப்படி நிரல் குறிமுறை செய்யணும்,

kids_array[0]

=> “குமரன்”

சதுர அடைப்பு குறியீடு என்பது ஒரு அணி வகையான மாறியில் உள்ள குறிப்பிட்ட இடம் சூட்டு எண்ணில் உள்ள மதிப்பை குறித்து எடுக்கும் வண்ணம் அமையும். இப்போது அதே மாதிரி சதுர அடைப்பு குறிகளுக்குள் அணியில் உள்ள மதிப்புகளையும் மாற்றலாம். இப்போது யாழினி ஊர் மாரி வேறு பள்ளிக்கு சென்றால் நமது அணியில் அவளது இடத்தை கருப்பாயி என்ற இன்னொரு

குழந்தைக்கு கொடுத்துவிட்டால் அதனை இப்படி நிரலில் எழுதலாம்.

kids_array = [“குமரன்”, “செல்வி”, “முகிலன்”, “யாழினி”]

kids_array[3] = “கருப்பாயி”

kids_array

=> [“குமரன்”, “செல்வி”, “முகிலன்”, “கருப்பாயி”]

இவர்களை அகரமுதலி படி வரிசைப்படுத்தலாம். இது ‘குமரன்’, ‘செல்வி’, ‘முகிலன்’, ‘யாழினி’ என்ற வரிசையில் வரும். புதிய நபர் ஒருவரை நமது அணி பட்டியலில் சேர்க்கலாம் - இவளது பெயர் அல்லி.

kids_array[4] = “அல்லி”

```
kids_array

=> [“குமரன்”, “செல்வி”, “முகிலன்”, “கருப்பாயி”,“அல்லி”]
```

அணிகள் என்பதில் சரம் சேர்த்தும் கழித்தும் பார்த்தோம் அனால் இவற்றில் எண்கள் மற்றும் வேறு அணிகள், போன்ற தரவமைப்புகளையும் சேற்றுக்கொள்ளலாம். கீழே உள்ள நிரல் துண்டில் பார்க்கலாம்

```
number_array = [1,2,3,4,5]

string_array = [“அல்லி”, “பாலன்”, “வள்ளி”, “தேவன்”]

mixed_array = [number_array, “a string”, 13]
```

ஒரு அணி மற்றோரு அணியை மதிப்பாகக் கொள்ளலாம். உதாரணத்துக்குக் கீழே உள்ள நிரலில்

அணி *lots_of_arrays* என்பதில் முதல் மதிப்பை எடுக்க எப்படி எழுதலாம்:

lots_of_arrays[0] இதன் மதிப்போ *[1,2]* என்ற அணி. இதன் இரண்டாம் இடம் சூட்டு மதிப்பை அணுக இப்படி எழுதவேண்டும் *lots_of_arrays[0][1]* இவை அனைத்தையும் சேர்த்து பார்த்தால் இப்படியான நிரல் துண்டு அமையும்.

lots_of_arrays = [[1,2],“சரம்”,“தேர்வுசெய்”]

lots_of_arrays[0][1]

=> 2

ரூபி மொழி நமக்கு அணிகள் என்ற தரவமைப்பில் பல உள்ளடக்கிய அணிகளுக்கான சார்புகளை தருகிறது. இவற்றில் வரிசைப்படுத்துவதற்கானது *sort* என்றும் இதனை செயல்படுத்தலாம். உதாரணம்,

string_array.sort

=> [“அல்லி”, “தேவன்”, “பாலன்”, “வள்ளி”]

இந்த அணி வரிசை படுத்தப்பட்டு கணினி திரையில் அச்சிட்டது. ஆனால் இது கணினி நினைவில் மாறாமல் இருக்கும். இதனை, அதாவது *string_array* என்ற மாறியை வரிசைப்படுத்தி அதே கணினி நினைவில் சேமிக்க *sort!* என்று அழைக்க வேண்டும்.

இதனை அழைக்கும் முன்பே பாருங்கள்

string_array

=> [“அல்லி”, “பாலன்”, “வள்ளி”, “தேவன்”]

அழைத்தபின் பாருங்கள்:

string_array.sort!

=> [“அல்லி”, “தேவன்”, “பாலன்”, “வள்ளி”]

அடுத்து ரூபி மொழியில் திணி *(shovel)* << என்கிற செயற்குறியைக் கொண்டு எப்படி ஒரு அணியின் கடைசியில் ஒரு மதிப்பை அந்த அணியில் இணைப்பது என்று பார்ப்போம்,

string_array

=> ["அல்லி", "தேவன்", "பாலன்", "வள்ளி"]

string_array << "கபிலன்"

=> ["அல்லி", "தேவன்", "பாலன்", "வள்ளி", "கபிலன்"]

இந்த மண்வாரி << என்ற செயற்குறி ஒவ்வொரு தரவமைப்புக்கும் தனிப்பட்ட செயல் செய்யும் வகை ரூபி மொழியில்

push என்கிற சார்பின் பயன்படுத்தி நமது அணியில் சேற்றிருக்கலாம். அதாவது:

string_array.push("கபிலன்")

=> ["அல்லி", "தேவன்", "பாலன்", "வள்ளி", "கபிலன்"]

.

2. தொடர்புரு அணி (அல்லது எண்ணிம அடைவு) (ஹாஸ் - Hash)

ரூபீ மொழியின் மற்றொரு தரவம்சம் Hash அல்லது எண்ணிம அடைவு என்பது. இதனை தொடர்புரு அணி அல்லது உருப்படி அகராதி என்றும் (இதன் நிறுவும் வடிவமைப்பை பொருத்து) அழைக்கலாம். அதாவது ஒரு சொல்-பொருள் தொடர்பை எப்படி அகராதி பிரதிபலிக்கிரதோ அதேபோல் இவ்வாறான தரவமைப்புகள் அனைத்தும் ஒரு உருப்படி அதனுடன் தொடர்புடைய மதிப்பையும் கணினியில் கையாள உதவுகின்றது. அணி என்பதில் உள்ள வரிசைப்படுத்தும்

குணம் இவ்வகை தொடர்புரு அணிகளில் கிடையாது - ஏனெனில் இவை சீரற்ற அமைக்கப்பட்டவை.

தொடர்புரு அணியில் உருப்படிகளை சுருளை அடைப்புக்குறி *'{' , '}'*, கொண்டு உருப்படியின் பெயர்-பொருள் என்று எழுதலாம். உதாரணம்: *{"உ-பெயர்" => "உ-பொருள்"}*. இங்கு *'=>'* என்ற குறியீடு பெயர் - பொருள் தொடர்பை குறிக்கும்.

ஒரே ரகத்தில் பல பொருள்கள் இருந்தால் அவற்றை தொடர்புரு அணியாக எழுதலாம். உதாரணம், வீட்டின் விளையாட்சாமான்களையும் அதன் மதிப்பையும் கொண்டு ஒரு தொடர்புரு அணி உறுவாக்கினால், இப்படி இருக்கும்:

```
toy_chest = {

"அன்னம்" => 12,
```

```
“பொம்மை” => 5,

“சொப்புச்சாமான்” => 514

}
```

இந்த மூன்று உருப்படிகளும் நமது மாறியில் சேமிக்கப்பட்டது. உருப்படி பொயர்களும் - அதாவது அன்னம், பொம்மை, சொப்புச்சாமான் மற்றும் அதன் நேர்தொடர்புடைய மதிப்புகளான (உருப்படி பொருட்கள்) 12, 5, 514 உம் சேமிக்கப்பட்டது. இந்தச் சேமிக்கப்பட்ட தகவல்களை அணுக சதுர அடைப்புக்குறிகளைப் பயன்படுத்தவேண்டும்.

```
toy_chest[“அன்னம்”]

=> 12

toy_chest[“பொம்மை”]
```

```
=> 5

toy_chest["சொப்பூச்சாமான்"]

=> 514
```

மேல்கண்டபடி புதிய உருப்படிகளை நமது மாறியில் சேக்கலாம். *'toy chest'* என்ற தொடர்புரு அணியில் சதுர அடைப்புக்குறிகளை கொண்டு புதிய உருப்படிகளை சேர்க்கலாம்:

```
toy_chest["விளையாட்டூப்பெட்டி"] = 7

=> 7

toy_chest

=> {"அன்னம்"=> 12, "பொம்மை"=> 5, "சொப்புச்சாமான்"=> 514,

"விளையாட்டுப்பந்து" => 7}
```

அதாவது “விளையாட்டுப்பந்து” என்று சதுர அடைப்புக்குறிகளுக்குள் எழுதினால், *“toy_-cars”* என்ற தொடர்புரு அணியில் *7* என்ற மதிப்பு “விளையாட்டுப்பந்து” என்ற பெயருடன் தொடர்பாக இணைக்கப்படும். ஒரு உருப்படியைத் தொடர்புரு அணியில் இருந்து அழிக்க *delete‘* என்ற நிரல்பாகத்தை பயன்படுத்தலாம். இது எப்படி செயல்படுகிறது என்பதை பற்றியும் நாம் அடுத்த கட்டத்தில் படிக்கலாம்.

```
toy_chest.delete(“அழி”)

=> 12

toy_chest

=> {“பொம்மை”=> 5, “சொப்புச்சாமான்”=> 514,

“விளையாட்டுப்பந்து” => 7}
```

மேல்கண்டதை விட கூடுதலாக ரூபீயில்

அடைவில் உருப்படிகளை சேர்க்கவும், நீக்கவும் மாற்றவும் அதிக வழிகள் உள்ளன். இவற்றை அனுகுவதற்கு முன் நிரல்பாகங்கள் *(methods)* என்பதைப்பற்றி பயிலவேண்டும். அடுத்துவரும் சில தொடர்புறு அணி உதாரணங்களை வசித்த பின் நீங்கள் மேற்படியாக அடுத்த பாடத்திற்கு செல்லலாம்!

பயிற்சி

அணிகள்

1. இந்தக் குழந்தைகளின் பெயர்களை *'kid_array'* என்ற ஒரு அணியில் திரட்டுங்கள்: சிவா, அலர்மேல் மங்கை, பூர்ணிமா, கவிதா, சேகர், நாச்சம்மை. ஆண் மற்றும் பெண் குழந்தைகளின் பெயர்களை அகரமுதலி வரிசைப்படுத்த; விடை என்ன ?
2. *'kid_array'* என்ற அணியில் இருந்து ஆண் மற்றும்

பெண் குழந்தை பெயர்களை தனி தனி அணிகளில் இடுக; மேலும் இந்த இரண்டு அணிகளையும் ஒரு "குழு" என்ற அணியில் இடுக. இதனை செய்ய எவ்வளவு சிறியதான நிரலை எழுதலாம் ?

