Python is slow

Brandt Bucher (February 22nd, 2025)

Python is slow

...let's make it faster!

Brandt Bucher (February 22nd, 2025)

Brandt Bucher

Brandt Bucher

- 2017: Started using Python.
- 2018: Contributed code to CPython.
- 2019: Joined Python's Triage Team.
- 2020: Joined Python's Core Development Team.
- 2021: Joined Microsoft's CPython Performance Engineering Team.
- 2022: Made CPython 3.11 25% faster!
- 2023: Implemented CPython's new JIT compiler.
- 2024: Worked on shipping the new JIT compiler in Python 3.14!

Microsoft's CPython Performance Engineering Team

The "Faster CPython" Project

The "Faster CPython" Project

- California
- Utah
- Washington
- Maryland
- United Kingdom
- Singapore

The "Faster CPython" Project

- California: Microsoft
- Utah: Microsoft
- Washington
- Maryland: Microsoft
- United Kingdom: Microsoft
- Singapore

The "Faster CPython" Project

- California: Microsoft
- Utah: Microsoft
- Washington: Snowflake
- Maryland: Microsoft
- United Kingdom: Microsoft, Arm
- Singapore: National University of Singapore

The "Faster CPython" Project

- California: Microsoft, ...?
- Utah: Microsoft
- Washington: Snowflake
- Maryland: Microsoft
- United Kingdom: Microsoft, Arm
- Singapore: National University of Singapore

github.com/faster-cpython/ideas

Results

Results (so far...)

Results

(so far...)

- 3.11: 25% faster
- 3.12: 4% faster
- 3.13: 7% faster
- 3.14: 8% faster

Results

(so far...)

- In less than 4 years, Python has gotten about 50% faster:
  - 93% of benchmarks have improved.
  - 48% of benchmarks are *over 50%* faster.
  - 14% of benchmarks are *over 100%* faster (including Pylint!)

Python

Python

- 34 years old!
- *Very* high-level
- *Very* widely used
- Dynamic
- Object-oriented
- Interpreted
- Automatic memory management
- Deep introspection and metaprogramming

Python

- 34 years old!
- *Very* high-level
- *Very* widely used
- Dynamic
- Object-oriented
- Interpreted
- Automatic memory management
- Deep introspection and metaprogramming

Python

- 34 years old!
- *Very* high-level
- *Very* widely used
- Dynamic
- Object-oriented
- Interpreted
- Automatic memory management
- Deep introspection and metaprogramming

Python

- Most objects have arbitrary mappings of attributes: `instance.__dict__`.
- Bytecode is a runtime object: `function.__code__`.
- *Frames* are runtime objects: `sys._getframe()`.
- Attribute/global name accesses and assignments can run arbitrary code.
- Even simple operators go through incredibly complex double-dispatching.
- A debugger can be entered *anywhere* and do *anything.*

Python

- Most objects have arbitrary mappings of attributes: `instance.__dict__`.
- Bytecode is a runtime object: `function.__code__`.
- *Frames* are runtime objects: `sys._getframe()`.
- Attribute/global name accesses and assignments can run arbitrary code.
- Even simple operators go through incredibly complex double-dispatching.
- A debugger can be entered *anywhere* and do *anything*.

Python

- Most objects have arbitrary mappings of attributes: `instance.__dict__`.
- Bytecode is a runtime object: `function.__code__`.
- *Frames* are runtime objects: `sys._getframe()`.
- Attribute/global name accesses and assignments can run arbitrary code.
- Even simple operators go through incredibly complex double-dispatching.
- A debugger can be entered *anywhere* and do *anything*.

Python

CPython

CPython

- Reference implementation of Python
- Used by the majority of Python programmers
- Reference-counted (augmented with cyclic stop-the-world GC)
- Has an incredibly rich ecosystem of third-party C extensions
- Maintained by a few dozen active "core developers"
- Free and open-source
- github.com/python/cpython

Optimizations

Optimizations

```
x = "foo"
print(x)
```

Optimizations

```
x = "foo"

(Pdb)

print(x)
```

Optimizations

```
x = "foo"

(Pdb) x = "bar"
(Pdb)

print(x)
```

Optimizations

```
x = "foo"

(Pdb) x = "bar"
(Pdb) continue

print(x)
```

Optimizations

```
x = "foo"

(Pdb) x = "bar"
(Pdb) continue

print(x)  # Prints "bar"!
```

