

Making CPython 3.11 Fast

Inside Python's new specializing, adaptive interpreter.

Brandt Bucher (October 24, 2022)

Making CPython 3.11 Especially Fast

Inside Python's new specializing, adaptive interpreter.

Brandt Bucher (October 24, 2022)

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 faster!

Bytecode

Bytecode

```
class Point:
    def __init__(self, x: float, y: float) -> None:
        self.x = x
        self.y = y

    def shifted(self, dx: float, dy: float) -> typing.Self:
        x = dx + self.x
        y = dy + self.y
        cls = type(self)
        return cls(x, y)
```

Bytecode

```
y = dy + self.y  
cls = type(self)
```

Bytecode

```
y = dy + self.y  
cls = type(self)
```

```
import dis  
dis.dis(Point.shifted)
```


Bytecode

```
dis.dis(Point.shifted)
```

```
y = dy + self.y  
cls = type(self)
```

Bytecode

```
dis.dis(Point.shifted)
```

```
y = dy + self.y  
cls = type(self)
```

```
LOAD_FAST      (dy)  
LOAD_FAST      (self)  
LOAD_ATTR      (y)  
BINARY_OP      (+)  
STORE_FAST     (y)  
LOAD_GLOBAL    (type)  
LOAD_FAST      (self)  
PRECALL        (1)  
CALL           (1)  
STORE_FAST     (cls)
```

Bytecode

`dis.dis(Point.shifted)`

<code>y = dy + self.y</code>	<code>LOAD_FAST</code>	<code>(dy)</code>	
<code>cls = type(self)</code>	<code>LOAD_FAST</code>	<code>(self)</code>	
	<code>LOAD_ATTR</code>	<code>(y)</code>	
	<code>BINARY_OP</code>	<code>(+)</code>	
	<code>STORE_FAST</code>	<code>(y)</code>	
	<code>LOAD_GLOBAL</code>	<code>(type)</code>	
	<code>LOAD_FAST</code>	<code>(self)</code>	<u>stack</u>
	<code>PRECALL</code>	<code>(1)</code>	
	<code>CALL</code>	<code>(1)</code>	
	<code>STORE_FAST</code>	<code>(cls)</code>	

Bytecode

```
dis.dis(Point.shifted)
```

```
y = dy + self.y  
cls = type(self)
```

```
LOAD_FAST      (dy)  
LOAD_FAST      (self)  
LOAD_ATTR      (y)  
BINARY_OP      (+)  
STORE_FAST     (y)  
LOAD_GLOBAL    (type)  
LOAD_FAST      (self)  
PRECALL        (1)  
CALL           (1)  
STORE_FAST     (cls)
```

stack
dy

Bytecode

```
dis.dis(Point.shifted)
```

y = dy + self.y	LOAD_FAST	(dy)
cls = type(self)	LOAD_FAST	(self)
	LOAD_ATTR	(y)
	BINARY_OP	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL	(type)
	LOAD_FAST	(self)
	PRECALL	(1)
	CALL	(1)
	STORE_FAST	(cls)

stack
self
dy

Bytecode

```
dis.dis(Point.shifted)
```

```
y = dy + self.y  
cls = type(self)
```

```
LOAD_FAST      (dy)  
LOAD_FAST      (self)  
LOAD_ATTR      (y)  
BINARY_OP      (+)  
STORE_FAST     (y)  
LOAD_GLOBAL    (type)  
LOAD_FAST      (self)  
PRECALL        (1)  
CALL           (1)  
STORE_FAST     (cls)
```

stack
self.y
dy

Bytecode

`dis.dis(Point.shifted)`

<code>y = dy + self.y</code>	<code>LOAD_FAST</code>	<code>(dy)</code>	
<code>cls = type(self)</code>	<code>LOAD_FAST</code>	<code>(self)</code>	
	<code>LOAD_ATTR</code>	<code>(y)</code>	
	<code>BINARY_OP</code>	<code>(+)</code>	
	<code>STORE_FAST</code>	<code>(y)</code>	
	<code>LOAD_GLOBAL</code>	<code>(type)</code>	<u>stack</u>
	<code>LOAD_FAST</code>	<code>(self)</code>	<code>dy + self.y</code>
	<code>PRECALL</code>	<code>(1)</code>	
	<code>CALL</code>	<code>(1)</code>	
	<code>STORE_FAST</code>	<code>(cls)</code>	