3. *'group' என்ற குழு அணியில் உள்ள 'boy_array' மற்றும் 'girl_array' என்பதை எப்படி வரிசை மாற்றலாம்?*

4. *நமது வகுப்பில் 'டிப்பாணி' என்ற சிறுமி அமெரிக்காவில் இருந்து சேர்ந்தாள்; இவளை எப்படி 'குழு அணியில்' சேர்ப்பீர்கள் ?*

5. *வகுப்பு வாத்யார் எப்படி வகுப்பில் சரிசமம் பையன்களும் பொண்ணுங்களும் இருப்பார்கள் என்று சரிபார்க்கலாம் ? இதற்க்கு என்ன நிரல் துண்டு உதவும். உதவி குறிப்பு: அணியின் நீளம் என்பது இங்கு எதனை குறிக்கும் என்று யோசித்து*

பாருங்கள்.

எண்குறி அடைவு

6. மாயாவியின் கைபையில் உள்ள உறுபடிகளை குறிக்க நிரலில் எண்குறி அடைவு மூலம் குறிக்க. இந்தப் பையில் 3 தவளைகள், 5 மூலிகைகள், மற்றும் 10 சுவடிகள் உள்ளன. “bag” என்ற பெயரில் ஒரு மாறிலியை உருவாக்குக. உங்கள் நிரல் எப்படி அமைந்தது ?

7. எண் குறி அடைவில் எந்த வகையான மதிப்புகளையும் சேமிக்கலாம் - வெறும் எண்கள் மட்டுமல்ல - இதனை நினைவில் கொள்ளுங்கள். மேல் உருவாக்கிய அடைவில் மாயாவியின் மந்திரம் ஒன்றை சேமியுங்கள். மாயாவியின் பையில் “shazam” என்ற மந்திரத்தை சேமியுங்கள் - அதன் விளைவை அடைவின்

மதிப்பாக சேமியுங்கள். இங்கு *“shazam”* என்ற மந்திரம் *“turns subject into a frog”* என்ற விளைவை உண்டாக்குவதால் அதனையே மதிப்பாக சேமியுங்கள். இப்போது கைபை நிரல் எப்படி இருக்கிறது ?

8. சரி இப்போ மாயாவி மனது மாறி அவர் பையில் உள்ள *“shazam”* என்ற மந்திரத்தை அழிக்க விரும்புகிறார்; எதனை எப்படி செய்வது?

9. மாயாவிக்கு சில புதிய லேகியங்கள் கிடைக்கின்றன; இவை 4 ஆரஞ்சு லேகியம், 5 நீலம், மற்றும் 7 சிவப்பு லேகியங்கள். உங்கள் பையில் லேகியங்கள் என்ற பெயரில் இந்த தகவல்களை எண்குறி அடைவாக சேமிக்கவும்.உதவி குறிப்பு: ஒரு அடைவு அதன் உறுபடிகளில் மற்றோரு எண்குறி அடைவை கொள்ளலாம்.

10. அடைவில் தற்போது நான்கு உறுபடிகள் உள்ளன (தவளை, மூலிகை, சுவடி மற்றும் லேகியம்); ஒரு புதிய மாயம் செய்ய மாயாவிக்கு 2 தவளைகள், 3 மூலிகைகள், 1 சுவடி மற்றும் 2 நீல லேகியங்கள் தேவை. இவற்றை கைப்பையில் இருந்து எடுங்கள் - மீதம் எவ்வளவு உள்ளது ?உதவி குறிப்பு : கழித்தல் '-' செயற்குறியைப் பயன்படுத்துங்கள்.

8. நிரல்பாகங்கள்

நிரலாக்கலில் பெரும் பகுதி என்பது, ஒரு பெருநிரலை ஒன்று சேர்ந்து இயங்கக்கூடிய, கட்டளைகளை சின்னஞ்சிறு பகுதிகளாக பிரிப்பது தான். இதைப்பற்றி புரிந்து கொள்ள, சராசரி வாழ்வில் நாம் காணும் நிகழ்வுகளுடன் ஒப்பிட்டுப் பார்ப்போம். உதாரணத்திற்கு நாம் கிரிக்கெட் விளையாடுவதாக கற்பனை செய்து கொள்வோம். மைதானத்திற்கு வெளியே நீங்கள் பந்தை விரட்டி அடிக்கும் பரபரப்பான தருணத்தை நீங்கள் அடைவதற்கு காரணம், நீங்கள் களமிறங்கியது முதல் கிடைத்த நேரத்தை சிறு சிறு பகுதிகளாக பிரித்து உங்களை தயார் படுத்திக் கொண்டது தான்.

முதலில் மட்டையைக் கையில் எடுத்திருப்பீர்கள். ஓரிரு முறை மட்டையை வீசிப் பயிற்சி

செய்திருப்பீர்கள். பின்னர், ஆடுகளத்தை நோக்கி நகர்ந்து, தயார் நிலைக்கு வந்து, பந்துவீச்சாளரை நோக்கிக் கவனத்தை செலுத்தியிருப்பீர்கள். இறுதியாக உள்மூச்செடுத்து பந்தை அடித்திருப்பீர்கள். இவ்வாறாகிய உங்கள் ஒவ்வொரு அசைவும் விளையாட்டின் பாகங்கள் (விளையாடுவதற்கான சிறு நெறிமுறைகள்). இவை தான் உங்களைச் சிறப்பான ஆட்டத்திற்கு தயார் செய்யும்.

இதை வேறு விதமாகவும் புரிந்து கொள்ளலாம். நீங்கள் காணொளி விளையாட்டில் புதிய உயர் நிலையை அடையும் போது, அந்த விளையாட்டில் உள்ள மெய்யான நிரல்பாகங்கள் அழைக்கப்படலாம். மேலும், அந்த விளையாட்டில் உள்ள உயிர்_-வாய்ப்புகள் எனும் நிரல்பாகம், நமக்கு விளையாட எத்தனை வாய்ப்புகள் உள்ளன என்பதை காட்டும். வேறொரு நிரல்பாகம், நமது மொத்த மதிப்பெண்களை

காண்பிக்கும். மற்றுமொன்று உங்களது மீதமுள்ள ஆயுளின் நலத்தை காண்பிக்கும். இதில் உள்ள அனைத்து நிரல்பாகங்களும், விளையாட்டில் உங்கள் செய்கைகளை கவனித்து உள்ளீடு *(input)* பெற்றுக் கொள்ளும். பின்னர், மற்ற நிரல்களை அணுகி தகவல்களை சேகரித்து, புதுப்பித்து சரியாக நமக்கு வழங்கும்.

நாமே ஒரு நிரல்பாகத்தை எழுதினால், அதைப்பற்றி இன்னும் சிறப்பாக நம்மால் புரிந்து கொள்ள முடியும். கீழே, சிறு தகவலை மட்டும் திரையில் காட்டும் ஓர் எளிய நிரல்பாகம் கொடுக்கப்பட்டுள்ளது. ஐஆர்பி-யின் *(IRB)* உள்ளே நாம் ஒரு நிரல்பாகத்தை எழுதிவிட முடியும், அதற்கு புதியொரு கோப்பை *(file)* உருவாக்க வேண்டியதில்லை. *repl.it/languages/Ruby* என்ற இணையதள முகவரிக்கு சென்றும் நாம் நிரல்பாகங்களை எழுதலாம்.

நிரல்பாகத்தை இடதுவசம் உள்ள பிரிவில் எழுதிவிட்டு, நிரலை ஓடவிட பொத்தானை அழுத்துங்கள்.

முனையத்தில் பின்வருமாறு தட்டச்சு செய்யுங்கள்:

```
$ irb # ரூபி மொழிமாற்றியைத் (nterpreter) திறக்க irb என தட்டச்சு செய்யுங்கள்

irb(main):001:0> def hello

irb(main):002:1> puts “நல்வரவு!”

irb(main):003:1> end

=> nil
```

ரூபியில் நாம் நிரல்பாகத்தை, *def* என்ற பதிப்புச்சொல்லால் *(keyword)* உருவாக்குகிறோம். நிரல்பாகத்தின் பெயர் *def* என்ற சொல்லுக்கு பின் வரவேண்டும். இதுதான் நம் *hello* நிரல்பாகம், இவை

எத்த உள்ளீடுகளையும் பெறவில்லை (அலுவலகம் செல்லும் போது, பெற்றோர்கள் உங்களை அழைத்துச் செல்ல மாட்டார்கள் அல்லவா, அது போலத்தான்). நிரல்பாகத்தின் உள்ளீடுகளைப் பற்றி இன்னும் சற்று நேரத்தில் பார்க்கலாம்.

திரைப்படத்தின் இறுதியிலோ, தாத்தா பாட்டி சொல்லும் கதைகளிலோ வரும் முற்றும் என்ற வார்த்தை போலவே, ரூபி மொழியின் நிரல்பாகத்தில் வரும் *end* எனும் வார்த்தை, அதன் முடிவை குறிக்கும். *def* மற்றும் *end* என்ற பதிப்புசொற்களுக்கு *(key-words)* இடையில் உள்ள நிரல்களை, கணினி இந்த நிரல்பாகத்தின் பகுதியாக இயக்கும். இந்த நிரல்வரிகள் தான் நிரல்பாகத்திற்கு இயங்க "தொகுப்பாக" நாம் வழங்குவதாகும்.

பல்வேறு நிரல்பாகங்களை நாம் ஒருசேர எழுதி இயக்குவதற்கு, அவற்றை ஒரு கோப்பில் சேமித்து

இயக்குவதே எளிமையானதாக இருக்கும். அதனா;ல், தற்போது முனையத்திலிருந்து வெளியேற exitஎன்ற சொல்லை தட்டச்சு செய்யுங்கள்.

உரை தொகுப்பியைத் *(text editor)* திறந்து கொள்ளுங்கள். எனக்கு ஸப்லைம் டெக்ஸ்ட் *(Sublime Text)* என்னும் தொகுப்பி பிடித்தமானது என்பதால், அதை பயன்படுத்துகிறேன். ஆனான், நீங்கள் வேறெது வேண்டுமானாலும் பயன்படுத்தலாம். பல்வேறு வகையான தொகுப்பிகள் இணயத்திலிருந்து இலவசமாக பதிவிறக்கம் செய்யும்படியாக கிடைக்கின்றன. இவை எதுவும் வேண்டாம் என்றாலும், நீங்கள் *repl.it* இணையதளத்தை பயன்படுத்திக் கொள்ளலாம். ஆனால், அதில் நிரல்களை சேமித்து வைக்க இயலாது.

