

PERL 빠른 참조

변수, 정규식, 파일 I/O, 레퍼런스, 모듈 핵심

기본

```

Hello World
#!/usr/bin/perl
use strict;
use warnings;
print "Hello, World!\n";
say "Hello, World!"; # with use feature 'say';

```

Perl 실행

```

perl script.pl # run a file
perl -e 'print "hi\n"' # run inline
perl -ne 'print' file # process file line by line

```

주석 및 문법

```

# single-line comment
=pod
Multi-line POD documentation
=cut

```

변수

시질

```

$scalar 단일 값 (문자열, 숫자, 레퍼런스)
@array 스칼라의 정렬된 목록
%hash 키-값 쌍
$array[0] 단일 배열 원소 접근 (스칼라 컨텍스트)
$hash{key} 단일 해시 값 접근 (스칼라 컨텍스트)
스칼라 변수

```

```

my $name = "Perl"; # string
my $version = 5.40; # number
my $count = 42; # integer
my $undef; # undefined (undef)
my $combined = "$name v$version"; # interpolation

```

컨텍스트

```

my @arr = (1, 2, 3);
my $count = @arr; # scalar context: 3
my @copy = @arr; # list context: (1, 2, 3)
my $len = scalar @arr; # force scalar context

```

특수 변수

```

$_ 기본 변수 (topic)
@_ 서브루틴 인수
$! 시스템 오류 메시지
$@ eval 오류
$? 프로그램 이름
@ARGV 명령행 인수
%ENV 환경 변수

```

연산자

비교 연산자

```

==, !=, <, >, <=, >= 숫자 비교
eq, ne, lt, gt, le, ge 문자열 비교
<<> 문자열 비교
cmp 문자열 우선순위 연산자 (-1, 0, 1 반환)
<=> 문자열 우선순위 연산자
~ 비트 NOT
!~ 비트 부정 매칭

```

문자열 연산자

```

my $full = "Hello". " ". "World"; # concatenation
my $line = " " x 40; # repetition
my $len = length($full); # 11

```

논리 연산자

```

&& / and 논리 AND (낮은 우선순위: `and`)
|| / or 논리 OR (낮은 우선순위, or)
// 정의-or 정의된 경우 왼쪽 반환)
! / not 논리 NOT
? : 삼항 조건

```

제어 흐름

조건문

```

if ($x > 0) { print "positive\n"; }
elsif ($x == 0) { print "zero\n"; }
else { print "negative\n"; }
print "yes\n" if $condition;
print "no\n" unless $condition; # postfix if

```

반복문

```

for my $i (0..9) { print "$i\n"; }
foreach my $item (@array) { print "$item\n"; }
while ($line = <STDIN>) { chomp $line; }
until ($done) { last if check(); }

```

반복문 제어

```

next 다음 반복문으로 건너뛸 ('continue' 유사)
last 반복문 종료 ('break' 유사)
redo 현재 반복 재시작
next LABEL 레이블된 반복문의 다음 반복으로
last LABEL 레이블된 반복문 종료

```

Given / When

```

use feature 'switch';
given ($status) {
    when ("ok") { say "success"; }
    when ("error") { say "failed"; }
    default { say "unknown"; }
}

```

서브루틴

기본 서브루틴

```

sub greet {
    my ($name) = @_;
    return "Hello, $name!";
}
my $msg = greet("Alice");

```

기본값 및 이름 있는 파라미터

```

sub connect {
    my ($opts) = @_;
    my $host = $opts{host} // "localhost";
    my $port = $opts{port} // 5432;
    return "$host:$port";
}
connect(host => "db.example.com", port => 3306);

```

서브루틴 레퍼런스

```

my $double = sub { return $_[0] * 2; };
print $double->(5); # 10
my @sorted = sort { $a <=> $b } @nums;

```

프로토타입 및 시그니처

```

use feature 'signatures';
sub add($a, $b) { return $a + $b; }
sub greet($name, $greeting = "Hello") {
    return "$greeting, $name!";
}

```

정규식

매칭

```

if ($str =~ /pattern/) { print "matched\n"; }
if ($str =~ /\d+/) { print "number: $1\n"; }
my @matches = ($str =~ /\d+/g); # all matches

```

치환

```

$str =~ s/old/new/; # first occurrence
$str =~ s/old/new/g; # global (all occurrences)
$str =~ s/^\s+|\s+$//g; # trim whitespace
(my $clean = $str) =~ s/\W//g; # non-destructive copy