Bytecode

`dis.dis(Point.shifted)`

<code>y = dy + self.y</code>	<code>LOAD_FAST (dy)</code>	<code>y = dy + self.y</code>
<code>cls = type(self)</code>	<code>LOAD_FAST (self)</code>	
	<code>LOAD_ATTR (y)</code>	
	<code>BINARY_OP (+)</code>	
	<code>STORE_FAST (y)</code>	
	<code>LOAD_GLOBAL (type)</code>	
	<code>LOAD_FAST (self)</code>	<u>stack</u>
	<code>PRECALL (1)</code>	
	<code>CALL (1)</code>	
	<code>STORE_FAST (cls)</code>	

Bytecode

```
dis.dis(Point.shifted)
```

```
y = dy + self.y  
cls = type(self)
```

```
LOAD_FAST      (dy)  
LOAD_FAST      (self)  
LOAD_ATTR      (y)  
BINARY_OP      (+)  
STORE_FAST     (y)  
LOAD_GLOBAL    (type)  
LOAD_FAST      (self)  
PRECALL        (1)  
CALL           (1)  
STORE_FAST     (cls)
```

```
y = dy + self.y
```

stack
type

Bytecode

```
dis.dis(Point.shifted)
```

y = dy + self.y	LOAD_FAST	(dy)	y = dy + self.y
cls = type(self)	LOAD_FAST	(self)	
	LOAD_ATTR	(y)	
	BINARY_OP	(+)	
	STORE_FAST	(y)	<u>stack</u>
	LOAD_GLOBAL	(type)	self
	LOAD_FAST	(self)	type
	PRECALL	(1)	
	CALL	(1)	
	STORE_FAST	(cls)	

Bytecode

```
dis.dis(Point.shifted)
```

```
y = dy + self.y  
cls = type(self)
```

```
LOAD_FAST      (dy)  
LOAD_FAST      (self)  
LOAD_ATTR      (y)  
BINARY_OP      (+)  
STORE_FAST     (y)  
LOAD_GLOBAL    (type)  
LOAD_FAST      (self)  
PRECALL        (1)  
CALL           (1)  
STORE_FAST     (cls)
```

```
y = dy + self.y
```

stack
self
type

Bytecode

`dis.dis(Point.shifted)`

<code>y = dy + self.y</code>	<code>LOAD_FAST (dy)</code>	<code>y = dy + self.y</code>
<code>cls = type(self)</code>	<code>LOAD_FAST (self)</code>	
	<code>LOAD_ATTR (y)</code>	
	<code>BINARY_OP (+)</code>	
	<code>STORE_FAST (y)</code>	
	<code>LOAD_GLOBAL (type)</code>	<u>stack</u>
	<code>LOAD_FAST (self)</code>	<code>type(self)</code>
	<code>PRECALL (1)</code>	
	<code>CALL (1)</code>	
	<code>STORE_FAST (cls)</code>	

Bytecode

```
dis.dis(Point.shifted)
```

y = dy + self.y	LOAD_FAST	(dy)	y = dy + self.y
cls = type(self)	LOAD_FAST	(self)	cls = type(self)
	LOAD_ATTR	(y)	
	BINARY_OP	(+)	
	STORE_FAST	(y)	
	LOAD_GLOBAL	(type)	
	LOAD_FAST	(self)	<u>stack</u>
	PRECALL	(1)	
	CALL	(1)	
	STORE_FAST	(cls)	

