Perl 6 For Neophytes

Elizabeth Mattijsen
17 November 2014

euro-t-guide.com - Summer 2006

http://learnxinyminutes.com/docs/perl6/

- Created by **vendethiel**++
- Contains too much info for this presentation
- So please check it out later!
- For now, here's some **code**!
- Well, actually, a **lot**!
- And please **DO ASK** questions!

Variables

```
my $sclr = 42;
my @arry = 1,2,3;
my %hash = a=>1,b=>2;

say $sclr;          # 42
say @arry[2];       # 3
say %hash{'a'}; # 1
```

Quoted Words

```
my @ar_y = 'a','b','c';
my @ar_y = <a b c>;

my %h = a => 1, b => 2;
say %h{'a','b'}; # 1 2
say %h<a b>;     # 1 2
```

Slices

```
my @a = 0..9;
say @a[3..5]; # 3 4 5
@a[3..5] = 42..44;
say @a; # 0 1 2 42 43…

my %h = a=>1,b=>2;
my %h; %h<a b> = 1,2;
```

Interpolation

```
my @ar-y = <a b c>;
say "@ar-y";    # @ar-y
say "@ar-y[]"; # a b c

my %h = a=>1,b=>2;
say "%h";    # %h
say "%h<>"; # a 1 b 2
```

Pairs

```
my $a = z => 42;
say $a.WHAT;   # (Pair)
say $a;        # z => 42
# :zip         zip     => True
# :!delete     delete  => False
# :$bar        bar     => $bar
# :$!attr      attr    => $!attr
# :nd(2)       nd      => 2
# :5th;        th      => 5
```

Adverbs

```
my %h;
%h{'a'..'f'} = 10..15;
%h<a b>;  # 10 11
%h<a b>:exists; # True True
%h<a z>:exists; # True False
%h<a b>:p;  # a=>10 b=>11
%h<a z>:kv; # a 10
%h<a z>:delete:k; # a
say %h; # b => 11, c => 12…
```

Subroutines

```
sub foo { say @_,%_ } # P5
foo( <a b c> ); # a b c
foo( n => 42 ); # n => 42
foo( :n(42) );  # n => 42
foo( <a b>, :n(42) );
  # a b n => 42
```

Signatures

```
sub foo($p,:$n) {say $p,$n}
foo( <a b c> ); # a b c (Any)
foo( 666,:n(42) ); # 666 42
foo( :n(42) ); # DIES

sub foo($p?,:$n!) {say $p,$n}
foo( :n(42) ); # (Any) 42

sub foo(*@_, *%_) { … } # P5
```

Default Parameters

```
sub foo($p = "Helsinki") {
    say "Hello $p";
}
foo("World"); # Hello World
foo();        # Hello Helsinki
foo;          # Hello Helsinki

sub foo($p = die “Must p”) {…}
```

Parameter Aliases

```
sub foo($p) { $p++ }  # DIES
sub bar($p is rw) { $p++ }

bar(42); # DIES
my $a=42; bar $a; say $a; # 43

sub baz(\p) { p++ }
my $b=42; baz $b; say $b; # 43
```

Variable Parameters

```
sub foo($p is copy) { ++$p }


my $a = 42;
say foo($a);   # 43
say $a;        # 42

say foo("z"); # aa
```

Slurpy Parameters

```
sub foo(*@a) { .say for @a }

foo(42);                # 42^N_L
foo(<a b c>);           # a^N_L b^N_L c^N_L
foo(<a b>,<e f>); # a^N_L b^N_L e^N_L f^N_L
foo( $(<a b>) );  # a b^N_L
```

Automatic Signatures

```
sub foo       { $^a.succ }
sub foo($a) {   $a.succ } #SAME


my $z = 42;
say foo($z);   # 43
say $z;        # 42

say foo("z"); # aa
```

Conditionals

```
if $x == 42 { # unless $x != 42
    say "The Answer";
}
else {
    say "$x is not good”;
}

say "The Answer" if $x == 42;
```

Ternaries

```
say $x == 42
   ?? "The Answer"
   !! "$x is not good”;
```

```
say $x == 42
   ?? "The Answer"
   !! $x == 666
      ?? "Number of the Beast"
      !! "$x is not good";
```

