

PDL Quick Reference

pip install prompt-declaration-language

pdl examples/hello/hello.pdl

LLM call with current context
model: watsonx/ibm/granite-13b-chat-v2 parameters: temperature: 0.1

LLM call with explicit input
model: watsonx/ibm/granite-13b-chat-v2 parameters: temperature: 0.1 input: array: - role: user content: Hello,

Reading from a file or stdin
read: # optionally, add file name message: Please enter an input. multiline: true # omit to stop at \n

Creating data (v1, v2 can be any block)
text: # outputs "v1v2" - v1 - v2
lastOf: # outputs v2 - v1 - v2
array: # outputs [v1, v2] - v1 - v2
object: # outputs {k1: v1, k2: v2} k1: v1 k2: v2
data: # outputs {k1: v1, model: v2} k1: v1 model: v2 # no LLM call

Including a PDL file
include: ./helper_defs.pdl
Declaring and calling functions
def: add function: x: int y: int return: \${x + y}
call: add args: x: 2 y: 2 pdl_context: [] # optional

Control constructs
if: \${x > 0} then: positive else: non-negative
for: # outputs 2_0_5 i: [1, 0, 1] j: [2, 3, 5] repeat: \${i * j} join: with: _ # optional
repeat: # outputs HiHiHi text: Hi num_iterations: 3
repeat: def: x read: until: \${ (x trim) == "stop" }

Executing code
lang: python code: result = "Hello, world!"

Optional keywords for any block
description: documentation text
def: x # define variable from block
defs: # define multiple variables x: v1 y: v2
role: user # or system or assistant
contribute: [result, context] # or less
parser: json # or jsonl, yaml, regex
spec: type # type specification

spec Types (shorthand for JSON Schema)	
Basic types	str, int, float, bool, null
Arrays	[int]
Objects	{x: int, y: int}
Enums	{enum: [red, green, blue]}

\${...} Expressions (subset of Jinja2)	
Basic values	"hi", 5, 3.1, true, none
Arrays	[1, 2, 3]
Objects	{"x": 4, "y": 5}
Variables	x, y[0], z.f
Operators	+, -, *, /, //, %, **, ~, and, or, not, ==, <, >, in
Tests	x if x is defined else 0
Filters	x default(0)