Bytecode

```
dis.dis(Point.shifted)
```

y = dy + self.y	LOAD_FAST	(dy)
cls = type(self)	LOAD_FAST	(self)
	LOAD_ATTR	(y)
	BINARY_OP	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL	(type)
	LOAD_FAST	(self)
	PRECALL	(1)
	CALL	(1)
	STORE_FAST	(cls)

stack

Bytecode

```
dis.dis(Point.shifted)
```

y = dy + self.y	LOAD_FAST	(dy)
cls = type(self)	LOAD_FAST	(self)
	LOAD_ATTR	(y)
	BINARY_OP	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL	(type)
	LOAD_FAST	(self)
	PRECALL	(1)
	CALL	(1)
	STORE_FAST	(cls)

Quickening

`dis.dis(Point.shifted)`

<code>y = dy + self.y</code>	LOAD_FAST	(dy)
<code>cls = type(self)</code>	LOAD_FAST	(self)
	LOAD_ATTR	(y)
	BINARY_OP	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL	(type)
	LOAD_FAST	(self)
	PRECALL	(1)
	CALL	(1)
	STORE_FAST	(cls)

Quickening

```
dis.dis(Point.shifted, adaptive=True)
```

<code>y = dy + self.y</code>	LOAD_FAST	(dy)
<code>cls = type(self)</code>	LOAD_FAST	(self)
	LOAD_ATTR	(y)
	BINARY_OP	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL	(type)
	LOAD_FAST	(self)
	PRECALL	(1)
	CALL	(1)
	STORE_FAST	(cls)

Superinstructions

```
dis.dis(Point.shifted, adaptive=True)
```

<code>y = dy + self.y</code>	LOAD_FAST	(dy)
<code>cls = type(self)</code>	LOAD_FAST	(self)
	LOAD_ATTR	(y)
	BINARY_OP	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL	(type)
	LOAD_FAST	(self)
	PRECALL	(1)
	CALL	(1)
	STORE_FAST	(cls)

Superinstructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST	(dy)
cls = type(self)	LOAD_FAST	(self)
	LOAD_ATTR	(y)
	BINARY_OP	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL	(type)
	LOAD_FAST	(self)
	PRECALL	(1)
	CALL	(1)
	STORE_FAST	(cls)

Superinstructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR	(y)
	BINARY_OP	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL	(type)
	LOAD_FAST	(self)
	PRECALL	(1)
	CALL	(1)
	STORE_FAST	(cls)

Adaptive Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

<code>y = dy + self.y</code>	<code>LOAD_FAST__LOAD_FAST</code>	<code>(dy, self)</code>
<code>cls = type(self)</code>	<code>LOAD_ATTR</code>	<code>(y)</code>
	<code>BINARY_OP</code>	<code>(+)</code>
	<code>STORE_FAST</code>	<code>(y)</code>
	<code>LOAD_GLOBAL</code>	<code>(type)</code>
	<code>LOAD_FAST</code>	<code>(self)</code>
	<code>PRECALL</code>	<code>(1)</code>
	<code>CALL</code>	<code>(1)</code>
	<code>STORE_FAST</code>	<code>(cls)</code>

Adaptive Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_ADAPTIVE	(y)
	BINARY_OP_ADAPTIVE	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_ADAPTIVE	(type)
	LOAD_FAST	(self)
	PRECALL_ADAPTIVE	(1)
	CALL_ADAPTIVE	(1)
	STORE_FAST	(cls)

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

<code>y = dy + self.y</code>	<code>LOAD_FAST__LOAD_FAST</code>	<code>(dy, self)</code>
<code>cls = type(self)</code>	<code>LOAD_ATTR_ADAPTIVE</code>	<code>(y)</code>
	<code>BINARY_OP_ADAPTIVE</code>	<code>(+)</code>
	<code>STORE_FAST</code>	<code>(y)</code>
	<code>LOAD_GLOBAL_ADAPTIVE</code>	<code>(type)</code>
	<code>LOAD_FAST</code>	<code>(self)</code>
	<code>PRECALL_ADAPTIVE</code>	<code>(1)</code>
	<code>CALL_ADAPTIVE</code>	<code>(1)</code>
	<code>STORE_FAST</code>	<code>(cls)</code>