```

수정자

```

/i 대소문자 무시
/g 전역 (모든 매치)
/m 여러 줄 (`^` 과 `$` 가 줄 경계 매칭)
/s 단일 줄 (`\`이 개행 문자 매칭)
/x 확장 (공백 및 주석 허용)

```

일일 패턴

```

\d, \w 숫자 / 비숫자
\w, \W 단어 / 비단어 문자
\s, \S 공백 / 비공백
\b, \B 단어 경계
(?: ... ) 캡처 없는 그룹
(?<name> ...) 이름 있는 캡처 ('${name}`으로 접근)

```

파일 I/O

열기 및 읽기

```

open(my $fh, '<', 'data.txt') or die "Cannot open: $!";
while (my $line = <$fh>) {
    chomp $line;
    print "$line\n";
}
close($fh);

```

쓰기 및 추가

```

open(my $fh, '>', 'out.txt') or die "Cannot open: $!";
print $fh "Hello\n";
close($fh);
open(my $fh, '>>', 'log.txt') or die "Cannot open: $!";
print $fh "entry\n";
close($fh);

```

전체 파일 읽기

```

use File::Slurp;
my $content = read_file('data.txt');
my @lines = read_file('data.txt', chomp => 1);

```

파일 테스트

```

-e $path 파일 존재 여부
-f $path 일반 파일 여부
-d $path 디렉터리 여부
-r / -w / -x 읽기 / 쓰기 / 실행 가능
-s $path 파일 크기 (바이트, 비어있으면 0)
-z $path 파일 크기가 0

```

배열 및 해시

배열

```

my @arr = (1, 2, 3, 4, 5);
push @arr, 6; # append
my $last = pop @arr; # remove last
my $first = shift @arr; # remove first
unshift @arr, 0; # prepend
my @slice = @arr[1..3]; # slice

```

배열 함수

```

scalar @arr 원소 수
push / pop 원소 추가/제거
shift / unshift 시작에서 제거/추가
splice(@a, 2, 1) 인덱스 2에서 원소 1개 제거
sort @arr 알파벳순 정렬
reverse @arr 순서 뒤집기
grep { /pat/ } @arr 패턴으로 필터링
map { $_ * 2 } @arr 각 원소 변환
join(',', @arr) 문자열로 합치기

```

해시

```

my %user = (name => "Alice", age => 30);
$user{email} = "a@b.com"; # add pair
delete $user{age}; # remove pair
my @keys = keys %user;
my @vals = values %user;

```

해시 반복

```

while (my ($k, $v) = each %hash) {
    print "$k => $v\n";
}
for my $key (sort keys %hash) {
    print "$key: $hash{$key}\n";
}

```

레퍼런스

레퍼런스 생성

```

my $scalar_ref = \$name;
my $array_ref = \@arr;
my $hash_ref = \%hash;
my $anon_arr = [1, 2, 3]; # anonymous array ref
my $anon_hash = {a => 1, b => 2}; # anonymous hash ref

```

역참조

```

print $$scalar_ref; # dereference scalar
print $$array_ref->[0]; # arrow notation
print $hash_ref->{key}; # arrow notation
my %copy = %$array_ref; # dereference to array
my %copy = %$hash_ref; # dereference to hash

```

복잡한 자료구조

```

my @users = (
    { name => "Alice", age => 30 },
    { name => "Bob", age => 25 },
);
print $users[0]->{name}; # "Alice"

```

ref() 함수

```

ref($r) eq 'SCALAR' 스칼라 레퍼런스
ref($r) eq 'ARRAY' 배열 레퍼런스
ref($r) eq 'HASH' 해시 레퍼런스
ref($r) eq 'CODE' 서브루틴 레퍼런스

```

모듈

모듈 사용

```

use strict;
use warnings;
use List::Util qw(sum max min);
use File::Basename;
use Cwd qw(abs_path);

```

모듈 생성

```

# MyModule.pm
package MyModule;
use Exporter 'import';
our @EXPORT_OK = qw(helper);
sub helper { return "help"; }
1; # module must return true

```

주요 코어 모듈

```

List::Util sum, max, min, reduce, any, all
File::Basename basename, dirname, fileparse
File::Path make_path, remove_tree
Getopt::Long 명령행 옵션 파싱
JSON encode_json, decode_json
LWP::Simple get(Surl) - 간단한 HTTP 클라이언트
Data::Dumper 자료구조 디버그 덤프
Carp croak, confess - 더 나은 오류 메시지
CPAN

```

```

cpan install Module::Name # install from CPAN
cpanm Module::Name # cpanminus (faster)
perldoc Module::Name # read module docs

```