

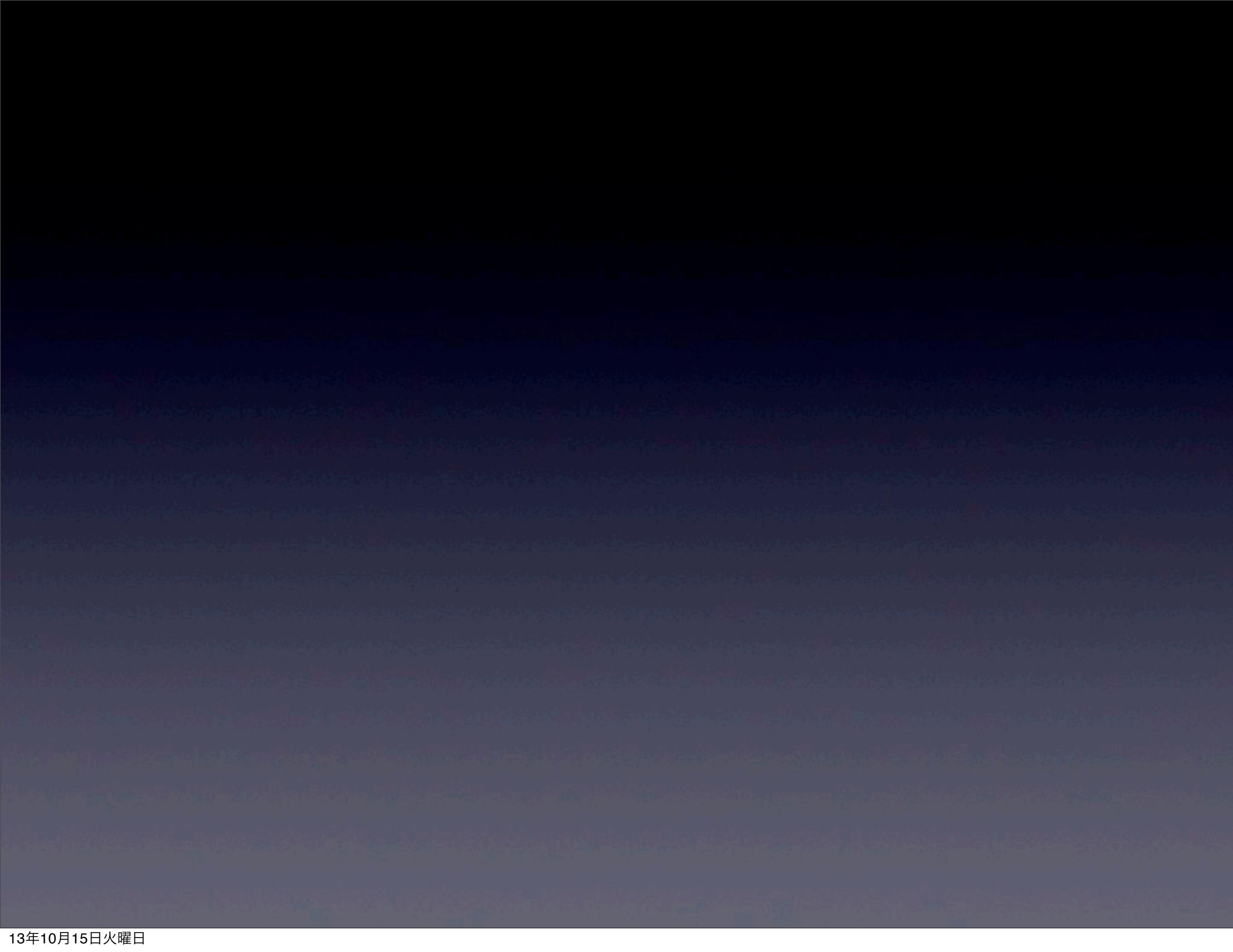
PHPのfloat型やら int型やらの話

id:hnw
y at hnw dot jp

PHP勉強会 SIDE-B (第1回) 発表資料

と、発表の前に…

- ・アンケートにご協力ください
- ・勉強会初参加の方！
- ・プレゼン未経験の方！

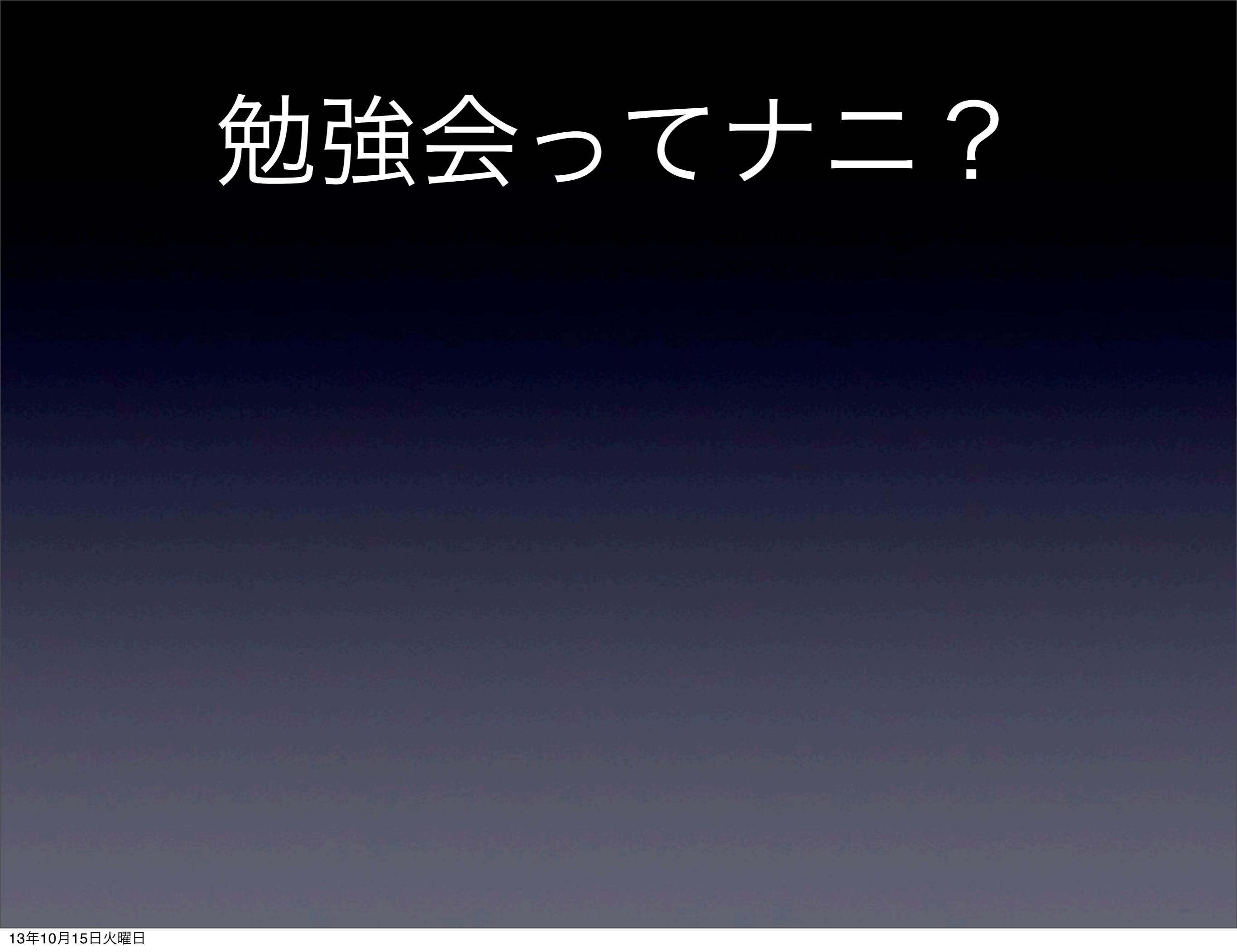


ようこそ！

ようこそ！

(主催者でも何でもありませんが)

勉強会ってナニ？



勉強会ってナニ？

- 主催者や発表者って何が楽しいの？
- 参加するまで僕もそう思ってました

勉強会ってナニ？

- 主催者や発表者って何が楽しいの？
- 参加するまで僕もそう思ってました
- 「理解できない→気持ち悪い」

勉強会ってナニ？

- 主催者や発表者って何が楽しいの？
- 参加するまで僕もそう思ってました
- 「理解できない→気持ち悪い」
- 初参加の人にその辺を伝えたい

勉強会のメリット



勉強会のメリット

- 新たな知識の獲得
- 強制力
- 自分のアンテナ外のことが聞ける

勉強会のメリット

- 新たな知識の獲得
- 強制力
- 自分のアンテナ外のことが聞ける

実はもっと凄いメリットがある！

すごいメリット(1)

- 懇親会！
 - 酔うと技術話が盛り上がる
 - 技術者はシャイなので酒重要
 - オフレコ話
- 単純に楽しい。有意義なことも。

すごいメリット(2)

- 発表者にもメリット
- 自分を知ってもらえる
- 懇親会でいじられるネタ提供
- 周辺知識を教えてもらったり
- 1vs多→フィードバックの期待値↑

まとめると

- ・他人への影響大→フィードバック大
- ・発表者のメリットが最大
- ・騙されたと思って一度発表側に！
- ・自分が面白いことを発表すればOK

自己紹介

- id:hnw
- カレー好き
- 先週は週4カレー

自己紹介

Hatena::Diary [] 日記 検索 最新の日記 記事一覧 日記を書く 管理 ログアウト ヘルプ

h n w の 日 記 AI RSS

<前の日 | 次の日> [プロフィール]

↑ PHPのround関数についての記事

自己紹介

- @halt(Ethna、 vimの人)
- @hiro_y(moonyの人)
- @hnw(round()の人)
- @ichii386(Ethnaの人)
- @iogi(PHP Extension勉強会の人)
- @iteman(Piece Frameworkの人)
- @junya(Sooey、 guessworkの人)
- @kensuu(ミルフィールの人)
- @komagata(plnet.jpの人)
- @koyhoge(とにかくすごい人)

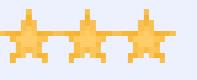
↑ 謎の紹介文

自己紹介

- @halt(Ethna、 vimの人)
- @hiro_y(moonyの人)
- @hnw(round()の人)
- @ichii386(Ethnaの人)
- @iogi(PHP Extension勉強会の人)
- @iteman(Piece Frameworkの人)
- @junya(Sooey、 guessworkの人)
- @kensuu(ミルフィールの人)
- @komagata(plnet.jpの人)
- @koyhoge(とにかくすごい人)

