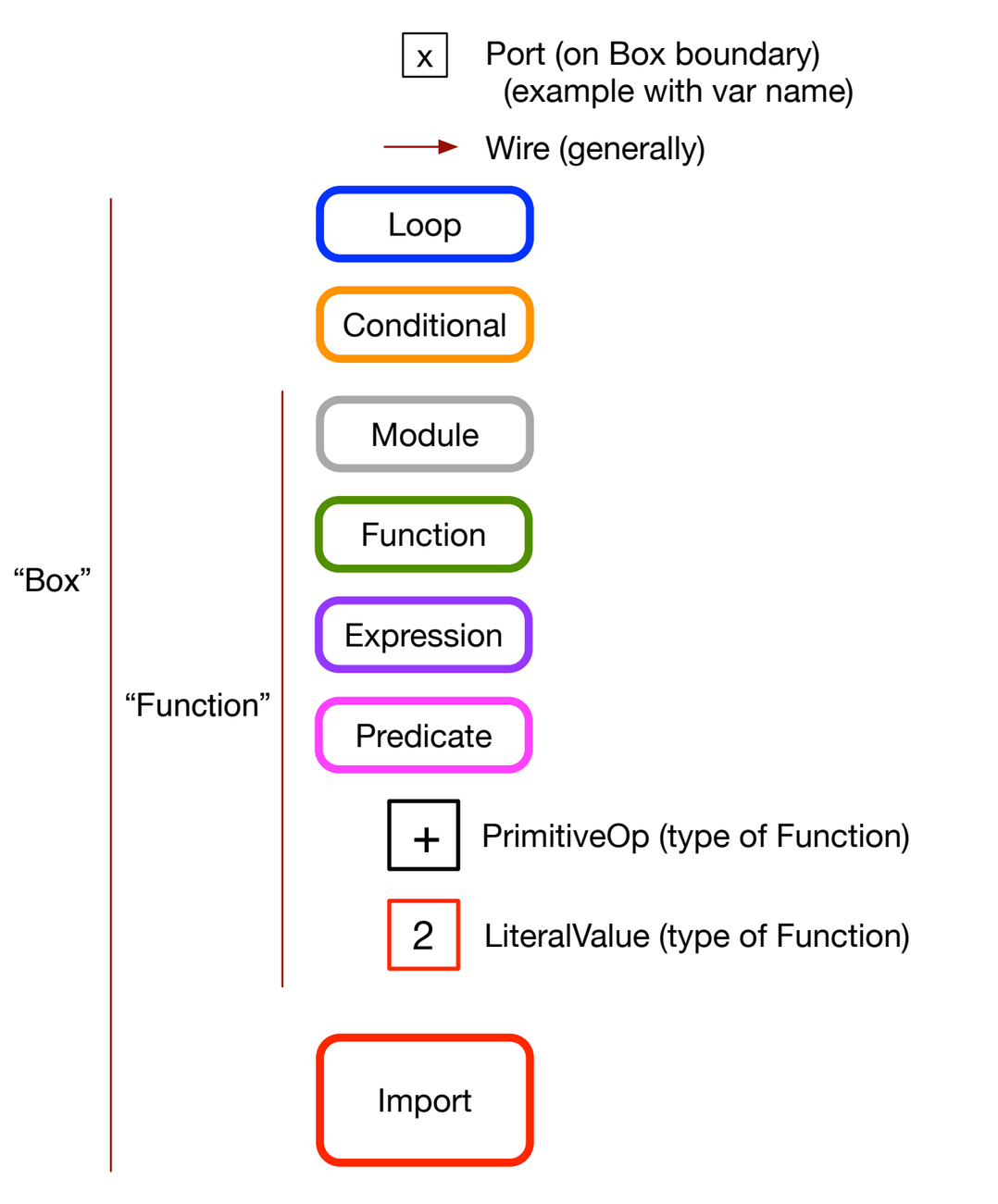
GroMEt $\mathcal{FN}$

GroMET FN Instance Notation:



AnnCAST to GroMET FN translation:

• Note about Wire directionality (where Wires are represented as "solid" arrows): In general, 'src' (tail of arrow) represents an element that will *receive* some value, and the 'tgt' (head of arrow) represents where that value comes from. As a consequence, in terms of "reading data flow", the arrows are "backwards"; think of the arrows as representing "where do values come from".
• "Dashed" arrows represent attribute value references (internally represented as a table index); E.g., the dashed arrows from a Box with ports but now internals is generally the *contents* attribute that holds the Integer (Natural number) index to the FN in the functions_networks table — but. in the drawings we draw the Fns that are referenced.
• Assignments (e.g., x=2) are generally identified with Expressions
• Variable assignments are identified with Output Ports (PortOut and OuterPortOut) of Functions
• LiteralValues are GroMEt LiteralValue (type of Function)
• [GroMEt notation]: Dashed arrows represent the field "contents" reference from a box call to a box function.
• The "name" field of a Port is used when that Port represents the declaration/assignment of a variable that has a named identifier in the original source code. Otherwise, the Port name should be anonymous. This include anonymous Functions/Expressions that are introduced (e.g., for the body of a Loop condition Predicate or a Loop Body Function/ Expression — the Outer Port Input Ports for those are generally all anonymous (not named).
The general rule of thumb here is that *sources of values that have been set* of variables that have corresponding identifiers in the original source code are what gets named, and these are usually either Outer Port Inputs OR Port Outputs of Boxes.
* The "name" field of a Box will generally be set for BoxFunctions that are the **Outer Boxes** of FN that implement
(1) a user-defined function
(2) an expression assigning a variable that has a name (e.g., contents)
(3) a function that is a constructor of a type