ஒரு புதிய கோப்பை உருவாக்கி, மேலே நாம் பார்த்த நிரல்பாகங்களை தட்டச்சு செய்யுங்கள்.

```
def hello
puts “நல்வரவு!”
end
```

இந்தக் கோப்பை *hello.rb* என்று மேசைதளத்தில் சேமித்துக் கொள்ளுங்கள். இனி இந்தக் கோப்பை நீங்கள் முனையத்திலிருந்து இயக்கிக் கொள்ளலாம். முனையத்தில் *cd (change directory)* கட்டளையை இயக்கி மேசைதளத்திற்குச் செல்லுங்கள்.

நீங்கள் சேமித்த கோப்பை முனையத்தில் கண்டுபிடிக்க முடியவில்லை என்றால் கலங்க வேண்டாம், நான் உங்களுக்கு ஒரு எளிய வழி கற்றுத்தருகிறேன். பின்வருமாறு முனையத்தில் தட்டச்சு செய்யுங்கள். (இது உங்களை இல்ல அடைவிக்கு அதாவது *home folder* அழைத்துச் சென்று பின்னர் மேசைதளாத்திற்கு கொண்டு செல்லும்)

```
Your-Computer-Name:$ cd

Your-Computer-Name:$ cd ~/Desktop

Your-Computer-Name:Desktop $
```

நாம் தற்போது மேசைதள அடைவில் இருப்பதால், நாம் *hello.rb* நிரலை *ruby* கட்டளையை பயன்படுத்தி இயக்கலாம்! முனையத்தில், *ruby hello.rb* என்று தட்டச்சு செய்யுங்கள். அது பின்வருமாறு இருக்கும்.

```
Your-Computer-Name:Desktop $ ruby hello.rb

Your-Computer-Name:Desktop $
```

திரையில் பிழைகள் எதுவும் தோன்றாவிட்டால், உங்கள் நிரல் வெற்றிகரமாக இயக்கப் பட்டது என்பது பொருள். வாழ்த்துக்கள்!

கொஞ்சம் பொறுங்கள், “நல்வரவு!” என்பது

ஏன் திரையில் தோன்றவில்லை? ஏனெனின், *hello* நிரல்பாகத்தை நாம் அழைக்கவில்லை. நாம் வெறுமனே *hello* நிரல்பாகம் உள்ள நிரலை இயக்கினோமே தவிர, அந்த நிரல்பாகம் கணினியால் இயக்கப்படுவதற்கு அதனை அழைக்காமல் விட்டுவிட்டோம். *hello.rb* நிரலின் சில மாற்றங்களை செய்து, அதை நாம் எளிதாக சரிசெய்து விடலாம்.

```
def hello

puts "நல்வரவு!"

end

hello()
```

இனி *hello.rb* கோப்பு இயக்கப்படும் போது, *hello* நிரல்பாகமானது ஐந்தாவது வரியில் இயங்கும். இந்த நிரல்பாகத்தை அழைக்க, அதன் பெயரை

மட்டும் எழுதினால் போதுமானது. இந்த நிரல்பாகத்திற்கு உள்ளீடுகள் ஏதும் இல்லை என்பதால், அடைப்புக்குறிகள் சேர்க்கவேண்டியது கட்டாயமில்லை. உள்ளீடுகள் இருக்குமேயானால், அவற்றை நிரல்பாகத்தின் பெயருக்கு பின்பாக அடைப்புக்குறிக்குள் கொடுக்க வேண்டும். இனி நாம் நிரலை இயக்கினால் பின்வரும் வெளியீடு நமக்கு கிடைக்கும்:

```
Your-Computer-Name:$ ruby hello.rb

நல்வரவு!

Your-Computer-Name:$
```

வாழ்த்துக்கள்! நிரல்பாகத்தை பயன்படுத்தி உங்கள் முதல் நிரலை நீங்கள் எழுதிவிட்டீர்கள். அது *puts* என்ற ரூபியின் உள் நிரல்பாகத்தை பயன்படுத்தி, வெற்றிகரமாக இயங்கி, “நல்வரவு!”

என்ற வெளியீட்டை நமக்கு திரையில் அளித்துள்ளது.

இனி உள்ளீடு பெறும் ஒரு நிரல்பாகத்தை எழுதலாம். நம் நிரல்பாகத்தை உபயோகித்து இரு எண்களை பெருக்க வேண்டும் என் வைத்துக் கொள்வோம். அதை நாம் பின்வருமாறு எழுதலாம்.

```
def multiply(num1,num2)

num1 * num2

end
```

அடைப்புகுறி பயன்படுத்தி இரு எண்களை *(num1 and num2)* உள்ளீடு செய்யுமாறு, *multiply* எனும் பெயர் சூட்டி நாம் நிரல்பாகத்தை எழுதலாம். அதன்பின் நாம் இலகுவாக ரூபியின் பெருக்கல் குறியை *(*)* இரு எண்களிலும் பயன்ன்படுத்தலாம். ரூபியில் இயல்பாக, ஒரு நிரல்பாகத்தின் இறுது கட்டளையின் விடை

திரையில் வெளியீடு செய்யப்படும். அதனால், நாம் எழுதிய நிரல்பாகத்தின் இறுதி கட்டளையின் விடை வெளியீடு செய்யப்பட்டுள்ளது.

```
multiply(2,3)

=> 6

multiply(4,2)

=> 8

multiply(“டிக்”,3)

=> “டிக்டிக்டிக்”
```

பார்க்கப்போனால், நாம் எழுதிய நிரல்பாகம் எண்கள் மட்டும் அல்லாது சரங்களையும் பெருக்கும் வல்லமை உள்ளது போலும்! நம் நிரலில் சில மாற்றங்களை செய்து, எண்களை மட்டும் உள்ளீடு

செய்ய அனுமதிக்கும்படி செய்துவிடலாம். ஆனால், அதைப் பற்றி நாம் இப்போது கவலைப்பட வேண்டாம். நமது, வயதை நாட்களில் கணக்கிடும் படியாக ஒரு புது உதாரணத்தை பார்க்கலாமா?

```
def years_to_days_old(years)

result = years * 365

puts “உங்கள் வயது #{result} நாட்கள் ஆகும்!”

end
```

மேல் குறிப்பிட்ட உதாரணத்தை சிறிது விளக்கமாக பார்க்கலாம். முதலாவதாக, நம் நிரல்பாகத்தின் பெயரை *years_to_days_old* என்று மாற்றியுள்ளோம். இது *years* என்னும் ஒரேயொரு மாறி / மாறிலி *(variable)* மட்டும் பெற்றுக் கொள்கிறது. நம் நிரல்பாகத்தின் முதல் வரி, *result* என்றொரு மாறிலியை உருவாக்கி, அதில் நாம்

உள்ளீடு செய்த *years* என்பதன் மதிப்பை 365-ஆல் பெருக்கி, அந்தப் பெருக்கல் மதிப்பைச் சேமிக்கிறது. இது நாம் உள்ளீடு செய்த ஆண்டுகளில் எத்தனை நாட்கள் உள்ளன என்பதை நமக்கு தெரிவிக்கும். பின்பு நாம் “உங்கள் வயது *x* நாட்கள் ஆகும்!” என்னும் சரத்தின் வாயிலாக விடையைத் திரையில் வெளியீடு செய்கிறோம். இதில் *x* என்பது நம் விடையைக் குறிக்கும்.

நம் நிரல்பாகத்திற்கு 10 ஆண்டுகள் என உள்ளீடு செய்தால் அது பின்வருமாறு இருக்கும்.

```
years_to_days_old(10)

உங்கள் வயது 3650 நாட்கள் ஆகும்!

=> nil
```

puts நிரல்பாகம், நம் விடையைத் தருவதற்கு

சரத்தினுள் உள்ள இடைச்செருகலைப் *(interpolation) #{ }*பயன்படுத்துகிறது. ரூபி, சரத்தினுள் சிறு நிரலைப் பயன்படுத்த வழங்கும் முறையைத் தான் நாம் இடைச்செருகல் *(interpolation)* என்கிறோம். அது ஒரு மாறிலியாகவோ, *#{ }* என்பதற்குள் இருக்கும் கணிதச் சமன்பாடாகவோ இருக்கலாம். இதோ, கீழே சில உதாரணங்கள்.

```
x = 15

puts “இதனுள் #{x} மாறிலி உள்ளது.”

=> இதனுள் 15 மாறிலி உள்ளது.

puts “இது சமன்பாட்டை கணிக்கிறது: #{x + 5}”

=> puts இது சமன்பாட்டை கணிக்கிறது: 20
```

நாம் *years_to_days_old* நிரல்பாகத்தில் வேறு சில கட்டளைகளைச் சேர்த்துவிட்டு, அதன் உள்ளீட்டை

நீக்கி விடலாம். நாம் ரூபியின் *gets* நிரல்பாகத்தைப் பயன்படுத்தினால், பயனரிடமிருந்தே நேரடியாக உள்ளீடு பெற்றுவிடலாம். நாம் முந்தைய பாடத்தில் செய்தது போல், *chomp* பயன்படுத்தி, புது வரியை *(new line character)* நீக்கிவிடலாம். *to_i* நிரல்பாகமானது நாம் உள்ளீடு செய்யும் சரத்தை எண்ணாக மற்றிவிடும். நாம் மாற்றம் செய்த நிரல்பாகம் பின்வருவது போல் காட்சியளிக்கும்.

```
def years_to_days_old
puts “தாங்கள் வயதை உள்ளீடு செய்யவும்!”
years = gets.chomp.to_i
days = 365 * years
puts “உங்கள் வயது #{days} நாட்கள்!”
end
```

இனி நாம் இதை *IRB* முனையத்தில் இயக்கிப் பார்க்கலாம். முதலாவதாக, இந்த நிரலை *age_in_days.rb*என்ற கோப்பில் சேமித்து, அதை மேசைதள அடைவிலோ, வேறு அடைவிலோ வைத்து விடுங்கள். இனி நீங்கள் முனையத்தை திறக்கும்போது, தாங்கள் சேமித்த கோப்பின் அடைவிற்கு சென்று *irb* என தட்டச்சு செய்யுங்கள். இனி நீங்கள் பின்வருமாறு உங்களது நிரலை இயக்கலாம்.

```
/Documents/Ruby/my_code_folder >

/Documents/Ruby/my_code_folder > irb

    load ‘age_in_days.rb’

=> true

    years_to_days_old

தாங்கள் வயதை உள்ளீடு செய்யவும்?
```

தற்போது நீங்கள் நிரல்பாகம் குறித்த அடிப்படை அறிவை பெற்றிருப்பீர்கள், இனி தானாக ஒரு நிரல்பாகத்தை எழுதி பாருங்கள். ஒரு சொல் விளையாட்டோ, அல்லது கணிப்பானையோ *(calculator)* எழுத முயன்று பார்க்கலாம். தானாக முயன்று பார்ப்பது நல்ல தொடக்கமாக அமையும்!