↑ 謎の紹介文

自己紹介

- 2008年06月26日  [TAKESAKO](#) php ミスター浮動小數点  

↑謎のブクマコメント

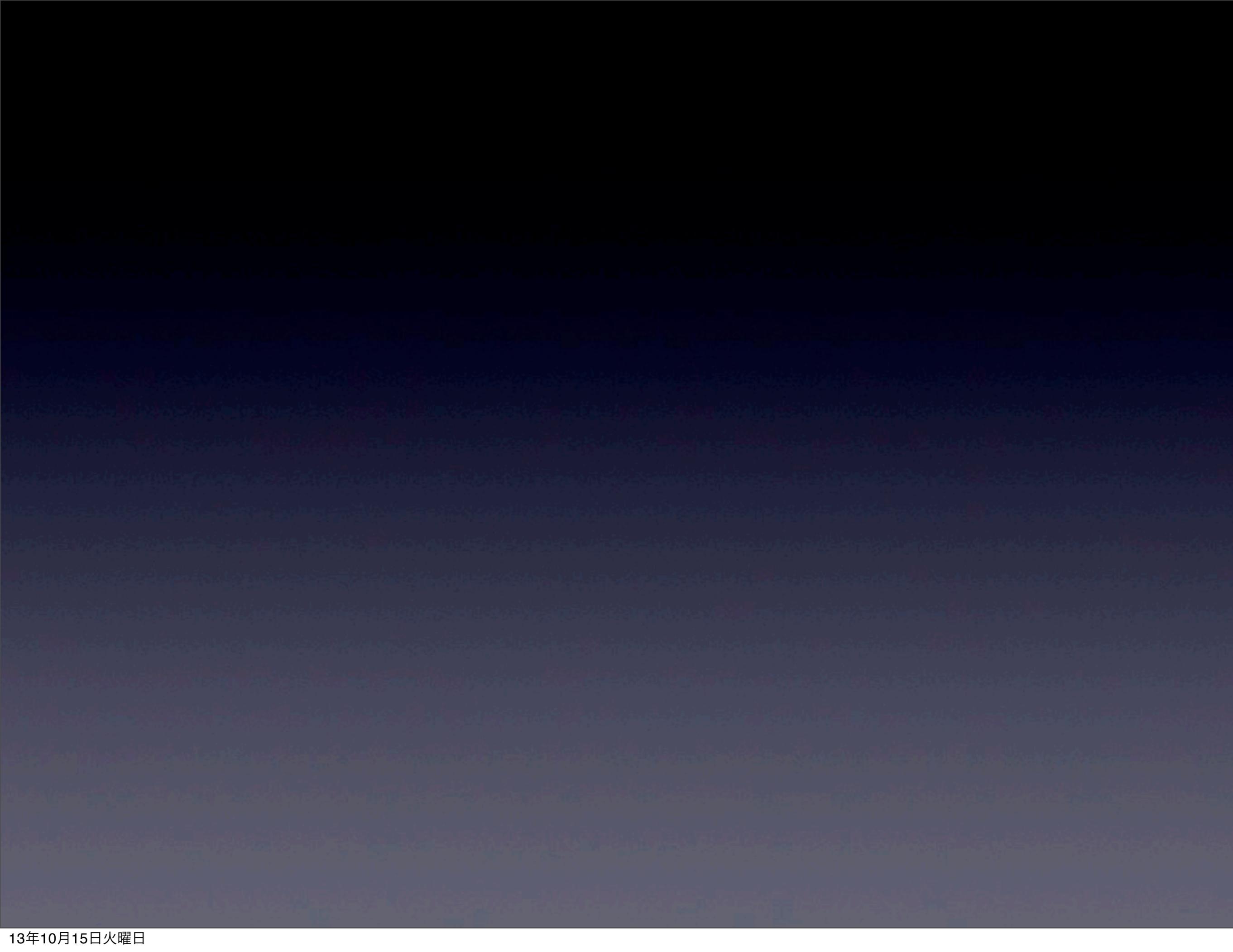
自己紹介

- 2008年06月26日  [TAKESAKO](#) php ミスター浮動小数点  

↑謎のブクマコメント

こんな相談を受けた

- ・「プログラム中で小数は使わない方がいい」って聞きました。なぜですか？
(千葉県・20歳・プログラマ)



ようやく

今月の

本題です

- intとfloat
- PHPでの実例
- その他の注意点
- まとめ

PHPの数値型

- int型
 - 32bit整数
 - float型
 - 64bit浮動小数点数
- PHP以外とも共通の道具

しばらく一般論用

CPUについて

- 使っているCPU : 32bitアーキテクチャ
 - 整数レジスタ長 = 32bit
 - 32bit整数同士の加減乗除は1命令

レジスタって？

- 計算結果の置き場。数十個程度?
 - 変数の実体みたいなもの
 - マシン語レベルでは頻出
 - レジスタ2個の和→レジスタ、など
 - CPUから見て最速のデータ格納領域