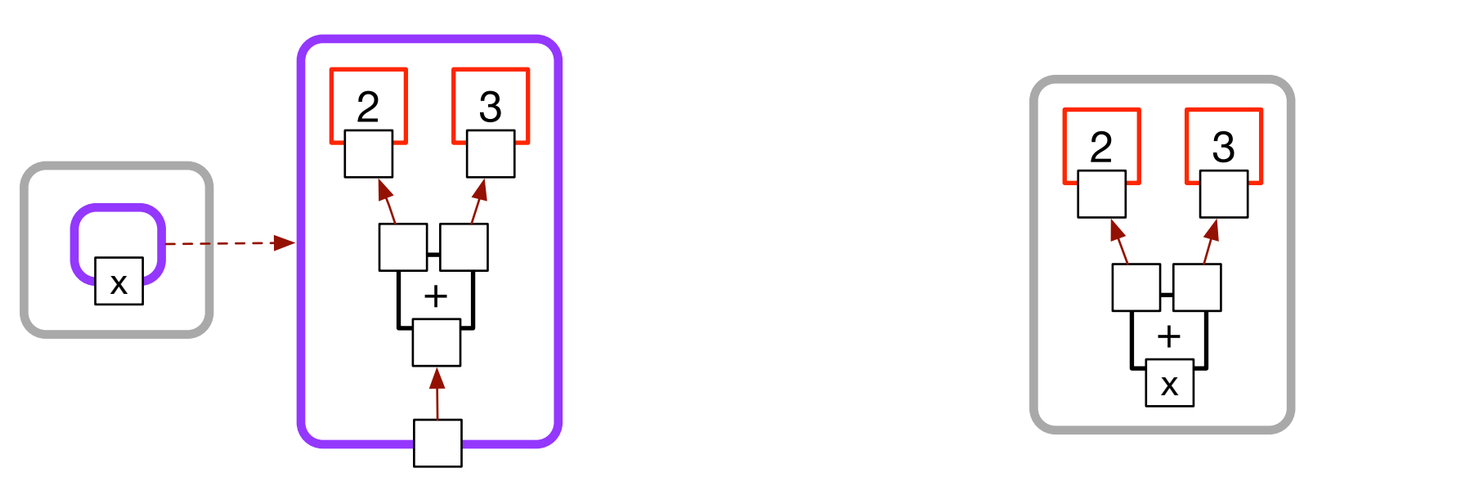
Python | GroMEt | Examples of substitution