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_ADAPTIVE	(y)
	BINARY_OP_ADAPTIVE	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_ADAPTIVE	(type)
	LOAD_FAST	(self)
	PRECALL_ADAPTIVE	(1)
	CALL_ADAPTIVE	(1)
	STORE_FAST	(cls)

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

<code>y = dy + self.y</code>	<code>LOAD_FAST__LOAD_FAST</code>	<code>(dy, self)</code>
<code>cls = type(self)</code>	<code>LOAD_ATTR_INSTANCE_VALUE</code>	<code>(y)</code>
	<code>BINARY_OP_ADAPTIVE</code>	<code>(+)</code>
	<code>STORE_FAST</code>	<code>(y)</code>
	<code>LOAD_GLOBAL_ADAPTIVE</code>	<code>(type)</code>
	<code>LOAD_FAST</code>	<code>(self)</code>
	<code>PRECALL_ADAPTIVE</code>	<code>(1)</code>
	<code>CALL_ADAPTIVE</code>	<code>(1)</code>
	<code>STORE_FAST</code>	<code>(cls)</code>

LOAD_ATTR_INSTANCE_VALUE

- Check if the class is the same as last time.
- Check if the object's `__dict__` keys are the same as last time.
- Reach directly into the `__dict__` values at the same offset as last time.
- Return the result.

LOAD_ATTR_INSTANCE_VALUE

- Check if the class is the same as last time.
- Check if the object's `__dict__` keys are the same as last time.
- Reach directly into the `__dict__` values at the same offset as last time.
- Return the result.

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADAPTIVE	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_ADAPTIVE	(type)
	LOAD_FAST	(self)
	PRECALL_ADAPTIVE	(1)
	CALL_ADAPTIVE	(1)
	STORE_FAST	(cls)

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADAPTIVE	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_ADAPTIVE	(type)
	LOAD_FAST	(self)
	PRECALL_ADAPTIVE	(1)
	CALL_ADAPTIVE	(1)
	STORE_FAST	(cls)

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADD_FLOAT	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_ADAPTIVE	(type)
	LOAD_FAST	(self)
	PRECALL_ADAPTIVE	(1)
	CALL_ADAPTIVE	(1)
	STORE_FAST	(cls)

BINARY_OP_ADD_FLOAT

- Check if the left object's class is `float`.
- Check if the right object's class is `float`.
- Add the values together.
- Return the result.

BINARY_OP_ADD_FLOAT

- Check if the left object's class is `float`.
- Check if the right object's class is `float`.
- Add the values together.
- Return the result.

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADD_FLOAT	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_ADAPTIVE	(type)
	LOAD_FAST	(self)
	PRECALL_ADAPTIVE	(1)
	CALL_ADAPTIVE	(1)
	STORE_FAST	(cls)

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADD_FLOAT	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_ADAPTIVE	(type)
	LOAD_FAST	(self)
	PRECALL_ADAPTIVE	(1)
	CALL_ADAPTIVE	(1)
	STORE_FAST	(cls)

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADD_FLOAT	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_BUILTIN	(type)
	LOAD_FAST	(self)
	PRECALL_ADAPTIVE	(1)
	CALL_ADAPTIVE	(1)
	STORE_FAST	(cls)

LOAD_GLOBAL_BUILTIN

- Check if the globals dictionary keys are the same as last time.
- Check if the built-in dictionary keys are the same as last time.
- Reach directly into the built-in values at the same offset as last time.
- Return the result.

LOAD_GLOBAL_BUILTIN

- Check if the globals dictionary keys are the same as last time.
- Check if the built-in dictionary keys are the same as last time.
- Reach directly into the built-in values at the same offset as last time.
- Return the result.