言語における整数

- intのサイズはアーキテクチャ都合
 - CPUにとって最適なサイズ
 - 最近のCPUだと32bit
 - そのうち64bitになるのかも

CPUと浮動小数点数

- 64bit浮動小数点数
 - IEEE 754で決まっている
- 浮動小数点数用に別レジスタ
- CPUの1命令で加減乗除できる

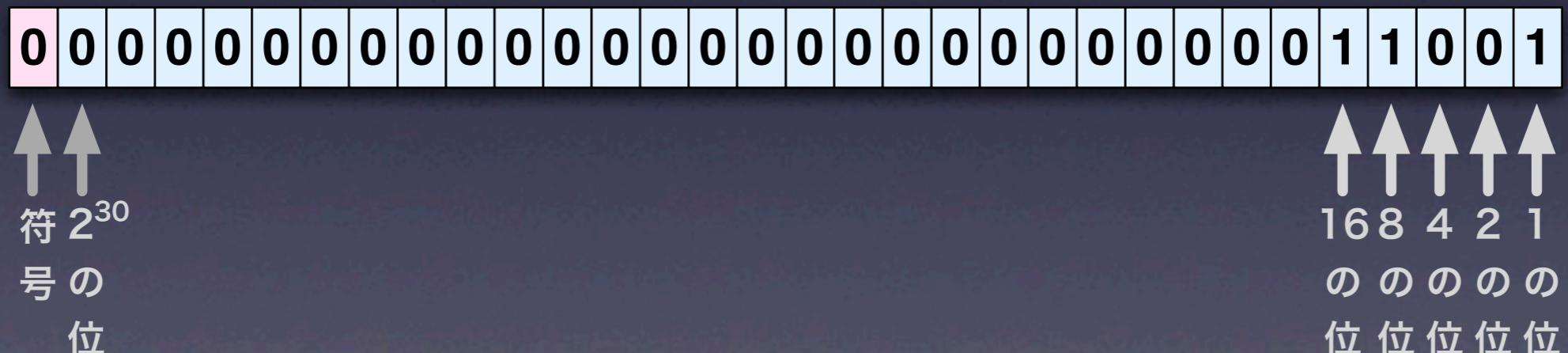
intもfloatも

CPUの都合で

決まっている

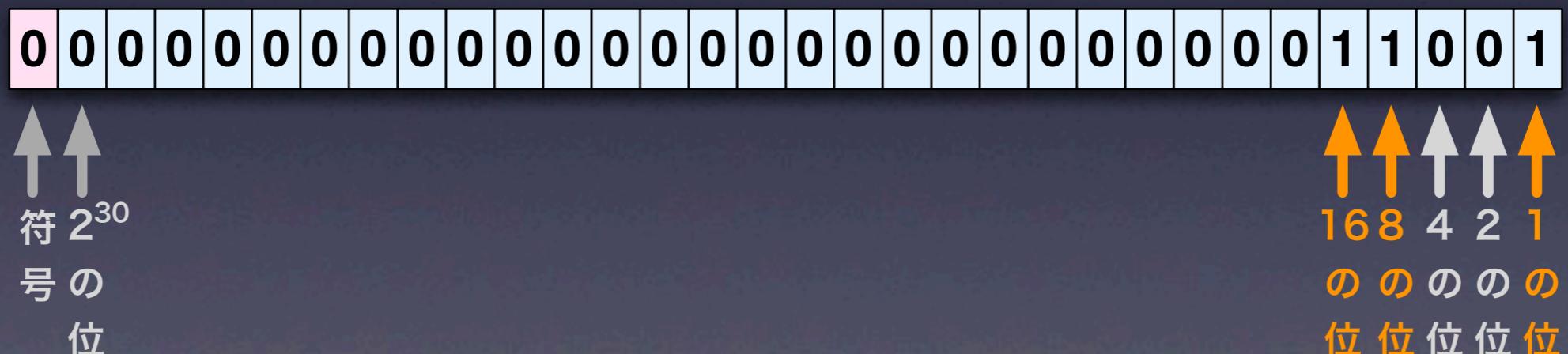
ビットパターン

- intの場合、符号が1bit／残りが数字
 - $-2^{31} \sim 2^{31}-1$ まで扱える。
 - 例： $25 = 16 + 8 + 1$



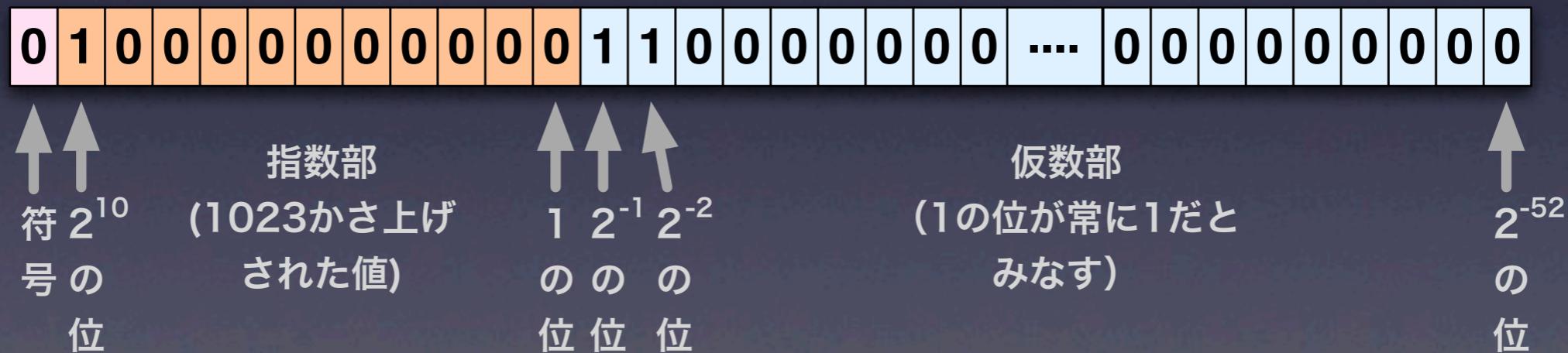
ビットパターン

- intの場合、符号が1bit／残りが数字
 - $-2^{31} \sim 2^{31}-1$ まで扱える。
 - 例： $25 = 16 + 8 + 1$



ビットパターン

- floatの場合
 - 符号1bit／指数部11bit／仮数部52bit
 - 例： $3.5 = 1.75 \times 2^{(1024-1023)}$



- intとfloat
- PHPでの実例
- その他の注意点
- まとめ

小数のトラブル

```
ringo:~ php -r 'var_dump(0.5+0.5);'  
float(1)  
ringo:~ php -r 'var_dump(0.5+0.1+0.1+0.1+0.1+0.1);'  
float(1)  
ringo:~ php -r 'var_dump(0.5+0.5==0.5+0.1+0.1+0.1+0.1+0.1);'  
bool(false)  
ringo:~
```

- ・ 両方1に見えるけど実は違います。
- ・ var_dumpは不正確。

小数のトラブル

```
ringo:~ php -r 'var_dump(0.5+0.5);'  
float(1)  
ringo:~ php -r 'var_dump(0.5+0.1+0.1+0.1+0.1+0.1);'  
float(1) 一見同じ値だけど…  
ringo:~ php -r 'var_dump(0.5+0.5==0.5+0.1+0.1+0.1+0.1+0.1);'  
bool(false)  
ringo:~
```

- ・ 両方1に見えるけど実は違います。
- ・ var_dumpは不正確。

小数のトラブル

```
ringo:~ php -r 'var_dump(0.5+0.5);'  
float(1)  
ringo:~ php -r 'var_dump(0.5+0.1+0.1+0.1+0.1+0.1);'  
float(1)  
ringo:~ php -r 'var_dump(0.5+0.5==0.5+0.1+0.1+0.1+0.1+0.1);'  
bool(false)  
ringo:  
違う！
```

- ・ 両方1に見えるけど実は違います。
- ・ var_dumpは不正確。

小数のトラブル

```
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.5))));'  
