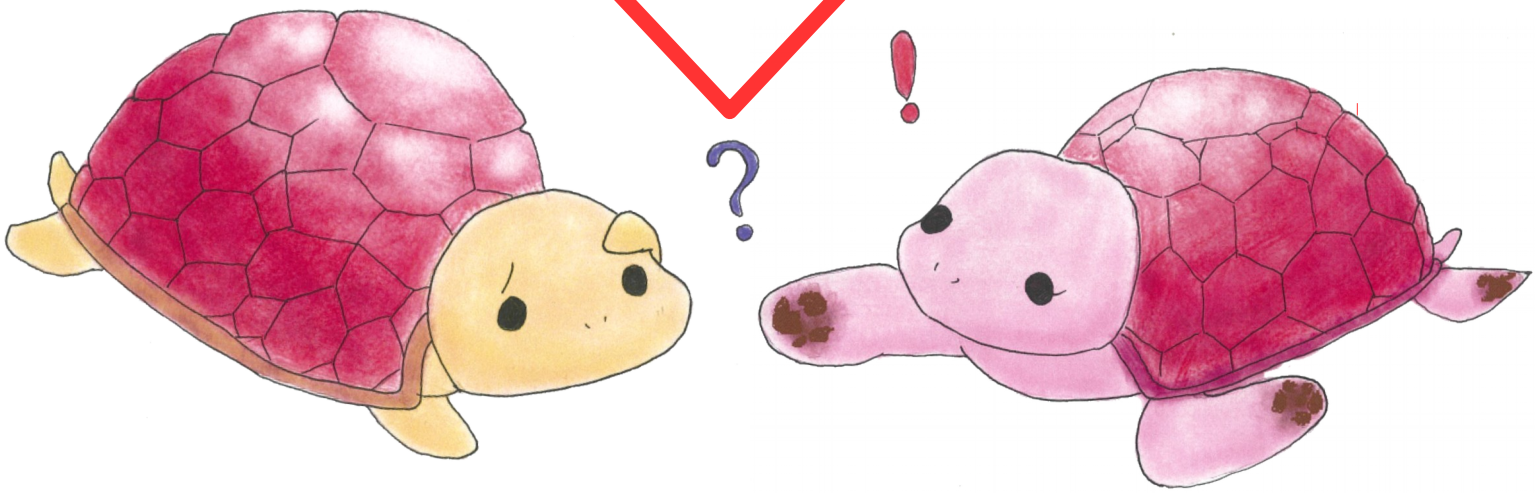
プログラミングで
考える力、伝える力を身につけよう!

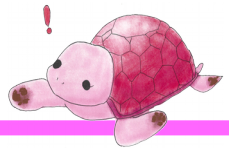
Ruby

てらこ・た



もくじ

1．みんなへのメッセージ

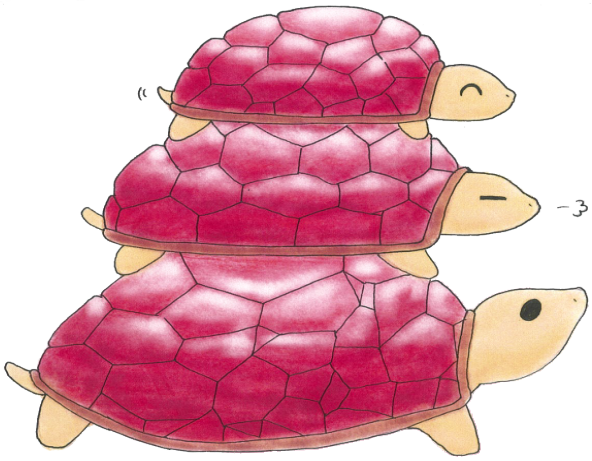


私達は、PC(パーソナルコンピュータ)やゲーム機、スマートフォンに囲まれた社会環境で育ち、このようなデジタル機器を使いこなしています。たとえば、ゲームやチャット、そしてインターネットでの情報収集などです。今やこんなの、あたりまえですよね。

しかし、ここには問題があります。多くの人は、ただデジタルメディアを使っているだけで、創造力を発揮しているわけではありません。ゲームやアニメーション、シミュレーションに触れて満足しているだけで、自分自身の作品をつくり出しているわけではありません。

「Ruby てらこった」ではプログラミングを通して、あなたの考えていることを形にすることができます。あなた自身のゲームや物語、社会に貢献できるプログラムをつくることができるのです。この学びの過程では、創造的に考え、それを伝える技を学ぶことができます。これは社会にでるためのすばらしいスタートです！

作品を創ったら、ぜひ周りのみんなに紹介しましょう！



user name
password

2．Hello！プログラミングの世界へ

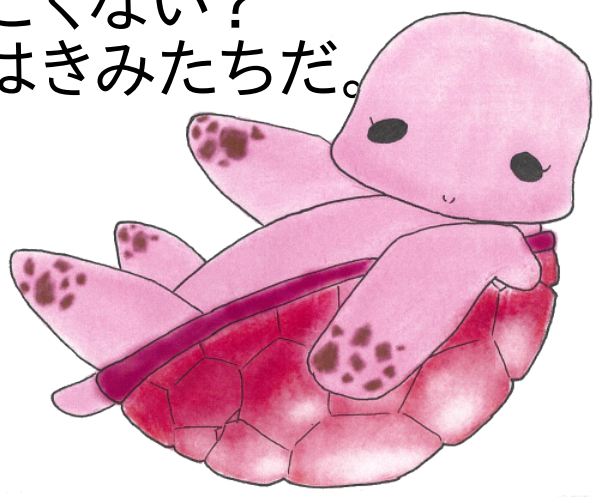
ようこそ、プログラミングの世界へ。

ここでは、プログラミングとはなにか、Ruby とはなにかについて簡単に紹介します。プログラミングとは、コンピュータに「これこれを、こうやって動いてほしい！」と伝えることです。そして、プログラミングによって作られるものがプログラムです。

Ruby はプログラムをつくるためのプログラミング言語です。周りにあるものをのぞいてみてください。どんなものでも最初からそこにあるわけではないです。誰かそれをつくった人がいます。そして、それをつくるためには材料や道具が必要です。図工の時間を思い出してみましょう。みんなが何かを作るときや絵を描くとき、ノリやハサミ、絵の具、筆で作り上げたと思います。

