
cisco
You make possible

How does the Python Interpreter interpret my Python?

Conor Murphy
Technical Solutions Architect – Data Centre

DEVLIT-4056

cisco Live!
Barcelona | January 27-31, 2020

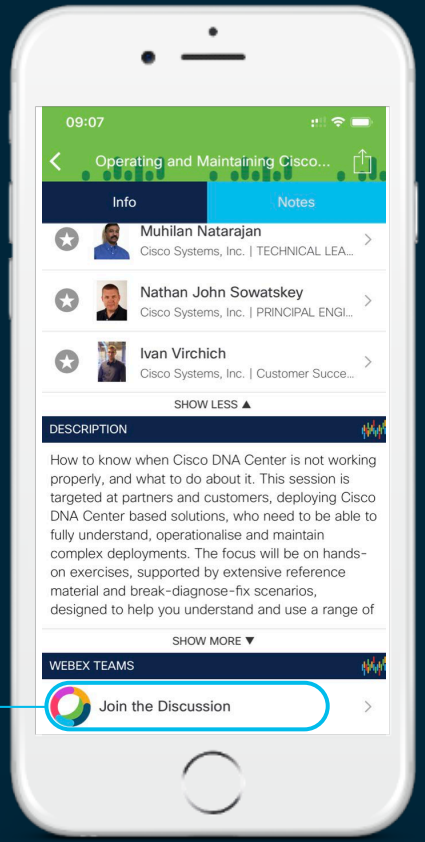
Cisco Webex Teams

Questions?

Use Cisco Webex Teams to chat with the speaker after the session

How

1. Find this session in the Cisco Events Mobile App
2. Click “Join the Discussion”
3. Install Webex Teams or go directly to the team space
4. Enter messages/questions in the team space



“A Python program is read by a parser. Input to the parser is a stream of tokens, generated by the lexical analyzer. ”

https://docs.python.org/3/reference/lexical_analysis.html

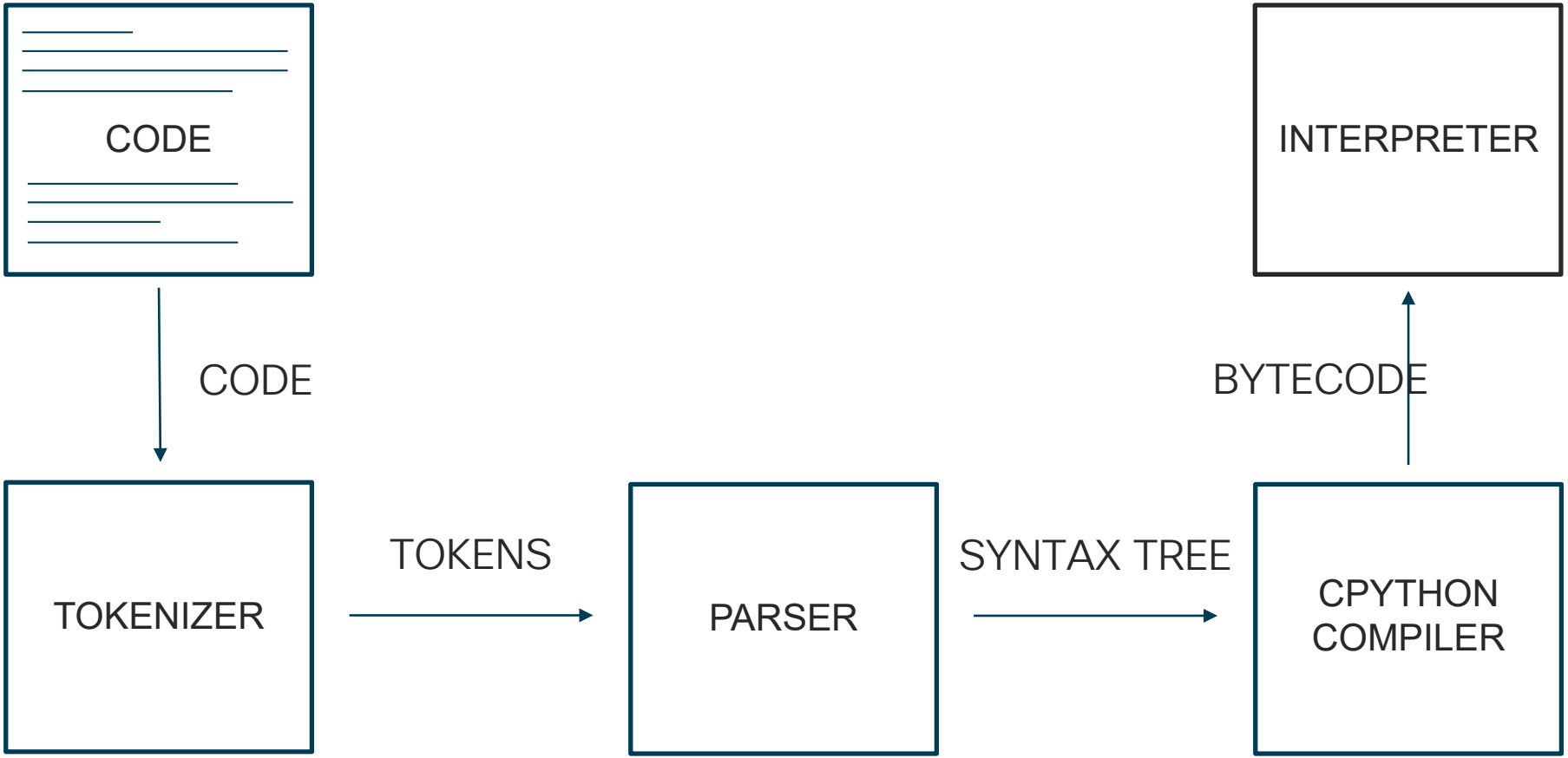
"CPython is the reference implementation of the Python programming language. Written in C and Python, CPython is the default and most widely-used implementation of the language"

https://en.wikipedia.org/wiki/CPython

"CPython can be defined as both an interpreter and a compiler as it compiles Python code into bytecode before interpreting it. "

https://en.wikipedia.org/wiki/CPython

https://devguide.python.org/compiler/



```python
'''This is an example app for the Cisco Live 2020 Barcelona Session - DEVLIT-4056'''

# Import the datetime library
from datetime import datetime

def whatIsTodaysDate():
    return datetime.now()

def concatenateTwoStrings(stringA, stringB):
    return stringA + stringB

todaysDate = whatIsTodaysDate()
print(todaysDate)

oneString = concatenateTwoStrings("DEVLIT-4056 ", "How Does The Python Interpreter Interpret My Python")
print(oneString)