exp0.py

```
x = 2
```

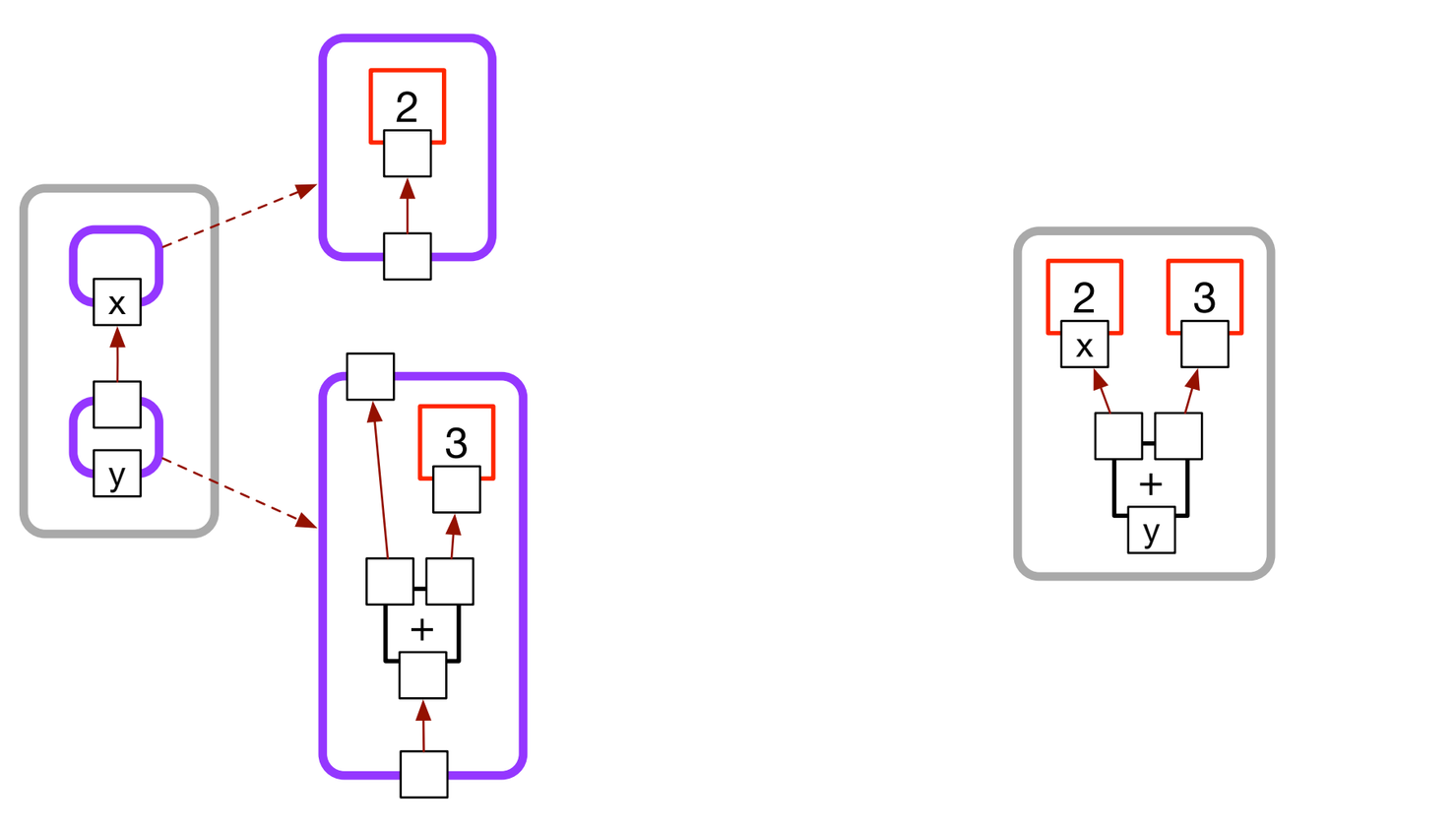
exp1.py

```
x = 2 + 3
```



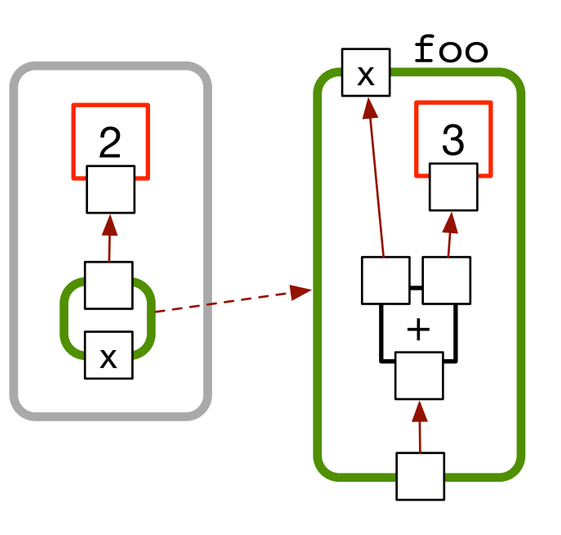
exp2.py

```
x = 2
y = x + 3
```



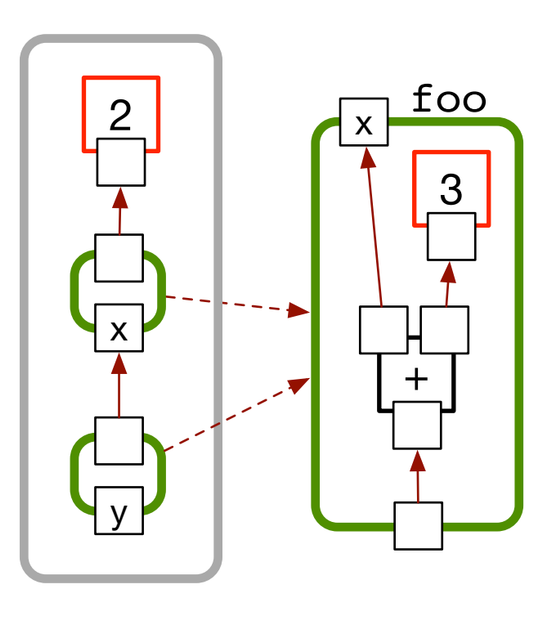
fun1.py

```
def foo(x):
    return x + 3
x = foo(2)
```