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADD_FLOAT	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_BUILTIN	(type)
	LOAD_FAST	(self)
	PRECALL_ADAPTIVE	(1)
	CALL_ADAPTIVE	(1)
	STORE_FAST	(cls)

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADD_FLOAT	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_BUILTIN	(type)
	LOAD_FAST	(self)
	PRECALL_ADAPTIVE	(1)
	CALL_ADAPTIVE	(1)
	STORE_FAST	(cls)

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADD_FLOAT	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_BUILTIN	(type)
	LOAD_FAST	(self)
	PRECALL_NO_KW_TYPE_1	(1)
	STORE_FAST	(cls)

PRECALL_NO_KW_TYPE_1

- Check if the object we're calling is still `type`.
- Get the argument's class.
- Return the result.

PRECALL_NO_KW_TYPE_1

- Check if the object we're calling is still `type`.
- Get the argument's class.
- Return the result.

PRECALL_NO_KW_TYPE_1

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADD_FLOAT	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_BUILTIN	(type)
	LOAD_FAST	(self)
	PRECALL_NO_KW_TYPE_1	(1)
	STORE_FAST	(cls)

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

<code>y = dy + self.y</code>	<code>LOAD_FAST__LOAD_FAST</code>	<code>(dy, self)</code>
<code>cls = type(self)</code>	<code>LOAD_ATTR_INSTANCE_VALUE</code>	<code>(y)</code>
	<code>BINARY_OP_ADD_FLOAT</code>	<code>(+)</code>
	<code>STORE_FAST</code>	<code>(y)</code>
	<code>LOAD_GLOBAL_BUILTIN</code>	<code>(type)</code>
	<code>LOAD_FAST</code>	<code>(self)</code>
	<code>PRECALL_NO_KW_TYPE_1</code>	<code>(1)</code>
	<code>STORE_FAST</code>	<code>(cls)</code>

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADD_FLOAT	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_BUILTIN	(type)
	LOAD_FAST	(self)
	PRECALL_NO_KW_TYPE_1	(1)
	STORE_FAST	(cls)

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADAPTIVE	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_BUILTIN	(type)
	LOAD_FAST	(self)
	PRECALL_NO_KW_TYPE_1	(1)
	STORE_FAST	(cls)

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

y = dy + self.y	LOAD_FAST__LOAD_FAST	(dy, self)
cls = type(self)	LOAD_ATTR_INSTANCE_VALUE	(y)
	BINARY_OP_ADD_INT	(+)
	STORE_FAST	(y)
	LOAD_GLOBAL_BUILTIN	(type)
	LOAD_FAST	(self)
	PRECALL_NO_KW_TYPE_1	(1)
	STORE_FAST	(cls)

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

<code>y = dy + self.y</code>	<code>LOAD_FAST__LOAD_FAST</code>	<code>(dy, self)</code>
<code>cls = type(self)</code>	<code>LOAD_ATTR_INSTANCE_VALUE</code>	<code>(y)</code>
	<code>BINARY_OP_ADD_INT</code>	<code>(+)</code>
	<code>STORE_FAST</code>	<code>(y)</code>
	<code>LOAD_GLOBAL_BUILTIN</code>	<code>(type)</code>
	<code>LOAD_FAST</code>	<code>(self)</code>
	<code>PRECALL_NO_KW_TYPE_1</code>	<code>(1)</code>
	<code>STORE_FAST</code>	<code>(cls)</code>

Specialized Instructions

```
dis.dis(Point.shifted, adaptive=True)
```

```
LOAD_FAST__LOAD_FAST      (dy, self)
LOAD_ATTR_INSTANCE_VALUE  (y)
BINARY_OP_ADD_INT         (+)
STORE_FAST                 (y)
LOAD_GLOBAL_BUILTIN       (type)
LOAD_FAST                  (self)
PRECALL_NO_KW_TYPE_1      (1)
STORE_FAST                 (cls)
```

