Sigils
```
$scalar
@array
%hash
&code
```

Major/minor contexts
```
item    list         sink
 Str     flat/slice
 Num     lazy/eager/hyper
 Bool
```

Access Arrays Hashes
```
   whole: @array[]      %hash{}
 element: @array[0]     %hash{'a'}
    (or)                %hash<a>
   slice: @array[0,2]   %hash{'a','b'}
    (or)                %hash<a b>
```

Twigils
```
$normal-lexical
$?compiler-constant
$*dynamic-or-global
$.public-accessor
$!private-attribute
$^positional-param
$:named-parameter
$=pod-info
$<named-match-capture>
$~slang-variable
```

Composers
```
[ ] array
{ } block/hash
< > quotewords
(,) parcel
:() signature
\() capture
```

Automatic dereference
```
&($foo)(1,2)          == $foo(1,2)
@($foo)[1]            == $foo[1]
%($foo){'bar'}        == $foo<bar>
@(@($foo)[1])[2]      == $foo[1][2]
```

Control syntax
```
for LIST { }                    # implicit $_ arg
for LIST -> $a, $b { }          # explicit args
while/until EXPR { }
repeat while/until EXPR { } # do at least once
loop { }   loop (a;b;c) { } # parens required!
if EXPR { } elsif EXPR { } else { }
unless EXPR { }                 # no else allowed!
given EXPR { when EXPR { } default { } }
EXPR if EXPR for LIST;     # list comprehension
next, last, redo                # loop controls
proceed, succeed                # switch controls
```

Operator precedence
```
.method .[] i
++ --
**
unary + - ~ ! ? ^
* / % %% div
+ -
x xx
~
&
| ^
sleep abs sin temp
<=> leg cmp .. but
~~ > == gt eq === eqv !op
&&
|| ^^ // min max
??!! ff
= := op= =>
so not
, :
X Xop Z Zop ...
say die map etc
and
or xor
<== ==>
```

Types
```
Bool Bit Int Rat FatRat UInt Num Complex int32, complex64 etc.
Str Cat Blob Char Byte Codepoint Grapheme Buf buf8 buf32 utf8
IO Mu Any Cool Junction Whatever Match
Parcel Capture Signature
Pair Range Set Bag
KeyHash KeySet KeyBag
Scalar Array Hash Code
Enum Order TrigBase
Block Routine Sub
Method Regex
Failure Exception
Instant Duration
Date DateTime
```

Scope declarators
```
       my   lexical scope
      our   package scope
      has   instance scope
     anon   no scope at all
    state   persistent lexical
  augment   benign parasitic
supersede   deadly parasitic
```

Operator domains
```
Numeric:  ==   !==(!=) + <      >     <=> <=      >=
Stringy:  eq   !eq(ne) ~ lt     gt    leg le      ge
  Value:  eqv  !eqv      before after cmp !after !before
ObjectID: ===  !===
```

Metaoperators
```
[op] reduce listop to A op B op C...
op=  A = A op B
!op  !(A op B)
»op« hyper/vectorize
Zop  zip with op
Xop  cross with op
Rop  reverse args
Sop  sequentialize
```

Links
```
perl6.org
rakudo.org
```

IRC
```
#perl6 irc.freenode.net
#parrot irc.perl.org
```

Regex metachars
```
^ $      string begin/end
^^ $$    line begin/end
+        one or more
*        zero or more
?        zero or one
**1..3   repeat in range
()       capture to $0,$1
[]       no capture
<foo>    subrule
<[]>     character class
|        parallel or
||       serial or
« »      word boundary
```

Regex modifiers
```
:i   ignore case
:m   ignore marks
:g   global
:r   ratchet
:s   sigspace
:4th nth occurrence
:4x  n times
```

Special variables
```
$_      current topic
$/      regex result
$!      error object
@*ARGS  command line
@*INC   include path
%*ENV   environment
$*PID   process id
```

Regex charclasses
```
.  == anychar, \N non \n
\s == <space>, \S non
\d == <digit>, \D non
\w == <+alpha+digit+[_]>
```