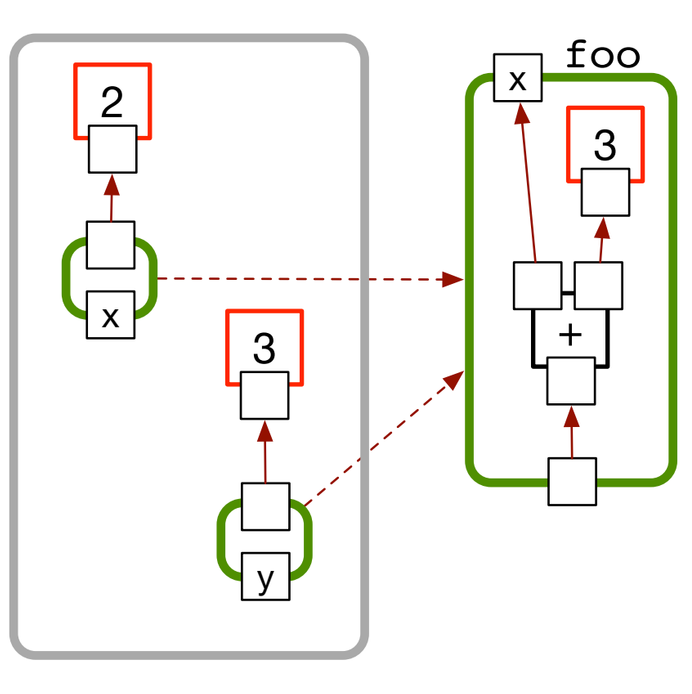
fun2.py

```
def foo(x):
    return x + 3
x = foo(2)
y = foo(x)
```



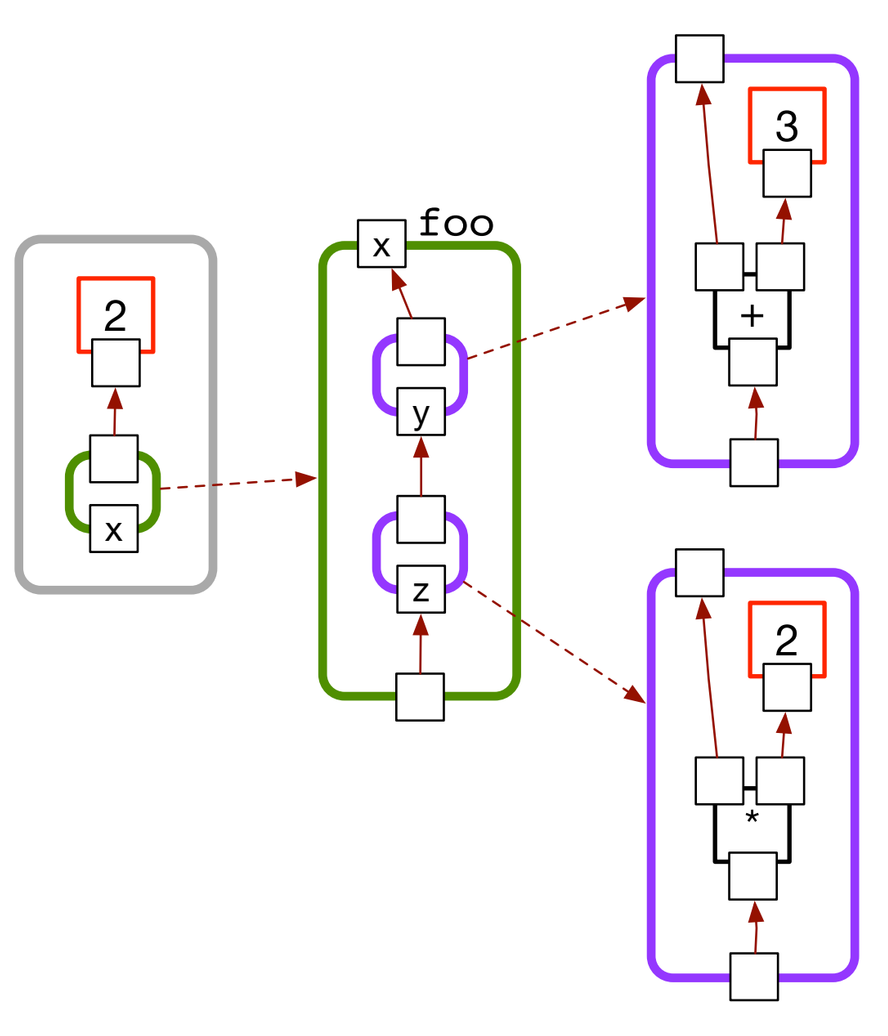
fun3.py

```
def foo(x):
    return x + 3
x = foo(2)
y = foo(3)
```