string(16) "3ff0000000000000"  
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.1+0.1+0.1+0.1+0.1))));'  
string(16) "3fefffffffffffff"  
ringo:~
```

小数のトラブル

```
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.5))));'  
string(16) "3ff0000000000000"  
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.1+0.1+0.1+0.1+0.1))))';  
string(16) "3fefffffffffffff"  
ringo:~
```

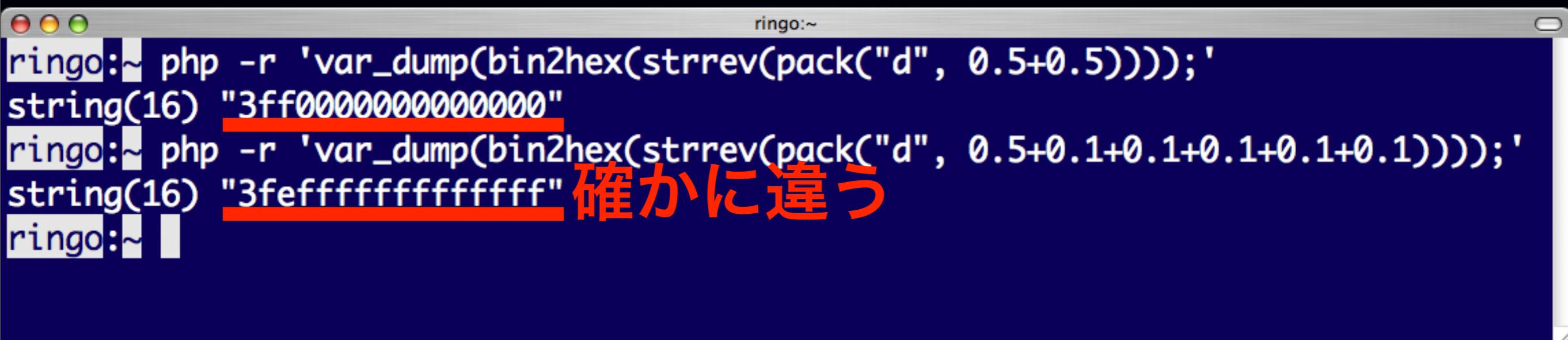
- 0.1を10回足しても1にならない！

小数のトラブル

```
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.5))));'  
string(16) "3ff0000000000000"  
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.1+0.1+0.1+0.1+0.1))));'  
string(16) "3fefffffffffffff" 確かに違う  
ringo:~
```

- 0.1を10回足しても1にならない！

小数のトラブル



```
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.5))));'  
string(16) "3ff0000000000000"  
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.1+0.1+0.1+0.1+0.1))))';  
string(16) "3fefffffffffffff" 確かに違う  
ringo:~
```

A screenshot of a terminal window titled "ringo:~". It shows two lines of PHP code using the var_dump function to convert floating-point numbers to their binary hex representations. The first line adds 0.5 + 0.5, resulting in a hex value of "3ff0000000000000". The second line adds 0.5 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1, resulting in a hex value of "3fefffffffffffff". A red box highlights the second hex value, and the text "確かに違う" (Indeed different) is overlaid in red.