```

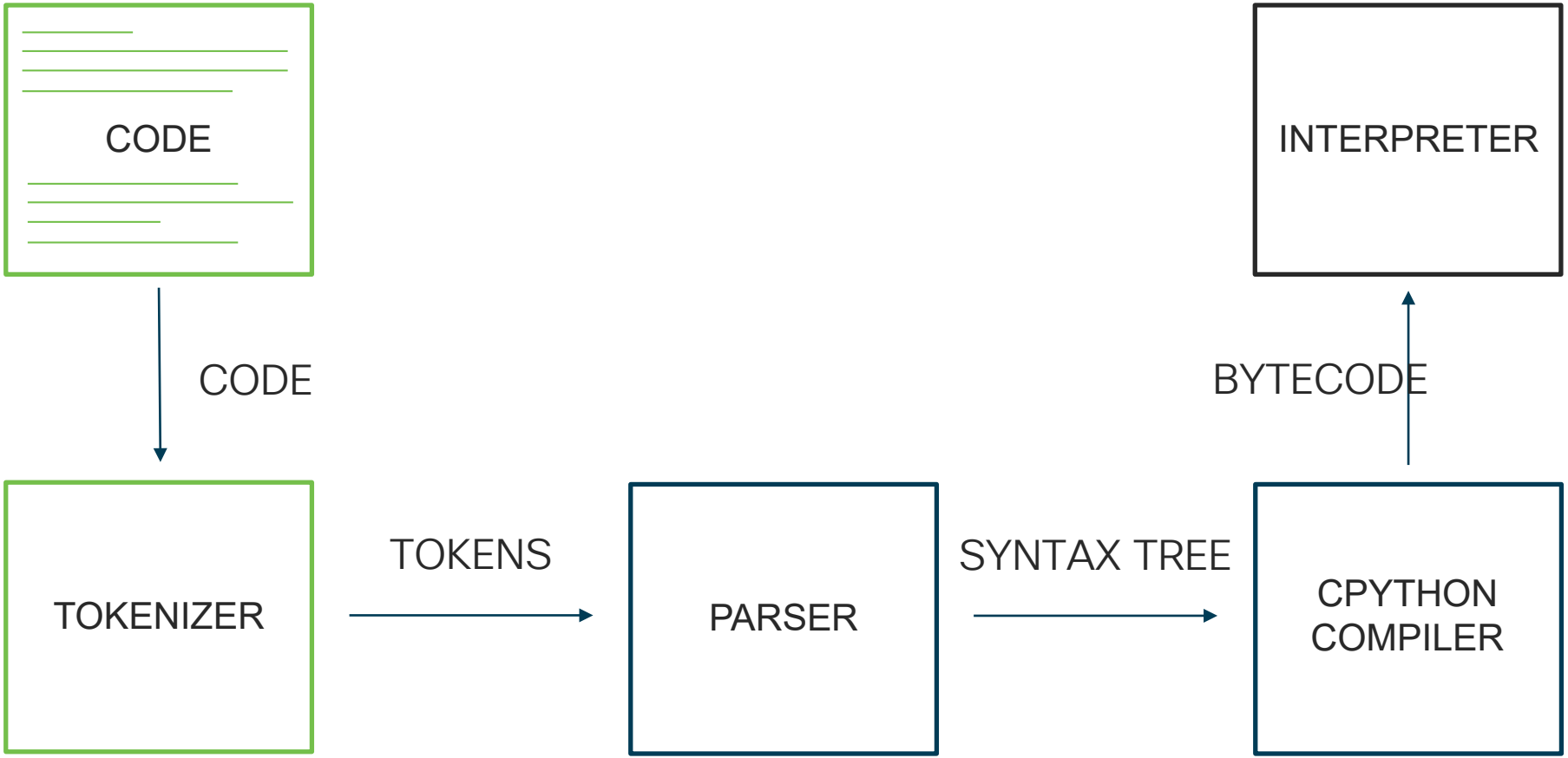
"The tokenize module provides a lexical scanner for Python source code, implemented in Python"

https://docs.python.org/3/library/tokenize.html

```
1   import tokenize, token
2   from tabulate import tabulate
3
4   if __name__ == '__main__':
5
6       outputTable = []
7
8       with open('example_app.py', 'rb') as f:
9           for line in tokenize.tokenize(f.readline):
10              outputTable.append([line.string,token.tok_name[line.type]])
11
12          print(tabulate(outputTable, headers=["Symbol","Type"]))
```

```
CONMURPH-M-J440:Python Interpreter conmurph$ python tokens.py
Symbol                                                                                  Type
--------------------------------------------------------------------------------------  ---------
utf-8                                                                                   ENCODING
'''This is an example app for the Cisco Live 2020 Barcelona Session - DEVLIT-4056'''    STRING
                                                                                        NEWLINE
                                                                                        NL
# Import the datetime library                                                           COMMENT
                                                                                        NL
from                                                                                    NAME
datetime                                                                                NAME
import                                                                                  NAME
datetime                                                                                NAME
                                                                                        NEWLINE
                                                                                        NL
def                                                                                     NAME
whatIsTodaysDate                                                                        NAME
(                                                                                       OP
)                                                                                       OP
:                                                                                       OP
                                                                                        NEWLINE
                                                                                        INDENT
return                                                                                  NAME
datetime                                                                                NAME
.                                                                                       OP
now                                                                                     NAME
(                                                                                       OP
)                                                                                       OP
                                                                                        NEWLINE
                                                                                        NL
                                                                                        DEDENT
def                                                                                     NAME
concatenateTwoStrings                                                                   NAME
(                                                                                       OP
stringA                                                                                 NAME
,                                                                                       OP
stringB                                                                                 NAME
)                                                                                       OP
:                                                                                       OP
                                                                                        NEWLINE
                                                                                        INDENT
return                                                                                  NAME
stringA                                                                                 NAME
+                                                                                       OP
stringB                                                                                 NAME
                                                                                        NEWLINE
                                                                                        NL
                                                                                        DEDENT
todaysDate                                                                              NAME
=                                                                                       OP
whatIsTodaysDate                                                                        NAME
(                                                                                       OP
)                                                                                       OP
                                                                                        NEWLINE
print                                                                                   NAME
```

```
Symbol                                                                                  Type
--------------------------------------------------------------------------------------  ---------
(                                                                                       OP
todaysDate                                                                              NAME
)                                                                                       OP
                                                                                        NEWLINE
                                                                                        NL
oneString                                                                               NAME
=                                                                                       OP
concatenateTwoStrings                                                                   NAME
(                                                                                       OP
"DEVLIT-4056 "                                                                          STRING
,                                                                                       OP
"How Does The Python Interpreter Interpret My Python"                                   STRING
)                                                                                       OP
                                                                                        NEWLINE
print                                                                                   NAME
(                                                                                       OP
oneString                                                                               NAME
)                                                                                       OP
                                                                                        NEWLINE
                                                                                        NL
                                                                                        ENDMARKER
```