Ruby はプログラムをつくるための道具です。プログラムを作ることで、ゲームをつくることができます。なんでもしてくれるコンピュータやお絵かきソフトだってつくれるかもしれません。なんでもできます。

これってすごくない？
そしてつくるのはきみたちだ。



さあ、プログラミングをはじめよう！

3．学習の手引き

プログラミングをはじめる前に、学習の手引きを確認しましょう！

てらこったの授業では、さまざまな記号や文字、もしくはそれらを組み合わせたものが出てきます。例えば、％や“ また、C-1 や C-2 などです。少し難しいように見えますが、安心してください！
覚えようとしなくても、教科書を読みながら学ぶことができます。

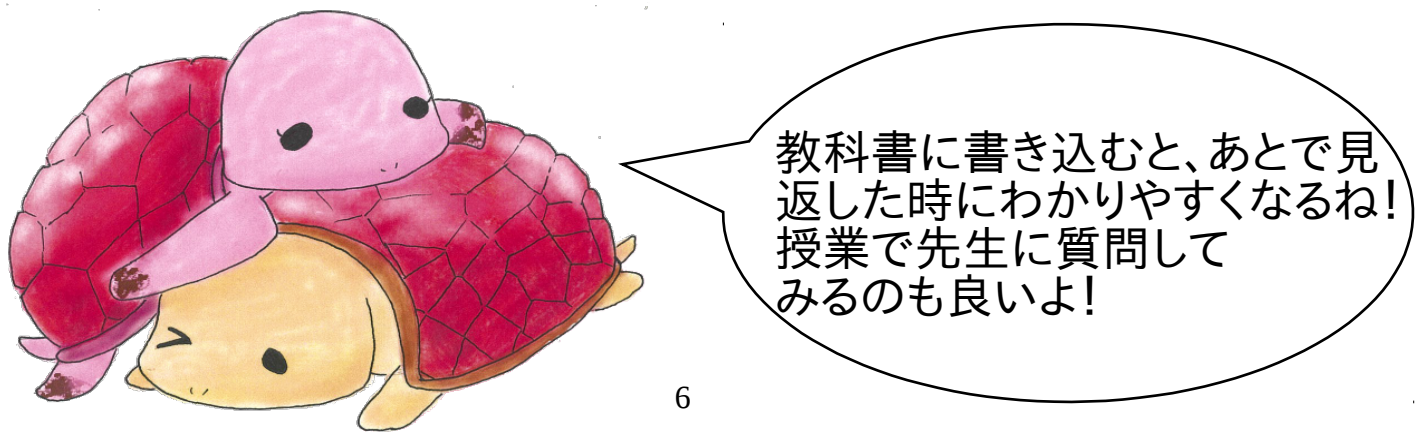
次の3つの手順を参考に学習を進めよう。

1．さらっとおまけを見ながら PC の使い方をマスター

2．どんどんプログラミング！

3．わからないところは、すかさず前のページを見る

この 3 つを繰り返して、プログラミングをマスターしよう！
プログラミングは実際に自分でプログラムを作成することで身につきますよ。



4．ファイルをつくろう

プログラミングをする前に、まずはファイルをつくろう。

絵を描くには、紙が必要だよね。プログラミングも同じです。
そのプログラミングの紙は、**ファイル**です！

部屋が散らかるのはあまり気持ちのいいものではないですよね。
ファイルも整理しておかないと、散らかってしまいます。

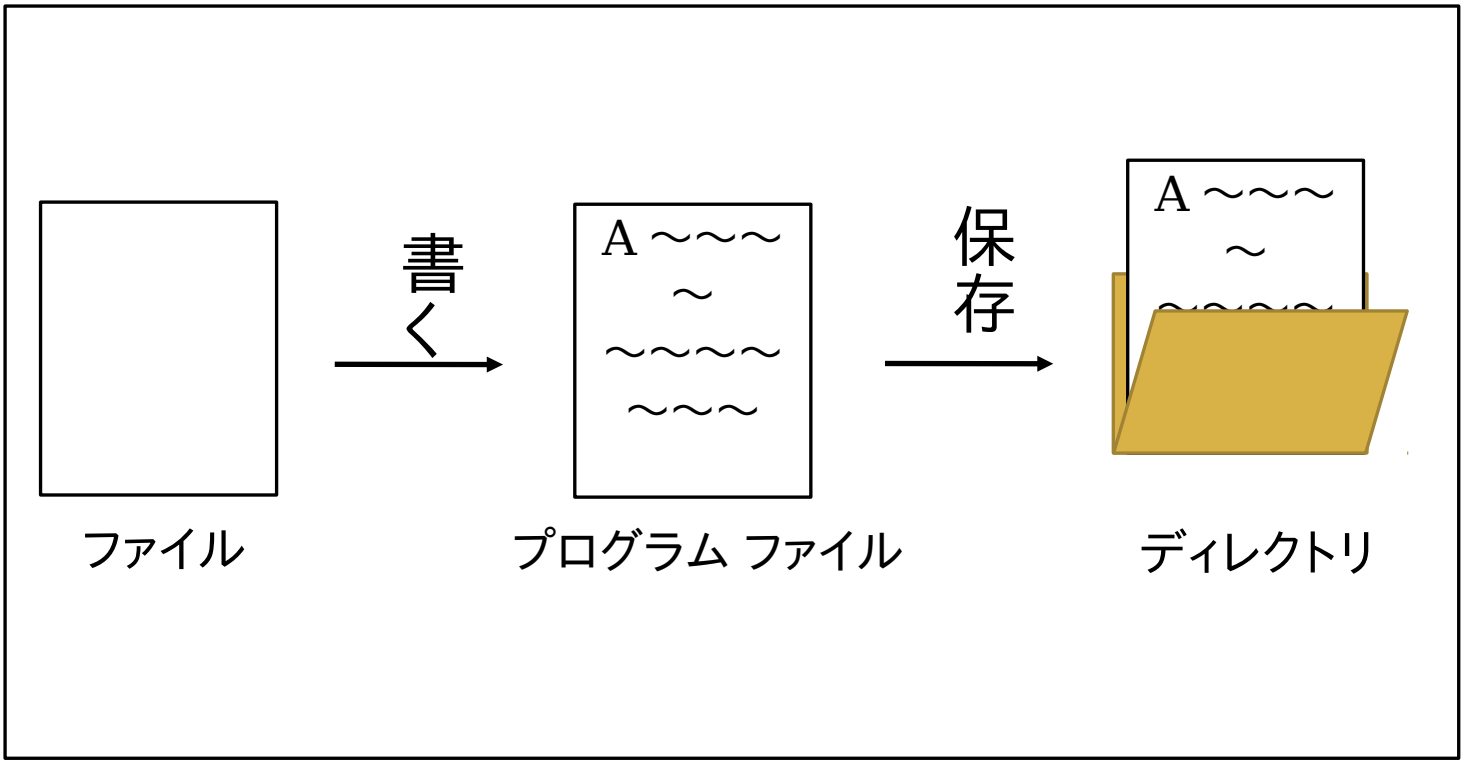
ディレクトリ: プログラムをしまっておく場所
ファイル　　: プログラムを作る場所（絵を描くときの紙）