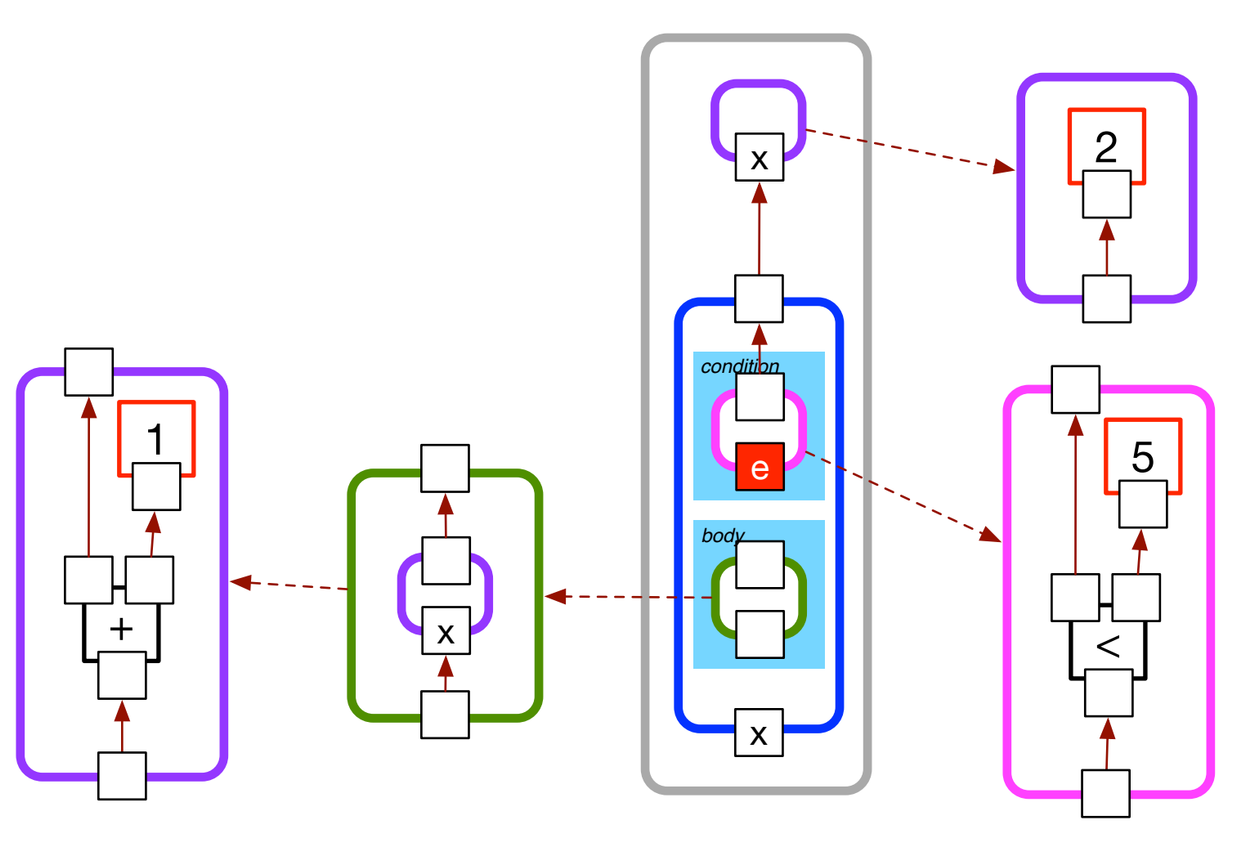
fun4.py

```
def foo(x):
    y = x + 3
    z = y * 2
    return z
x = foo(2)
```



