Tail Call “Optimization”

October 11

Tail Call ~~Optimization~~ Implementation

October 11

Rust Demo…

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

```
fact_v1(4)
```

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

```
fact_v1(4)
```

fact_v1(4)

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

fact_v1(4)

```
fact_v1(4) ==> if 4 <= 1: 1 else 4 * fact_v1(3)
```

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

fact_v1(4)

```
fact_v1(4) ==> if 4 <= 1: 1 else 4 * fact_v1(3)
==> 4 * fact_v1(3)
```

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

fact_v1(4)

```
fact_v1(4) ==> if 4 <= 1: 1 else 4 * fact_v1(3)
==> 4 * fact_v1(3)
==> 4 * (if 3 <= 1: 1 else 3 * fact_v1(2))
```

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

fact_v1(4)

```
fact_v1(4) ==> if 4 <= 1: 1 else 4 * fact_v1(3)
==> 4 * fact_v1(3)
==> 4 * (if 3 <= 1: 1 else 3 * fact_v1(2))
==> 4 * (3 * fact_v2(2))
```

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

fact_v1(4)

```
fact_v1(4) ==> if 4 <= 1: 1 else 4 * fact_v1(3)
==> 4 * fact_v1(3)
==> 4 * (if 3 <= 1: 1 else 3 * fact_v1(2))
==> 4 * (3 * fact_v2(2))
==> 4 * (3 * (if 2 <= 1: 1 else 2 * fact_v1(1)))
```

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

fact_v1(4)

```
fact_v1(4) ==> if 4 <= 1: 1 else 4 * fact_v1(3)
==> 4 * fact_v1(3)
==> 4 * (if 3 <= 1: 1 else 3 * fact_v1(2))
==> 4 * (3 * fact_v2(2))
==> 4 * (3 * (if 2 <= 1: 1 else 2 * fact_v1(1)))
==> 4 * (3 * (2 * fact_v1(1)))
```

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

fact_v1(4)

```
fact_v1(4) ==> if 4 <= 1: 1 else 4 * fact_v1(3)
==> 4 * fact_v1(3)
==> 4 * (if 3 <= 1: 1 else 3 * fact_v1(2))
==> 4 * (3 * fact_v2(2))
==> 4 * (3 * (if 2 <= 1: 1 else 2 * fact_v1(1)))
==> 4 * (3 * (2 * fact_v1(1)))
==> 4 * (3 * (2 * (if 1 <= 1: 1 else 1 * fact_v1(0))))
```

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

fact_v1(4)

```
fact_v1(4) ==> if 4 <= 1: 1 else 4 * fact_v1(3)
==> 4 * fact_v1(3)
==> 4 * (if 3 <= 1: 1 else 3 * fact_v1(2))
==> 4 * (3 * fact_v2(2))
==> 4 * (3 * (if 2 <= 1: 1 else 2 * fact_v1(1)))
==> 4 * (3 * (2 * fact_v1(1)))
==> 4 * (3 * (2 * (if 1 <= 1: 1 else 1 * fact_v1(0))))
==> 4 * (3 * (2 * (1)))
```

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

fact_v1(4)

```
fact_v1(4) ==> if 4 <= 1: 1 else 4 * fact_v1(3)
==> 4 * fact_v1(3)
==> 4 * (if 3 <= 1: 1 else 3 * fact_v1(2))
==> 4 * (3 * fact_v2(2))
==> 4 * (3 * (if 2 <= 1: 1 else 2 * fact_v1(1)))
==> 4 * (3 * (2 * fact_v1(1)))
==> 4 * (3 * (2 * (if 1 <= 1: 1 else 1 * fact_v1(0))))
==> 4 * (3 * (2 * (1)))
==> 4 * (3 * 2)
```

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

fact_v1(4)