作成手順

C-1 で Emacs を選び、C-x C-f でファイルを開く。ファイル名は my.rb とする。

```
Find file: ~/my.rb
```

ファイル名の最後にある .rb は「ruby プログラムのファイル」という意味です！



4.1 つくってみよう～print, puts, gets～

プログラムの作成は、以下の 2 つのステップで作成する
1. プログラムファイルの作成
2. プログラムの実行

手順

1. プログラムファイルの作成

C-1 で Emacs を選び、C-x C-f でファイルを開く。ファイル名は ~/my.rb とする。

```
Find file: ~/my.rb
```

ファイルの内容は以下のようにする

```
print “こんにちは!”
print “ruby てらこったで修行中のウミです。\n”
puts ‘海は広いっすねー’
```

書き終わったら、C-x C-s で保存する。

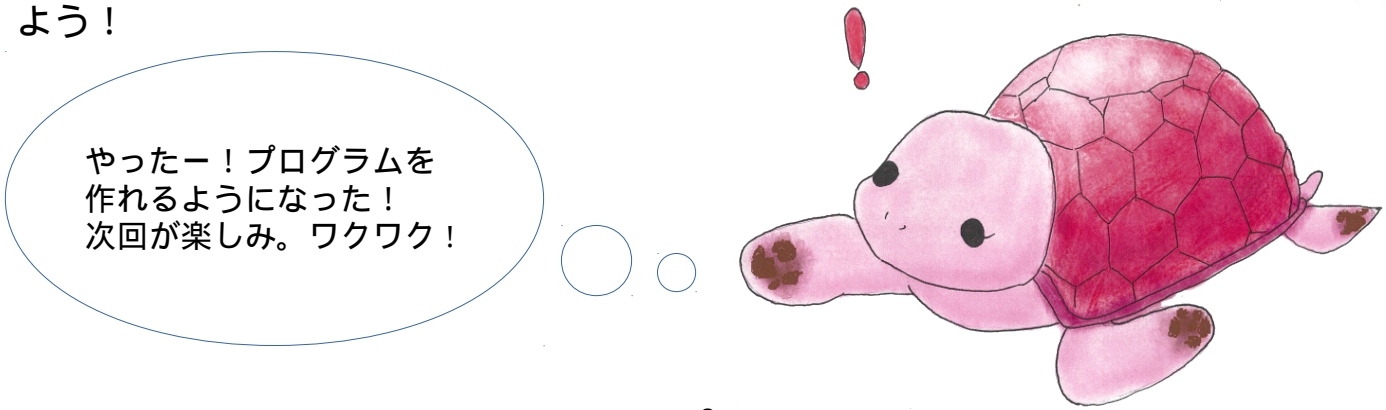
2. プログラムの実行

C-2 で kterm を選び Ruby プログラムを実行する。

```
% ruby my.rb
```

まとめ

・ファイルの作り方をしっかり覚えよう
・print, puts はそれぞれ何をするものかな？
・\n は何を意味するか考えてみよう
・さらにたくさん print や puts を使っておもしろい文を出すように変えてみよう！



プログラム例 (ascii.rb)

```
#!/usr/bin/env ruby
# coding: utf-8

print " アスキーアートを作成してみよう。\n"
print "\n"
print(" 作った作品は「ハート」です。\n")
print("\n")
print "  ●●●     ●●●  \n"
print("●●●●●  ●●●●●\n")
print("●●●●●●●●●●●\n")
print("●●●●●●●●●●●\n")
print("  ●●●●●●●●●  \n")
print("    ●●●●●●●    \n")
print("      ●●●●●      \n")
print("        ●●●        \n")
print("          ●          \n")
print("\n")

print(" 少し頑張って作るとこのような作品も作れます。\n")
print(" 作った作品は「スヌーピー」です。\n")
print("\n")
print("              ＿＿＿＿ \n")
print("       ＿ ／         ▲ \n")
print("  ／       ヽ   ■■ \n")
print("  ●               ■■ \n")
print("  ヽ＿＿＿＿       ■■ \n")
print("              )＝| \n")
print("            ／   || \n")
print("      ∩∩＿＿＿とノ \n")
print("      しし―――┘  \n")
print("\n")

print"
     ###     ###
   #####  #####
   ###########
     #########
       #######
         #####
           ###
             # \n\n"
```

4.2 つくってみよう～while（ワイル） 文～

○gets とは

キーボードに打ち込んだ値を文字列として取得するときには使う。

文字列なら → .chomp を付けて改行文字('\n') を切り取る。

数値なら → .to_i をつけて文字列から整数に変換する。

○変数

値につける名前のことである。好きな名前をつけることができる 。(使える文字は英数字とアンダースコア(_)で 最初の文字は英小文字でなければならない)