- 0.1を10回足しても1にならない！
- 0.1は2進数では不正確

0.1が不正確？

- 分数と小数の関係
 - $1/3$ を10進小数で表そうとする
と $0.33333\dots$ と無限桁になる
 - 同様に、 $0.1 = 1/10$ を2進小数で
表そうとすると無限桁になる

小数の処理

- 誤差が蓄積することがある
 - 対策は？

小数を扱う(1)

- ・ 整数で扱えないか？
- ・ 例：小数点以下第2位まで表示
- ・ 内部的には100倍して計算
- ・ 出力時に1/100して表示

小数を扱う(2)

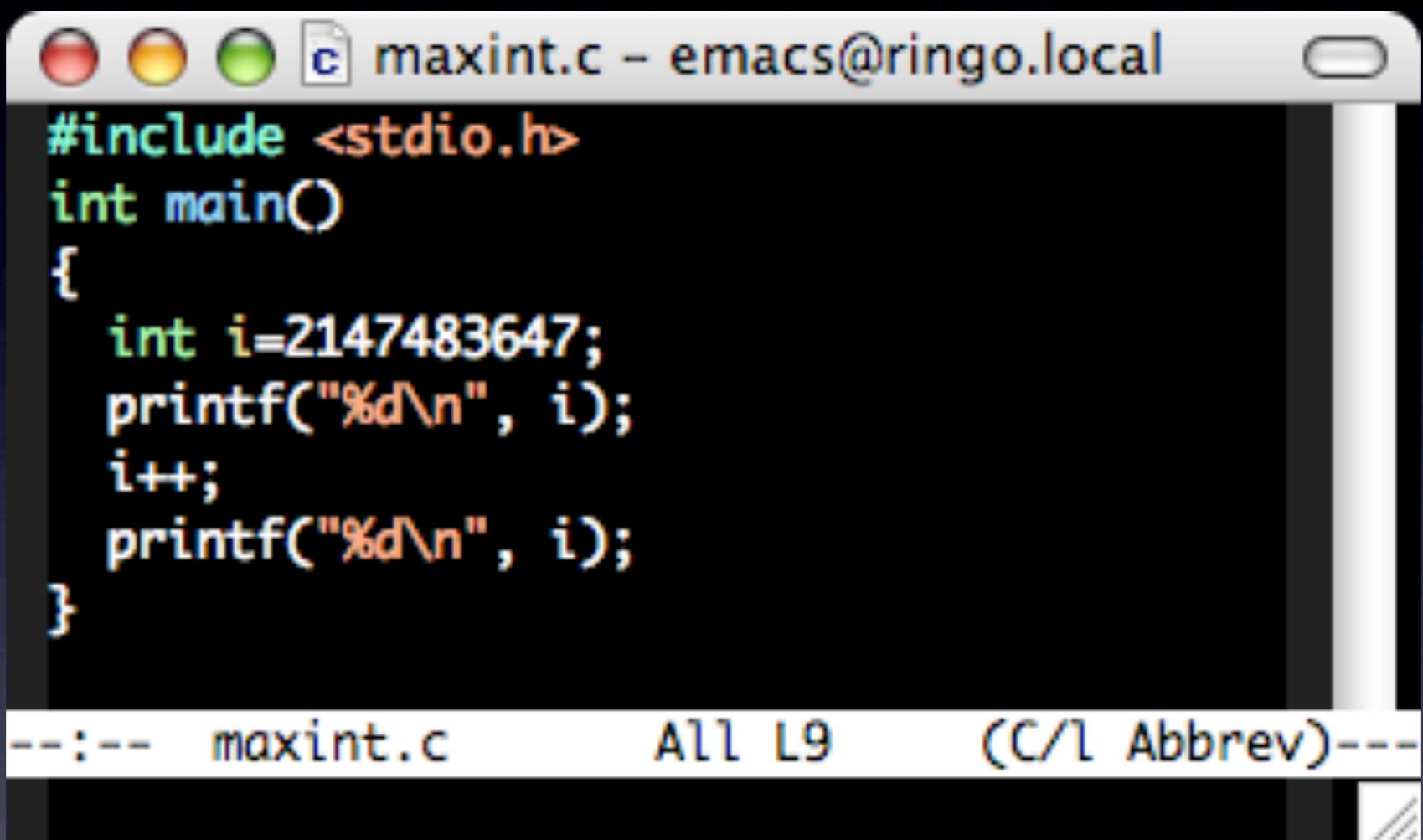
- 誤差を真面目に計算する
- 任せられる人が居て、情報共有も問題なれば。
- 浮動小数点数を扱う限り誤差の話題からは逃れられない

