創好リナの

# Programming Seminar

## 2bit

# Sequential Processing

2018.08.15

# プログラミングの基礎の基礎

- 処理の流れの大前提
- 標準出力 / 標準入力
- データの型
- 変数
- 式 / 演算子

全項目がそれぞれ絡み合って
全て大事なので初めての人は
2回見ると理解しやすいかも

全部**覚える**必要はない

やっているうちに**理解**できる

言葉ではなく心で理解するッ！

# 処理の流れの大前提

# 上から下に処理される

- Program

```
print("Hello")
print("World")
```

- Result

```
Hello
World
```

Twitterになれてる人は注意

2chになれてる人はそのまま

# 標準出力 / 標準入力

# 標準出力

- 最も単純な出力機能
- 文字列を画面に出力する
- `print()` ←これ
- 他にも`term.write()`や`io.write()`がある
- `()`をつけて実行されるのは関数という

# 3つの標準出力関数

- `print()`
  - 最も簡単な標準出力
  - 自動で改行される
- `io.write()`
  - 改行はされない
- `term.write()`
  - CC専用
  - termAPIで設定した色が適応される

# 標準入力

- 最も単純な入力機能
- キーボードから文字を入力する
- `read()` ←これ
- 他にも`io.read()`がある

# 2つの標準入力

- `read()`
  - CC専用
  - そのまま実行で`io.read()`と同じ
  - 便利機能がいくつかある(後日説明)
- `io.read()`
  - そのまま実行でEnterが押されるまでの入力を文字列として返却
  - formatを指定できる(後日説明)

# 標準入出力 例

- Program

```
local str = read()
print(str)
```

- Result

```
abcd[Enter]
abcd
```

# 変数

# 変数

- データを保持しておくための箱
- グローバル変数とローカル変数がある
  - 基本はローカル変数を使う
  - ローカル変数は有効期限がある

# データを保持する箱

- Program

```
local str = "aaaaa"   -- 変数を作成 + "aaaaa"を代入
print(str)
str = "bbbbb"   -- "bbbbb"で変数の中身を上書き
print(str)
```

- Result

```
aaaaa
bbbbb
```

# グローバル変数/ローカル変数

## グローバル変数

- どこで作ってもどこでも呼び出せる
  - 理解していないと思わぬ事故につながる
- プログラム終了まで消えない(消さない限り)
  - メモリを専有する
- `local`を付けないで変数を作るとグローバル変数になる

# グローバル変数/ローカル変数

## ローカル変数

- 作った場所でしか使えない
- 作った場所を抜けると自動で消える
- 特別な理由がない限りこれを使う
- 変数を作る時に`local`をつける

# データ型

# データ型

- プログラム内で使用する値には型がある
- Luaでは変数に型を指定しない
  - 自由に入れることができるが無闇に使うな
- 型を変換する方法がそれぞれある

# Luaの型一覧

| 型名 | 説明 | 例 |
| --- | --- | --- |
| nil | 何もない 無 | nil |
| boolean | 論理型 / true or false | true |
| number | 数値 | 10 |
| string | 文字列 | "abcd" |
| table | テーブル (配列等) | {1, 2, 3} |
| function | 関数 | function ... end |
| userdata | ユーザ定義型 | ※今回使わない |
| thread | スレッド | ※今回使わない |

# nil

- なんでもない型
- 無
- 基本演算もできない(エラー)

# boolean

- 論理型
- true か false しかない

# number

- 数値型
- 少数も扱う
  - 他言語ではint, float, doubleなど別れているがLuaではnumberでまとめられている

# string

- 文字列型
- 文字列は`"`で囲って表現する
  - Luaでは`[[ ]]`を使うこともある
- 文字列は文字が連続していると考えるとわかりやすい
- `\n`や`\t`と言った特殊文字がある
  - `\n`は改行文字 (`print()`は末尾に自動で`\n`を付与)
  - `\t`はタブ文字

# table

- テーブル型
- 値をいくつも入れられる
- Luaでは配列も連想配列もtable型
- `{}`で囲って表現する
- Luaはtableが強い!!!
- 長くなるのであとで解説

# function

- 関数型
- `function` ~ `end`で表現
- Luaでは関数も第一級オブジェクト
  - JavascriptやPythonと同じ感じ
- `print`や`read`も最初から宣言済みの関数型
- 詳しいことは後日