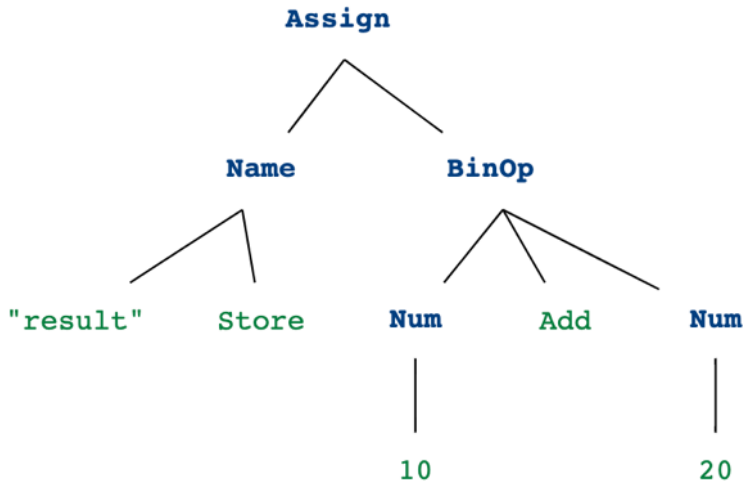
"Abstract syntax trees are data structures widely used in compilers to represent the structure of program code"

https://en.wikipedia.org/wiki/Abstract_syntax_tree

"A complete traversal of the tree allows verification of the correctness of the program"

https://en.wikipedia.org/wiki/Abstract_syntax_tree

result = 10 + 20



“The ast module helps Python applications to process trees of the Python abstract syntax grammar“

https://docs.python.org/3/library/ast.html

“An AST unparser for Python“

https://astunparse.readthedocs.io/en/latest/

ast.parse()

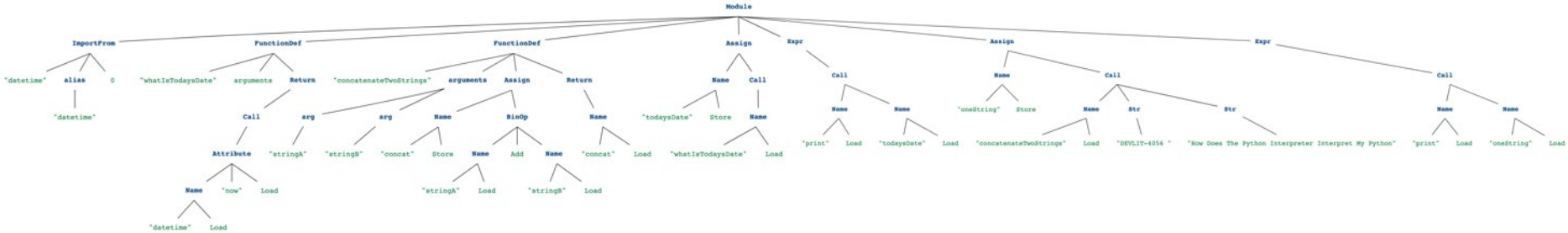
Builds an ast from code stored as a string

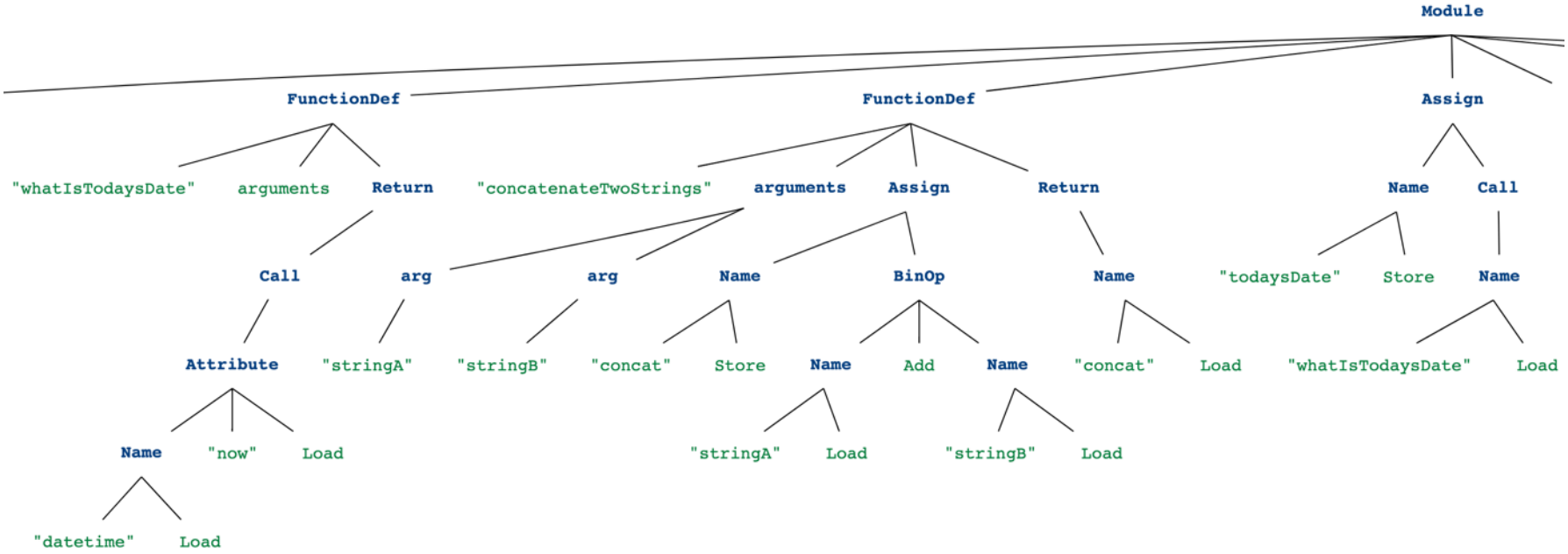
https://greentreesnakes.readthedocs.io/en/latest/tofrom.html


```
1   import ast
2   import astunparse
3
4   if __name__ == '__main__':
5
6       with open('example_app/example_app.py', 'rb') as f:
7           sourceCode = f.read()
8           tree = ast.parse(sourceCode)
9           print(astunparse.dump(tree))
10
```

```
CONMURPH-M-J440:Python Interpreter conmurph$ python parser.py
Module(body=[
  Expr(value=Str(s='This is an example app for the Cisco Live 2020 Barcelona Session - DEVLIT-4056')),
  ImportFrom(
    module='datetime',
    names=[alias(
      name='datetime',
      asname=None)],
    level=0),
  FunctionDef(
    name='whatIsTodaysDate',
    args=arguments(
      args=[],
      vararg=None,
      kwonlyargs=[],
      kw_defaults=[],
      kwarg=None,
      defaults=[]),
    body=[Return(value=Call(
      func=Attribute(
        value=Name(
          id='datetime',
          ctx=Load()),
        attr='now',
        ctx=Load()),
      args=[],
      keywords=[]))],
    decorator_list=[],
    returns=None),
  FunctionDef(
    name='concatenateTwoStrings',
    args=arguments(
      args=[
        arg(
          arg='stringA',
          annotation=None),
        arg(
          arg='stringB',
          annotation=None)],
      vararg=None,
      kwonlyargs=[],
      kw_defaults=[],
      kwarg=None,
      defaults=[]),
    body=[Return(value=BinOp(
      left=Name(
        id='stringA',
        ctx=Load()),
      op=Add(),
      right=Name(
        id='stringB',
        ctx=Load())))],
    decorator_list=[],
    returns=None),
  Assign(
```

```
    targets=[Name(
      id='todaysDate',
      ctx=Store())],
    value=Call(
      func=Name(
        id='whatIsTodaysDate',
        ctx=Load()),
      args=[],
      keywords=[])),
  Expr(value=Call(
    func=Name(
      id='print',
      ctx=Load()),
    args=[Name(
      id='todaysDate',
      ctx=Load())],
    keywords=[])),
  Assign(
    targets=[Name(
      id='oneString',
      ctx=Store())],
    value=Call(
      func=Name(
        id='concatenateTwoStrings',
        ctx=Load()),
      args=[
        Str(s='DEVLIT-4056 '),
        Str(s='How Does The Python Interpreter Interpret My Python')],
      keywords=[])),
  Expr(value=Call(
    func=Name(
      id='print',
      ctx=Load()),
    args=[Name(
      id='oneString',
      ctx=Load())],
    keywords=[]))])
```