- intとfloat
- PHPでの実例
- その他の注意点
- まとめ

intを超えた場合

- intには範囲があります。
- Q：範囲外の大きさになつたら
何が起こる？

例：Cの場合

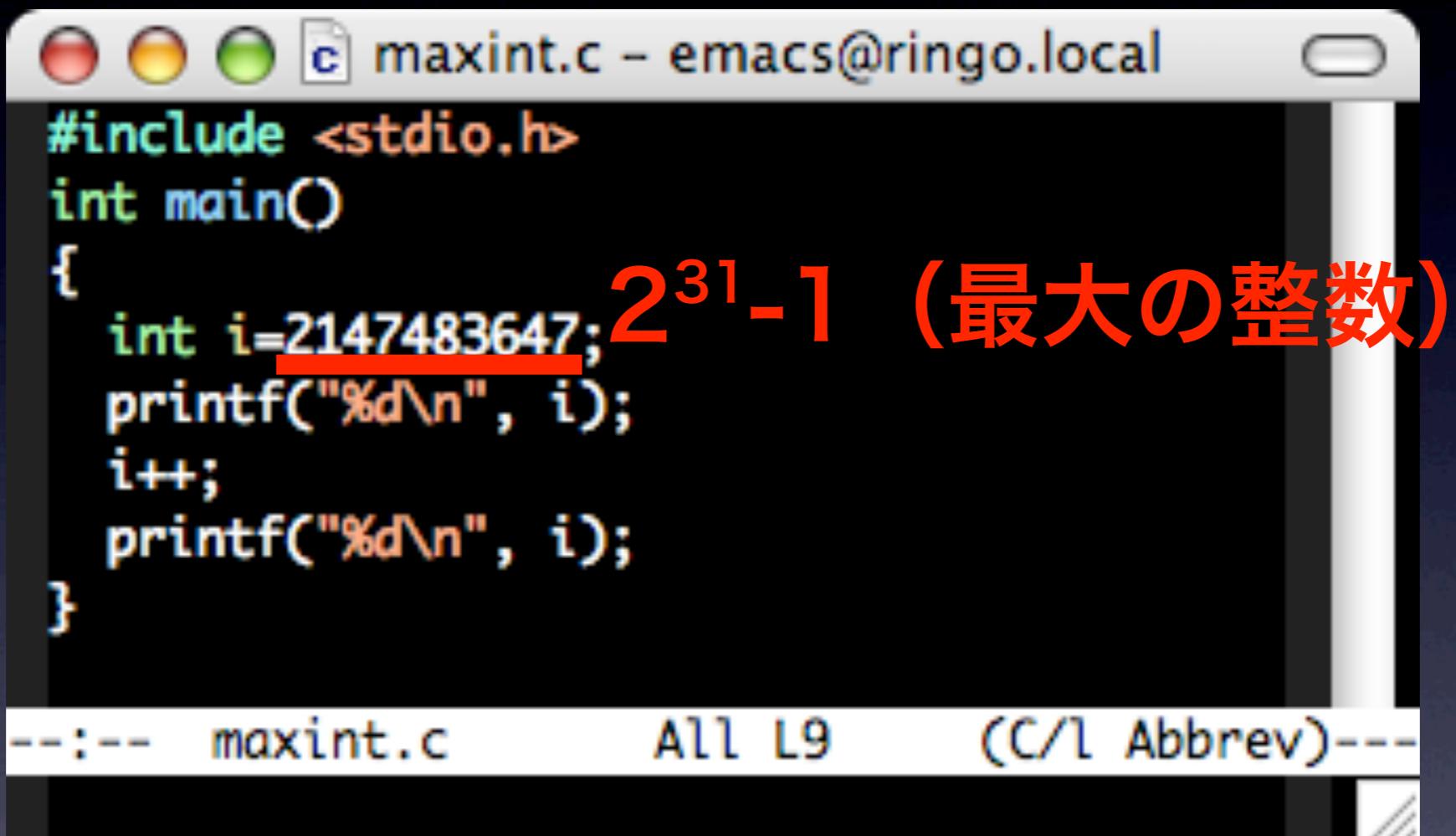


The screenshot shows an Emacs window titled "maxint.c - emacs@ringo.local". The buffer contains the following C code:

```
#include <stdio.h>
int main()
{
    int i=2147483647;
    printf("%d\n", i);
    i++;
    printf("%d\n", i);
}
```

The status bar at the bottom displays the file name "maxint.c", the mode "All L9", and the abbrev mode "(C/L Abbrev)".

例：Cの場合



The screenshot shows an Emacs window titled "maxint.c - emacs@ringo.local". The buffer contains the following C code:

```
#include <stdio.h>
int main()
{
    int i=2147483647;231-1 (最大の整数)
    printf("%d\n", i);
    i++;
    printf("%d\n", i);
}
```

The variable assignment `i=2147483647;` is highlighted with a red underline. To the right of the assignment, the text $2^{31}-1$ is written in red, followed by the Japanese text "(最大の整数)". The status bar at the bottom of the Emacs window shows the file name "maxint.c", mode "All L9", and abbreviations "(C/L Abbrev)".

- 最大のintをインクリメント

例：Cの場合



A screenshot of a Mac OS X terminal window titled "ringo:~/c". The window contains the following text:

```
ringo:~/c gcc -o maxint maxint.c
ringo:~/c ./maxint
2147483647
-2147483648
ringo:~/c |
```

例：Cの場合



A screenshot of a terminal window titled "ringo:~/c". The window contains the following text:

```
ringo:~/c gcc -o maxint maxint.c
ringo:~/c ./maxint
2147483647
-2147483648 -231 (最小の整数)
ringo:~/c
```

The output "-2³¹ (最小の整数)" is highlighted with a red box.

