これはPerl？　それともRuby？
クイズ〜〜〜〜〜！！！

YAPC::Hakodate LT

@moznion

「Perl書けるならば Ruby書けるよね？」

「Ruby書けるならPerl書けるよね？」

どちらも人権問題であると
広く認識されている

さておき……

クイズ～～～～！！！！！

回答候補

1. Perl
2. Ruby
3. 両方
4. どちらでもない
5. 両方で動くが結果が違う

Q1

```
print "yapc"
```

```
prin两为apc"
```

Q2

```
say "yapc"
```

sayPerl"

Q3

```
print "yapc"
print "hakodate"
```

```
print "yapc"
print "hakodate"
```

Ruby

Q4

```
print '0' ? 'true' : 'false'
```

両方で動くが結果が違う

```
print('hoge' in 'hoge')
```

Q5

```
@list = (1..5);
print @list;
```

```
@list = (1..5);
print @list;
```

両方で動くが結果が違う

Q6

```
$_a = '5';
$_b = '10';
print $_a + $_b;
```

```
$_a = '5';
$_b = '10';
print $_a + $_b;
```

両方で動くが結果が違う

Q7

```
print $_
```

両方で動くが結果が違う

Q8

```
class Example {
}
```

```
class Example {
}
```

Perl

Q9

```
print true
```

両方
`print true`
（Perlのバージョン依存）

```
use v5.40;
print true;
```

Q10

```
class Person
    attr_accessor :name
    attr_accessor :age
end
```

```
class Person
    attr_accessor :name
    attr_accessor :age
end
```

Ruby, ところによりPerl

```
package Filter;

use Filter::Util::Call;

sub import {

    filter_add([]);

}

sub filter {

    my ($self) = @_;

    my $status = filter_read();

    return $status if $status <= 0;

    s/\bclass\s+(\w+)/package $1 {/g;

    s/\bend\b/}/g;

    s/\battr_accessor\s+:(\w+)/sub $1 { my \$self = shift; \@_ ? \$self->{\$1} = shift : \$self->{\$1} }/g;

    return $status;

};
```

Q11

```
attr_accessor 'yapc', 'hakodate'
```

```
attr_accessor :name, :birthdate
```

Ruby, ところにより Perl

```
use Rubyish::Class;
attr_accessor 'yapc', 'hakodate'
```

Q12

```
def add(a, b)
  a + b
end
```

```
def add(a, b)
  a + b
end
```

Ruby, ところによりPerl

```
use Inline::Ruby;
__END__
__Ruby__
def add(a, b)
  a + b
end
```

いかがでしたか？

RubyとPerlって似てますね