showast

pypi package 0.2.4 | receives 0.00 USD/week

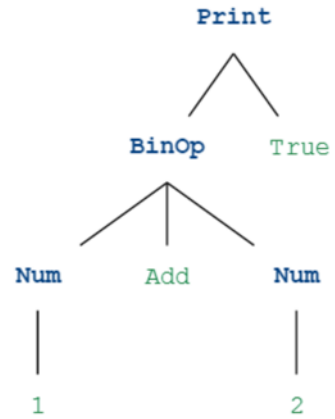
An IPython/Jupyter notebook plugin for visualizing abstract syntax trees.

Example usage

Examples can be found in this IPython notebook.

```
import showast
```

```
%%showast
print 1 + 2
```



https://github.com/hchasestevens/show_ast

ast.walk(node)

Recursively yield all descendant nodes in the tree starting at *node* (including *node* itself), in no specified order

This is useful if you only want to modify nodes in place and don't care about the context

https://docs.python.org/3/library/ast.html

```python
import ast
from tabulate import tabulate

if __name__ == '__main__':

    outputTable = []

    with open('example_app/example_app.py', 'rb') as f:
        sourceCode = f.read()
        tree = ast.parse(sourceCode)

        for node in ast.walk(tree):
            outputTable.append([type(node).__name__ , node])

        print(tabulate(outputTable, headers=["Type","Node"]))
```

```
CONMURPH-M-J440:Python Interpreter conmurph$ python ast_walk.py
Type         Node
-----------  ----------------------------------------
Module       <_ast.Module object at 0x106d90c18>
Expr         <_ast.Expr object at 0x106e80978>
ImportFrom   <_ast.ImportFrom object at 0x106e809e8>
FunctionDef  <_ast.FunctionDef object at 0x106e80a90>
FunctionDef  <_ast.FunctionDef object at 0x106e80cc0>
Assign       <_ast.Assign object at 0x106e80f60>
Expr         <_ast.Expr object at 0x106e8a080>
Assign       <_ast.Assign object at 0x106e8a160>
Expr         <_ast.Expr object at 0x106e8a2b0>
Str          <_ast.Str object at 0x106e809b0>
alias        <_ast.alias object at 0x106e80a20>
arguments    <_ast.arguments object at 0x106e80ac8>
Return       <_ast.Return object at 0x106e80b70>
arguments    <_ast.arguments object at 0x106e80cf8>
Assign       <_ast.Assign object at 0x106e80da0>
Return       <_ast.Return object at 0x106e80ef0>
Name         <_ast.Name object at 0x106e80f98>
Call         <_ast.Call object at 0x106e80fd0>
Call         <_ast.Call object at 0x106e8a0b8>
Name         <_ast.Name object at 0x106e8a198>
Call         <_ast.Call object at 0x106e8a1d0>
Call         <_ast.Call object at 0x106e8a2e8>
Call         <_ast.Call object at 0x106e80ba8>
arg          <_ast.arg object at 0x106e80d30>
arg          <_ast.arg object at 0x106e80d68>
Name         <_ast.Name object at 0x106e80dd8>
BinOp        <_ast.BinOp object at 0x106e80e48>
```

```
Name         <_ast.Name object at 0x106e80f28>
Store        <_ast.Store object at 0x106da3ef0>
Name         <_ast.Name object at 0x106e8a048>
Name         <_ast.Name object at 0x106e8a0f0>
Name         <_ast.Name object at 0x106e8a128>
Store        <_ast.Store object at 0x106da3ef0>
Name         <_ast.Name object at 0x106e8a208>
Str          <_ast.Str object at 0x106e8a240>
Str          <_ast.Str object at 0x106e8a278>
Name         <_ast.Name object at 0x106e8a320>
Name         <_ast.Name object at 0x106e8a358>
Attribute    <_ast.Attribute object at 0x106e80be0>
Store        <_ast.Store object at 0x106da3ef0>
Name         <_ast.Name object at 0x106e80e80>
Add          <_ast.Add object at 0x106da8748>
Name         <_ast.Name object at 0x106e80eb8>
Load         <_ast.Load object at 0x106da3e48>
Load         <_ast.Load object at 0x106da3e48>
Load         <_ast.Load object at 0x106da3e48>
Load         <_ast.Load object at 0x106da3e48>
Load         <_ast.Load object at 0x106da3e48>
Load         <_ast.Load object at 0x106da3e48>
Load         <_ast.Load object at 0x106da3e48>
Name         <_ast.Name object at 0x106e80c18>
Load         <_ast.Load object at 0x106da3e48>
Load         <_ast.Load object at 0x106da3e48>
Load         <_ast.Load object at 0x106da3e48>
Load         <_ast.Load object at 0x106da3e48>
```

isinstance(object, classinfo)

Return *True* if the object argument is an instance of the classinfo argument, or of a . . . subclass thereof

If object is not an object of the given type, the function always returns *False*

. . .