Inline Caches

```
dis.dis(Point.shifted, adaptive=True)
```

```
LOAD_FAST__LOAD_FAST      (dy, self)
LOAD_ATTR_INSTANCE_VALUE  (y)
BINARY_OP_ADD_INT         (+)
STORE_FAST                 (y)
LOAD_GLOBAL_BUILTIN       (type)
LOAD_FAST                  (self)
PRECALL_NO_KW_TYPE_1      (1)
STORE_FAST                 (cls)
```

Inline Caches

```
dis.dis(Point.shifted, adaptive=True, show_caches=True)
```

```
LOAD_FAST__LOAD_FAST      (dy, self)
LOAD_ATTR_INSTANCE_VALUE  (y)
BINARY_OP_ADD_INT          (+)
STORE_FAST                 (y)
LOAD_GLOBAL_BUILTIN        (type)
LOAD_FAST                  (self)
PRECALL_NO_KW_TYPE_1       (1)
STORE_FAST                 (cls)
```

Inline Caches

dis.dis(Point.shifted, adaptive=True, show_caches=True)

```
LOAD_FAST__LOAD_FAST      (dy, self)
LOAD_ATTR_INSTANCE_VALUE  (y)
CACHE                      (counter)
CACHE                      (version)
CACHE                      (version)
CACHE                      (index)
BINARY_OP_ADD_INT          (+)
CACHE                      (counter)
STORE_FAST                 (y)
LOAD_GLOBAL_BUILTIN        (type)
CACHE                      (counter)
CACHE                      (index)
CACHE                      (version)
CACHE                      (version)
CACHE                      (version)
LOAD_FAST                  (self)
PRECALL_NO_KW_TYPE_1       (1)
CACHE                      (counter)
STORE_FAST                 (cls)
```

Inline Caches

```
dis.dis(Point.shifted, adaptive=True)
```

```
LOAD_FAST__LOAD_FAST      (dy, self)
LOAD_ATTR_INSTANCE_VALUE  (y)
BINARY_OP_ADD_INT         (+)
STORE_FAST                 (y)
LOAD_GLOBAL_BUILTIN       (type)
LOAD_FAST                 (self)
PRECALL_NO_KW_TYPE_1      (1)
STORE_FAST                 (cls)
```

Inline Caches

dis.dis(Point.shifted)

```
LOAD_FAST      (dy)
LOAD_FAST      (self)
LOAD_ATTR      (y)
BINARY_OP      (+)
STORE_FAST     (y)
LOAD_GLOBAL    (type)
LOAD_FAST      (self)
PRECALL        (1)
CALL           (1)
STORE_FAST     (cls)
```


Bytecode

`dis.dis(Point.shifted, adaptive=True)`

RESUME_QUICK		LOAD_GLOBAL_BUILTIN	(type)
LOAD_FAST__LOAD_FAST	(dx)	LOAD_FAST	(self)
LOAD_FAST	(self)	PRECALL_NO_KW_TYPE_1	(1)
LOAD_ATTR_INSTANCE_VALUE	(x)	CALL_ADAPTIVE	(1)
BINARY_OP_ADD_FLOAT	(+)	STORE_FAST	(cls)
STORE_FAST__LOAD_FAST	(x)	PUSH_NULL	
LOAD_FAST__LOAD_FAST	(dy)	LOAD_FAST__LOAD_FAST	(cls)
LOAD_FAST	(self)	LOAD_FAST__LOAD_FAST	(x)
LOAD_ATTR_INSTANCE_VALUE	(y)	LOAD_FAST	(y)
BINARY_OP_ADD_FLOAT	(+)	PRECALL_ADAPTIVE	(2)
STORE_FAST	(y)	CALL_ADAPTIVE	(2)
		RETURN_VALUE	

Bytecode

dis.dis(Point.shifted, adaptive=True)