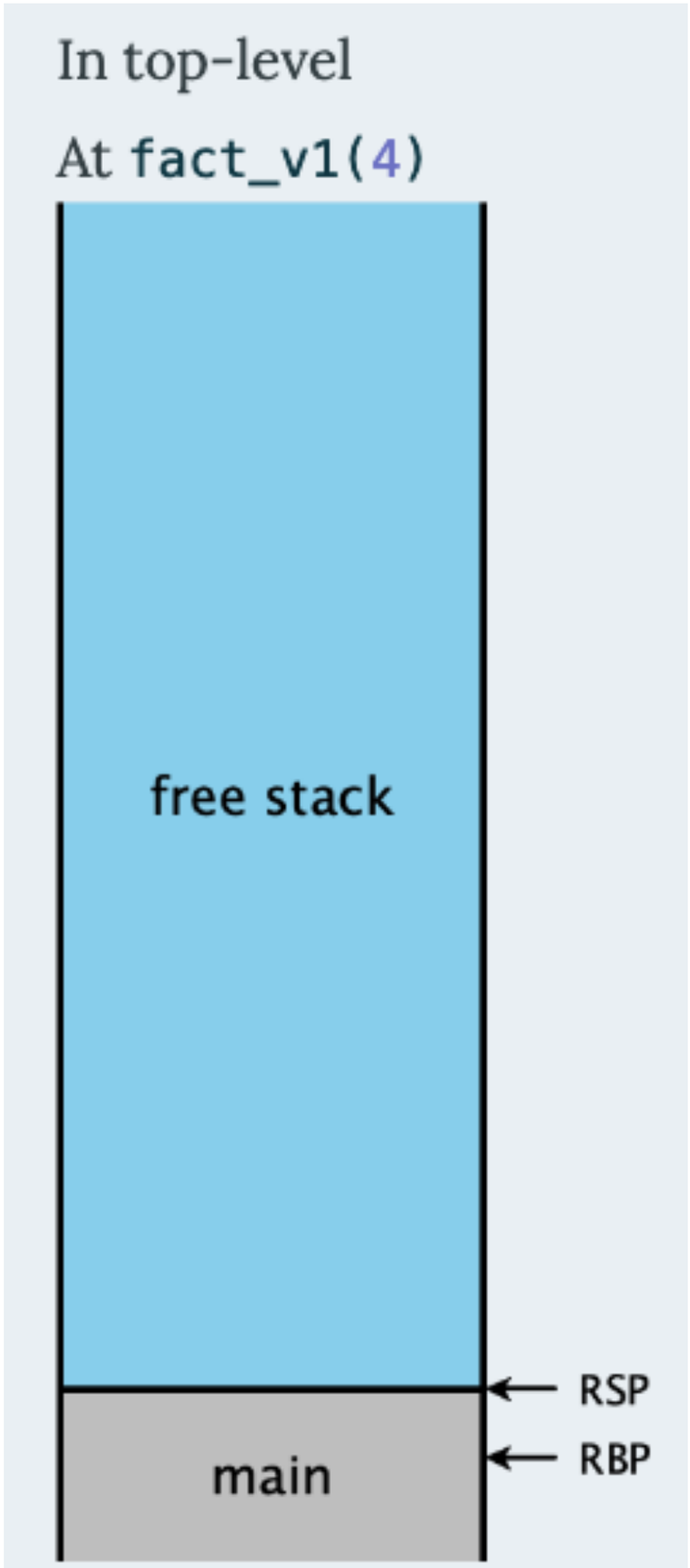
```
fact_v1(4) ==> if 4 <= 1: 1 else 4 * fact_v1(3)
==> 4 * fact_v1(3)
==> 4 * (if 3 <= 1: 1 else 3 * fact_v1(2))
==> 4 * (3 * fact_v2(2))
==> 4 * (3 * (if 2 <= 1: 1 else 2 * fact_v1(1)))
==> 4 * (3 * (2 * fact_v1(1)))
==> 4 * (3 * (2 * (if 1 <= 1: 1 else 1 * fact_v1(0))))
==> 4 * (3 * (2 * (1)))
==> 4 * (3 * 2)
==> 4 * 6
```

```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end
```

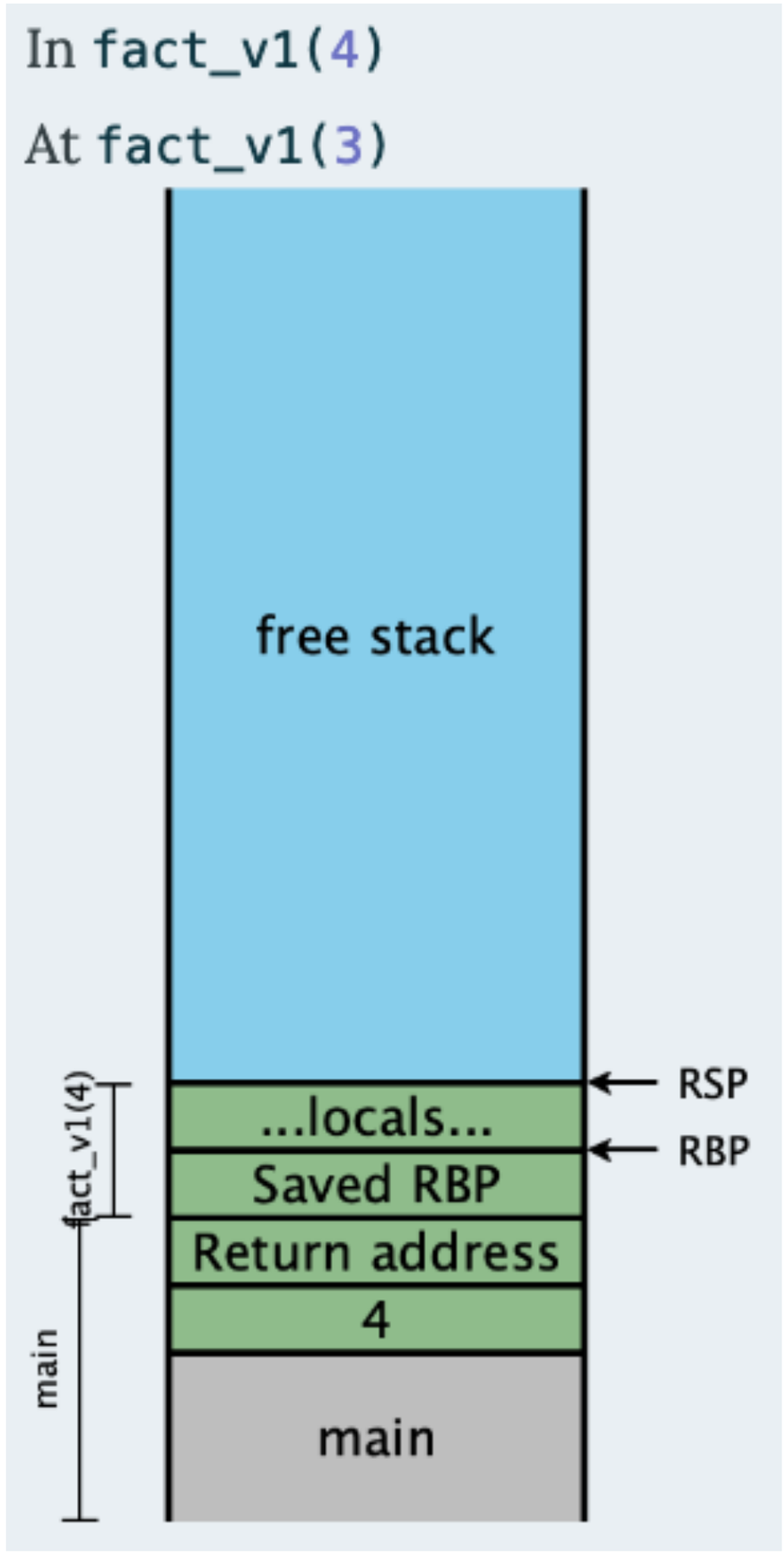
fact_v1(4)

```
fact_v1(4) ==> if 4 <= 1: 1 else 4 * fact_v1(3)
           ==> 4 * fact_v1(3)
           ==> 4 * (if 3 <= 1: 1 else 3 * fact_v1(2))
           ==> 4 * (3 * fact_v2(2))
           ==> 4 * (3 * (if 2 <= 1: 1 else 2 * fact_v1(1)))
           ==> 4 * (3 * (2 * fact_v1(1)))
           ==> 4 * (3 * (2 * (if 1 <= 1: 1 else 1 * fact_v1(0))))
           ==> 4 * (3 * (2 * (1)))
           ==> 4 * (3 * 2)
           ==> 4 * 6
           ==> 24
```