- 最小のintになる

どういうこと？

$2^{31} - 1$

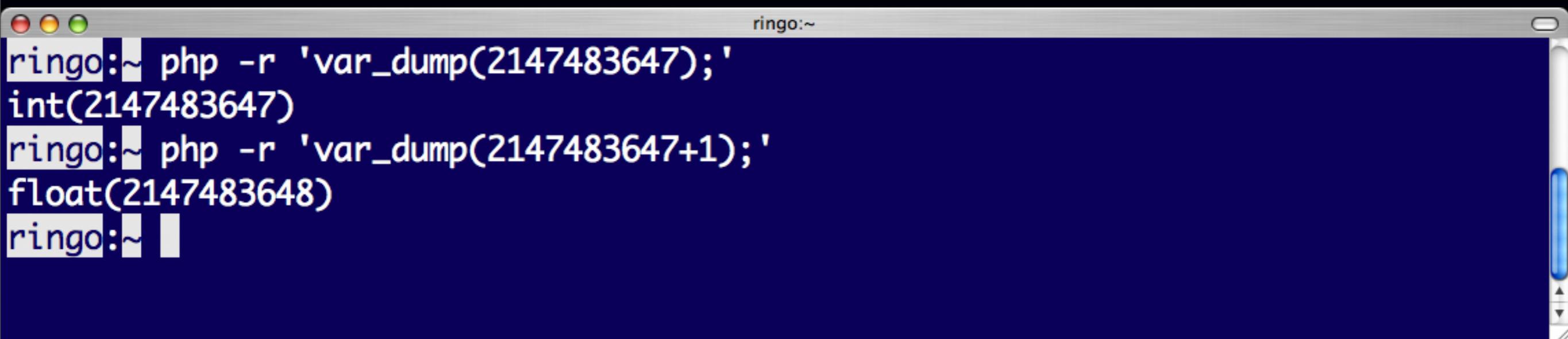
-2³¹

↑↑ 符2³⁰ 号の位

↑↑↑↑↑
168 4 2 1
の の の の
位 位 位 位

- ・繰り上がって符号が反転しただけ

PHPの場合

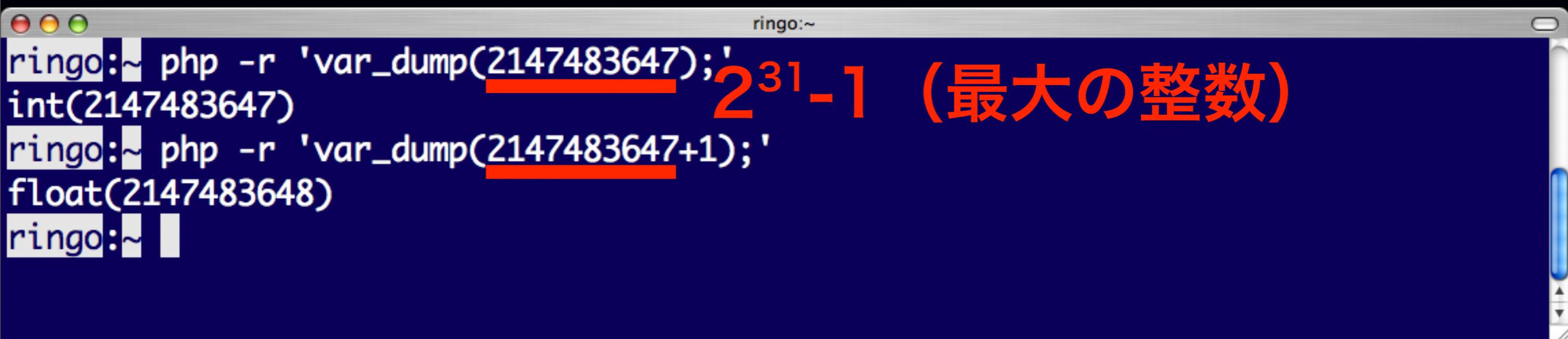


```
ringo:~ php -r 'var_dump(2147483647);'  
int(2147483647)  
ringo:~ php -r 'var_dump(2147483647+1);'  
float(2147483648)  
ringo:~
```

A screenshot of a terminal window titled "ringo:~". It shows two lines of PHP code being run with the command "php -r". The first line outputs "int(2147483647)". The second line outputs "float(2147483648)". The terminal has a dark blue background and standard OS X-style window controls.

- floatになります。
- 数としては（まだ）正確です。

PHPの場合

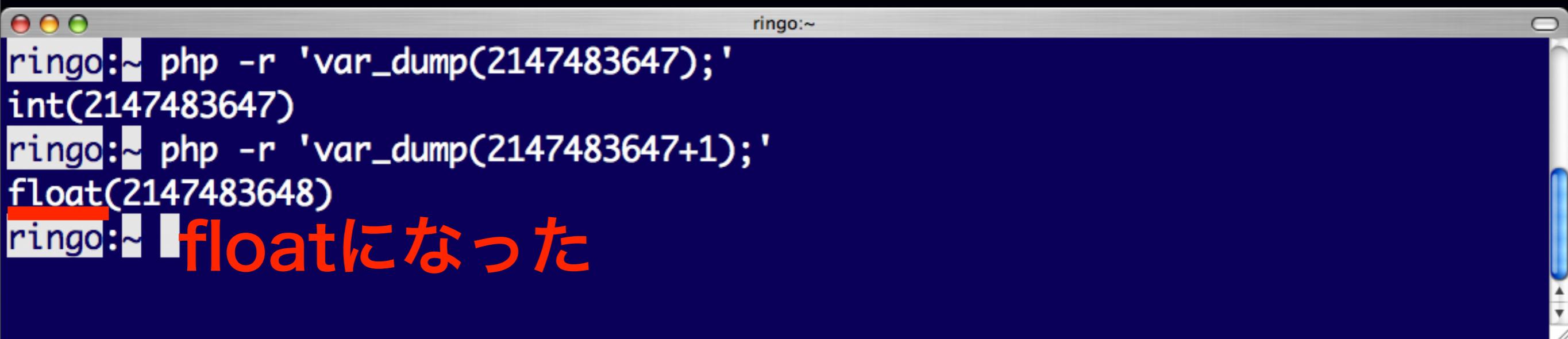


```
ringo:~ php -r 'var_dump(2147483647);'  
int(2147483647) 231-1 (最大の整数)  
ringo:~ php -r 'var_dump(2147483647+1);'  
float(2147483648)  
ringo:~
```

A screenshot of a terminal window titled "ringo:~". It shows two lines of PHP code using the var_dump function. The first line outputs the integer 2147483647 as an int. The second line outputs 2147483648 as a float, with the text "2³¹-1 (最大の整数)" overlaid in red, indicating that the original integer value has been converted to a floating-point number due to PHP's integer overflow handling.

- floatになります。
- 数としては（まだ）正確です。

PHPの場合

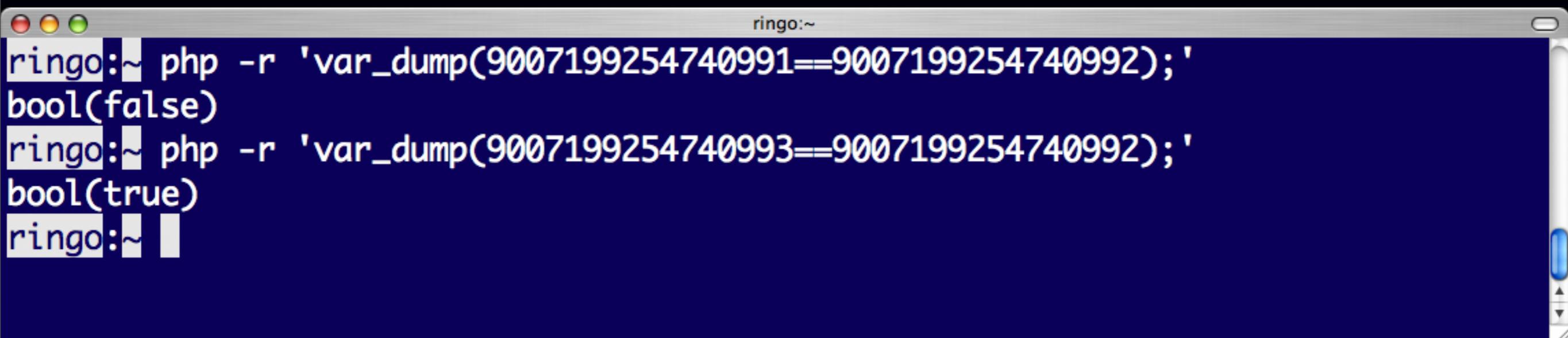


```
ringo:~ php -r 'var_dump(2147483647);'  
int(2147483647)  
ringo:~ php -r 'var_dump(2147483647+1);'  
float(2147483648)  
ringo:~ floatになった
```

A screenshot of a terminal window titled "ringo:~". It shows two lines of PHP code using the -r command and var_dump() function. The first line outputs an integer (int). The second line, which adds 1 to the previous integer, outputs a float. The word "float" is highlighted in red in the terminal output.