RESUME_QUICK		LOAD_GLOBAL_BUILTIN	(type)
LOAD_FAST__LOAD_FAST	(dx)	LOAD_FAST	(self)
LOAD_FAST	(self)	PRECALL_NO_KW_TYPE_1	(1)
LOAD_ATTR_INSTANCE_VALUE	(x)	CALL_ADAPTIVE	(1)
BINARY_OP_ADD_FLOAT	(+)	STORE_FAST	(cls)
STORE_FAST__LOAD_FAST	(x)	PUSH_NULL	
LOAD_FAST__LOAD_FAST	(dy)	LOAD_FAST__LOAD_FAST	(cls)
LOAD_FAST	(self)	LOAD_FAST__LOAD_FAST	(x)
LOAD_ATTR_INSTANCE_VALUE	(y)	LOAD_FAST	(y)
BINARY_OP_ADD_FLOAT	(+)	PRECALL_ADAPTIVE	(2)
STORE_FAST	(y)	CALL_ADAPTIVE	(2)
		RETURN_VALUE	

Bytecode

dis.dis(Point.shifted, adaptive=True)

RESUME_QUICK		LOAD_GLOBAL_BUILTIN	(type)
LOAD_FAST__LOAD_FAST	(dx)	LOAD_FAST	(self)
LOAD_FAST	(self)	PRECALL_NO_KW_TYPE_1	(1)
LOAD_ATTR_INSTANCE_VALUE	(x)	CALL_ADAPTIVE	(1)
BINARY_OP_ADD_FLOAT	(+)	STORE_FAST	(cls)
STORE_FAST__LOAD_FAST	(x)	PUSH_NULL	
LOAD_FAST__LOAD_FAST	(dy)	LOAD_FAST__LOAD_FAST	(cls)
LOAD_FAST	(self)	LOAD_FAST__LOAD_FAST	(x)
LOAD_ATTR_INSTANCE_VALUE	(y)	LOAD_FAST	(y)
BINARY_OP_ADD_FLOAT	(+)	PRECALL_ADAPTIVE	(2)
STORE_FAST	(y)	CALL_ADAPTIVE	(2)
		RETURN_VALUE	

Source Code

```
RESUME_QUICK
LOAD_FAST__LOAD_FAST      (dx)
LOAD_FAST                  (self)
LOAD_ATTR_INSTANCE_VALUE  (x)
BINARY_OP_ADD_FLOAT        (+)
STORE_FAST__LOAD_FAST     (x)
LOAD_FAST__LOAD_FAST      (dy)
LOAD_FAST                  (self)
LOAD_ATTR_INSTANCE_VALUE  (y)
BINARY_OP_ADD_FLOAT        (+)
STORE_FAST                 (y)

LOAD_GLOBAL_BUILTIN      (type)
LOAD_FAST                 (self)
PRECALL_NO_KW_TYPE_1     (1)
CALL_ADAPTIVE             (1)
STORE_FAST                (cls)
PUSH_NULL
LOAD_FAST__LOAD_FAST     (cls)
LOAD_FAST__LOAD_FAST     (x)
LOAD_FAST                 (y)
PRECALL_ADAPTIVE          (2)
CALL_ADAPTIVE             (2)
RETURN_VALUE
```

Source Code

```
x = dx + self.x
```

```
LOAD_FAST__LOAD_FAST      (dy)
LOAD_FAST                  (self)
LOAD_ATTR_INSTANCE_VALUE  (y)
BINARY_OP_ADD_FLOAT        (+)
STORE_FAST                 (y)
```

```
LOAD_GLOBAL_BUILTIN      (type)
LOAD_FAST                 (self)
PRECALL_NO_KW_TYPE_1     (1)
CALL_ADAPTIVE             (1)
STORE_FAST                (cls)
PUSH_NULL
LOAD_FAST__LOAD_FAST     (cls)
LOAD_FAST__LOAD_FAST     (x)
LOAD_FAST                 (y)
PRECALL_ADAPTIVE          (2)
CALL_ADAPTIVE             (2)
RETURN_VALUE
```