○while 文

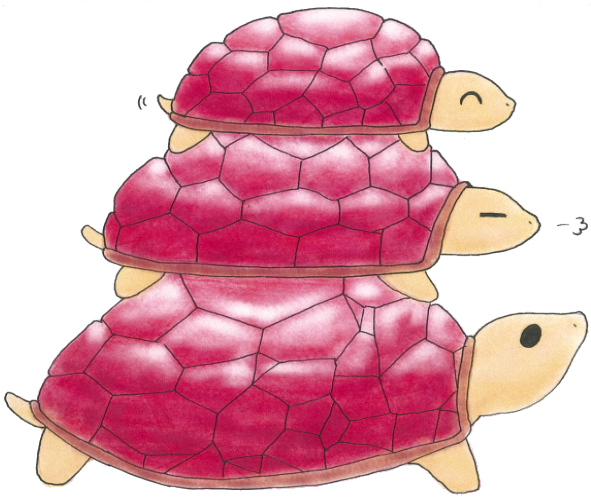
繰り返しの処理を行うときに使う。

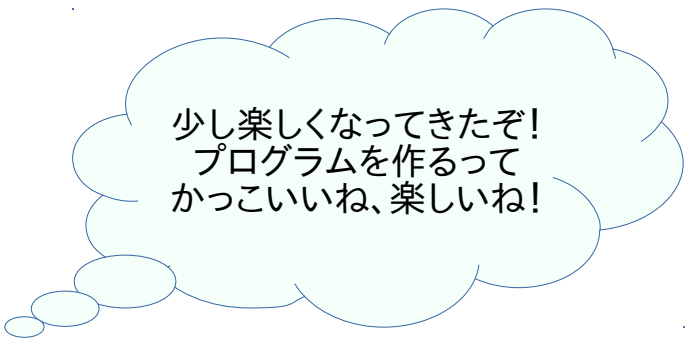
```
while a1
  こんにちは
end
```

条件 a1 がtrue である間「こんにちは」の部分を繰り返し実行する。
何度も繰り返す処理 のことを「ループ」という。

まとめ

・gets について理解できたかな?「 .chomp」「 .to_i」の違いは分かったかな？
・while 文を使ってどんなプログラムが作れるか考えてみよう！
・変数は次回以降の授業でも大切になってくるので復習しておこう！



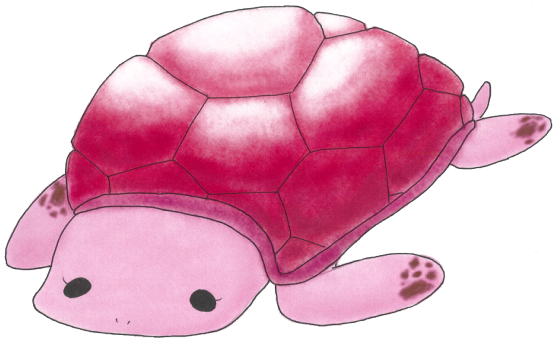


プログラム例 (register.rb)

```
#!/usr/bin/env ruby
# coding: utf-8

puts " ここはスーパーマーケットのレジだよ。
 買ったものの値段を入れてね !(100 円→ 100 と入力 )"


□ = 0
while true
  print " 値段は ?:( 終わりたい時は q を押してね )"
  △ = gets.chomp
  if △ == "q" then
    break
    end
  □ +=△.to_i
  printf(" 今の合計は %d 円だよ !\n",□)
end
printf(" 今回の合計は %d 円です。お買い上げありがとうございました !\n",□)
```

ここを while で繰り返してるよ。while true の中を一行づつ読むとわかりやすいかな

スーパーのレジはこんな感じで作られているのかなぁ。もっとすごいレジを作れそう !?



4.3 つくってみよう～配列、乱数～

配列とは

変数の中身を一列にならべるのが配列。

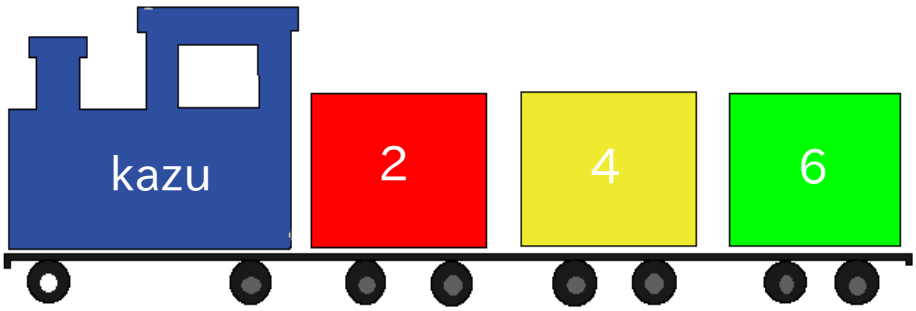
配列の書き方は下に書いてあるようにする。

```
kazu = [2 , 4 , 6]
kyoka = [“国語”, “算数”, “英語”]
```