Optimizations

```
x = "foo"
print(x)
```

Optimizations

```
class Sneaky:
    def __del__(self):
        import sys
        sys._getframe(1).f_locals["x"] = "bar"

x = "foo"
print(x)
```

Optimizations

```
class Sneaky:
    def __del__(self):
        import sys
        sys._getframe(1).f_locals["x"] = "bar"

x = Sneaky()

x = "foo"
print(x)
```

Optimizations

```
class Sneaky:
    def __del__(self):
        import sys
        sys._getframe(1).f_locals["x"] = "bar"

x = Sneaky()

x = "foo"
print(x)  # Prints "bar"!
```

Optimizations

```
def f(x, y):
    x.attr = "foo"
    y.attr = "bar"
    if x.attr == y.attr:
        do_something()
```

Optimizations

```
def f(x, y):
    x.attr = "foo"
    y.attr = "bar"
    if x.attr == y.attr:
        do_something()
```

Optimizations

```
def f(x, y):
    # assert sys.gettrace() is None
    x.attr = "foo"
    y.attr = "bar"
    if x.attr == y.attr:
        do_something()
```

Optimizations

```
def f(x, y):
    # assert sys.gettrace() is None
    # assert "attr" not in x.__dict__
    # assert "attr" not in y.__dict__
    x.attr = "foo"
    y.attr = "bar"
    if x.attr == y.attr:
        do_something()
```

Optimizations

```
class Sneaky:
    def __getattr__(self, name): return "baz"
    def __setattr__(self, name, value): pass

def f(x, y):
    # assert sys.gettrace() is None
    # assert "attr" not in x.__dict__
    # assert "attr" not in y.__dict__
    x.attr = "foo"
    y.attr = "bar"
    if x.attr == y.attr:
        do_something()
```

Optimizations

```
def f(x, y):
    # assert sys.gettrace() is None
    # assert "attr" not in x.__dict__
    # assert "attr" not in y.__dict__
    x.attr = "foo"
    y.attr = "bar"
    if x.attr == y.attr:
        do_something()
```

Optimizations

```
# class Boring:
#     pass


def f(x, y):
    # assert sys.gettrace() is None
    # assert "attr" not in x.__dict__
    # assert "attr" not in y.__dict__
    # assert x.__class__ is y.__class__ is Boring
    x.attr = "foo"
    y.attr = "bar"
    if x.attr == y.attr:
        do_something()
```

Optimizations

```
def f(x, y):
    x.attr = "foo"
    y.attr = "bar"
    if x.attr == y.attr:
        do_something()
```

Optimizations

```
def f(x, y):
    x.attr = "foo"
    y.attr = "bar"
    if x.attr == y.attr:
        do_something()

f(a, a)
```

Optimizations

Runtime Optimizations

Runtime Optimizations

- Build IR
- Check types
- Optimize
- Compile

Runtime Optimizations

- Build IR
- Check types
- Optimize
- Compile

Runtime Optimizations

- Build IR
- Profile
- Optimize
- Compile

Runtime Optimizations

```
def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        a, b = b, a + b
    return a
```

Runtime Optimizations

```
for _ in range(n):
    a, b = b, a + b
```

Runtime Optimizations

CPython 3.10: Bytecode

```
for _ in range(n):    FOR_ITER
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST
                      BINARY_OP
                      STORE_FAST_STORE_FAST
                      JUMP_BACKWARD
```

Runtime Optimizations

CPython 3.10: Bytecode

```
for _ in range(n):    FOR_ITER
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST
                      BINARY_OP
                      STORE_FAST_STORE_FAST
                      JUMP_BACKWARD                stack
```

Runtime Optimizations

CPython 3.10: Bytecode

```
for _ in range(n):    FOR_ITER
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST
                      BINARY_OP
                      STORE_FAST_STORE_FAST          stack
                      JUMP_BACKWARD                iterator
```