https://docs.python.org/3/library/functions.html#isinstance

```python
import ast
import astunparse
from tabulate import tabulate

if __name__ == '__main__':

    outputTable = []

    with open('example_app/example_app.py', 'rb') as f:
        sourceCode = f.read()
        tree = ast.parse(sourceCode)

        for node in ast.walk(tree):
            if isinstance(node, ast.FunctionDef):
                print("Nodetype: {} {} ".format(type(node).__name__,node))
                print(astunparse.unparse(node))
```

```
CONMURPH-M-J440:Python Interpreter conmurph$ python ast_isinstance.py
Nodetype: FunctionDef <_ast.FunctionDef object at 0x103526e10>


def whatIsTodaysDate():
    return datetime.now()

Nodetype: FunctionDef <_ast.FunctionDef object at 0x1035a2b70>


def concatenateTwoStrings(stringA, stringB):
    concat = (stringA + stringB)
    return concat
```

```
Python Interpreter$  python ast_isinstance_count.py
 Function: whatIsTodaysDate
 Function: concatenateTwoStrings

 Total Functions: 2
```

ast.NodeVisitor

Base class that walks the abstract syntax tree and calls a visitor function for every node found

https://docs.python.org/2/library/ast.html#ast.NodeVisitor

```python
import ast
import astunparse

class codeTree(ast.NodeVisitor):

    def generic_visit(self, node):
        ast.NodeVisitor.generic_visit(self, node)

    def visit_FunctionDef(self, node):
        print("Function Name: {}".format(node.name))
        print(astunparse.unparse(node))
        self.generic_visit(node)

if __name__ == '__main__':

    with open('example_app/example_app.py', 'rb') as f:
        sourceCode = f.read()
        tree = ast.parse(sourceCode)

        codeTree().visit(tree)
```

Class Inheritance

The codeTree class is a child of the NodeVisitor class and inherits functions from NodeVisitor

Overriding parent function from the NodeVisitor class

visit() is a function within the NodeVisitor class

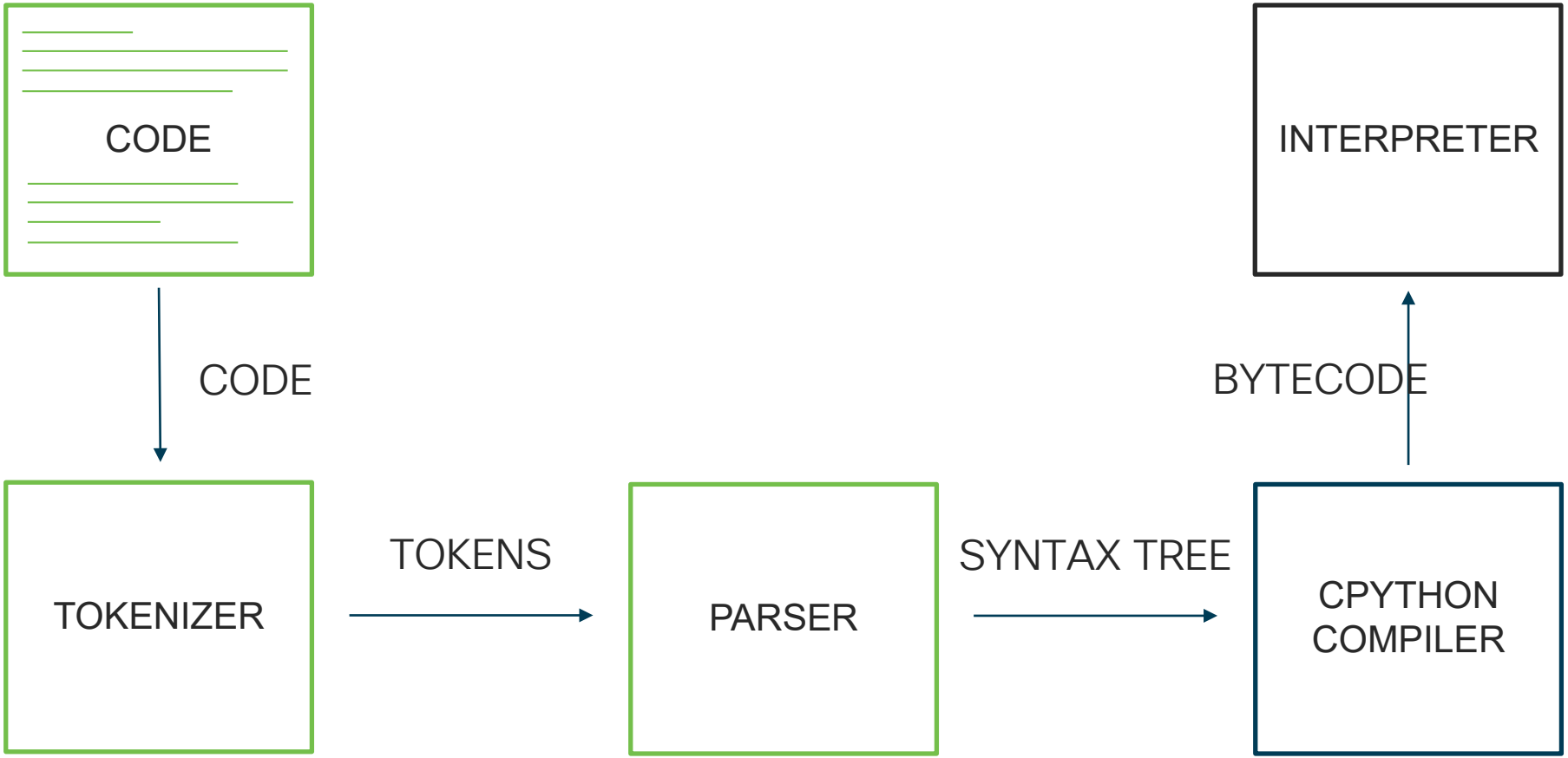
https://docs.python.org/3/tutorial/classes.html

```
CONMURPH-M-J440:Python Interpreter conmurph$ python ast_visit.py
Function Name: whatIsTodaysDate


def whatIsTodaysDate():
    return datetime.now()

Function Name: concatenateTwoStrings


def concatenateTwoStrings(stringA, stringB):
    concat = (stringA + stringB)
    return concat
```