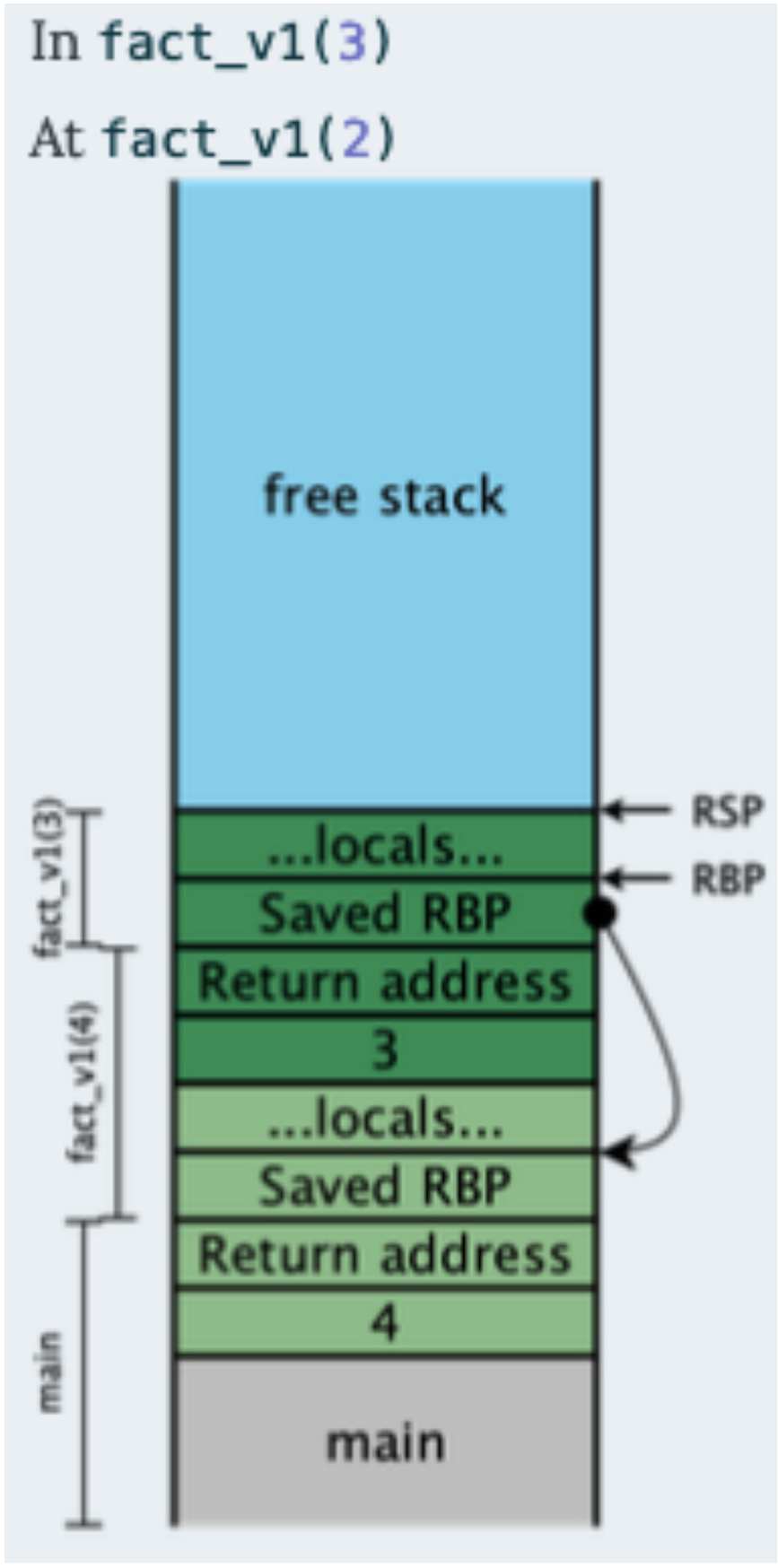
```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end

fact_v1(4)
```