Runtime Optimizations

CPython 3.10: Bytecode

```
for _ in range(n):    FOR_ITER
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST
                      BINARY_OP
                      STORE_FAST_STORE_FAST
                      JUMP_BACKWARD
```

stack

next(iterator)

iterator

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):      FOR_ITER
    a, b = b, a + b     STORE_FAST
                        LOAD_FAST_LOAD_FAST
                        LOAD_FAST
                        BINARY_OP
                        STORE_FAST_STORE_FAST
                        JUMP_BACKWARD
```

stack

next(iterator)

iterator

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):      FOR_ITER_RANGE
    a, b = b, a + b     STORE_FAST
                        LOAD_FAST_LOAD_FAST
                        LOAD_FAST
                        BINARY_OP
                        STORE_FAST_STORE_FAST
                        JUMP_BACKWARD
```

stack

next(iterator)

iterator

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST
                      BINARY_OP
                      STORE_FAST_STORE_FAST        stack
                      JUMP_BACKWARD              iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST
                      BINARY_OP                    stack
                      STORE_FAST_STORE_FAST          b
                      JUMP_BACKWARD                iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST                    stack
                      BINARY_OP                      a
                      STORE_FAST_STORE_FAST          b
                      JUMP_BACKWARD               iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST          stack
                      LOAD_FAST                      b
                      BINARY_OP                      a
                      STORE_FAST_STORE_FAST          b
                      JUMP_BACKWARD                iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST                    stack
                      BINARY_OP                    a + b
                      STORE_FAST_STORE_FAST          b
                      JUMP_BACKWARD                iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST                    stack
                      BINARY_OP                    a + b
                      STORE_FAST_STORE_FAST          b
                      JUMP_BACKWARD                iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):      FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b     STORE_FAST
                        LOAD_FAST_LOAD_FAST
                        LOAD_FAST                    stack
                        BINARY_OP_ADD_INT            a + b
                        STORE_FAST_STORE_FAST          b
                        JUMP_BACKWARD              iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):
    a, b = b, a + b
```

```
(Pdb) a = ""
(Pdb) b = "🐰🐰"
(Pdb) continue
```

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_ = next(iterator)
```

stack

a + b

b

iterator

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):          FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b         STORE_FAST
                            LOAD_FAST_LOAD_FAST
(Pdb) a = ""                LOAD_FAST                       stack
(Pdb) b = "🐰🐰"            BINARY_OP_ADD_INT               a + b
(Pdb) continue              STORE_FAST_STORE_FAST             b
                            JUMP_BACKWARD                 iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):
    a, b = b, a + b
```

```
(Pdb) a = ""
(Pdb) b = "🐰🐰"
(Pdb) continue
```

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_UNICODE
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_ = next(iterator)
```

stack

a + b

b

iterator

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):      FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b     STORE_FAST
                        LOAD_FAST_LOAD_FAST
                        LOAD_FAST                    stack
                        BINARY_OP_ADD_INT            a + b
                        STORE_FAST_STORE_FAST          b
                        JUMP_BACKWARD              iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):      FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b     STORE_FAST              b = a + b
                        LOAD_FAST_LOAD_FAST
                        LOAD_FAST
                        BINARY_OP_ADD_INT              stack
                        STORE_FAST_STORE_FAST            b
                        JUMP_BACKWARD                iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):      FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b     STORE_FAST              b = a + b
                        LOAD_FAST_LOAD_FAST     a = b
                        LOAD_FAST
                        BINARY_OP_ADD_INT
                        STORE_FAST_STORE_FAST       stack
                        JUMP_BACKWARD             iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b   STORE_FAST              b = a + b
                      LOAD_FAST_LOAD_FAST     a = b
                      LOAD_FAST
                      BINARY_OP_ADD_INT
                      STORE_FAST_STORE_FAST       stack
                      JUMP_BACKWARD             iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
CALL_ALLOC_AND_ENTER_INIT
CALL_BOUND_METHOD_EXACT_ARGS
CALL_BOUND_METHOD_GENERAL
CALL_BUILTIN_CLASS
CALL_BUILTIN_FAST
CALL_BUILTIN_FAST_KEYWORDS
CALL_BUILTIN_O
CALL_ISINSTANCE
CALL_LEN
CALL_LIST_APPEND
CALL_METHOD_DESCRIPTOR_FAST
CALL_METHOD_DESCRIPTOR_FAST_KW
CALL_METHOD_DESCRIPTOR_NOARGS
CALL_METHOD_DESCRIPTOR_O
CALL_NON_PY_GENERAL
CALL_PY_EXACT_ARGS
CALL_PY_GENERAL
CALL_STR_1
CALL_TUPLE_1
CALL_TYPE_1