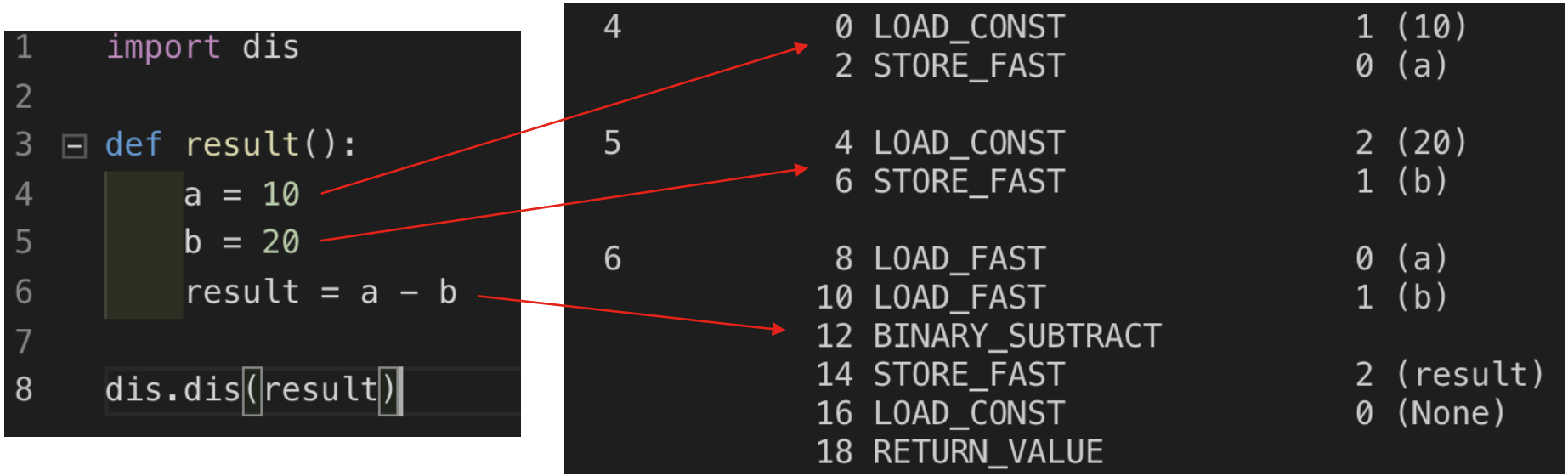


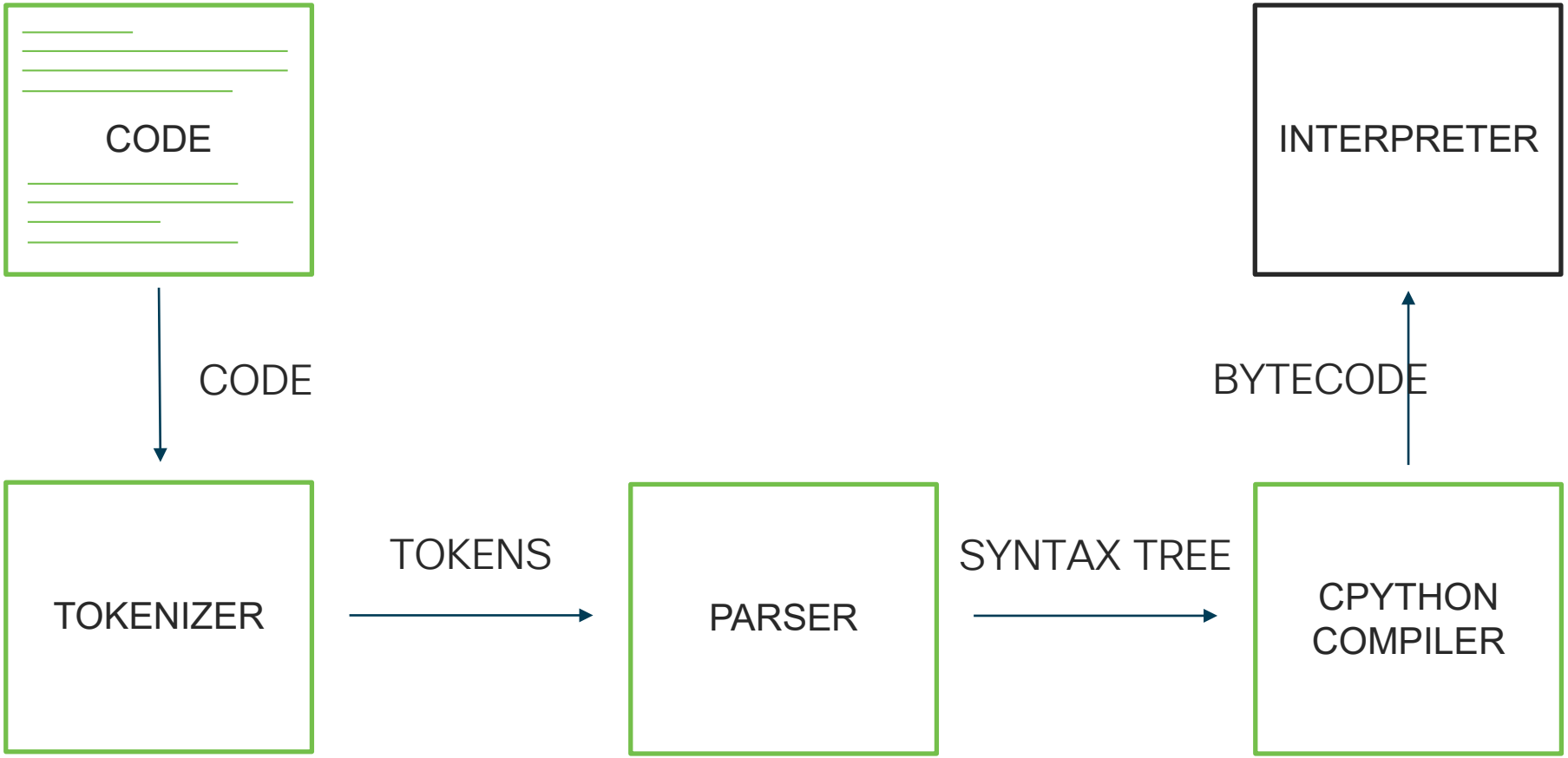
“The dis module supports the analysis of CPython bytecode by disassembling it.“