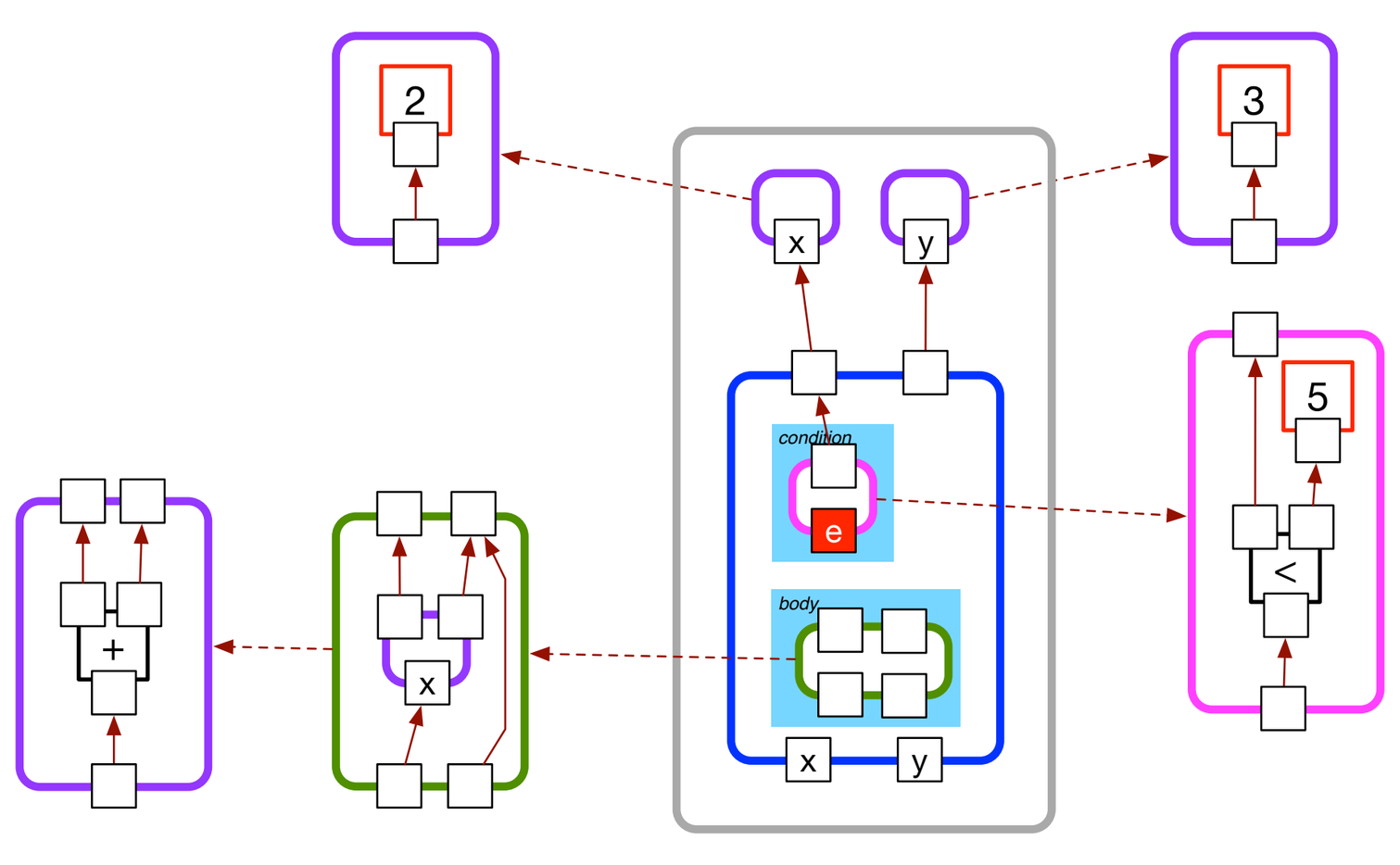
while1.py

```
x = 2
while x < 5:
    x = x + 1
```



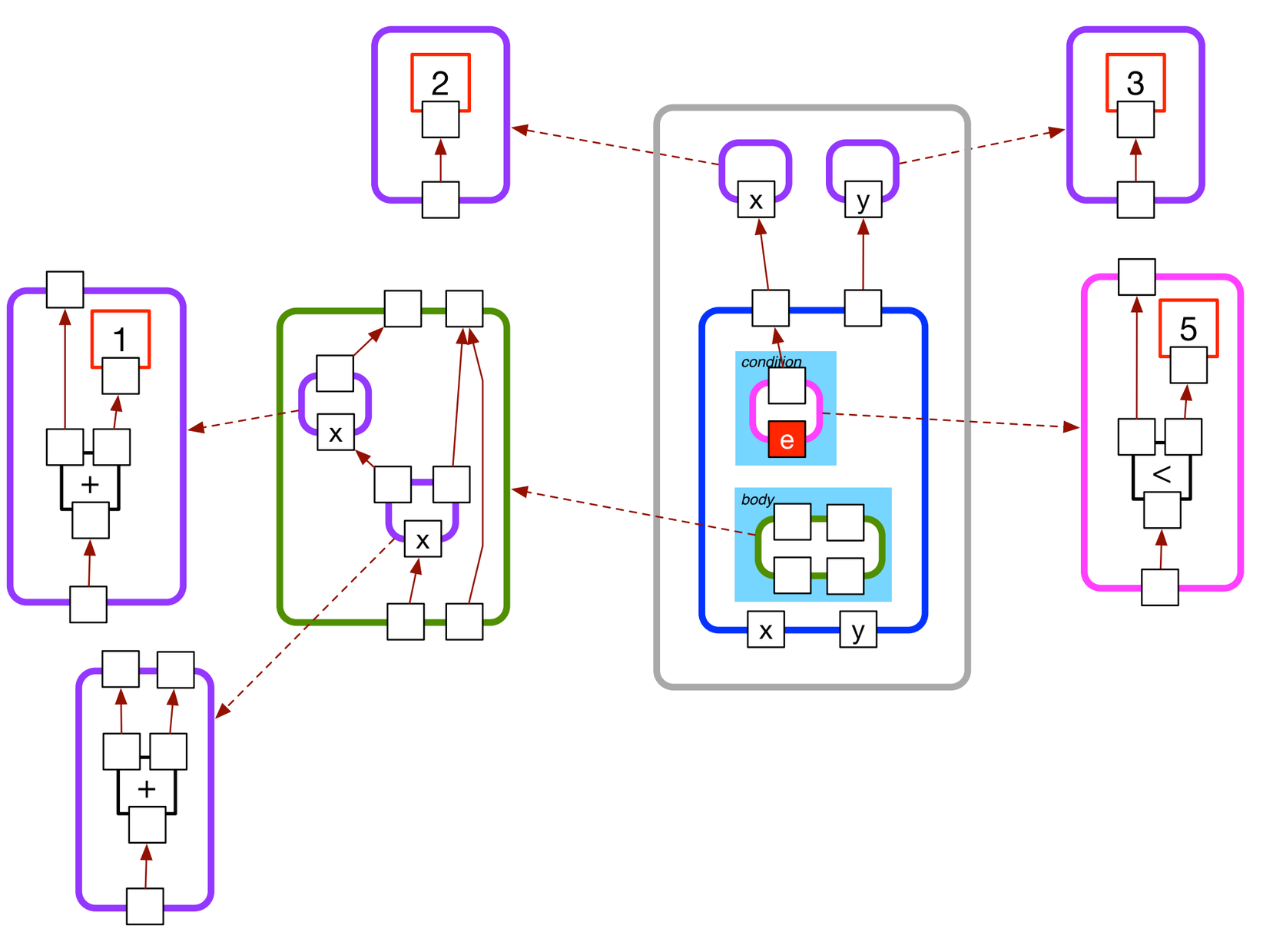
while2.py

```
x = 2
y = 3
while x < 5:
    x = x + y
```