பயிற்சி

1. இந்த (கீழ் உள்ள) நிரல்பாகத்தின் சார்பு பெயர் என்ன ?

```
def multiply(num1,num2)

num1 * num2

end
```

2. இந்த நிரல்பாகத்தின் சார்பில் எத்தனை உள்ளீடுகள் உள்ளன ?

```
def meeting(place, time, day)

...

end
```

3. இந்த நிரல்பாகத்தின் சார்பு வெளியீடு என்ன ?

```
def calculus

numbers = (25 * 37) / 42

numbers / 12 * 25

“Programming is not math”

end
```

4. ஒரு பெயரை உள்ளீடாக கொள்ளும் நிரல்பாக சார்பை எழுதுங்கள். இந்த நிரல் வெளியீடு உங்களது உள்ளீடு வார்த்தையைக் கொண்டு

” சூப்பெராக உள்ளது!” என்று அளிக்க வேண்டும். இதற்கு ஏற்றவாறு நிரலை எழுதுங்கள் பார்க்கலாம்!

9. எண்வகை தரவமைப்புகள்

Enumerable தொகுதி என்பது நீங்கள் தேடல் செய்ய, வரிசைப்படுத்த, தேட மற்றும் தொகுப்புகளை கையாள அனுமதிக்கும் ஒரு ரூபி முறைகள் உள்ளமைக்கப்பட்ட ஒரு தொகுப்பு ஆகும். சேகரிப்புகள் இரண்டு வடிவங்களில் வரும் அவையாவன எண்ணிமஅடைவு மற்றும் தொர்புரு அணி என்பதை நினைவில் கொள்ளுங்கள். இந்த அத்தியாயத்தில் நான்கு அடிப்படைக் *Enumerable* தொகுதிகளை நாம் விபரிக்க உள்ளோம்:

-each (ஒவ்வொரு)

-each_with_index

-தேர்வு

-தொர்புரு அணி

1. *ஒவ்வொன்றாக (Each)*

'each' என்கின்ற முறை (method) உங்களது அடைவில் உள்ள ஒவ்வொரு உருப்படியிலும் கொடுக்கப்பட்ட நிரல் துண்டினை இயக்கும். உதாரணமாக, ஒரு பொம்மை களஞ்சியத்தை அணியில் கொண்டு நிரல் எழுதினால் இவ்வாறு 'each' என்ற முறையினை செயல்படுத்தலாம்: இதனை கொண்டு நமது பொம்மை அணியில் உள்ள உருப்படிகளின் பெயர்களை திரையில் இடலாம். அதாவது:

```
toys = [

"கார்",

"பந்து",

"வீரன் பொம்மை",
```

```
“வீட்டு விலங்கு”

]

toys.each { |toy| puts toy }
```

‘#’ தற்போது ஒவ்வொறு பொம்மையின் பெயரும் ‘toys’ என்ற அணியில் இருந்து திரையிடப்படும்:

```
கார்

பந்து

வீரன் பொம்மை

வீட்டு விலங்கு
```

each முறைக்குள்ளே நாம் என்ன செய்திருக்கின்றோம் என்பதைப் பார்க்கலாம். இந்த முறை, வரிசை வகுப்பை சார்த்திருக்கும் வரை இதனை எங்களுடைய பொம்மை வரிசையில் பயன்படுத்த

முடியும். பின்னர் எங்களுடைய *each* முறையினை நிரலினூடாக செயற்படுத்த முடியும். இங்கே நாங்கள் *puts* முறையினைப் பயன்படுத்துகின்றோம். எங்களுடைய *|pipes|* இல் நாங்கள் *toy* என்னும் தற்காலிக மாறியை வரையறுத்து *puts* என்னும் முறையினை ஒவ்வொரு *toy* குள்ளும் பயன்படுத்தும் நிரலாக்கத்தை கொண்டிருக்கின்றோம்.

எங்களுடைய *each* முறையால் வரிசை ஒன்றிலுள்ள எல்லா கூறுகளையும் பார்க்கமுடியும். இது வரிசயில் மீண்டும் மீண்டும் தேடல் என அழைக்கப்படும்(வரிசயில் மீண்டும் மீண்டும் தேடல் என்பது தொகுதியிலுள்ள ஒவ்வொரு *toy* இனையும் பார்ப்பதற்கான சிறந்த வழியாகும்) வரிசையிலுள்ள ஒவ்வொரு கூறுகளின் பெறுமதியையும் *|pipes|* இனுள்ளே நாங்கள் சேமிக்கலாம். அத்தோடு ஒவ்வொரு கூறுகளிலும் *puts* முறையினை

பயன்படுத்தலாம்.நாங்கள் கீழுள்ளவாறு நிரலை எழுதியுள்ளோம்.

```
toys.each do |toy|

puts toy

end
```

சுருளி அடைப்புக்களைப் பயன்படுத்தி *do and end* இனை ஒரு வரியில் எழுதமுடியும். நீங்கள் அதனை கட்டாயமாகப் பயன்படுத்த வேண்டியதில்லை. ஆனால் இந்தக் குறுகிய முறையினை தேவைப்படின் பயன்படுத்தலாம். அத்தோடு ரூபியின் மூலம் நிரலொன்றை எங்களுடைய *do end or { }* என்ற நிரலினுள்ளே செயற்படுத்த முடியும்.இப்போது நாங்கள் *each* முறையினை எவ்வாறு பயன்படுத்த வேண்டும் என்று பார்க்கலாம்.

எங்களுடைய சேகரிப்பில் ஒன்றுக்கு மேற்பட்ட *toy* இருப்பதைக் கற்பனை செய்து பாருங்கள்.(உங்களுடைய பெற்றோர் உங்களுக்கு ரூபியில் நிரல் எழுத கற்பித்திருந்தால் உங்களுடைய சேகரிப்பில் ஒன்றுக்கு மேற்பட்ட *toy* இருக்க வாய்ப்புள்ளது.நீங்கள் அதிஷ்டாசாலி)எங்களுடைய பொம்மைப் பெட்டியினை சிறப்பாக ஒழுங்குபடுத்த அல்லது காட்சிப்படுத்த எண்ணிம அடைவினைக் கட்டாயமாகப் பயன்படுத்த வேண்டும்.*each* இனைப் பயன்படுத்துவதன் மூலம் *toy* இனையும் *toy* இன் எண்ணிக்கையையும் காட்டிட முடியும்.

```
toys = {“car” => 1, “ball” => 3, “action figure” => 2,

“stuffed animal” => 8}

toys.each { |key, value| puts “#{key} => #{value}” }

car => 1
```

```
ball => 3

action figure => 2

stuffed animal => 8
```

எண்ணிம அடைவில் *each* முறையினைப் பயன்படுத்தும் போது வரிசையிலுள்ள ஒவ்வொரு கூறுகளும் *key and a value* உள்ளது என்பதை ரூபி தெரிந்து வைத்துள்ளது.*|pipes|* னுடைய *key and value pairs* இற்கு பெயரிட நீங்கள் எதை வேண்டுமானாலும் பயன்படுத்தலாம். உதாரணமாக *|x, y|*. ஆனால் *key and value* இனை அவற்றின் பெயரினால் அடையாளப் படுத்துவது இலகுவானது.பின்னர் நாங்கள் மீண்டும் *puts* முறையினை ஒவ்வொரு கூறுகளும் பயன்படுத்தி, எங்களுடைய *interpolation #{ }* மூலம் எங்களுடைய மாறிகளின் பெறுமதியினை சரம்களில் சேமிக்கமுடியும். அத்தோடு இறுதியாக *toy* களின் சாரம்களையும் *toy*

களின் எண்ணிக்கையையும் காட்சிப்படுத்த முடியும்.

2. ஒவ்வொன்றாக இடம்காட்டியுடன் *(Each_with_index)*

இந்த எண்ணிலடங்கா முறை ஒவ்வொன்றும் மற்றொன்றைப் போலவே செயல்படுகிறது, தவிர அது குறிப்பிட்ட அணிகளின் குறியீட்டை அதன் அழைப்புக்கு தகுந்தவாறு பிடிக்கிறது. ஒரு அணி என்பது தரவுகளை அல்லது தகவல்களை ஒரு குறியீட்டு (வரிசையாக்க அமைப்பு)முறையில் வரிசைப்படுத்த உதவ பயன்படுகின்றது என்பதை நினைவில் வைத்துக்கொள்ளுங்கள். ஒரு அணி மற்றும் ஐந்து உருப்படிகள் சேர்த்து *0* இலிருந்து *4* வரை ஐந்து குறியீடுகளை கொண்டுள்ளன, ஒவ்வொரு குறியீடும் ஒவ்வொரு தனிமத்தின் நிலையையும் குறிப்பிடுகிறது.

my_array = ['பந்து', 'மட்டை', 'தொப்பி']

my_array[2]

=> தொப்பி

my_array[0]

=> பந்து

மேலே உள்ள எடுத்துக்காட்டில், *my_array* இல் உள்ள ஒவ்வொரு உறுப்பையும் அதன் குறியீட்டால் அழைக்க முடியும். பந்து *(ball)* குறியீட்டு *0* இல் அமைந்துள்ளது, *Bat* குறியீடு *1* இல் உள்ளது, மற்றும் குறியீடு *2* இல் *Hat* அமைந்துள்ளது. நாங்கள் ஒவ்வொரு பிற உருப்படியையும் அச்சிட விரும்பினால், ஒவ்வொரு குறியீட்டு முறையையும் இந்த மாதிரி பயன்படுத்தலாம்:

my_array = ['பந்து', 'மட்டை', 'தொப்பி', 'சீருடை', 'சப்பாத்து']

```
my_array.each_with_index do |item, index|
if index % 2 == 0
puts item
end
end
```

modulo என்பது ஒரு எண்ணில் மீதமுள்ள அளவை (இடது பகுதிக்கு மேல்) பார்க்கும் ஒரு வழி என்பதை நினைவில் கொள்ளுங்கள். எனவே இங்கே, நாம் உருப்படிகள் குறியீட்டினை இரண்டால் வகுக்கும் போது பூஜ்ஜியத்தினை எஞ்சியதாக கொண்டிருக்கும்