FOR_ITER_GEN
FOR_ITER_LIST
FOR_ITER_RANGE
FOR_ITER_TUPLE
```

```
BINARY_OP_ADD_FLOAT
BINARY_OP_ADD_INT
BINARY_OP_ADD_UNICODE
BINARY_OP_EXTEND
BINARY_OP_INPLACE_ADD_UNICODE
BINARY_OP_MULTIPLY_FLOAT
BINARY_OP_MULTIPLY_INT
BINARY_OP_SUBSCR_DICT
BINARY_OP_SUBSCR_GETITEM
BINARY_OP_SUBSCR_LIST_INT
BINARY_OP_SUBSCR_STR_INT
BINARY_OP_SUBSCR_TUPLE_INT
BINARY_OP_SUBTRACT_FLOAT
BINARY_OP_SUBTRACT_INT

TO_BOOL_ALWAYS_TRUE
TO_BOOL_BOOL
TO_BOOL_INT
TO_BOOL_LIST
TO_BOOL_NONE
TO_BOOL_STR

CALL_KW_BOUND_METHOD
CALL_KW_NON_PY
CALL_KW_PY
```

```
LOAD_ATTR_CLASS
LOAD_ATTR_CLASS_WITH_METACLASS
LOAD_ATTR_GETATTRIBUTE
LOAD_ATTR_INSTANCE_VALUE
LOAD_ATTR_METHOD_LAZY_DICT
LOAD_ATTR_METHOD_NO_DICT
LOAD_ATTR_METHOD_WITH_VALUES
LOAD_ATTR_MODULE
LOAD_ATTR_NONDESCRIPTOR_NO_DICT
LOAD_ATTR_NONDESCRIPTOR_VALUES
LOAD_ATTR_PROPERTY
LOAD_ATTR_SLOT
LOAD_ATTR_WITH_HINT

COMPARE_OP_FLOAT
COMPARE_OP_INT
COMPARE_OP_STR

STORE_ATTR_INSTANCE_VALUE
STORE_ATTR_SLOT
STORE_ATTR_WITH_HINT

UNPACK_SEQUENCE_LIST
UNPACK_SEQUENCE_TUPLE
UNPACK_SEQUENCE_TWO_TUPLE
```

```
CONTAINS_OP_DICT
CONTAINS_OP_SET

JUMP_BACKWARD_JIT
JUMP_BACKWARD_NO_JIT

LOAD_CONST_IMMORTAL
LOAD_CONST_MORTAL

LOAD_GLOBAL_BUILTIN
LOAD_GLOBAL_MODULE

LOAD_SUPER_ATTR_ATTR
LOAD_SUPER_ATTR_METHOD

STORE_SUBSCR_DICT
STORE_SUBSCR_LIST_INT

RESUME_CHECK

SEND_GEN
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
CALL_ALLOC_AND_ENTER_INIT
CALL_BOUND_METHOD_EXACT_ARGS
CALL_BOUND_METHOD_GENERAL
CALL_BUILTIN_CLASS
CALL_BUILTIN_FAST
CALL_BUILTIN_FAST_KEYWORDS
CALL_BUILTIN_O
CALL_ISINSTANCE
CALL_LEN
CALL_LIST_APPEND
CALL_METHOD_DESCRIPTOR_FAST
CALL_METHOD_DESCRIPTOR_FAST_KW
CALL_METHOD_DESCRIPTOR_NOARGS
CALL_METHOD_DESCRIPTOR_O
CALL_NON_PY_GENERAL
CALL_PY_EXACT_ARGS
CALL_PY_GENERAL
CALL_STR_1
CALL_TUPLE_1
CALL_TYPE_1

FOR_ITER_GEN
FOR_ITER_LIST
FOR_ITER_RANGE
FOR_ITER_TUPLE
```

```
BINARY_OP_ADD_FLOAT
BINARY_OP_ADD_INT
BINARY_OP_ADD_UNICODE
BINARY_OP_EXTEND
BINARY_OP_INPLACE_ADD_UNICODE
BINARY_OP_MULTIPLY_FLOAT
BINARY_OP_MULTIPLY_INT
BINARY_OP_SUBSCR_DICT
BINARY_OP_SUBSCR_GETITEM
BINARY_OP_SUBSCR_LIST_INT
BINARY_OP_SUBSCR_STR_INT
BINARY_OP_SUBSCR_TUPLE_INT
BINARY_OP_SUBTRACT_FLOAT
BINARY_OP_SUBTRACT_INT