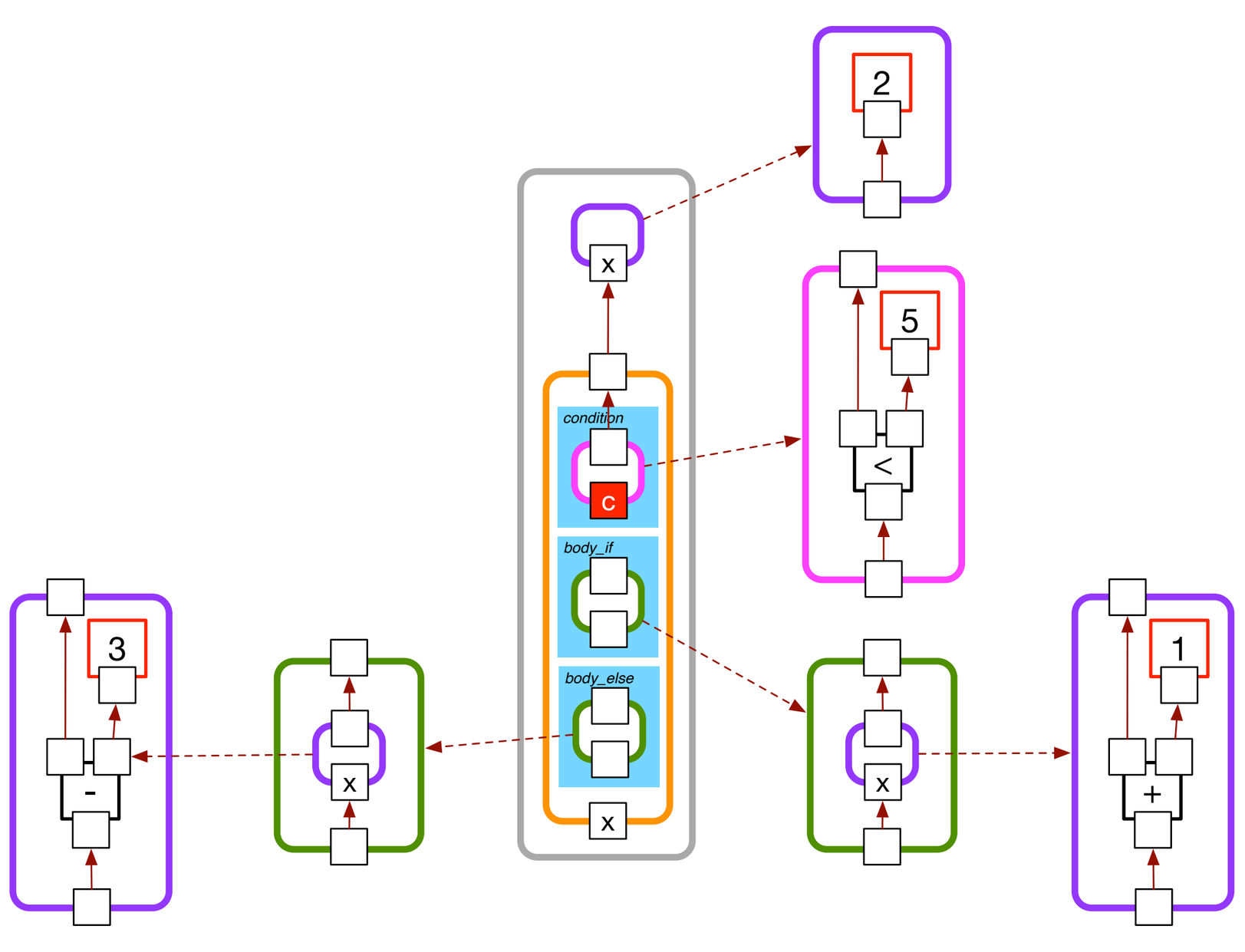
while3.py

```
x = 2
y = 3
while x < 5:
    x = x + 1
    x = x + y
```