Given / When

```
my $x = "this is foo";
given $x {                  # $_ = $x
   when 42                  # $_ ~~ 42
      { say "The Answer" }
   when /foo/               # $_ ~~ /foo/
      { say "Not bar" }
   when .chars>50   # $_ ~~ T/F
      { say "Too long" }
   default { say "Huh?" }
}
```

Looping

```
loop { # forever, until stopped
    last if $done;
}

while $count-- {
    say "Still $count to go";
}

until $done {
    say “Still not done”;
}
```

Old / New Looping

```
loop (my $i=0; $i<5; $i++){
    say "at iteration #$i";
}

for ^5 -> $i { # 0..^5 -> 0..4
    say "at iteration #$i";
}

say "at iteration #$_" for ^5;
```

Iterating

```
for @values -> $v {
    say "processing $v";
    ...             # yada yada yada
    next if $skip;
    redo if $again;
    last if $done;
}
```

Nested Iterations

```
RECORD:
for @records.kv -> $o,$r {
   say "Processing record #$o";
   for @$r -> $field {
      RECORD.last
         if $field eq "DONE";
}
```

Pointy Blocks

```
if ultimate-question() -> $r {
    say "result = $r"; # 42 :-)
}

my $b = -> $x { say $x * 2 }
say $b.WHAT;   # (Block)
$b(21);        # 42

my &c = -> $x = 666 { say $x/2 }
c(84); c() # 42ᴺᴸ333
```

Sorting

```
<1 2 3 10>.sort.say; # 1 10 2 3

<1 2 3 10>.sort( -> $x,$y {
   $x <=> $y
} ).say;                    # 1 2 3 10

<1 2 3 10>.sort( {
   $^b <=> $^a
} ).say;                    # 10 3 2 1
```

Operators

```
say 1 + 2;   # 3
say 1 - 2;   # -1
say 6 * 7;   # 42  etc. etc.

# + - * / ** ~ x xx
# += -= *= /= **= ~= .=
# < > <= >= lt gt le ge
# == ~~ === =:= <=> leg eq gt …
# != !== !~~ !=== !=:=
```

Binding

```
my $a = 42;
say $a;              # 42
my $b := $a;
$b = 666;
say $a;              # 666

say $a =:= $b; # True

my $c ::= $a;
$c = 21;  # FAIL, but NYI :-(
```

Gradual Typing

```
my Int $i = 42;     # ok
my Str $s = 42;     # FAIL

my $a; say $a;  # (Any)

sub foo(Int $a) { … };
foo("42"); # FAIL at compile time
foo($s);   # FAIL at compile time
foo(~$i);  # FAIL at runtime
foo(+$s);  # ok if was "42"
```

Multi Subs

```
multi sub a(Int $i) { "int: $i" }
multi sub a(Str $s) { "str: $s" }

say a(42);     # int: 42
say a("Foo"); # str: Foo
say a(π);      # FAIL
say π.WHAT;    # (Num)

multi sub a(Int $i where * > 100)
 { ... }
```

Lexically Scoped

```
{
  sub a($a) { say $a }
  a(42);  # 42
}
a(666); # FAIL at compile time

{
  use Test;
  ok True, "Success";
}
ok True; # FAIL at compile time
```

Dynamic Variables

```
sub foo() { say $*a // 666 }

{
    my $*a = 42;
    foo();      # 42
}

foo();          # 666

say $*VM.name;  # moar
say ~$*USER;    # liz
```

Roles

```
role Foo {
  has $.foo;
  method foofoo { $.foo ~ $.foo }
}

my $a = Foo.new( foo => 42 );
my Foo $a .= new(:foo(42));  # same

say $a.foo;    # 42
say $a.foofoo; # 4242
```

Classes

```
class Bar does Foo {
  has $.times;
  method ftw { $.foo xx $.times }
}

my $foo = "HEL";
my $a = Bar.new( :$foo, :4times );
say $a.foo;     # HEL
say $a.foofoo; # HELHEL
say $a.ftw;     # HEL HEL HEL HEL
say $a.times;  # 4
```

Support?

- fine folks of the #perl6 channel on irc.freenode.org
- blogs: http://pl6anet.org
- from Perl 5: http://perlgeek.de/en/article/5-to-6
- in Y minutes: http://learnxinyminutes.com/docs/perl6
  - (only done ~ 40% in this presentation)
- the nitty gritty: http://perlcabal.org/syn/

Perl 6 For Neophytes

Questions?

Elizabeth Mattijsen

17 November 2014

euro-t-guide.com - Summer 2006