TO_BOOL_ALWAYS_TRUE
TO_BOOL_BOOL
TO_BOOL_INT
TO_BOOL_LIST
TO_BOOL_NONE
TO_BOOL_STR

CALL_KW_BOUND_METHOD
CALL_KW_NON_PY
CALL_KW_PY
```

```
LOAD_ATTR_CLASS
LOAD_ATTR_CLASS_WITH_METACLASS
LOAD_ATTR_GETATTRIBUTE
LOAD_ATTR_INSTANCE_VALUE
LOAD_ATTR_METHOD_LAZY_DICT
LOAD_ATTR_METHOD_NO_DICT
LOAD_ATTR_METHOD_WITH_VALUES
LOAD_ATTR_MODULE
LOAD_ATTR_NONDESCRIPTOR_NO_DICT
LOAD_ATTR_NONDESCRIPTOR_VALUES
LOAD_ATTR_PROPERTY
LOAD_ATTR_SLOT
LOAD_ATTR_WITH_HINT

COMPARE_OP_FLOAT
COMPARE_OP_INT
COMPARE_OP_STR

STORE_ATTR_INSTANCE_VALUE
STORE_ATTR_SLOT
STORE_ATTR_WITH_HINT

UNPACK_SEQUENCE_LIST
UNPACK_SEQUENCE_TUPLE
UNPACK_SEQUENCE_TWO_TUPLE
```

```
CONTAINS_OP_DICT
CONTAINS_OP_SET

JUMP_BACKWARD_JIT
JUMP_BACKWARD_NO_JIT

LOAD_CONST_IMMORTAL
LOAD_CONST_MORTAL

LOAD_GLOBAL_BUILTIN
LOAD_GLOBAL_MODULE

LOAD_SUPER_ATTR_ATTR
LOAD_SUPER_ATTR_METHOD

STORE_SUBSCR_DICT
STORE_SUBSCR_LIST_INT

RESUME_CHECK

SEND_GEN
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
CALL_ALLOC_AND_ENTER_INIT
CALL_BOUND_METHOD_EXACT_ARGS
CALL_BOUND_METHOD_GENERAL
CALL_BUILTIN_CLASS
CALL_BUILTIN_FAST
CALL_BUILTIN_FAST_KEYWORDS
CALL_BUILTIN_O
CALL_ISINSTANCE
CALL_LEN
CALL_LIST_APPEND
CALL_METHOD_DESCRIPTOR_FAST
CALL_METHOD_DESCRIPTOR_FAST_KW
CALL_METHOD_DESCRIPTOR_NOARGS
CALL_METHOD_DESCRIPTOR_O
CALL_NON_PY_GENERAL
CALL_PY_EXACT_ARGS
CALL_PY_GENERAL
CALL_STR_1
CALL_TUPLE_1
CALL_TYPE_1

FOR_ITER_GEN
FOR_ITER_LIST
FOR_ITER_RANGE
FOR_ITER_TUPLE
```

```
BINARY_OP_ADD_FLOAT
BINARY_OP_ADD_INT
BINARY_OP_ADD_UNICODE
BINARY_OP_EXTEND
BINARY_OP_INPLACE_ADD_UNICODE
BINARY_OP_MULTIPLY_FLOAT
BINARY_OP_MULTIPLY_INT
BINARY_OP_SUBSCR_DICT
BINARY_OP_SUBSCR_GETITEM
BINARY_OP_SUBSCR_LIST_INT
BINARY_OP_SUBSCR_STR_INT
BINARY_OP_SUBSCR_TUPLE_INT
BINARY_OP_SUBTRACT_FLOAT
BINARY_OP_SUBTRACT_INT

TO_BOOL_ALWAYS_TRUE
TO_BOOL_BOOL
TO_BOOL_INT
TO_BOOL_LIST
TO_BOOL_NONE
TO_BOOL_STR

CALL_KW_BOUND_METHOD
CALL_KW_NON_PY
CALL_KW_PY
```

```
LOAD_ATTR_CLASS
LOAD_ATTR_CLASS_WITH_METACLASS
LOAD_ATTR_GETATTRIBUTE
LOAD_ATTR_INSTANCE_VALUE
LOAD_ATTR_METHOD_LAZY_DICT
LOAD_ATTR_METHOD_NO_DICT
LOAD_ATTR_METHOD_WITH_VALUES
LOAD_ATTR_MODULE
LOAD_ATTR_NONDESCRIPTOR_NO_DICT
LOAD_ATTR_NONDESCRIPTOR_VALUES
LOAD_ATTR_PROPERTY
LOAD_ATTR_SLOT
LOAD_ATTR_WITH_HINT