ஒரு உருப்படியினை திரையில் உடசெலுத்துவோம். இவற்றில் *0, 2,* மற்றும் *4* ஆகிய குறியீடுகள் மீதமின்றி இரண்டினால் வகுப்படும். ஆகவே இக் குறியீடுகளுக்குரிய உருப்படிகளை மட்டுமே

உட்செலுத்துவோம் .

enumerable முறைமையை எளிமைப்படுத்த *even?* முறையும் சுருள் அடைப்புக்குறிகளும் பயன்படும்.

```
my_array.each_with_index do |item, index|

puts item if index.even?

end
```

```
பந்து #item at index 0

தொப்பி #item at index 2

சப்பாத்து #item at index 4
```

ரூபி இரட்டை எண் *(even?)*எனில் நம்மை இம் முறை மட்டுப்படுத்தப்பட்ட தொகுதி பகுதியை தவிர்க்க அனுமதிக்கிறது. இரட்டை எண் *(Even?)*, ஒரு இலக்கம் இரட்டை எண் எனில் அது *true* எனும்

கட்டளைக்கு திருப்பி அழைக்கப்படும். ஒவ்வொன்றும் ஒவ்வொரு படியிலும் குறியீட்டு முறையினைப் பயன்படுத்தி கணினியை எவ்வாறு செயல்படுத்துகிறது என்பதைப் பார்ப்பதற்கான வழி கீழே தரப்படுள்ளது.

```
my_array = [‘பந்து’, ‘மட்டை’, ‘தொப்பி’, ‘சீருடை’, ‘சப்பாத்து’]

my_array.each_with_index do |item, index|

puts item if index.even?

end
```

|‘ball’, 0| if 0 is even?, then puts ‘பந்து’

true => puts ‘பந்து’

|‘bat’, 1| if 1 is even?, then puts ‘மட்டை’

false

|'hat', 2| if 2 is even?, then puts 'தொப்பி'

true => puts 'தொப்பி'

|'uniform', 3| if 3 is even?, then puts 'சீருடை'

false

|'shoes', 4| if 4 is even?, then puts 'சப்பாத்து'

true => puts 'சப்பாத்து'

இந்த முறையானது அணி ஒன்றின் உருப்படி மற்றும் இதன் குறியீட்டில் இருந்து தொடங்குகிறது. பின்னர் நாம் வரிசைப்படுத்தியுள்ள *do* மற்றும் *end* காரணிகளின் இடையே குறித்த பகுதிக்குரிய கட்டளையைச் செயல்படுத்த வேண்டும். இந்த எடுத்துக்காட்டில் குறியீட்டு எண் இரட்டை எண் என்றால் மட்டுமே அது உருப்படியைப் பெயரிடும். இல்லையெனில், *if* கட்டளை தவறானது என திரும்பி

வரும் மற்றும் அந்தக் குறிப்பிட்ட கட்டளையைக்குரிய எந்த தகவலையும் ரூபி திரையில் காண்பிக்காது.

3. தேர்வு *(Selection)*

சரி, இது புரிந்து கொள்ள ஒரு சிறிய சிக்கலாக இருக்கலாம். ஆனால் எங்களை நம்புங்கள். இதக்ச் சில பயிற்சிகளின் மூலம் எளிதாக அடையலாம். *select* செயன்முறையில் அதை எப்படித் தெரிவுசெய்வது என்பதைத் தெரிந்து கொள்ளப் பார்க்கலாம். ஒரு தொகுதியின் கடந்து செல்லும் குறியீட்டைக் கண்டறிவதன் மூலம் *'Select'* வேலை செய்கின்றது,தொகுதி *true* என மதிப்பிடப்பட்டால் மட்டுமே *select* இலுள்ள கூறுகளை விடையாகத் தரும்.

```
$ [1,2,3,4].select { false }

=> []
```

```
$ [1,2,3,4].select { true }

=> [1, 2, 3, 4]
```

மேலே உள்ள எடுத்துக்காட்டில், தேர்ந்தெடுக்கப்பட்ட முறை வரிசை *(1,2,3* மற்றும் *4)* இல் உள்ள ஒவ்வொரு உறுப்புகளுடனும் பார்க்கப்படுகிறது, மேலும் எங்கள் தொகுதி உண்மையாக இருந்தால் அந்த உறுப்பு விடையாக கிடைக்கும் .எங்கள் தொகுதி தவறான பூலியன் என்பதால், *select* முறை உண்மை மதிப்பீடு இல்லை மற்றும் விடையாக எதனையும் தராது ஆனால் அது ஒரு வெற்று வரிசையைத் தரும். அடுத்த வரியில் நாம் தொகுதிக்கு *true* ஐ அனுப்புகின்றோம் மற்றும் *select* முறை வரிசையில் உள்ள ஒவ்வொரு உருப்படியையும் விடையாகத் தருகின்றது.

select முறையை எவ்வாறு பயன்படுத்துவது

மற்றும் மிகவும் பயனுள்ள வழியில் எவ்வாறு பயன்படுத்தப்படலாம் என்பதற்கான உதாரணம் இங்கே தரப்பட்டுள்ளது. பரீட்சை மதிப்பெண்களின் தொகுப்பைக் கற்பனை செய்து பாருங்கள், 80 புள்ளிகளுக்கு மேலாக மட்டுமே மதிப்பெண்களைத் தேர்ந்தெடுக்க வேண்டும்.

```
test_scores = [23,80,34,99,54,82,95,78,85]
test_scores.select do |score|
score > 80
end
=> [99, 82, 95, 85]
```

இங்கே வரிசையிலுள்ள ஒவ்வொரு கூறுகளையும் பார்ப்பதற்கு *select* இனை நாங்கள் பயன்படுத்துகின்றோம்.*score* என்னும் மாறியில்

இந்த மதிப்பு தற்காலிகமாக சேமிக்கப்படும்.பின்னர் ஒவ்வொரு மதிப்பெண்ணும் *80* ஐ விட அதிகமாக உள்ளதா என சோதிக்கப்படும்.பின்னர் *80* ஐ விட அதிகமான மதிப்பெண்கள் சேகரிக்கப்பட்டு வரிசைக்கு அனுப்பப்படும்.

```
test_results = {

"Bobby" => 80, "Jane" => 95,

"Adam" => 99, "Doug" => 65,

"Sue" => 89, "Kim" => 91

}

good_students = test_results.select do |student, score|

score > 80

end
```

இங்க எங்களிடம் மாணவர்களது பெயர் மற்றும் மதிப்பெண்களைக் கொண்ட *test_results* எனப்படும் எண்ணிம அடைவு உள்ளது.*80* மதிப்பெண்களுக்கு அதிகமாகப் பெற்ற மாணவர்களையும் அவர்களது மதிப்பெண்களையும் பெற *select* என்னும் முறையினை நாங்கள் பயன்படுத்த முடியும்.இதனை நாங்கள் செயற்படுத்தும் போது *good_students* என்ற எண்ணிம அடைவு இப்படி இருக்கும்.

$ good_students

=> {"Jane"=>95, "Adam"=>99, "Sue"=>89, "Kim"=>91}

வகுப்பிலுள்ள நூற்றுக்கணக்கான மாணவர்களை அவர்களது மதிப்பெண்களின் அடிப்படையில் வகைப்படுத்த ஒரு எளிய *select* முறை உபயோகப்படுவதைக் நீங்கள் கற்பனை

செய்து பாருங்கள். தரவுகளை வரிசைப்படுத்த, எண்ணிக்கையைப் பெற, வகைப்படுத்த மற்றும் ஒழுங்கமைக்க ரூபி பயன்படும். நூற்றுக்கணக்கான மாணவர்களை அவர்களது பெயர் மற்றும் மதிப்பெண்களின்அடிப்படையில் வகைப்படுத்த ஒரு மனிதனுக்குப் பல மணி நேரங்கள் எடுக்கும் ஆனால் ஒரு சிறந்த கணினி நிரல் உடனடியாக இதனைச் செய்து முடிக்கும்.

4. தொடர்புரு அணி *(map)*

அறுதியும் இறுதியுமாக தொர்புரு அணி*(map)* முறை பாருங்கள். நம் சேகரிப்புகளிலுள்ள கூறுகளை*(elements)* வேறுபட்ட முறைகள் எவ்வாறு பாதிக்கின்றன என்பதை இதுவரை பார்த்தோம்.தொர்புரு அணி முறை மூலம் குறிப்பிட்ட கூறுகளைத் தேர்ந்தெடுத்து ஒரே நேரத்தில் திருத்தலாம். உதாரணமாக

[“dog”, “cat”, “snake”, “mouse”].map do |animal| animal.capitalize end => [“Dog”, “Cat”, “Snake”, “Mouse”]

இங்கே நாம் ஒரு விலங்குகளின் அணியை வைத்திருக்கிறோம். அணியிலுள்ள ஒவ்வொரு விலங்குகளின் முதல் எழுத்தையும் ஆங்கில பெரிய எழுத்துக்களால் மாற்றீடு செய்ய ரூபியிலுள்ள தொர்புருஅணி*(map)* முறைமையை பயன்படுத்துவதோடு *capitalize* முறையையும் பயன்படுத்துகின்றோம் *.capitalize* முறைமை அணியிலுள்ள விலங்குகள் எல்லாவற்றினதும் முதல் எழுத்துக்களை ஆங்கில பெரிய எழுத்துக்களால் மாற்றீடு செய்யும்.

தொர்புரு அணி முக்கியமாக அணியிலுள்ள தரவை கண்டுபிடித்து கையாளுகிறது அல்லது அணியிலுள்ள கூறுகளை வரைபடமாக்குகிறது ரூபியில் தொர்புரு அணியுடன் ஒரு அணியை நிரந்தரமாக மாற்றுவதற்கு,

ஏற்கனவே உள்ள வரிசைக்கு ஒரு ஆச்சரிய குறியை சேர்க்கவும் .

animals = [“dog”,“cat”,“snake”,“mouse”] animals.map! { |animal| animal.capitalize } => [“Dog”, “Cat”, “Snake”, “Mouse”]

Animals => [“Dog”, “Cat”, “Snake”, “Mouse”]

இந்த நேரத்தில் நிரலாக்க மொழிகளில் இரண்டு அடிப்படை சேகரிப்புகளை நீங்கள் புரிந்துகொள்கிறீர்கள். *Enumerable* தொகுதியை கற்றுக்கொண்டதன் பின்னர் அணிகள் மற்றும் எண்ணிம அடைவு கொண்ட சேகரிப்புகளுக்கு பொதுவாக நான்கு முறைகள் பயன்படுத்தப்படும் அவை *each, each_with_index, select and map*

இப்போது சில உதாரணங்களோடு உங்களது அறிவை மேம்படுத்த சில பயிற்சிகளை

மேற்க்கொள்ளலாம்.