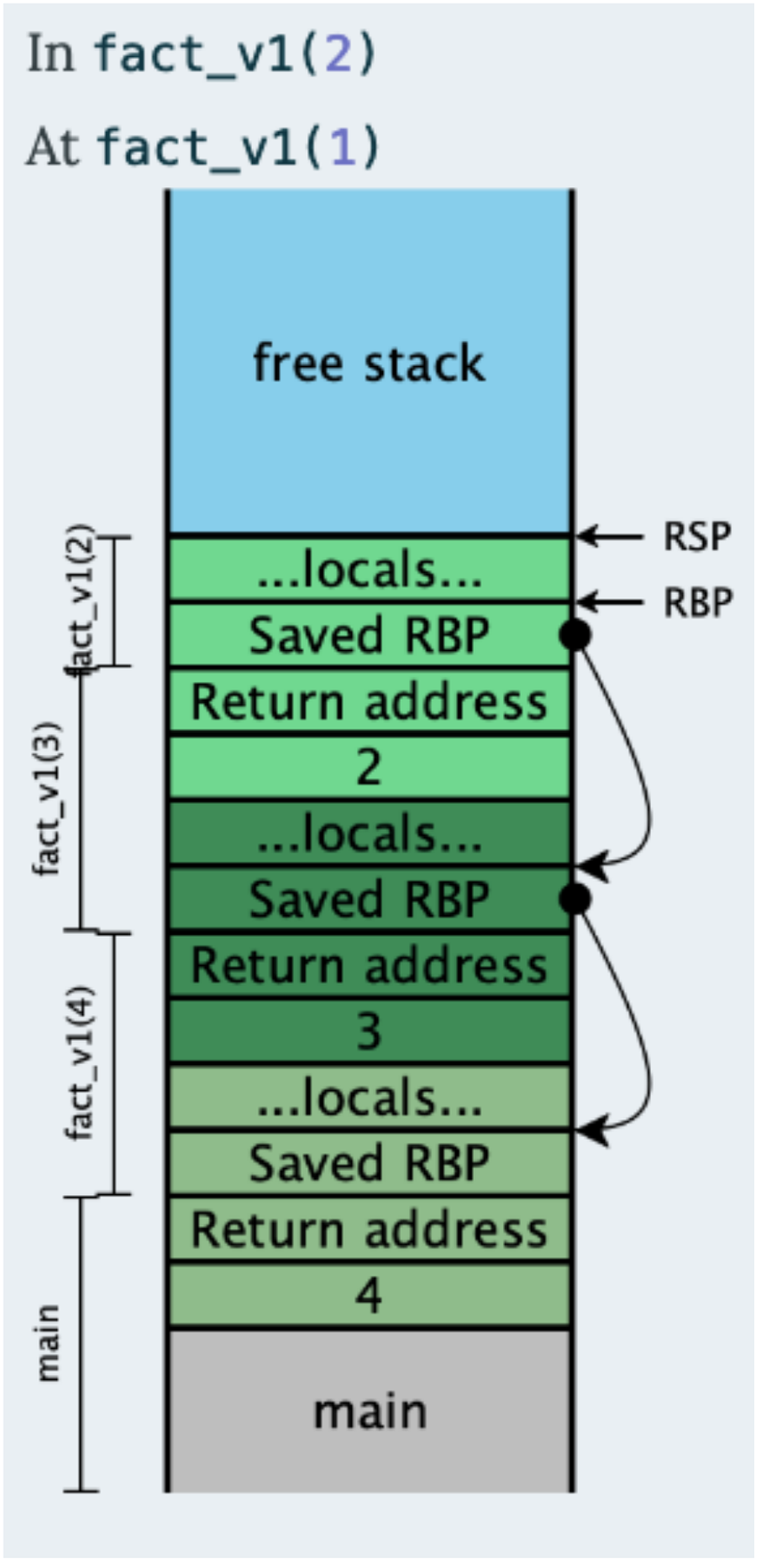
```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end

fact_v1(4)
```



```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end

fact_v1(4)
```