COMPARE_OP_FLOAT
COMPARE_OP_INT
COMPARE_OP_STR

STORE_ATTR_INSTANCE_VALUE
STORE_ATTR_SLOT
STORE_ATTR_WITH_HINT

UNPACK_SEQUENCE_LIST
UNPACK_SEQUENCE_TUPLE
UNPACK_SEQUENCE_TWO_TUPLE
```

```
CONTAINS_OP_DICT
CONTAINS_OP_SET

JUMP_BACKWARD_JIT
JUMP_BACKWARD_NO_JIT

LOAD_CONST_IMMORTAL
LOAD_CONST_MORTAL

LOAD_GLOBAL_BUILTIN
LOAD_GLOBAL_MODULE

LOAD_SUPER_ATTR_ATTR
LOAD_SUPER_ATTR_METHOD

STORE_SUBSCR_DICT
STORE_SUBSCR_LIST_INT

RESUME_CHECK

SEND_GEN

...?
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: object<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: object<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: object<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: object<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: object<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: int<?>
stack_0: range_iterator<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: int<?>
stack_0: range_iterator<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: object<b>
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: object<b>
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: object<a>
stack_1: object<b>
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: object<a>
stack_1: object<b>
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: object<b>
stack_2: object<a>
stack_1: object<b>
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: object<b>
stack_2: object<a>
stack_1: object<b>
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: int<b>
stack_2: object<a>
stack_1: int<b>
stack_0: range_iterator<?>

_: int<?>
b: int<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: int<b>
stack_2: int<a>
stack_1: int<b>
stack_0: range_iterator<?>

_: int<?>
b: int<b>
a: int<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: int<a+b>
stack_1: int<b>
stack_0: range_iterator<?>

_: int<?>
b: int<b>
a: int<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: int<a+b>
stack_1: int<b>
stack_0: range_iterator<?>

_: int<?>
b: int<b>
a: int<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: int<b>
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: int<b>
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_SET_IP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_SET_IP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

```
_SET_IP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
_GUARD_FAST_INT
_GUARD_FAST_INT
_BINARY_OP_ADD_INT
```

```
_SET_IP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_ITER_CHECK_RANGE
_GUARD_FAST_INT
_GUARD_FAST_INT
_MAKE_WARM
_CHECK_VALIDITY
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
_BINARY_OP_ADD_INT
```

```
_NOP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_ITER_CHECK_RANGE
_GUARD_FAST_INT
_GUARD_FAST_INT
_MAKE_WARM
_CHECK_VALIDITY
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
_BINARY_OP_ADD_INT
_NOP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
_START_EXECUTOR
_ITER_CHECK_RANGE
_GUARD_FAST_INT
_GUARD_FAST_INT
_MAKE_WARM
_CHECK_VALIDITY
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
_BINARY_OP_ADD_INT
_NOP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    s = ""
    if i % 3:
        if i % 5:
            print(s or i)
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    s = ""
    if i % 3:
        if i % 5:
            print(s or i)
        else:
            ...
    else:
        ...
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    s = ""
    if i % 3:
        if i % 5:
            print(s or i)
        else:
            ...
    else:
        ...
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    if i % 3:
        if i % 5:
            print(i)
        else:
            ...
    else:
        ...
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    if i % 3:
        if i % 5:
            print(i)
        else:
            ...
    else:
        s = ""
        s += "fizz"
        if i % 5:
            print(s or i)
        else:
            ...
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    if i % 3:
        if i % 5:
            print(i)
        else:
            ...
    else:
        s = ""
        s += "fizz"
        if i % 5:
            print(s or i)
        else:
            ...
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    if i % 3:
        if i % 5:
            print(i)
        else:
            ...
    else:
        if i % 5:
            print("fizz")
        else:
            ...
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    if i % 3:
        if i % 5:
            print(i)
        else:
            s = ""
            s += "buzz"
            print(s or i)
    else:
        if i % 5:
            print("fizz")
        else:
            ...
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    if i % 3:
        if i % 5:
            print(i)
        else:
            s = ""
            s += "buzz"
            print(s or i)
    else:
        if i % 5:
            print("fizz")
        else:
            ...
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    if i % 3:
        if i % 5:
            print(i)
        else:
            print("buzz")
    else:
        if i % 5:
            print("fizz")
        else:
            ...
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    if i % 3:
        if i % 5:
            print(i)
        else:
            print("buzz")
    else:
        if i % 5:
            print("fizz")
        else:
            s = ""
            s += "fizz"
            s += "buzz"
            print(s or i)
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    if i % 3:
        if i % 5:
            print(i)
        else:
            print("buzz")
    else:
        if i % 5:
            print("fizz")
        else:
            s = ""
            s += "fizz"
            s += "buzz"
            print(s or i)
```