注意：
ここでかっこの形に注意しよう。
丸いかっこ( )ではなく角ばったかっこ [ ] を使う。

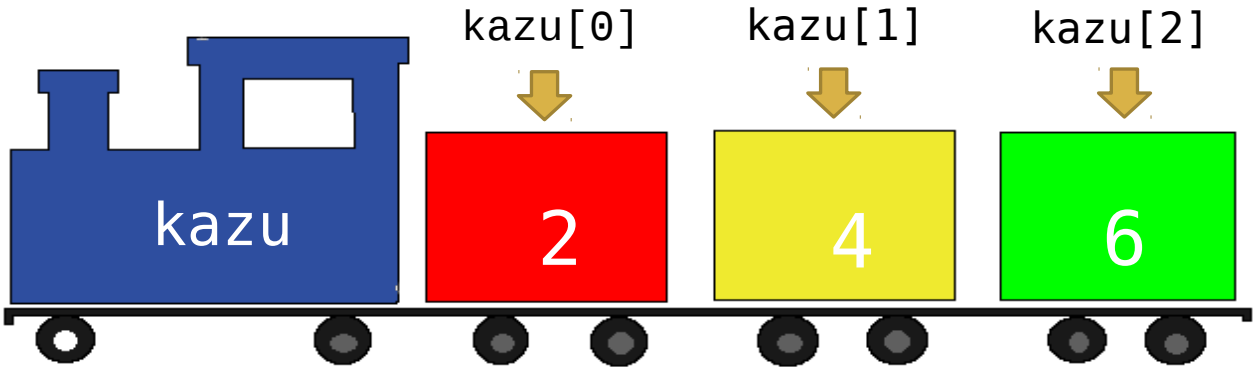
・配列のイメージは「貨物列車」

kazu = [2 , 4 , 6]を貨物列車で表すと



・配列の中の一つを表示させたいときは、配列の名前を書いて左から数えて何番目かを[ ]の中に書く。

注意：
コンピュータが数字を数えるときは 1 からではなく 0 から数えることに注意しよう。



乱数

・配列の中からランダムに１つ取り出すことを乱数という。
　お祭り屋台にあるおみくじ屋さんからくじを1枚とるイメージ。乱数は何が出るかわからない。
・乱数は、これから乱数を使いますよという合図を書くこと。
・乱数を使いますという合図の書き方は下に書いてあるようにする。

```
suzi = rand(3)
```

・suzi は乱数を表す変数。変数は好きに決めてよい。rand( )は乱数を使いますという合図。ここは変えてはダメ。かっこの中の数字は配列に並んでいる個数。
・乱数を表示させる書き方は下に書いてあるようにする。

```
kyouka[suzi]
```

・配列の名前のあとに乱数を表す suzi を入れる。

注意：
乱数を表す変数を入れるかっこは角ばったかっこ [ ] を使う。

○プログラムを～秒間ストップさせる

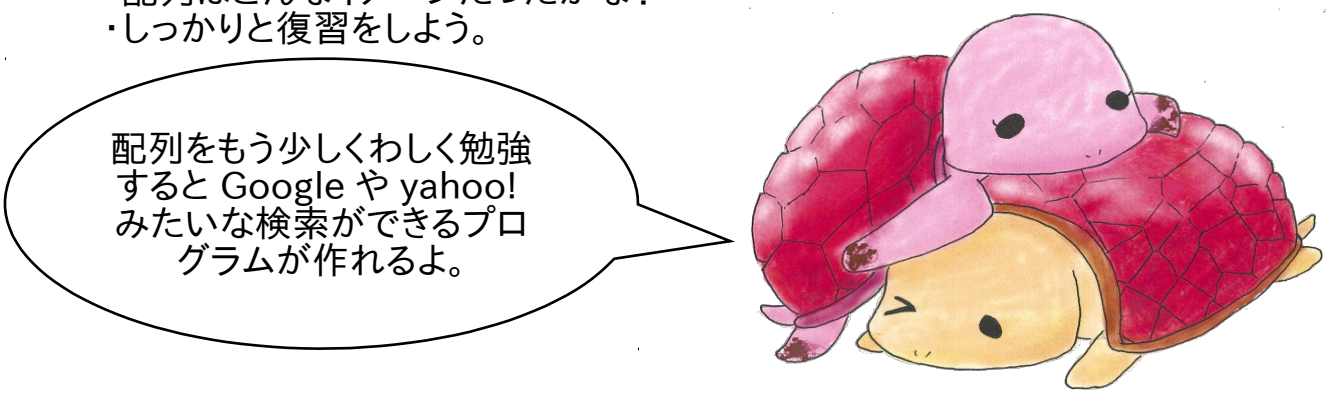
・プログラムは実行したらすぐに表示されるすごさがあるが、逆にすこしプログラムを止めることができる。止めるためには sleep を使う。
・プログラムを止めるときの書き方は下に書いてあるようにする。

```
sleep(1)
```

・かっこの中の数字は止めたい時間を入れる。

まとめ

・配列の書き方、乱数の書き方をしっかり覚えよう。
・配列はどんなイメージだったかな？
・しっかりと復習をしよう。



プログラム例 (array.rb)

```
#!/usr/bin/env ruby
# coding: utf-8

kazu = [2,4,6]
kyoka = [" 国語 "," 算数 "," 英語 "," 理科 "," 社会 "]

#  配列を表示する
printf("%d\n",kazu[0])

printf("%s\n",kyoka[0])


#  乱数を表示する
suzi = rand(3)          # ( )  には配列の個数を入れる
printf("%d\n",kazu[suzi])

benkyo = rand(5)        # ( )  には配列の個数を入れる
printf("%s\n",kyoka[benkyo])


#  プログラムを止める
print(" うとうと ...\n")
print("zzz\n")

sleep(3) #  かっこの中の秒数分だけプログラムを止めることができる。

print(" はっ !! 寝てません !!\n")
```