பயிற்சி

1. இந்த நிரல் துண்டு என்ன எண்களை வெளியீடு செய்யும்?

```
[1,2,3,4,5].each { |num| puts num if num.odd? }
```

2. கீழே உள்ள *‘each’* என்ற சார்பு என்ன வெளியீடு தரும் ?

```
“ThisdmakesdmoredsensedwithoutdD’s”.split(“d”).each {

|letter| puts letter

}
```

குறிப்பு : இங்கு சார்புகளை கோர்வையாக இணைக்கிறோம் *(‘method chaining’* என்று இதனை கூறுவார்கள்). முதலில் ஒரு சரம் அணி என்பதனை

உருவாக்கி பின்பு அந்த அணியில் *‘each’* என்ற செயல்பாடை இயக்குகிறோம். இதில் அணி மறைமுகமாகவே உள்ளது.

3. *‘select’* என்பது தேர்ந்தெடு வாக்கியம் இந்த எண் சுருக்கு?

   ```
   food = {
   “apple” => “fruit”,
   “carrot” => “vegetable”,
   “tomato” => “fruit”
   }
   food.select do |item, category|
   category == “vegetable”
   ```

```
end
```

4. கீழ் கொடுக்க பட்ட தொர்புறு சார்பில் *(map)* என்ன மதிப்பு கணிக்கப்பட்டும்*?*

```
numbers = [1,2,3,4,5]

numbers.map { |num| num * 5 }
```

5. கேள்வி *4-*இல் கணிக்கப்பட்ட மதிப்பு என்ன *?*

10. கோப்புகள்

முழு புரிதல்

இதுவரை, நாம் ரூபி மொழியின் அடிப்படைகள் பலவற்றை கற்றுக்கொண்டோம். எந்த ஒரு நிரல் மொழிக்கும், நாம் கற்றுக்கொண்ட சரங்கள், எண்கள், மாறிலிகள், தரவமைப்புகள் மற்றும் நிரல்பாகங்கள் மிக மிக முக்கியமானவை. இவைகள் தாம் நிரல்களின் சின்னஞ்சிறு கட்டமைப்புகள் ஆகும். ரூபியையோ வெறு நிரல் மொழிகளையோ கற்றுக் கொள்வதற்கு பிற்காலத்தில் இவை உங்களுக்கு பயன்படும்.

நீங்கள் நிரல்கள் எழுத துவங்குவதற்கு முன் மேலும் ஒன்றை கற்க வேண்டி உள்ளது. இதுவரை நாம் சரங்களையும் எண்களையும் திருத்தவும், மாற்றங்கள் செய்யவும் முனையத்தை தான் பயன்படுத்தினோம்.

ஆனால், கோப்பில் நாம் நேரடியாக மாற்றம் செய்ய வேண்டுமாயின்?

கொப்புகளோடு நாம் தொடர்பு ஏர்படுத்திக் கொள்ள, ரூபியின் *File* வகுப்பை *(File Class)* நாம் உபயோகிக்க வேண்டும். கோப்புகளை உருவாக்கி, திறந்து, தகவல்களை சேமிப்பதற்கான நிரல் பின்வருமாறு தோற்றமளிக்கும்.

```
the_file = File.open(“my_file.txt”, “w”)

the_file.puts “கோப்பின் முதல் வரி.”

the_file.close
```

மேலுள்ள எடுத்துக்காட்டில், *File* வகுப்பையும் *open* நிரல்பாகத்தையும் பயன்படுத்தி அதற்கு இரு உள்ளீடுகளை அனுப்பியுள்ளோம். அந்த உள்ளீடுகளில் முதலாவதானது, *my_file.txt*, இது

நாம் உருவாக்கப்போகும் கோப்பின் பெயர் ஆகும். இரண்டாவது உள்ளீடு, *w*. இது, நாம் கோப்பில் தகவல்களை எழுத போகிறோம் என்னும் செய்தியை ரூபிக்கு தெரிவிக்கும். அடுத்த வரியில் உள்ள *puts*, நாம் அளிக்கும் சரங்களை கோப்பினுள் எழுதும். இறுதியாக, *close* நிரல்பாகத்தை பயன்படுத்தி நாம் கோப்பை மூடிவிடுகிறோம்.

நீங்கள் முனையத்தை திறந்து ரூபியின் *'Interactive Shell (IRB)'*-ற்கு சென்றால், மேல் குறிப்பிட்டுள்ள கட்டளைகள் அனைத்தையும் இயக்கலாம். *IRB*-ல் உள்ளபோது, உங்கள் கோப்பை காண்பதற்கு *exit*என்று தட்டச்சு செய்து வெளியேறி விட்டு, பின்பு *ls* என தட்டச்சு செய்யுங்கள். இது உங்கள் அனைத்து கோப்புகளையும் பட்டியலிட்டு காண்பிக்கும். அதில் *my_file.txt* இடம் பெற்றிருப்பதை நீங்கள் காண்பீர்கள். இனி *open my_file.txt* என தட்டச்சு செய்தால், உங்கள்

கோப்பு திறக்கும்.

இனி இந்தக் கோப்பில் நாம் மாற்றம் செய்ய வேண்டுமானால், மேல் குறிப்பிட்டுள்ள அதே நிரலில், கோப்பை திறப்பதற்கான *open* கட்டளையின் இரண்டாம் உள்ளீடாக *'a'* பயன்படுத்த வேண்டும். இதற்கு பதிலாக நாம் *w* பயன்படுத்தினால், கோப்பில் உள்ள தகவல்களை அழிக்கப்பட்டு முதலிலிருந்து எழுதப்படும். நாம் *'a'* பயன்படுத்துவதன் மூலம், கோப்பில் முன்னிருந்த தகவல்களுக்கு கீழ் புதிய தகவல்கள் சேமிக்கப்படும்.

நாம் கோப்பில் உள்ள தகவக்களை படிக்க வேண்டுமானால், அதற்கு இரு வழிமுறைகள் உள்ளன. முதலாவது நாம் கோப்பில் உள்ள அனைத்து தகவல்களையும் ஒருசேர படிக்கலாம், அல்லது ஒவ்வொரு வரியாக படிக்கலாம். நாம் பின்வருமாறு *'r'* உள்ளீடை பயன்படுத்தி கோப்புகளை படிக்கலாம்.

‘r’ என்பது நாம் கோப்பை படிக்க போகிறோம் என்பதற்கான குறியீடாகும்.

```
file = File.open(“our_file.txt”, “r”)

entire_file = file.read

puts entire_file

கோப்பின் முதல் வரி.

இது இரண்டாம் வரி.

மேலும், இது மூன்றாம் வரி ஆகும்.

=> nil
```

‘r’ பயன்படுத்துவதன் மூலம் நாம் கோப்பை *read-only mode* அதாவது படிக்க மட்டுமே வகை செய்யும் விதமாக திறக்கிறோம். இவ்விதமாக திறந்த கோப்பில் நாம் எழுத முற்பட்டால், அச்செயல் தோல்வியடையும்.

ஆனால், நாம் *read* நிரல்பாகத்தை அழைத்து கோப்பின் முழு தகவலையும் ஒரு மாறிலியில் சேமித்து, பின்னர் திரையில் வெளியீடு செய்யலாம்.

ஒவ்வொரு வரியாக படிக்க வேண்டுமென்றால், *readlines* நிரல்பாகத்தை பயன்படுத்தலாம்.

```
File.open(“our_file.txt”).readlines.each do |line|

puts line

end
```

இரு வழிமுறைகளிலும், கோப்பு படிக்கப்பட்டு ஒவ்வொரு வரியாக வெளியீடு செய்யப்படும். ஆனாலும், அவ்விரு நிரல்பாகங்களின் வெளியீட்டிலும் பெரும் வித்தியாசம் உள்ளது. *read*நிரல்பாகத்தை பயன்படுத்தும் போது, கோப்பில் உள்ள முழு தகவலும் படிக்கப்பட்டு *nil* வெளியீடு செய்யப்படுகிறது. ஆனால்,

readlines நிரல்பாகத்தை உபயோகித்தால் ஒவ்வொரு வரியும் ஓர் அணியில் சேமிக்கப்பட்டு, இறுதியாக அந்த அணி வெளியீடு செய்யப்படுகிறது.

அது பின்வருமாறு தோற்றமளிக்கும்.

```
file_array = File.open(“our_file.txt”).readlines

=> [“கோப்பின் முதல் வரி.\n”,

“இது இரண்டாம் வரி.\n”,

“மேலும், இது மூன்றாம் வரி ஆகும்.”]
```

நாம் கோப்பை படிக்க, File.open-ஆல் readlines நிரல்பாகத்தை அழைக்கிறோம். கோப்பு முழுவதுமாக படித்து முடிக்கப்படும் வரை அனைத்து வரிகளும் ஓர் அணியில் சேர்க்கப்படுகிறது. நம் சரங்களுக்கு இடையில் `\n` எனும் புதுவரி எழுத்தை கவனித்தீர்களா? நாம் each நிரல்பாகத்தின் உதவியுடன், ஒரு தொகுதிக்குள்

ஒவ்வொரு வரியையும் செலுத்தி *chomp* பயன்படுத்தி அந்தப் புதுவரிகளை எளிதாக நீக்கிவிடலாம்.

```
file_array = File.open(“our_file.txt”).readlines

.each { |line| line.chomp! }

=> [“கோப்பின் முதல் வரி.”,

“இது இரண்டாம் வரி.”,

“மேலும், இது மூன்றாம் வரி ஆகும்.”]
```

அற்புதம்! கோப்புகளை திறப்பதற்கும், அதில் எழுதுவதற்கும், அவற்றை படிப்பதற்கும் நீங்கள் கற்றுக் கொண்டீர்கள்! மேலும், *close* நிரல்பாகத்தை பயன்படுத்தி கோப்புகளை மூடுவதற்கு மறந்து விடாதீர்கள். கோப்புகளை திறத்துவிட்டு அப்படியே விட்டுவிடுவது நல்லதல்ல, ஏனெனில் கோப்பை மூடும்போது தான் கணினி நாம் எழுதிய

தகவல்களை அதில் சேமிக்கும். இதற்கு மேலாக, திறந்து வைக்கப்பட்ட கோப்புகள் கணினியில் நினைவு திறத்தில் அழுத்தத்தை ஏற்படுத்தி அதன் செயல்பாட்டை குறைத்துவிடலாம்.