Source Code

```
x = dx + self.x
```

```
y = dy + self.y
```

```
LOAD_GLOBAL_BUILTIN    (type)
LOAD_FAST               (self)
PRECALL_NO_KW_TYPE_1   (1)
CALL_ADAPTIVE           (1)
STORE_FAST              (cls)
PUSH_NULL
LOAD_FAST__LOAD_FAST   (cls)
LOAD_FAST__LOAD_FAST   (x)
LOAD_FAST              (y)
PRECALL_ADAPTIVE        (2)
CALL_ADAPTIVE           (2)
RETURN_VALUE
```

Source Code

```
x = dx + self.x
```

```
y = dy + self.y
```

```
cls = type(self)
```

```
LOAD_FAST__LOAD_FAST (cls)
```

```
LOAD_FAST__LOAD_FAST (x)
```

```
LOAD_FAST (y)
```

```
PRECALL_ADAPTIVE (2)
```

```
CALL_ADAPTIVE (2)
```

```
RETURN_VALUE
```

Source Code

```
x = dx + self.x
```

```
y = dy + self.y
```

```
cls = type(self)
```

```
return cls(x, y)
```


Source Code

```
class Point:
    def __init__(self, x: float, y: float) -> None:
        self.x = x
        self.y = y

    def shifted(self, dx: float, dy: float) -> typing.Self:
        x = dx + self.x
        y = dy + self.y
        cls = type(self)
        return cls(x, y)
```

Source Code

```
class Point:
    def __init__(self, x: float, y: float) -> None:
        self.x = x
        self.y = y

    def shifted(self, dx: float, dy: float) -> typing.Self:
        x = dx + self.x
        y = dy + self.y
        cls = type(self)
        return cls(x, y)
```

Specialist

```
class Point:
    def __init__(self, x: float, y: float) -> None:
        self.x = x
        self.y = y

    def shifted(self, dx: float, dy: float) -> typing.Self:
        x = dx + self.x
        y = dy + self.y
        cls = type(self)
        return cls(x, y)
```

Specialist

\$ specialist --blue point.py

```
class Point:
    def __init__(self, x: float, y: float) -> None:
        self.x = x
        self.y = y

    def shifted(self, dx: float, dy: float) -> typing.Self:
        x = dx + self.x
        y = dy + self.y
        cls = type(self)
        return cls(x, y)
```

Specialist

\$ specialist --blue point.py

```
import typing

class Point:
    def __init__(self, x: float, y: float) -> None:
        self.x = x
        self.y = y

    def shifted(self, dx: float, dy: float) -> typing.Self:
        x = dx + self.x
        y = dy + self.y
        cls = type(self)
        return cls(x, y)

def actually_run_something() -> None:
    p = Point(3.14, 2.72)
    for i in range(100):
        p = p.shifted(19.95, 12.06)

if __name__ == "__main__":
    actually_run_something()
```

Specialist

```
$ specialist --blue point.py
```

```
def actually_run_something() -> None:
    p = Point(3.14, 2.72)
    for i in range(100):
        p = p.shifted(19.95, 12.06)

if __name__ == "__main__":
    actually_run_something()
```

Specialist

brandtbucher/specialist

Specialist

LATEST

V0.4.1

RELEASED

OCTOBER

BUILD

PASSING

ISSUES

1

Specialist uses [fine-grained location](#) information to create visual representations of exactly *where* and *how* CPython 3.11's new [specializing, adaptive interpreter](#) optimizes your code.

```
def encode_decode(key: str, text: str) -> str:
    out = []
    for i, t in enumerate(text):
        k = key[i % len(key)]
        out.append(chr(ord(t) ^ ord(k)))
    return "".join(out)
```

Installation

Specialist supports CPython 3.11+ on all platforms.

To install, just run:

```
$ pip install specialist
```

PEP 659: Specializing Adaptive Interpreter

Thank you!

@brandtbucher

Thank you!

@brandtbucher | brandt@python.org