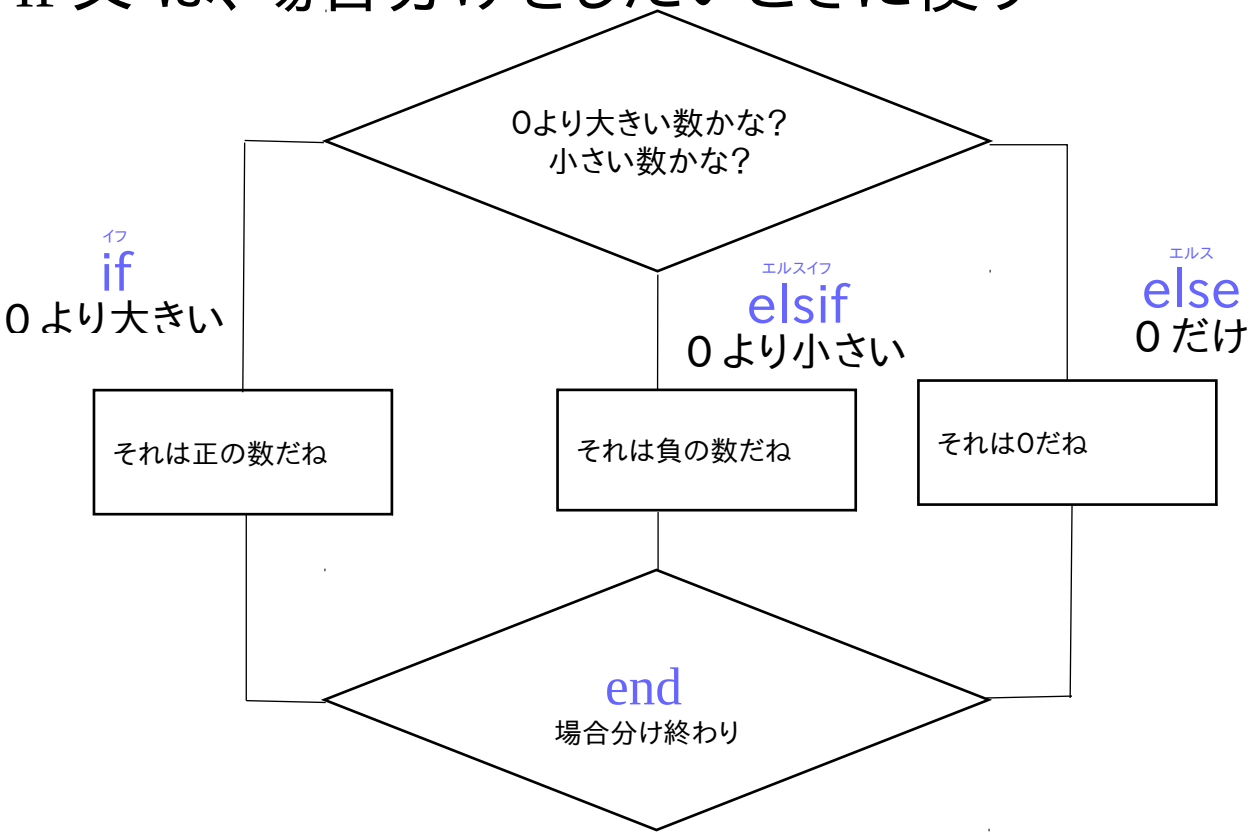
[数] と [“ 文字”] に注意しよう!
ちなみに # はコメント文で、プログラムの実行には反映されないよ!

(rand.rb)

```
#!/usr/bin/env ruby

boke = [" ハンバーーーグ !!" , " 角野卓造じゃねーーよ !!" , " はい、ひょっこりはん "]
print(" 爆笑ギャグまで \n")
sleep(1)

puts "3 秒前 "
sleep(1)
puts "2 秒前 "
sleep(1)
puts "1 秒前 "
sleep(1)

srand
erabu = rand(3)
printf("%s\n",boke[erabu])
```

4.4 つくってみよう～if-end～

if 文 は、場合分けをしたいときに使う



プログラムの中身は図のように理解しよう。

if, elsif, else で場合分けをしても最後はみんな同じところに帰ってくるから end は 1 つで大丈夫。

次のページでプログラム例を見てみよう!

まとめ

・if と elsif と else の使い方を覚えよう。
・選択肢の入れ間違いがないようにしよう。
・自分のオリジナルの問題を作れたかな？
・puts と elsif を使って選択肢を増やしてみよう!

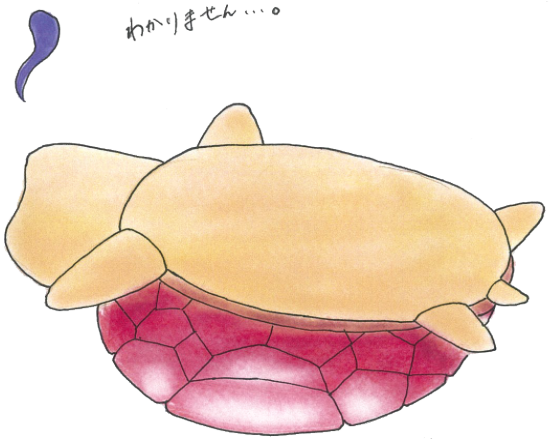
プログラムの例

```
puts (“問題です!”)
puts (“ここに問題文を入れる。”)
puts (“選択肢1をいれる”)
puts (“選択肢2をいれる”)
puts (“選択肢3をいれる”)

△ = gets.chomp.to_i

if △ == 1
  puts (“選択肢1番の返しを入れる。正解なら正解!不正解なら不正解!”)

elsif △ == 2
  puts (“選択肢2番の返しを入れる。”)

elsif △ == 3
  puts (“選択肢3番の返しを入れる。”)

else
  puts (“その選択肢はないよ!”)

end
```

puts を入れれば
何個でも選択肢を
加えることができるよ!

elsif をいれれば何個でも選択肢への返しが入れられるよ！

puts でつくった分の返しをちゃんといれよう！



最初はみんなわからない!
繰り返し作って理解しよう

5.もっと楽しもう

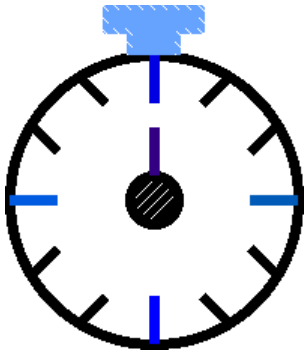
プログラムをもっと面白くするひみつへいき！

1. Time.now.to_i

いまの時間を受け取ることができるよ！
これを使うとストップウォッチみたいなのが作れちゃう！

・ストップウォッチ (stopwatch.rb)

```
print“ストップウォッチスタート(Enterを押すと止まるよ) : ”
start = Time.now.to_i
stop = gets.chomp
finish = Time.now.to_i
time = stop - start
printf(“タイムは%d秒です。\n”, time)
```



・プログラムを動かすと...

```
% ruby stopwatch.rb
ストップウォッチスタート(Enterを押すと止まるよ) :
タイムは2秒です。
```

2. system “banner moji”

moji の部分に好きな文字を入れると簡単に文字のアスキーアートが作れるよ！
ただし、英語しか使えないから注意！

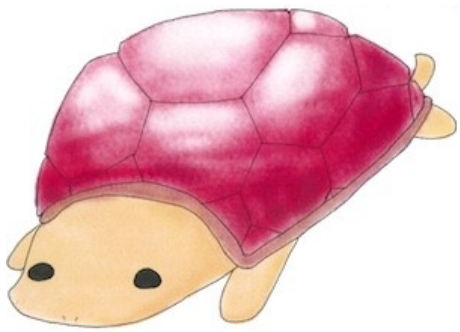
・簡単なアスキーアートプログラム (banner.rb)

```
system “banner Hello!”
```