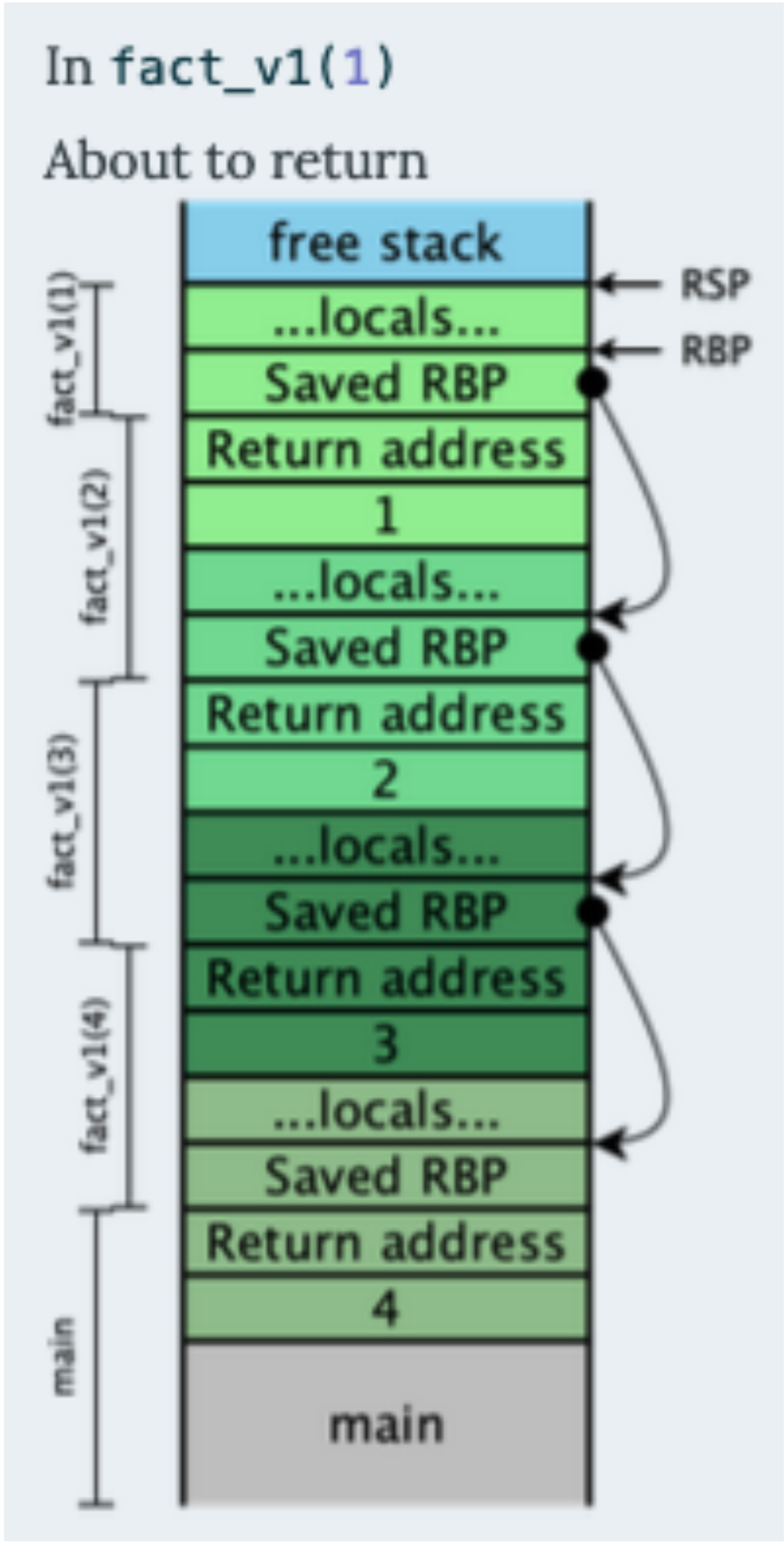


```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end

fact_v1(4)
```



```
def fact_v1(n):
  if n <= 1:
    1
  else:
    n * fact_v1(n - 1)
end

fact_v1(4)
```

```
  4 * (3 * (2 * 1))
```



```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = n * acc in
    fact_tail(n - 1, acc)
end
```

```
def fact_v2(n):
  fact_tail(n, 1)
end


fact_v2(4)
```

```
fact_v2(4)
```

```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = n * acc in
    fact_tail(n - 1, acc)
end
```

```
def fact_v2(n):
  fact_tail(n, 1)
end


fact_v2(4)
```

```
fact_v2(4) ==> fact_tail(4, 1)
```

```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = n * acc in
    fact_tail(n - 1, acc)
end
```

```
def fact_v2(n):
  fact_tail(n, 1)
end


fact_v2(4)
```

```
fact_v2(4) ==> fact_tail(4, 1)
           ==> if 4 <= 1: 1 else fact-tail(4 - 1, 4 * 1)
```

```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = n * acc in
    fact_tail(n - 1, acc)
end
```

```
def fact_v2(n):
  fact_tail(n, 1)
end


fact_v2(4)
```

```
fact_v2(4) ==> fact_tail(4, 1)
           ==> if 4 <= 1: 1 else fact-tail(4 - 1, 4 * 1)
           ==> fact_tail(3, 4)
```

```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = n * acc in
    fact_tail(n - 1, acc)
end
```

```
def fact_v2(n):
  fact_tail(n, 1)
end


fact_v2(4)
```

```
fact_v2(4) ==> fact_tail(4, 1)
           ==> if 4 <= 1: 1 else fact-tail(4 - 1, 4 * 1)
           ==> fact_tail(3, 4)
           ==> if 3 <= 1: 4 else fact-tail(3 - 1, 3 * 4)
```

```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = n * acc in
    fact_tail(n - 1, acc)
end
```

```
def fact_v2(n):
  fact_tail(n, 1)
end


fact_v2(4)
```

```
fact_v2(4) ==> fact_tail(4, 1)
           ==> if 4 <= 1: 1 else fact-tail(4 - 1, 4 * 1)
           ==> fact_tail(3, 4)
           ==> if 3 <= 1: 4 else fact-tail(3 - 1, 3 * 4)
           ==> fact_tail(2, 12)
```

```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = n * acc in
    fact_tail(n - 1, acc)
end
```

```
def fact_v2(n):
  fact_tail(n, 1)
end


fact_v2(4)
```

```
fact_v2(4) ==> fact_tail(4, 1)
           ==> if 4 <= 1: 1 else fact-tail(4 - 1, 4 * 1)
           ==> fact_tail(3, 4)
           ==> if 3 <= 1: 4 else fact-tail(3 - 1, 3 * 4)
           ==> fact_tail(2, 12)
           ==> if 2 <= 1: 12 else fact-tail(2 - 1, 2 * 12)
```

```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = n * acc in
    fact_tail(n - 1, acc)
end
```

```
def fact_v2(n):
  fact_tail(n, 1)
end


fact_v2(4)
```

```
fact_v2(4) ==> fact_tail(4, 1)
           ==> if 4 <= 1: 1 else fact-tail(4 - 1, 4 * 1)
           ==> fact_tail(3, 4)
           ==> if 3 <= 1: 4 else fact-tail(3 - 1, 3 * 4)
           ==> fact_tail(2, 12)
           ==> if 2 <= 1: 12 else fact-tail(2 - 1, 2 * 12)
           ==> fact_tail(1, 24)
```

```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = n * acc in
    fact_tail(n - 1, acc)
end
```

```
def fact_v2(n):
  fact_tail(n, 1)
end


fact_v2(4)
```

```
fact_v2(4) ==> fact_tail(4, 1)
           ==> if 4 <= 1: 1 else fact-tail(4 - 1, 4 * 1)
           ==> fact_tail(3, 4)
           ==> if 3 <= 1: 4 else fact-tail(3 - 1, 3 * 4)
           ==> fact_tail(2, 12)
           ==> if 2 <= 1: 12 else fact-tail(2 - 1, 2 * 12)
           ==> fact_tail(1, 24)
           ==> if 1 <= 1: 24 else fact-tail(1 - 1, 1 * 24)
```