- floatになります。
- 数としては（まだ）正確です。

PHPの場合

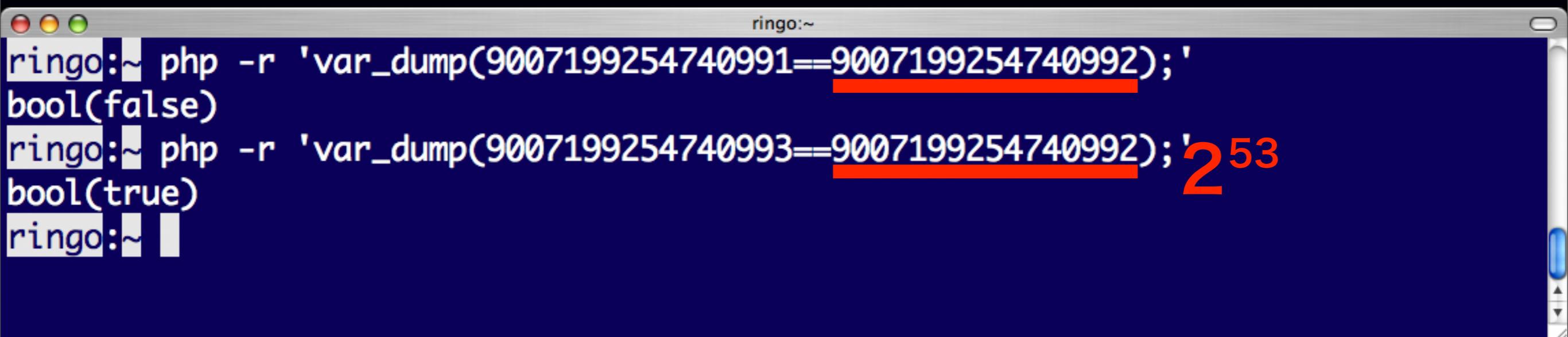


A screenshot of a Mac OS X terminal window titled "ringo:~". The window contains the following text:

```
ringo:~ php -r 'var_dump(9007199254740991==9007199254740992);'  
bool(false)  
ringo:~ php -r 'var_dump(9007199254740993==9007199254740992);'  
bool(true)  
ringo:~
```

- 2^{53} あたりから不正確
 - 途中計算にも注意が必要
 - 仕事で扱うには色々怖いですね

PHPの場合

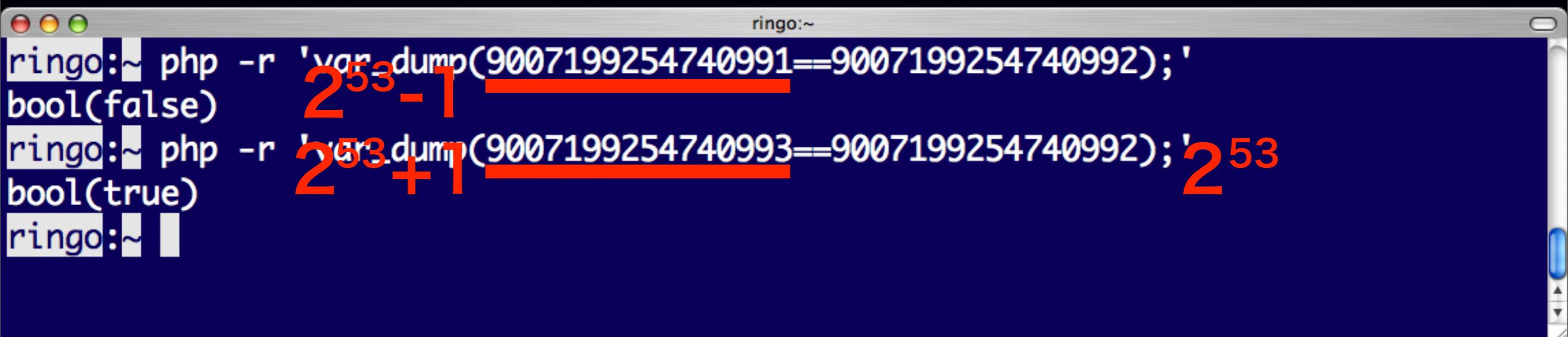


```
ringo:~ php -r 'var_dump(9007199254740991==9007199254740992);'  
bool(false)  
ringo:~ php -r 'var_dump(9007199254740993==9007199254740992);'  
bool(true)  
ringo:~
```

The terminal window shows two executions of PHP code. The first execution compares the integer 9007199254740991 with itself, resulting in a bool(false) output. The second execution compares 9007199254740993 with itself, resulting in a bool(true) output. The number 9007199254740992 is highlighted in red in both comparisons. A large red '2⁵³' is overlaid on the terminal window.

- 2^{53} あたりから不正確
 - 途中計算にも注意が必要
 - 仕事で扱うには色々怖いですね

PHPの場合



```
ringo:~ php -r 'var_dump(9007199254740991==9007199254740992);'  
bool(false)  
253-1  
ringo:~ php -r 'var_dump(9007199254740993==9007199254740992);'  
bool(true)  
ringo:~
```

The screenshot shows a terminal window with the following content:

```
ringo:~ php -r 'var_dump(9007199254740991==9007199254740992);'  
bool(false)  
253-1  
ringo:~ php -r 'var_dump(9007199254740993==9007199254740992);'  
bool(true)  
ringo:~
```

The output shows that the comparison of two numbers (one being $2^{53} - 1$ and the other 2^{53}) results in `bool(false)`, indicating they are not equal. The number $2^{53} + 1$ is also shown, which is greater than $2^{53} - 1$.

- 2^{53} あたりから不正確
 - 途中計算にも注意が必要
 - 仕事で扱うには色々怖いですね

PHPの場合

```
ringo:~ php -r 'var_dump(9007199254740991==9007199254740992);'  
bool(false)  
ringo:~ php -r 'var_dump(9007199254740993==9007199254740992);'  
bool(true)  
ringo:~ 同じ数とみなされている
```

- 2^{53} あたりから不正確
 - 途中計算にも注意が必要
 - 仕事で扱うには色々怖いですね

- intとfloat
- PHPでの実例
- その他の注意点
- まとめ

まとめ

- PHPのint型=Cのint型
- PHPのfloat型=Cのdouble型
 - 10進小数の常識が通用しない世界
 - 大きすぎる数も注意
 - 無限に正確に表せるわけではない

ご清聴
ありがとうございました