Runtime Optimizations

Tracing and Abstract Interpretation

```
for i in range(n):
    s = ""
    if not i % 3:
        s += "fizz"
    if not i % 5:
        s += "buzz"
    print(s or i)
```

```
for i in range(n):
    if i % 3:
        if i % 5:
            print(i)
        else:
            print("buzz")
    else:
        if i % 5:
            print("fizz")
        else:
            print("fizzbuzz")
```

Just-In-Time Compilation

Just-In-Time Compilation

- Technical goals:
  - Compile traces to optimized machine code
  - Remove interpretive overhead

- Deployment goals:
  - Broad platform support
  - Few runtime dependencies
  - Low implementation complexity

Just-In-Time Compilation

- Technical goals:
  - Compile traces to optimized machine code
  - Remove interpretive overhead

- Deployment goals:
  - Broad platform support
  - Few runtime dependencies
  - Low implementation complexity

Just-In-Time Compilation

Copy-And-Patch Compilation

Copy-And-Patch Compilation

- Haoran Xu and Fredrik Kjolstad. 2021. Copy-and-Patch Compilation: A Fast Compilation Algorithm for High- Level Languages and Bytecode. Proc. ACM Program. Lang. 5, OOPSLA, Article 136 (October 2021), 30 pages. https://doi.org/10.1145/3485513

- Haoran Xu. 2023. Building a baseline JIT for Lua automatically. (12 March 2023). Retrieved from https://sillycross.github.io/2023/05/12/2023-05-12/.

- Automatically turns an interpreter written in C into a fast JIT compiler.

Copy-And-Patch Compilation

```
case _LOAD_FAST:
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
    break;
...
```

Copy-And-Patch Compilation

```
case _LOAD_FAST:
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
    break;
...
```

Copy-And-Patch Compilation

```
case _LOAD_FAST:
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
    break;
...
```

Copy-And-Patch Compilation

```
case _LOAD_FAST:
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
    break;
...
```

Copy-And-Patch Compilation

```
case _LOAD_FAST:
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
    break;
...
```

Copy-And-Patch Compilation

```
switch (opcode) {
    case _LOAD_FAST:
        PyObject *value = frame->localsplus[oparg];
        Py_INCREF(value);
        *stack_pointer++ = value;
        break;
    ...
}
```

Copy-And-Patch Compilation

```
while (true) {
    switch (opcode) {
        case _LOAD_FAST:
            PyObject *value = frame->localsplus[oparg];
            Py_INCREF(value);
            *stack_pointer++ = value;
            break;
        ...
    }
    ...
}
```

Copy-And-Patch Compilation

```
PyObject *value = frame->localsplus[oparg];
Py_INCREF(value);
*stack_pointer++ = value;
```

Copy-And-Patch Compilation

```
int
_LOAD_FAST(_PyInterpreterFrame *frame, PyObject **stack_pointer)
{
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
}
```

Copy-And-Patch Compilation

```
b8 00 00 00 00          mov $0x0, %eax
48 8b 44 c7 48          movq 0x48(%rdi,%rax,8), %rax
8b 08                   movl (%rax), %ecx
ff c1                   incl %ecx
74 02                   je 0x12
89 08                   movl %ecx, (%rax)
48 89 06                movq %rax, (%rsi)
48 83 c6 08             addq $0x8, %rsi


01: R_X86_64_32         oparg
```