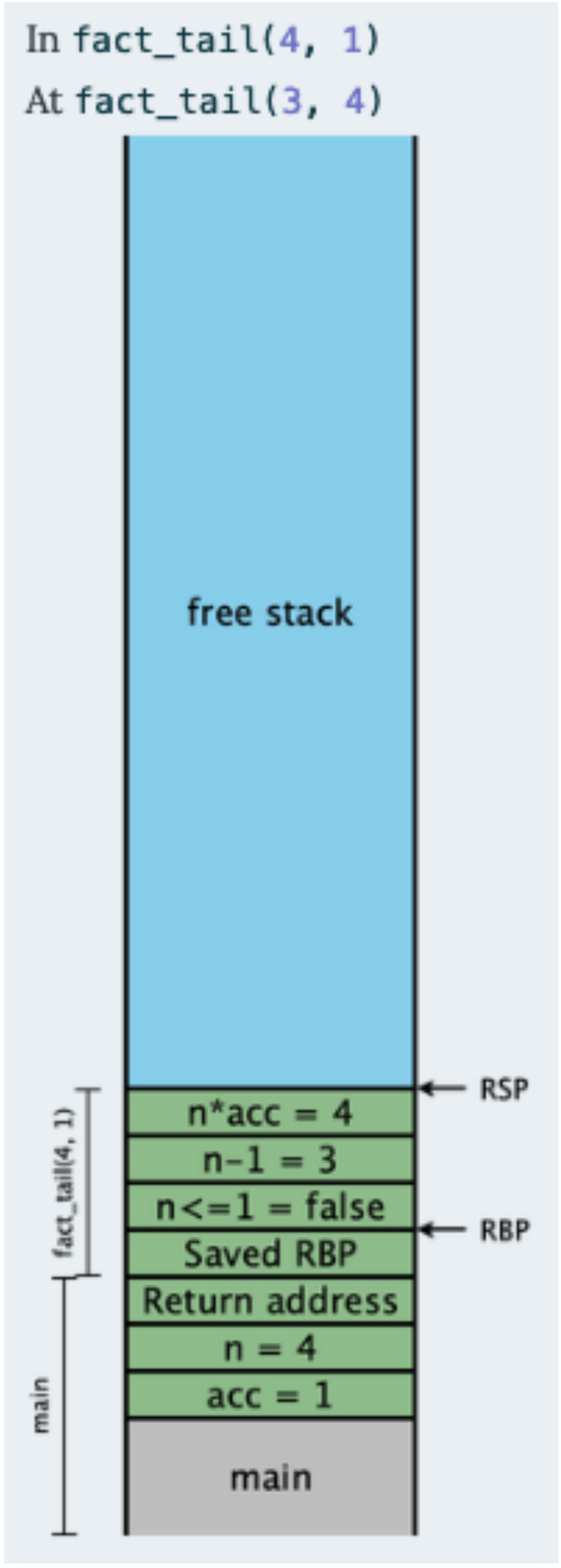
```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = n * acc in
    fact_tail(n - 1, acc)
end
```

```
def fact_v2(n):
  fact_tail(n, 1)
end


fact_v2(4)
```