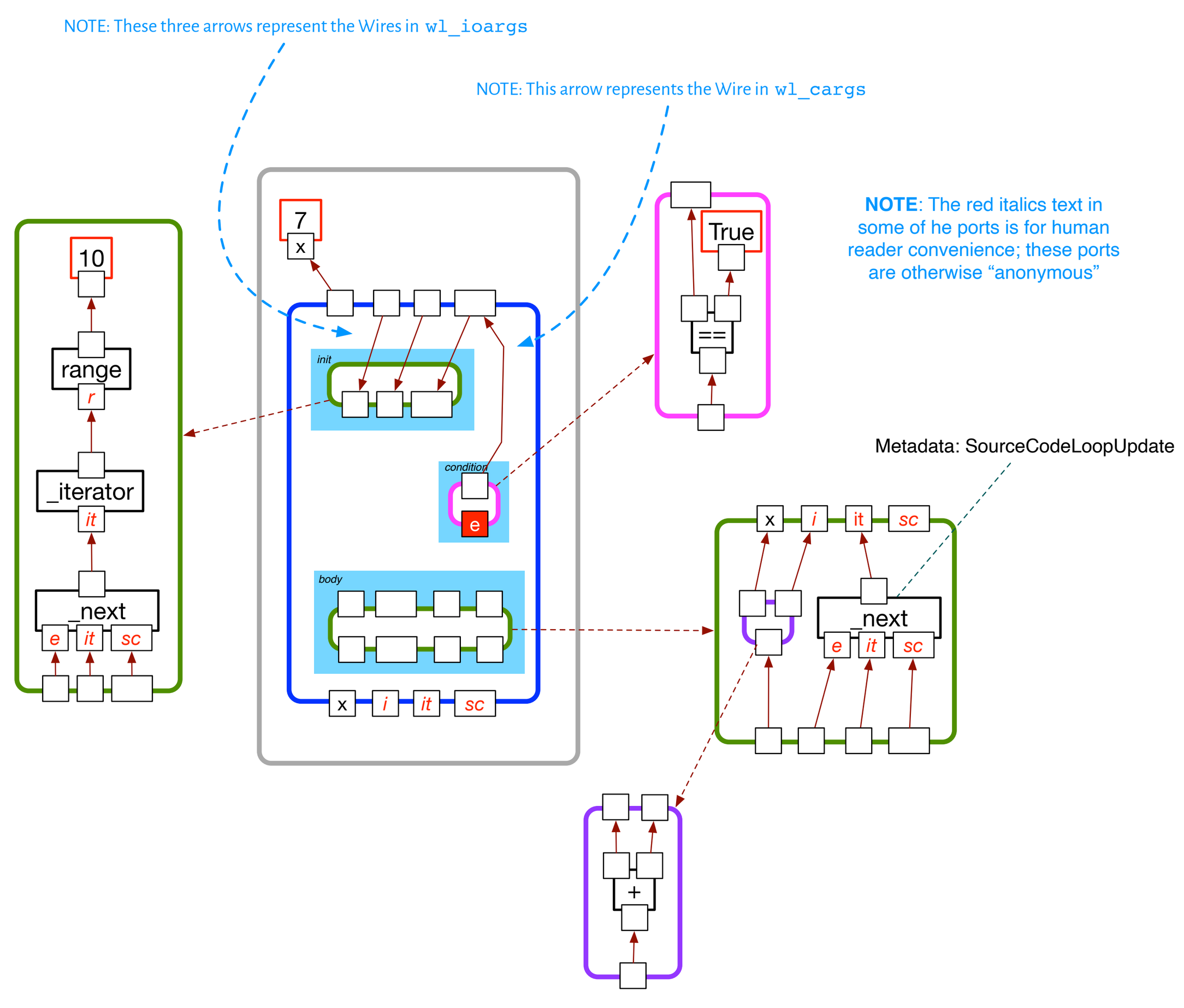
cond1.py

```
x = 2
if x < 5:
    x = x + 1
else:
    x = x - 3
```



for1.py

```
x = 7
for i in range(10):
    x = x + i
```



for2.py

```
x = 7

y = x

for i in range(10):
    x = x + i

print(x)
print(y)
```

import1.py

```
import sys
import numpy

x = numpy.array([1,2,3])
sys.exit()
```