Copy-And-Patch Compilation

```
b8 00 00 00 00          mov $0x0, %eax
48 8b 44 c7 48          movq 0x48(%rdi,%rax,8), %rax
8b 08                   movl (%rax), %ecx
ff c1                   incl %ecx
74 02                   je 0x12
89 08                   movl %ecx, (%rax)
48 89 06                movq %rax, (%rsi)
48 83 c6 08             addq $0x8, %rsi


01: R_X86_64_32         oparg
```

Copy-And-Patch Compilation

```
b8 00 00 00 00          mov $0x0, %eax
48 8b 44 c7 48          movq 0x48(%rdi,%rax,8), %rax
8b 08                   movl (%rax), %ecx
ff c1                   incl %ecx
74 02                   je 0x12
89 08                   movl %ecx, (%rax)
48 89 06                movq %rax, (%rsi)
48 83 c6 08             addq $0x8, %rsi


01: R_X86_64_32         oparg
```

Copy-And-Patch Compilation

```
b8 00 00 00 00          mov $0x0, %eax
48 8b 44 c7 48          movq 0x48(%rdi,%rax,8), %rax
8b 08                   movl (%rax), %ecx
ff c1                   incl %ecx
74 02                   je 0x12
89 08                   movl %ecx, (%rax)
48 89 06                movq %rax, (%rsi)
48 83 c6 08             addq $0x8, %rsi


01: R_X86_64_32         oparg
```

Copy-And-Patch Compilation

```
void
emit__LOAD_FAST(unsigned char *code, _PyUOpInstruction *uop)
{
    const unsigned char code_body[25] = {
          b8,   00,   00,   00,   00,   48,   8b,   44,
          c7,   48,   8b,   08,   ff,   c1,   74,   02,
          89,   08,   48,   89,   06,   48,   83,   c6,
          08,
    };
    memcpy(code, code_body, sizeof(code_body));
    patch_32(code + 0x1, uop->oparg);
}
```

Copy-And-Patch Compilation

```
void
emit__LOAD_FAST(unsigned char *code, _PyUOpInstruction *uop)
{
    const unsigned char code_body[25] = {
        0xb8, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8b, 0x44,
        0xc7, 0x48, 0x8b, 0x08, 0xff, 0xc1, 0x74, 0x02,
        0x89, 0x08, 0x48, 0x89, 0x06, 0x48, 0x83, 0xc6,
        0x08,
    };
    memcpy(code, code_body, sizeof(code_body));
    patch_32(code + 0x1, uop->oparg);
}
```

Platform Support

Platform Support

Platform Support

x86-64

- `x86_64-apple-darwin/clang`
- `x86_64-pc-windows-msvc/msvc`
- `x86_64-unknown-linux-gnu/clang`
- `x86_64-unknown-linux-gnu/gcc`

Platform Support

x86 and x86-64

- i686-pc-windows-msvc/msvc
- x86_64-apple-darwin/clang
- x86_64-pc-windows-msvc/msvc
- x86_64-unknown-linux-gnu/clang
- x86_64-unknown-linux-gnu/gcc

Platform Support

AArch64, x86, and x86-64

- aarch64-apple-darwin/clang
- aarch64-pc-windows-msvc/msvc
- aarch64-unknown-linux-gnu/clang
- aarch64-unknown-linux-gnu/gcc
- i686-pc-windows-msvc/msvc
- x86_64-apple-darwin/clang
- x86_64-pc-windows-msvc/msvc
- x86_64-unknown-linux-gnu/clang
- x86_64-unknown-linux-gnu/gcc

Platform Support

AArch64, x86, and x86-64

- `aarch64-apple-darwin/clang`
- `aarch64-pc-windows-msvc/msvc`
- `aarch64-unknown-linux-gnu/clang`
- `aarch64-unknown-linux-gnu/gcc`
- `i686-pc-windows-msvc/msvc`
- `x86_64-apple-darwin/clang`
- `x86_64-pc-windows-msvc/msvc`
- `x86_64-unknown-linux-gnu/clang`
- `x86_64-unknown-linux-gnu/gcc`

Other Projects

Other Projects

- Better benchmarks, with more emphasis on modern idioms.
- Reduced reference counting overhead.
- Improving the object model.
- True function inlining.
- GC improvements.
- Integer unboxing.
- Subinterpreters.
- Free-threading.
- ...?

Thank you!

@brandtbucher

Thank you!

@brandtbucher | brandt@python.org