https://docs.python.org/3/library/dis.html

```
1  import dis
2
3  if __name__ == '__main__':
4      with open('example_app/__pycache__/example_app.cpython-37.pyc', 'rb') as f:
5          sourceCode = f.read()
6          dis.dis(sourceCode)
```

```
conmurph$ python bytecode.py example_app/__pycache__/example_app.cpython-37.pyc
     >>    2 <13>
     >>    4 <0>
           6 <0>
     >>    8 <252>                137
          10 <247>                 93
          12 <211>                  1
     >>   14 <0>
          16 <227>                  0
          18 <0>
          20 <0>
          22 <0>
          24 <0>
          26 <0>
          28 <0>
          30 <0>
     >>   32 <0>
          34 <0>
          36 <0>
          38 GET_ITER
          40 <0>
          42 LOAD_CONST             0 (0)
          44 STORE_NAME             0 (0)
     >>   46 LOAD_CONST             1 (1)
          48 LOAD_CONST             2 (2)
          50 IMPORT_NAME            1 (1)
          52 IMPORT_FROM            1 (1)
          54 STORE_NAME             1 (1)
          56 POP_TOP
          58 LOAD_CONST             3 (3)
          60 LOAD_CONST             4 (4)
          62 MAKE_FUNCTION          0
          64 STORE_NAME             2 (2)
          66 LOAD_CONST             5 (5)
          68 LOAD_CONST             6 (6)
          70 MAKE_FUNCTION          0
          72 STORE_NAME             3 (3)
          74 LOAD_NAME              2 (2)
          76 CALL_FUNCTION          0
          78 STORE_NAME             4 (4)
          80 LOAD_NAME              5 (5)
          82 LOAD_NAME              4 (4)
          84 CALL_FUNCTION          1
          86 POP_TOP
          88 LOAD_NAME              3 (3)
          90 LOAD_CONST             7 (7)
          92 LOAD_CONST             8 (8)
          94 CALL_FUNCTION          2
          96 STORE_NAME             6 (6)
          98 LOAD_NAME              5 (5)
     >>  104 POP_TOP
         106 LOAD_CONST             9 (9)
     >>  108 RETURN_VALUE
     >>  110 <41>
     >>  112 SETUP_FINALLY         78 (to 192)
     >>  114 IMPORT_STAR
     >>  116 BUILD_MAP             115
         118 <32>
         120 POP_JUMP_IF_TRUE       32
         122 STORE_GLOBAL           110 (110)
         124 <32>
         126 SETUP_LOOP             97 (to 225)
         128 IMPORT_FROM            112 (112)
         130 IMPORT_NAME            101 (101)
         132 <32>
         134 JUMP_IF_TRUE_OR_POP    112
         136 <32>
         138 JUMP_IF_FALSE_OR_POP   114
         140 <32>
         142 BUILD_SET              101
         144 <32>
         146 BUILD_MAP              115
         148 <99>                   111
         150 <32>
         152 BUILD_MAP              118
         154 LOAD_NAME              32 (32)
         156 GET_AITER
         158 GET_AITER
         160 <32>
         162 STORE_GLOBAL           114 (114)
         164 <99>                   101
         166 IMPORT_NAME            111 (111)
         168 JUMP_FORWARD           97 (to 267)
         170 <32>
         172 LOAD_NAME              115 (115)
         174 POP_JUMP_IF_TRUE       105
         176 JUMP_IF_FALSE_OR_POP   110
         178 <32>
         180 <32>
         182 GET_YIELD_FROM_ITER
         184 INPLACE_RSHIFT
         186 IMPORT_STAR
         188 BEFORE_ASYNC_WITH
         190 <53>
     >>  192 <233>                   0
         194 <0>
         196 <0>
         198 POP_TOP
         200 <8>
         202 STORE_GLOBAL           116 (116)
         204 LOAD_NAME              116 (116)
         206 BUILD_MAP              109
         358 LOAD_FAST              0 (0)
         360 LOAD_FAST              1 (1)
         362 BINARY_ADD
         364 STORE_FAST             2 (2)
         366 LOAD_FAST              2 (2)
         368 RETURN_VALUE
         370 <41>
         372 INPLACE_XOR
         374 ROT_THREE
         376 <0>
         378 <41>
         380 STORE_NAME             7 (7)
         382 POP_JUMP_IF_TRUE       116
         384 POP_JUMP_IF_FALSE      105
         386 JUMP_FORWARD           103 (to 491)
         388 BINARY_XOR
         390 <7>
         392 LOAD_GLOBAL            114 (114)
         394 BUILD_MAP              110
         396 BUILD_LIST             66
         398 <218>                  6
         400 <99>                   111
         402 JUMP_FORWARD           99 (to 503)
         404 STORE_GLOBAL           116 (116)
         406 POP_JUMP_IF_FALSE      3
         408 <0>
         410 <0>
         412 ROT_THREE
         414 <0>
         416 POP_JUMP_IF_FALSE      4
         418 <0>
         420 <0>
         422 <21>
         424 JUMP_IF_FALSE_OR_POP   110
         426 <99>                   97
         428 LOAD_GLOBAL            101 (101)
         430 JUMP_FORWARD           97 (to 529)
         432 LOAD_GLOBAL            101 (101)
         434 IMPORT_STAR
         436 JUMP_IF_FALSE_OR_POP   83
         438 LOAD_GLOBAL            114 (114)
         440 BUILD_MAP              110
         442 BUILD_LIST             115
         444 NOP
         446 <0>
         448 POP_JUMP_IF_TRUE       4
         450 <0>
         452 <0>
```





python / cpython

Sponsor | Watch 1.1k | Star 28.7k | Fork 13k

Code | Pull requests 1,054 | Actions | Security | Insights

Branch: 3.6 cpython / Python / ceval.c

Find file | Copy path

miss-islington bpo-35444: Fix error handling when fail to look up builtin "getattr". (... be6ec44 on 11 Dec 2018