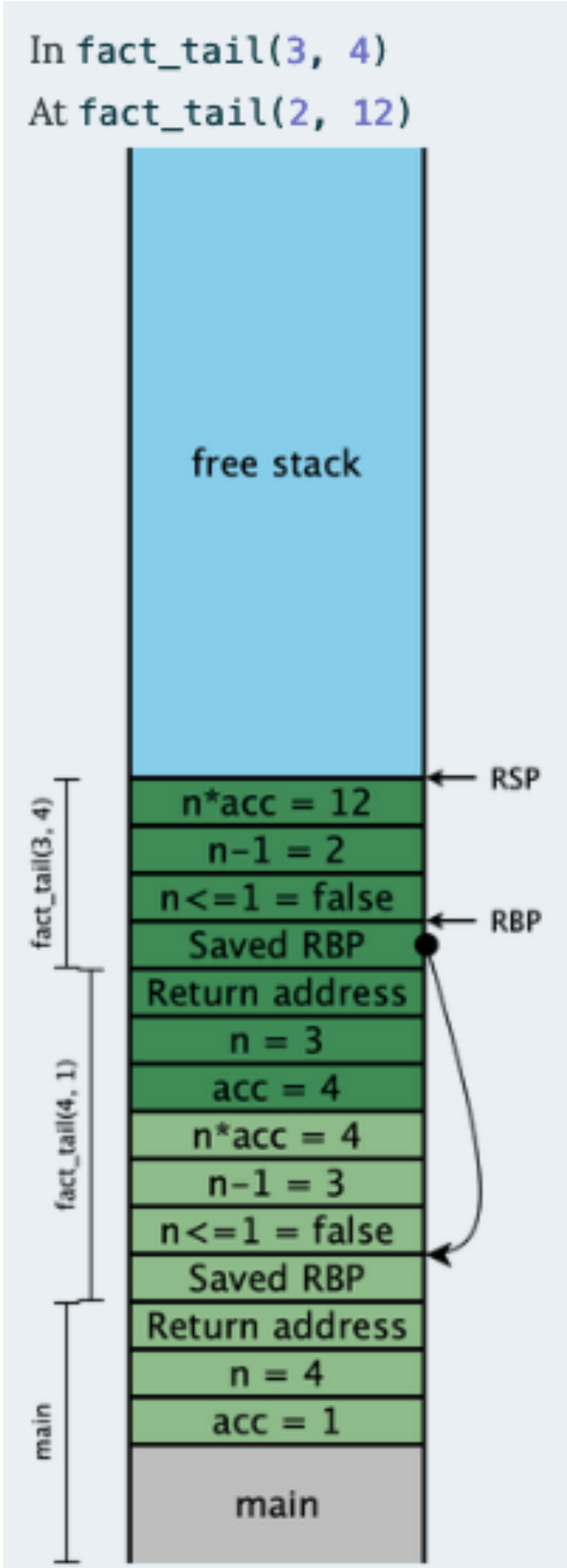
```
fact_v2(4) ==> fact_tail(4, 1)
           ==> if 4 <= 1: 1 else fact-tail(4 - 1, 4 * 1)
           ==> fact_tail(3, 4)
           ==> if 3 <= 1: 4 else fact-tail(3 - 1, 3 * 4)
           ==> fact_tail(2, 12)
           ==> if 2 <= 1: 12 else fact-tail(2 - 1, 2 * 12)
           ==> fact_tail(1, 24)
           ==> if 1 <= 1: 24 else fact-tail(1 - 1, 1 * 24)
           ==> 24
```

```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = acc * n in
    fact_tail(n - 1, acc)
end

def fact_v2(n):
  fact_tail(n)
end

fact_v2(4)
```

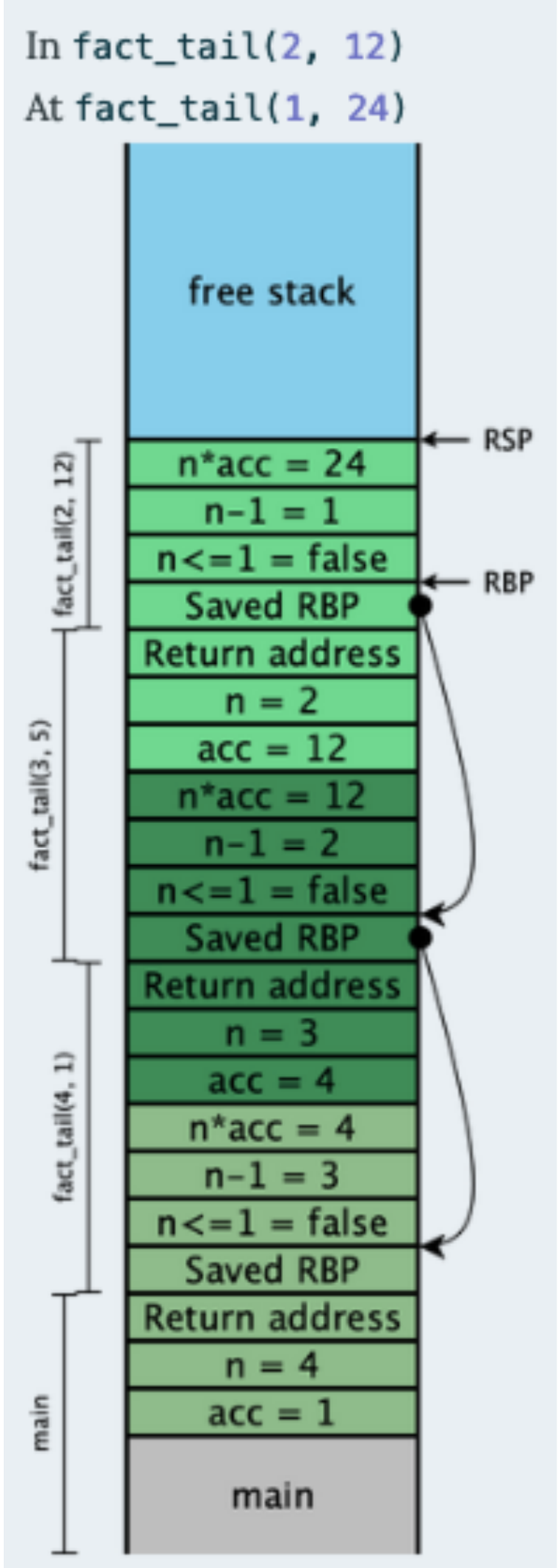
```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = acc * n in
    fact_tail(n - 1, acc)
end

def fact_v2(n):
  fact_tail(n)
end

fact_v2(4)
```