முடிவு

வாழ்த்துக்கள்!

நீங்கள் கலாசலாக சிறுவகர்ளுக்கென எழுதப்பட்ட முதல் ரூபி புத்தகத்தை படித்து முடித்துவிட்டீர்கள்! சபாஷ்! அடுத்து வங்கியில் லோன் எடுத்து பெங்களூரில் ஒரு நிறுவனத்தை அரமித்துவிடவேண்டியது தானே - இல்லை ஒரு வேளை அதையும் பண்ணலாம்.

சற்று தீவிரமாக யோசித்தால் நிரலாக்கல் என்பதில் என்றும் கற்றுக்கொள்ள கடலளவு உள்ளது. உங்கள் முயற்சி இந்தக் கடலில் கால் நனைத்தது போலவே அமையும். உங்களுக்கு நேரம் கிடைக்கையில் நிரலாக்கல் பற்றி கணினியில் கற்று கொள்ளுங்கள் - மற்ற நிரலாக்கல் மொழிகள் *Python*, எழில், *C*, *C++*, *Java* என்றும் கற்று கொள்ளலாம் - சில பயிற்சி

படங்களையும் முயலுங்கள், அதில் கற்றதில் நிரல்களை செயலிகளை உருவாக்குங்கள், உங்கள் திறமையை மேம்படுத்துங்கள்.

வாழ்க நிரலாக்கல். உங்கள் பயனம் வெற்றிகரமாக அமைய வாழ்த்துக்கள்!

11. பின் இணைப்புகள்

Appendix

இ.1 பயிற்சி கேள்விகளின் விடைகள்

பாடம் 2 - எண்கள்

1. *2 + 3 + 5 => 10*

2. *10 - 3 => 7*

3. *9 / 3 => 3*

4. *4 * 2 => 8*

5. *4 ** 2 => 16*

6. இந்தக் கணக்கின் *11 % 5* விடை என்ன? *=> 1*

11 ஐ *5* இன் இரண்டு மடக்கில் அடக்கலாம். *5* இன் இரண்டு மடங்கு *10.* ஆகவே *11 - 10* விடையாக *1* இனைத் தரும்

7. *14 % 3* விடை என்ன? *=> 2*

8. ஒரு மந்திரவாதி தன் ஒவ்வொரு கையிலும் இரண்டு இலக்கங்களை கொண்டுள்ளான் (ஒன்று இரட்டை எண் மற்றது ஒற்றை எண்). அவர் தன் கைகளை திறந்து உங்களுக்கு அவற்றை காட்டப்போவது இல்லை. ஆனால் அவர் உங்களை *modulo* முறையை பாவிக்க அனுமதிக்கிறார். உங்களுடைய பணி என்னவென்றால் எந்தக் கையில் இரட்டை எண் மற்றும் ஒற்றை எண் உள்ளது என்பதை

கண்டு பிடிப்பதாகும்.

ஒவ்வொரு இலக்கத்தையும் *x* என குறியிட்டால் நாம் *modulo* இனை பயன்படுத்தலாம். *x % 2 == 0* உண்மை*(true)* என்றால் இது ஒரு இரட்டை எண். *x % 2 == 0* என்பது தவறு என்றால் இது ஒரு ஒற்றை எண்.

பாடம் 3 - சரங்கள்

1. *puts* "இதன் விடை" + " என்னவாக " + "வெளிப்படும் ?" இதன் விடை என்னவாக வெளிப்படும்? *=> nil*

மேல்கொடுக்கப்பட்ட *put* முறையானது சரங்களை ஒன்றிணைத்து திரையில் விடையை காண்பித்து வெறுமையாக திரும்பிகிறது

2. *"This string minus" - "That string"*

NoMethodError: undefined method '-' for

"This string minus":String

சரங்களுக்கு என எந்த '-' முறையும்*(method)* இல்லை. எனவே இந்தச் செயல்பாடு செயற்படுத்தப் பட முடியாது. ஞாபகத்தில் வைத்திருக்கவும் – *(minus)*

என்பது இலக்கங்களுக்கு *(numbers)* மட்டுமே!

```
“1234.55”.to_i => 1234

“1234.55”.to_f => 1234.55

“Not a number”.to_i => 0

puts “1\n 2 \n 3\n”

1

2

3

=> nil
```

3. இலக்கங்களாக*(numbers)* குறிப்பிட முடியாத வெளியீடான பூஜியத்தை சரத்தின் மேல் தந்தும் ஏன் *‘to integer’ (to_i)* முறையானது பயனுள்ளதாக

கருதப்படுகிறது?

"to_i" இலக்கங்கள் இல்லாமல் கூட இன்னும் சரங்களால் அழைக்கப்படலாம், மேலும் இது பூஜியத்தை வெளியீடாக தரும். இது குறித்த வெற்றிகரமான அழைப்பு மற்றும் பூச்சிய மதிப்பால் சரத்தில் பூசிய இலக்கங்களை குறித்து காட்ட முடியும்.

4. *Length method* என்ன செய்கின்றது என நினைக்கிறீர்கள்? *"Count".length => 5* சரத்திலுள்ள எழுத்துக்களின் எண்ணிக்கையை அல்லது உருப்படிகளை கணக்கிடுகிறது.

5. *split method* பற்றி என்ன நினைக்கின்றீர்கள்? *"Count".split("") => ["C", "o", "u", "n", "t"]* *'split'* முறையினை இடைவெளிகளின்றி *("")* பயன்படுத்தும் போது சரம் ஒன்றின் ஒவ்வொரு எழுத்துக்களையும் தனித்தனியாக வரிசை ஒன்றில்

பெற முடியும்.

6. துண்டுகள் (slice) என்ன செய்கின்றன என நினைகிறீர்கள்?

“Count”.slice(2) => “u”

Slice முறையினூடாக அனுப்பப்படும் கட்டளைக்கு அமைவாக சரமொன்றின் எழுத்துக்களின் குறியீட்டின் அடிப்படையில் எழுத்து தேர்வு செய்யப்படும். இங்கே குறியீடு 2 ஆக்கவுள்ள எழுத்து U ஆகும்.

string: [“C”, “o”, “u”, “n”, “t”] indexes: [0 , 1 , 2 , 3 , 4]

பாடம் 4 - மாறிகள்

1. *x* என்னும் மாறியில் *54 / 3* என்ற பெறுமதி சேமித்து வைக்கப்பட்டிருந்தால் *x* இன் பெறுமதி என்ன?*x = 54 / 3 => 18*

2. மாறி *x* இன் பெறுமதியைப் புதிய மாறி *y* இற்கு வழங்குக ,பின்னர் மாறி *x* இனை *3* ஆல் வகுக்குக. இப்போது மாறி *x* இன் பெறுமதி என்ன? *y* இன் பெறுமதி என்ன?

y = x

=> 18 # y is assigned the x value, 18

x = x / 3 # or written as x /= 3

=> 6 # x is now the value of 18 / 3

3. நாம் நம் மாறிகளில்சில கணித செயன் முறைகளை செய்து பார்த்தால் என்ன? X க்கு பெறுமானம் 12 ஐ கொடுபோம். இதனை இப்போது 3 ஆல் வகுத்தால் பெறுமானம் என்னவாக இருக்கும்?

$x = 12$

$x / 3$

=> 4

4. இப்போது x இன் மதிப்பு என்ன? => 12 நாங்கள் x இனை 3ஆல் வகுக்கும் போது வரும் விளைவை சேமித்து வைக்கவில்லை. அதனால் X இனுடைய மதிப்பு மாறாமல் முன்பு வழங்கப்பட்ட அதே 12 என்ற மதிப்பையே கொண்டிருக்கும்.

பாடம் 5 - If and Else

1. *Is 3 > 5 ? => false*
2. *3 < 5 => true*
3. *5 == 5 => true*
4. *10 >= 10 => true*
5. *10 <= 12 => true*
6. *10 != 10 => false*
7. *10.object_id == 10.object_id => true*

   இந்தச் சரங்களைப் பற்றி*? 8) “dog” == “cat” => false*

9. *“cat” == “cat” => true*
10.

```
“cat”.object_id == “cat”.object_id

=> false
```

12. Numbers have object id's that don't change.

ஒவ்வொரு முறையும் சரம் உருவாக்கபடுகிறது. எப்படியோ அந்த சரத்துக்குரிய ஒரு புதிய *object_id* உருவாக்கப்பட்டுவிடுகிறது.

1. *நேர மாறியின் தற்போதைய மதிப்பு என்ன?? => 8*

எமது இறுதி விளைவு 7 ஆக இருந்தாலும் அடுத்த குறியீட்டு தொகுதி செயற்படுத்தபட்டது. (current_time = current_time + 1), மேலும் எமது மாறிகள் 1 ஆல் அதிகரித்து 8 க்கு சமமாக மாறியுள்ளது.

பாடம் 7 - Collections

1)மந்திரவாதியோருவரின் மந்திர கைப்பைக்கு ஒரு எண்ணிம அடைவை உருவாக்கவும். நாம் பையினுள் 3 தவளை, 5 மூலிகைகள் , மற்றும் சுருள்களை போடுவோம். பைக்கு ஒரு எண்ணிம அடைவை எழுதுங்கள்.இப்போது உங்களுடைய எண்ணிம அடைவு எவ்வாறு காட்சியளிக்கும்?

```
bag = {

“frogs” => 3,

“herbs” => 5,

“scrolls” => 10

}
```

*2)*ஞாபகத்தில் வைத்திருக்கவும், எண்ணிம அடைவானது இலக்கங்களை மட்டும் அன்றி எந்தவொரு தரவுகளின் முக்கிய மதிப்பு சோடிகளை உள்ளடக்கி இருக்கும். மந்திரவாதியின் மந்திரத்தையும் அதன விளைவையும் சேர்ப்போம் (இது மந்திரவாதியின் நினைவில் ஒருபோதும் இருக்காது). மந்திரவாதியின் பைக்குள் *"shazam"* எனும் மந்திர சொல்லை சேர்க்கவும். மேலும் இந்த மதிப்பு எந்தப் பொருளையும் தவளையாக மாற்றும். இப்போது எமது பையினுள் என்ன உள்ளது?

```
bag["shazam"] = "turns subject into a frog"

bag

=> {"frogs" => 3,

"herbs" => 5,
```

```
"scrolls" => 10,

"shazam" => "turns subject into a frog"}
```

3)எங்களுடைய மந்திரவாதி இனி யாரையும் எப்போதும் தவளையாக மாற்ற விரும்பவில்லை. ஆகவே நாம் எவ்வாறு "shazam" எனும் மந்திரத்தை மந்திரவாதியின் பையினுள் இருந்து நீக்கலாம்?

```
bag.delete("shazam")

=> "turns subject into a frog"
```

4)எமது மந்திரவாதி சமீபத்தில் மூன்று வெவ்வேறு வகையான போஷன் வாங்கியுள்ளார். 4 ஆரஞ் போஷன், 5 நீல போஷன் மற்றும் 7 சிவப்பு போஷன். நாம் இப்போது எமது பைக்குள் எவ்வாறு வேறு வகையான எண்ணிம அடைவு போஷனை சேர்க்கலாம்?