・プログラムを動かすと...

```
% ruby banner.rb

#     #                                    ###
#     #  ######  #       #        ####    ###
#     #  #       #       #       #    #   ###
#######  #####   #       #       #    #    #
#     #  #       #       #       #    #
#     #  #       #       #       #    #   ###
#     #  ######  ######  ######   ####    ###
```

これでもっと面白いプログラムがいっぱい書けちゃうね！



プログラム例(fukusyu.rb)

```
#!/usr/bin/env ruby

system"banner Hello!"
puts" さぁ、復習の時間だぁ! "
puts" 解けるものなら解いてみろ! "
seikaisuu = 0
start=Time.now.to_i

puts"- 第 1 問 -"
puts"1 回目に習った kterm に文字を出すときに使う言語は? "
puts"1,puts 2,exit 3,cntrol"
answer = gets.to_i

if answer == 1
puts" 正解!他に print や printf があるよ! "
seikaisuu += 1
else
puts" 残念!正解は puts だよ。他には print や printf があるよ。使い分けできるようにね。"
end

puts"- 第 2 問 -"
puts"gets の後につけるもので文字列の時につけるものは? "
puts"1,to_i 2,.chomp 3,.to_f"
answer = gets.to_i

if answer == 2
puts" 正解!流石!! "
seikaisuu += 1
else
puts" 残念!! .chomp だよ "
end

puts"- 第 3 問 -"
puts" ループの時に使うものは? "
puts"1,if 2,while 3,else"
answer = gets.to_i

if answer == 2
puts" 正解!ちなみに「ワイル」と読むよ "
seikaisuu += 1
else
puts" 残念! while だよ "
end
```

#次ページに続く

```
puts"- 第 4 問 -"
puts" 配列は何番目から始まるでしょう？ "
puts"1,0 番目  2,1 番目  3,10 番目 "
answer = gets.to_i

if answer == 1
puts" 正解!ちゃんと覚えてて good ! "
seikaisuu += 1
else
puts" 残念!! "
end

puts"- 第 5 問 -"
puts" 乱数の時に使うものは?ランダムに1つ取り出すときに使うものだよ "
puts"1,iland 2,land 3,rand"
answer = gets.to_i

if answer == 3
puts" 正解!!すごいねぇ! "
seikaisuu += 1
else
puts" 残念! rand だよ "
end

puts"Enter を押してね "
stop=gets.chomp
finish=Time.now.to_i
time=finish-start
printf(" タイムは、%d 秒です。\n" ,time)
sleep 1
puts" 正解数は ....."
sleep 1
printf("5 問中、%d 問正解!! \n", seikaisuu )
```

#(janken.rb)

```
#!/usr/bin/env ruby

puts" コンピュータとじゃんけんをしよう !"
puts"5 回中 3 回勝てればクリアだよ !"
sleep(3)
puts""
print" それじゃ  GAME START\n\n"

win = 0
kaisuu = 1
janken = [" だしてないよー ", " グー ", " チョキ ", " パー "]

while kaisuu <= 5
  sleep(2)
```

```
  printf("%d 回目 !( あなたの勝利数 :%d)\n\n", kaisuu, win)
  print" じゃーんけーん !( グーなら「 1 」チョキなら「 2 」パーなら「 3 」を押してね ): "

  while true
    you = gets.to_i
    if you >= 4
       you = 0
    end
    com = rand(3) + 1
    printf(" ぽん ! ( あなたは「 %s 」で コンピュータは「 %s 」 )\n", janken[you], janken[com])
    sleep(1)
    if (you == 1 && com == 2) || (you == 2 && com == 3) || (you == 3 && com == 1)
      puts" あなたの勝ち !"
      win += 1
      kaisuu += 1
      break
    elsif (you == 1 && com == 3) || (you == 2 && com == 1) || (you == 3 && com == 2)
      puts" あなたの負け !"
      kaisuu += 1
      break
    elsif you == com
      print" あーいこーで ( グーなら「 1 」チョキなら「 2 」パーなら「 3 」を押してね ): "
      redo
    else
      puts" なにもだしてないからあなたの負け !"
      kaisuu += 1
      break
    end
  end
end

sleep(3)
printf(" あなたの勝利数は「 %d  回」 \n", win)
sleep(3)

if win >= 3
  puts" おめでとう !"
  sleep(2)
  puts"+--------------------------------------------------+"
  system 'banner YOU WIN!!'
  puts"+--------------------------------------------------+"
else
  puts" 残念 ..."
  sleep(2)
  puts"+--------------------------------------------------+\n"
  system 'banner GAME OVER'
  puts"+--------------------------------------------------+\n"
end
```

6.おまけ

コマンド集

＊Emacs(イーマックス)編

Emacs のコマンドを入力するには、コントロールキー(CONTROL(コントロール) とか Ctrl(コントロール) とか CTL(コントロール))を使う。
そこで、Ctrl を書く代わりに、次のような書き方をする。

C-<文字>　Ctrl キーを押しながら<文字>キーを押す。
例）C-f は Ctrl キーを押しながら f のキーを押す

・ファイルをつくろう！

1. C-x C-f の順番で入力
2. ファイル名を入力して、Enter を押す

・ファイルを保存しよう！

1. C-x C-s の順番で入力
2. Emacs の左下 にある U:**- が U:--- に変わったら保存完了！

※ **印を残したまま Emacs を終了すると、編集した成果が台無しになるので注意すること。

・取り消し(UNDO)

もし、文章を変えたあとで「あ、間違った！」と思ったら、

C-x u

で変更の一つ前に戻るよ。

・もし Emacs が反応しなくなったら…

キーを押していてわけがわからなくなったら、とりあえず C-g を押してみる。

こまったら C-g

コマンド集(簡単 Ver)