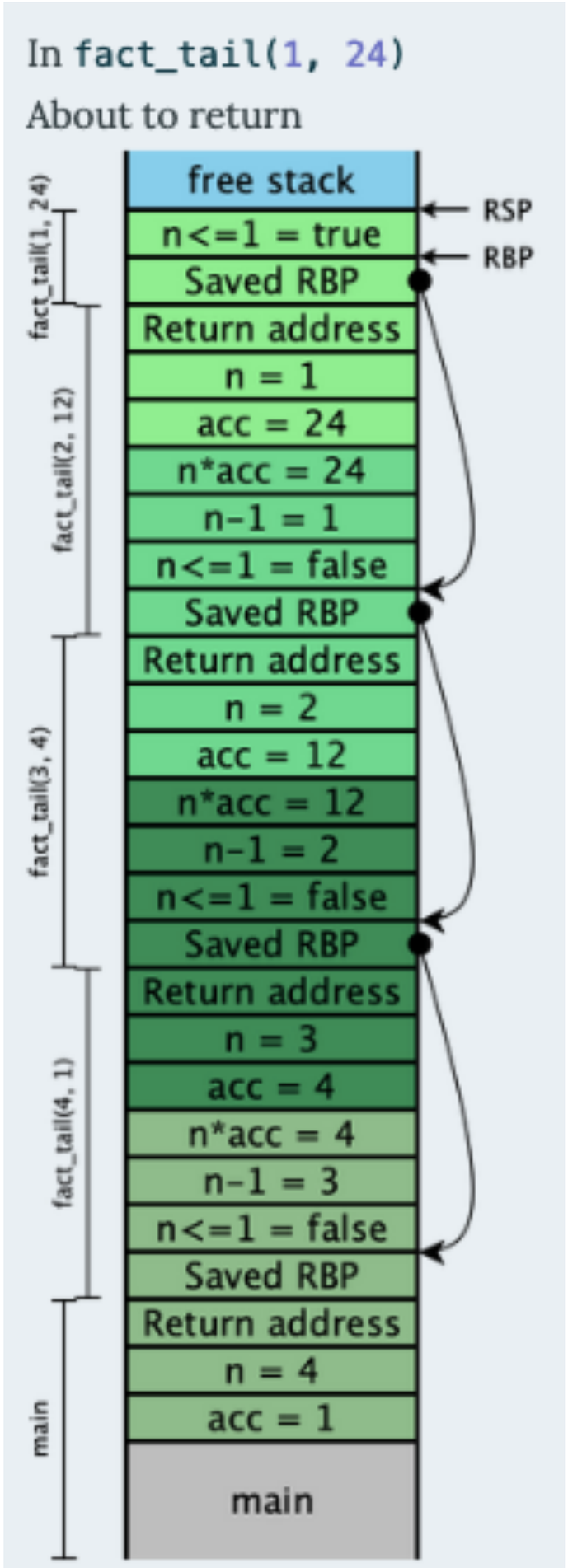
```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = acc * n in
    fact_tail(n - 1, acc)
end

def fact_v2(n):
  fact_tail(n)
end

fact_v2(4)
```



```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = acc * n in
    fact_tail(n - 1, acc)
end

def fact_v2(n):
  fact_tail(n)
end

fact_v2(4)
```



```
def fact_tail(n, acc):
  if n <= 1:
    acc
  else:
    let acc = acc * n in
    fact_tail(n - 1, acc)
end

def fact_v2(n):
  fact_tail(n)
end

fact_v2(4)
```



Assembly Demo…

Which Expressions are in Tail Position?

1. The main expression is in tail position
2. The body of a function is in tail position
3. If a let is in tail position, so is its body, the bindings are never tail position
4. If an if is in tail position, the branches are in tail position
5. Arguments to a prim or call are never in tail position

Pitfalls of Tail Call Impl

1. Overwriting parameters
2. Changing # of arguments