# tableちょっとだけ詳しく

# 配列 #とは

- 値を何個も入れておく連続した箱
- それぞれの箱には番号が振り分けられる
- ArrayとかListとかよく言われる

# 配列 例

- Program

```
local array = {1, 5, 10, 50, 100, 500, 1000, 5000, 10000}
print(array[3])
```

- Result

```
10
```

# 連想配列 #とは

- 配列の箱一つ一つに名前をつけたやつ
- 順番が担保されない
- Mapとかよく言われる

# 連想配列 例

- Program

```
local map = { hoge = 10, foo = 30, bar = 100}
print(map["hoge"])
print(map.bar)
```

- Result

```
10
100
```

# 式

# 式 #とは

- 何かの値を持つものの事
  - リテラル (直接書かれた値: `1`, `"aaa"`, `true`, `nil`)
  - 無名関数宣言
  - テーブルのコンストラクタ
  - 変数の参照
  - 関数呼び出し
  - `()`でくくられたもの
  - 単項演算子がついたもの (`-1`, `^num`)
  - 2つの式を二項演算子で繋げたもの (`num + 2`)

# 演算子

# 演算子

- `+`とか`-`とかのやつ
- 式に付けたり式同士を繋げて一つの式にする
- 演算子には優先順位がある
- 演算子を制したものがプログラムを制す

# 算術演算子

## 数字を扱う

| 演算子 | 名前 | 例 | 備考 |
| --- | --- | --- | --- |
| + | 加算 | a + b | |
| - | 減算 | a - b | |
| * | 乗算 | a * b | |
| / | 除算 | a / b | 余りは出さず少数になる |
| % | 剰余 | a % b | 整数で除算した時の余り |
| ^ | べき乗 | a ^ b | 指数は整数 / 右結合 |
| - | 反数 | -a | |

# 関係演算子

## Booleanを返す

| 演算子 | 例 | 説明 |
| --- | --- | --- |
| == | a == b | 型、値が等しいとtrue |
| ~= | a ~= b | ==の逆 |
| < | a < b | bの方が大きいとtrue |
| > | a > b | aの方が大きいとtrue |
| <= | a <= b | bの方が大きいか等しいとtrue |
| >= | a >= b | aの方が大きいか等しいとtrue |

# 論理演算子

| 演算子 | 例 | 備考 |
| --- | --- | --- |
| and | a and c | 左が`偽値`で左、`真値`なら右を返却 |
| or | a or c | 左が`真値`で左、`偽値`なら右を返却 |
| not | not a | true or falseのみ返却 |

- Luaでは`nil`と`false`以外は全て`真値`扱い

# 結合演算子

| 演算子 | 例 | 備考 |
| --- | --- | --- |
| .. | a .. b | 文字列を連結する |

- 文字列でないなら文字列表現に変換されてから連結される

# 長さ演算子

| 演算子 | 例 | 備考 |
| --- | --- | --- |
| # | #a | 文字列ならバイト長、配列なら配列の長さが返却 |

# 演算子優先順位

| 優先度 | 演算子 |
| --- | --- |
| 1 | () |
| 2 | ^ |
| 3 | not # -(反数) |
| 4 | * / % |
| 5 | + -(減算) |
| 6 | .. |
| 7 | < > <= >= ~= == |
| 8 | and |
| 9 | or |

# 盛りだくさんでした

疲れました😢

今日やった内容は
全ての言語の基礎なので
覚えるのではなく
理解しましょう

一気に理解する必要なく、
今後やっていくうちに
自然とわかります

# 今日使うCCの関数

- `turtle.forward()`
  - タートルを一つ前に動かす
- `turtle.turnRight()`
  - タートルを右に90度回転
- `turtle.turnLeft()`
  - タートルを左に90度回転

# 今日の練習問題

# 2-1

- 以下の出力をするプログラムを書きなさい(1行)

```
創好リナ
かわいい
```

# 2-2

- 以下の入出力をするプログラムを書きなさい(2行)

```
創好リナ[Enter]
創好リナはかわいい
```

# 2-3

- 変数aに24、変数bに10を代入、aとbの加算,減算,乗算,除算,剰余の結果を以下のような出力をせよ

```
24 + 10 = 34
24 - 10 = 14
24 * 10 = 240
24 / 10 = 2.4
24 % 10 = 4
```

# 2-4

- タートルに将棋の桂馬の動きをさせなさい(右前)