Emacs(イーマックス)へ移動	kterm(ケータァム)へ移動	console(コンソール)へ移動
C-1	C-2	C-3

Emacs 編

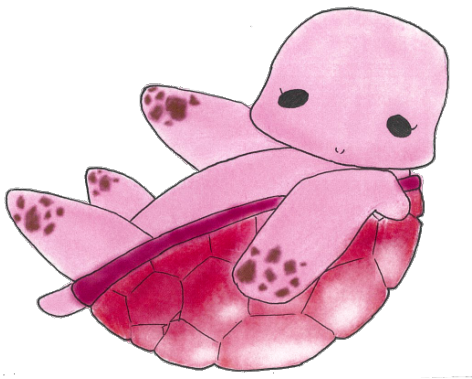
C-x C-f	ファイルの作成 / ファイルを開く
C-x C-s	ファイルの保存

kterm 編

cd ディレクトリ	[ディレクトリ] に移動
cd	ホームディレクトリに移動
mkdir ディレクトリ	[ディレクトリ] を作成
ls	今いるディレクトリの中にあるファイルを表示
ruby ファイル名	[ファイル名] を実行

console 編

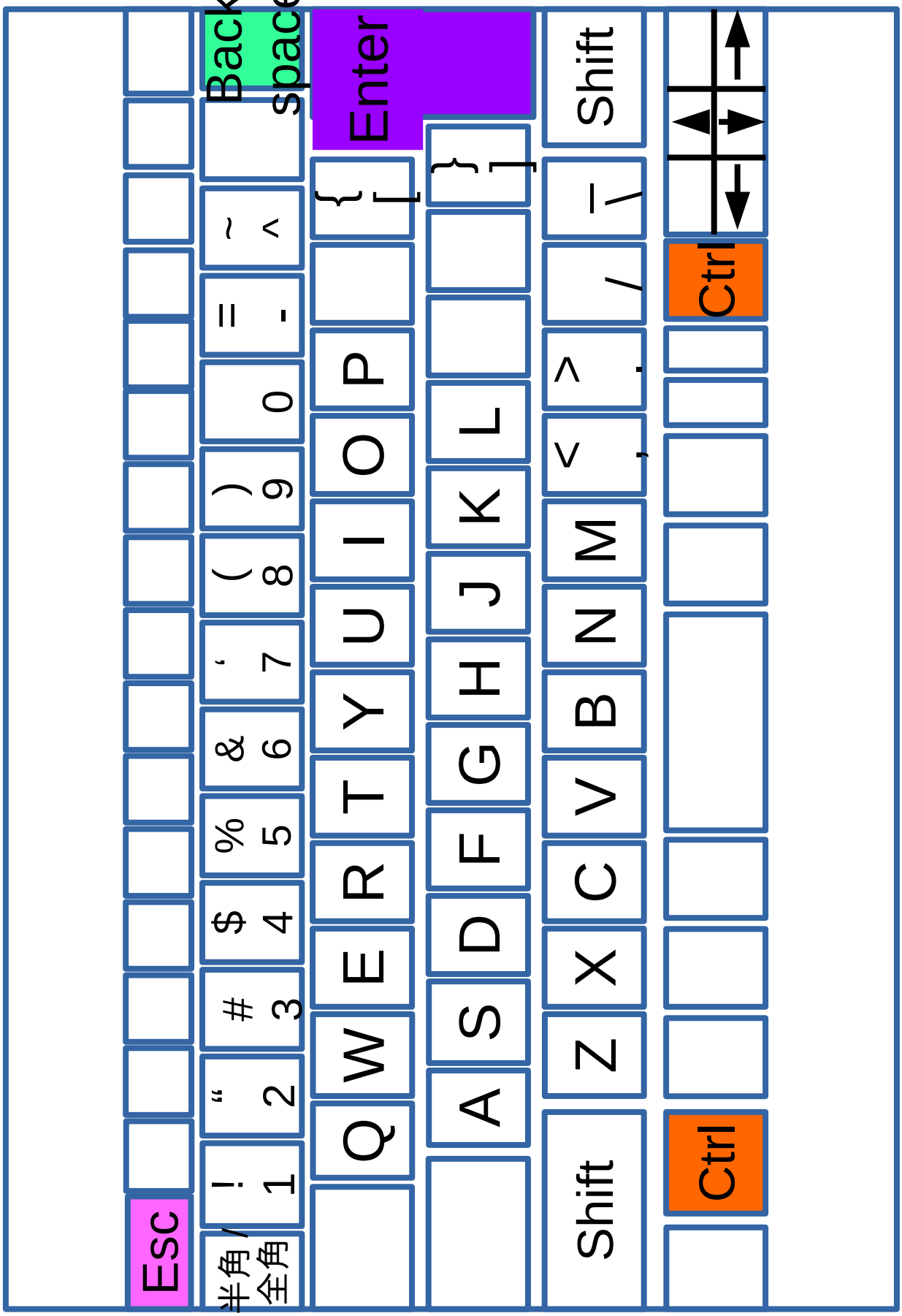
exit	ログアウト
shutdown -p now	電源をおとす

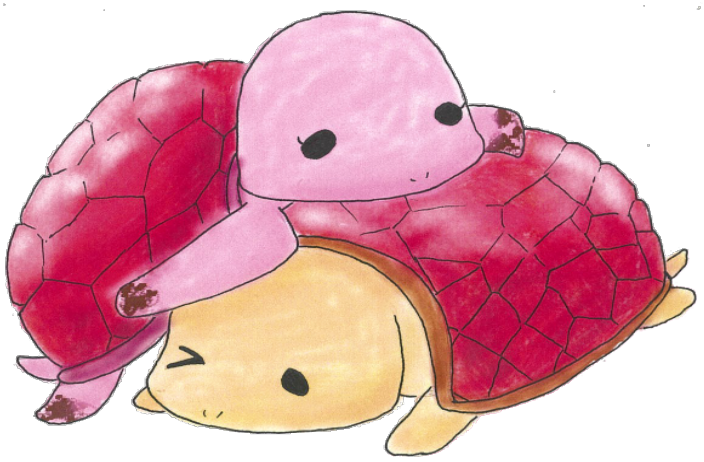


コマンドってこんなにあるのか…。
でも、コマンドを知っているとプログラムの作成がスムーズになるよ!

キーボード早見表

(横にしてつかってね)





Let’s enjoy programming !!

NAME