குறிப்பு : collection இற்குள் collections களை நாம்

வைத்திருக்கலாம் என்பதை நினைவில் வைத்திருங்கள்.

```
bag[“potions”] = {
“orange” => 4,
“blue” => 5,
“red” => 10
}
bag
=> { “frogs”=>3, “herbs”=>5,
“scrolls”=>10,
“potions”=> {
“orange”=>3,
```

```
    "blue"=>5,
    "red"=>10
  }
}
```

5)இப்போது எமது பைக்குள் நான்கு சாவிகள் உள்ளன (தவளை, மூலிகை, சுருள், போஷன்). நாம் இவற்றை பாவித்து எமது தரவுகளை அணுகலாம். ஒரு புதிய மந்திரத்தை உருவாக்க எமது மந்திரவாதிக்கு 2 தவளைகள், 3 மூலிகைகள், 1 சுருள் மற்றும் 2 நீல போஷன் தேவைப்படுகிறது. இவற்றை எவ்வாறு எமது எண்ணிம அடைவில் இருந்து நாம் நீக்கலாம்?

குறிப்பு : நாங்கள் சாவிக உரிய மதிப்புக்களை வழங்க முடிவதோடு நீக்கப்பட்ட உருப்படிகளை கழிக்கவும்

```
bag[“frogs”] = bag[“frogs”] - 2
=> 1
bag[“herbs”] = bag[“herbs”] - 3
=> 2
bag[“scrolls”] = bag[“scrolls”] - 1
=> 9
bag[“potions”][“blue”] = bag[“potions”][“blue”] - 2
=> 3
bag
=> {
“frogs” => 1,
```

```
“herbs” => 2,

“scrolls” => 9,

“potions” => {

“orange” => 4,

“blue” => 3,

“red” => 10

}

}
```

பாடம் 8 - Methods

1)கீழ் கொடுக்கபட்டுள்ள முறையின் (method) பெயர் என்ன?

```
def multiply(num1,num2)

num1 * num2

end

=> multiply
```

2)எத்தனை வாதங்களை இந்த முறை கொண்டுள்ளது?

```
def meeting(place, time, day)

...
```

```
end

=> 3
```

3)இந்த முறை எதை வெளியீடு செய்யும்? (இதன் வெளியீடுகள் என்ன?)

```
def calculus

numbers = (25 * 37) / 42

numbers / 12 * 25

“Programming is not math”

end

=> “Programming is not math”
```

Ruby methods return their last line by default

4)வாதமாக எடுத்துக்கொள்ள கூடிய ஒரு

வார்த்தையைக் கொண்ட முறையை எழுதுக. இந்த முறையை பாவித்து அந்த வார்த்தையை வெளியீடாகக் கிடைக்குமாறு செய்யவும், மீளும் சரம் *“is awesome!”* என வர வேண்டும்.

```
def awesomify(word)

word + ” is awesome!”

end
```

பாடம் 9 - Enumerables

1)எந்த இலக்கங்கள் பின்வரும் தொகுதியின் வெளியீடுகளாக அமையும்?

```
[1,2,3,4,5].each { |num| puts num if num.odd? }

1

3

5

=> [1, 2, 3, 4, 5]
```

2)இந்தச் சரத்திலிருந்து ஒவ்வொரு முறைக்கான வெளியீடு என்னவாக இருக்கும்?

```
“ThisdmakesdmoredsensedwithoutdD’s”.split(“d”).each {
```

```
|letter| puts letter
}
This
makes
more
sense
without
D’s
=> [“This”, “makes”, “more”, “sense”, “without”, “D’s”]
```

3)தேர்வு முறையானது எண்ணிம அடைவுக்கு என்ன செய்கிறது?

```
food = {
```

```
“apple” => “fruit”,
“carrot” => “vegetable”,
“tomato” => “fruit”
}
food.select do |item, category|
category == “vegetable”
end
=> {“carrot”=>“vegetable”}
```

4)இங்கு தொர்புரு அணி (map) என்ன செய்யும்?

```
numbers = [1,2,3,4,5]
numbers.map { |num| num * 5 }
```

=> [5, 10, 15, 20, 25]

5)இப்போது இந்த இலக்கங்களின் பெறுமானம் என்ன?

=> [1, 2, 3, 4, 5]

13. இ.2 தமிழ்க் கணிமைச் சொற்கள்

தமிழ்-ஆங்கிலம் கணிமை சொற்கள் பட்டியலை இங்கு காணலாம்.

1. ரூபி = *ruby*
2. தரவு = *data*
3. தொர்புரு அணி /(அல்லது) எண்ணிம அடைவு = *map/hash*
4. அணி = *array*
5. பட்டியல் = *list*
6. முனை-ஓரம் அடைவு = *graph*
7. தரவமைப்பு = *data structure*

8. தகவல் = *information*

9. கணினி = *computer*

10. சரம் = *string*

11. முழுஎண் / முழுயெண் = *integer*

12. செயல்பாடு / செயற்பாடு = *operation*

13. செயற்குறி = *operator*

14. ஏரணம் / தருக்கம் = *logic*

15. செயலுருபு = *parameter*

16. மதிப்பு = *value*

17. மதிப்பிலி = *null*

18. வெற்று = *empty*

19. மாறி = *variable*

20. மாறிலி = *constant*

21. மாற்றுறா = *immutable*

22. மாற்றுறா வகை = *immutable type*

23. மாற்றுறாமை = *immutability*

24. மாற்றறு = *readonly*

25. தேக்கு = *store*

26. தேக்கம் = *storage*

27. சேமி = *save*

28. இயக்கு = *execute*

29. முகவரி = *address*

30. நினைவகம் = *memory*

31. குறியுரு / வரியுரு = *character*

32. குறிப்பெயர் = *identifier*

33. நிலையுரு = *literal*

34. குறிச்சொல் = *keyword*

35. வில்லை - டோக்கன் = *token*

36. சிறப்புசொல் = *reserved word*

14. இ.3 தொடர்புகளுக்கு

ஏதேனும் பின்னூட்டல் அல்லது வினா தொடர்பாக என்னைத் தொடர்புகொள்ள விரும்பினால் நீங்கள் என்னை *‘mrdougwright’* என்பதை பயன்படுத்தி தொடர்பு கொள்ளலாம்.

GitHub > @mrdougwright

மின்னஞ்சல் > *mrdougwright@gmail.com*

மொழிபெயர்ப்பு பற்றி

எழில் மொழி அறக்கட்டளை இந்த நூலை மொழி பெயர்த்தது. பிழைகள் மற்றும் கருத்துக்களுக்கு இங்குள்ள *ezhillang@gmail.com* அஞ்சலில் தொடர்பு கொள்ளலாம்.

பங்களித்தவர்கள் பற்றி:

1. திரு. விமலன் குமரகுலசிங்கம், இலங்கை
2. திரு. முத்து அண்ணாமலை, அமெரிக்கா
3. பயிலகம் திரு. முத்து, சென்னை, இந்தியா
4. திரு. கருத்தன், இந்தியா
5. திரு. ஜோபைன், இந்தியா

15. இந்தப் புத்தகம் RubyKin.com என்ற ஆங்கிலப் புத்தகத்தின் தமிழ் மொழியாக்கம்

இந்தப் புத்தகம் மாணவர்கள் நிரலாக்கம் பற்றி அறிந்து கொள்ள துணைக் கையேடாக இருக்கும். பெரியவர்களும் இதனைப் பயன்படுத்தலாம்.

பத்தே அத்தியாயங்களில் எளிமையான முறையில் கணினி நிரலாக்க மொழிகளில் உள்ள அடிப்படைக் கோட்பாடுகளையும் நீங்கள் கற்றுக் கொள்ளலாம்; நிரலாக்க அடிப்படைகளான மாறிலி *(Constant)*, மாறி *(variable)*, சரம்*(string)*, எண்கள்*(numbers)*, சார்புகள்/நிரல்பாகங்கள்*(functions/methods)*, தரவமைப்புகள் *(collections and data structures)*

போன்றவற்றையும் நீங்கள் கற்றுக்கொள்ளலாம்.

இந்தப் புத்தகத்தைப் படித்து முடித்ததும் நீங்கள் மேற்கண்ட கணிணிக்கோட்பாடுகளையும் பயன்பாட்டையும் அறிந்துகொள்வீர்கள் என்று நம்புகிறேன்.

16. வாருங்கள், இந்தப் பயணத்தைத் தொடரலாம்!

உரிமம்

படைப்பாக்கப் பொதும உரிமம் வழியில் உருவாக்கியும் மொழிபெயர்க்ப்பட்டது.

இந்த ஆக்க வேலையானது படைப்பாக்கப் பொதும உரிமம் என்ற வணிக நோக்கம் இல்லாத பன்னாட்டு உரிமையின் கீழ் வெளியிடப்பட்டுள்ளது. எந்தவொரு நோக்கத்திற்காகவும் படியெடுக்க, விநியோகிக்க, காட்சிப்படுத்த, மாற்றம் செய்ய, மற்றும் பயன்படுத்துவதற்கான உரிமையை இங்கு வழங்கியுள்ளது. வணிக நோக்கங்களுக்காக அனுமதி தேவை எனில் எம்மிடமிருந்து பெற்றுக்கொள்ள,

17. நூல் வடிவமைப்பு

வெளியீடு: ஜூன் 23, 2019. கலிபோனியா, வட அமெரிக்கா.

எழுத்துரு: தமிழ் சங்கம் எம். என்.

மென்பொருள்: ஒப்பன் ஆபீஸ், கூகிள் ஆபீஸ், மற்றும் ஒப்பன் தமிழ்.