69 contributors  and others

5621 lines (5059 sloc) | 172 KB

Raw | Blame | History

```
1
2  /* Execute compiled code */
3
```

https://github.com/python/cpython/blob/3.6/Python/ceval.c

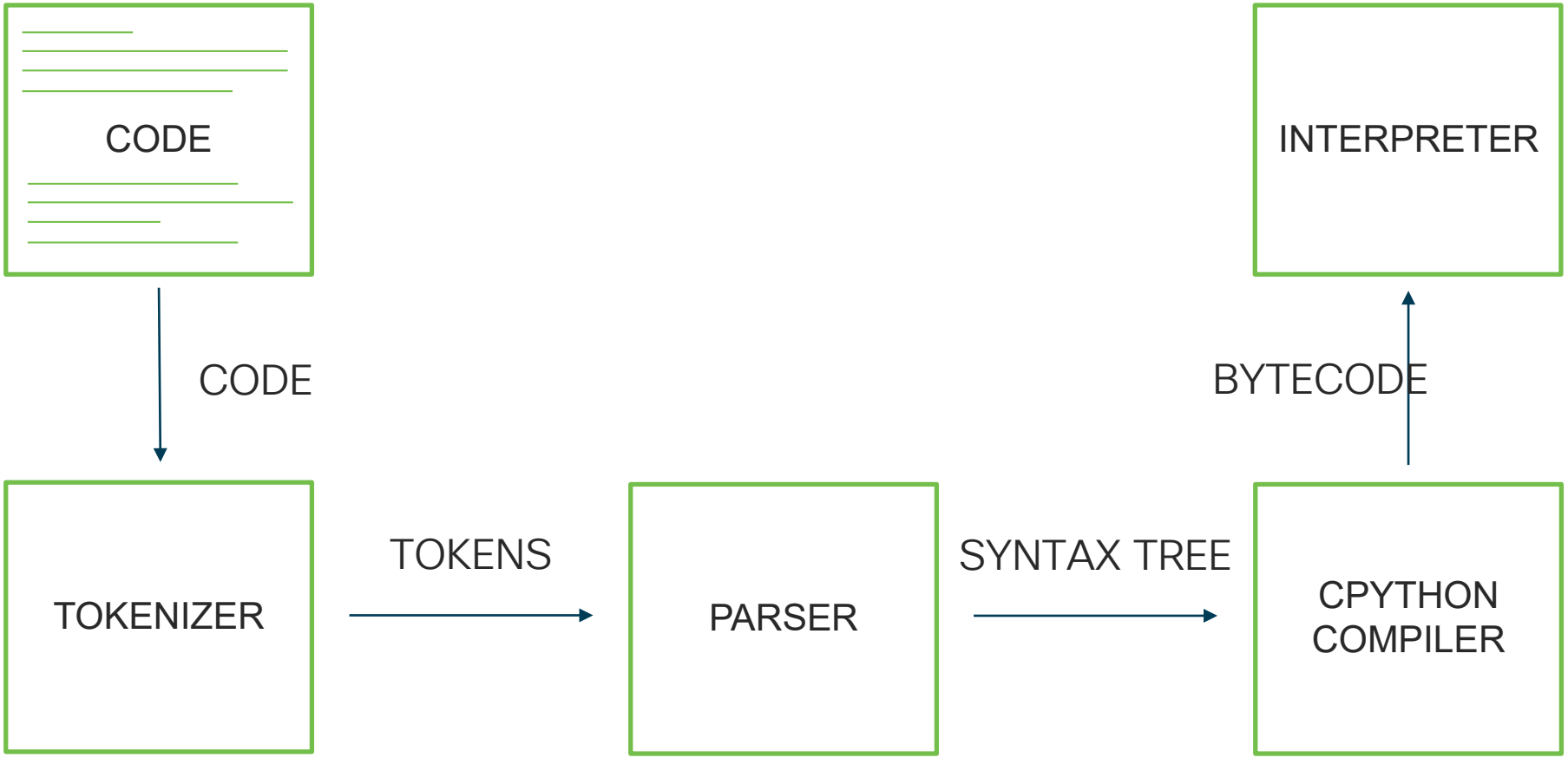
```
740  /* Interpreter main loop */
741
742  PyObject *
743  PyEval_EvalFrame(PyFrameObject *f) {
744      /* This is for backward compatibility with extension modules that
745         used this API; core interpreter code should call
746         PyEval_EvalFrameEx() */
747      return PyEval_EvalFrameEx(f, 0);
748  }
749
750  PyObject *
751  PyEval_EvalFrameEx(PyFrameObject *f, int throwflag)
752  {
753      PyThreadState *tstate = PyThreadState_GET();
754      return tstate->interp->eval_frame(f, throwflag);
755  }
756
```

https://github.com/python/cpython/blob/3.6/Python/ceval.c

```
1274        switch (opcode) {
1275
1276        /* BEWARE!
1277           It is essential that any operation that fails sets either
1278           x to NULL, err to nonzero, or why to anything but WHY_NOT,
1279           and that no operation that succeeds does this! */
1280
1281        TARGET(NOP)
1282            FAST_DISPATCH();
1283
1284        TARGET(LOAD_FAST) {
1285            PyObject *value = GETLOCAL(oparg);
1286            if (value == NULL) {
1287                format_exc_check_arg(PyExc_UnboundLocalError,
1288                                     UNBOUNDLOCAL_ERROR_MSG,
1289                                     PyTuple_GetItem(co->co_varnames, oparg));
1290                goto error;
1291            }
1292            Py_INCREF(value);
1293            PUSH(value);
1294            FAST_DISPATCH();
1295        }
1296
1297        PREDICTED(LOAD_CONST);
1298        TARGET(LOAD_CONST) {
1299            PyObject *value = GETITEM(consts, oparg);
1300            Py_INCREF(value);
1301            PUSH(value);
1302            FAST_DISPATCH();
1303        }
```

```
TARGET(BINARY_SUBTRACT) {
    PyObject *right = POP();
    PyObject *left = TOP();
    PyObject *diff = PyNumber_Subtract(left, right);
    Py_DECREF(right);
    Py_DECREF(left);
    SET_TOP(diff);
    if (diff == NULL)
        goto error;
    DISPATCH();
}
```

https://github.com/python/cpython/blob/3.6/Python/ceval.c



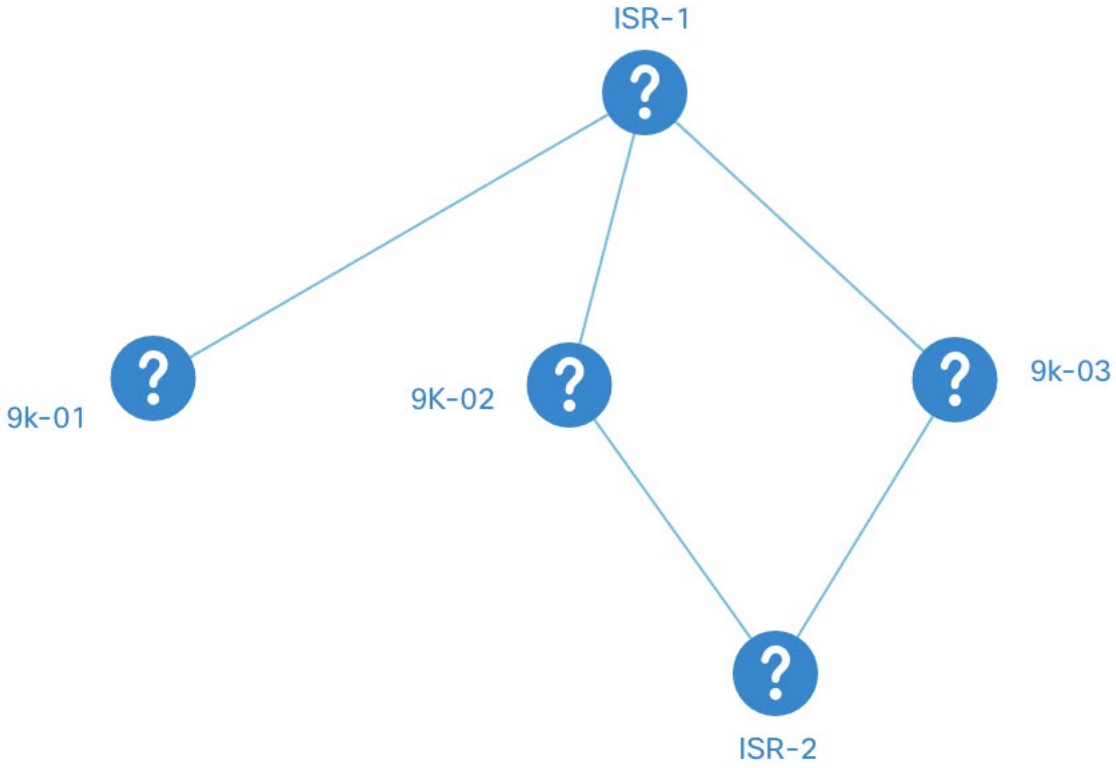
What Else Can I Do?

Learning

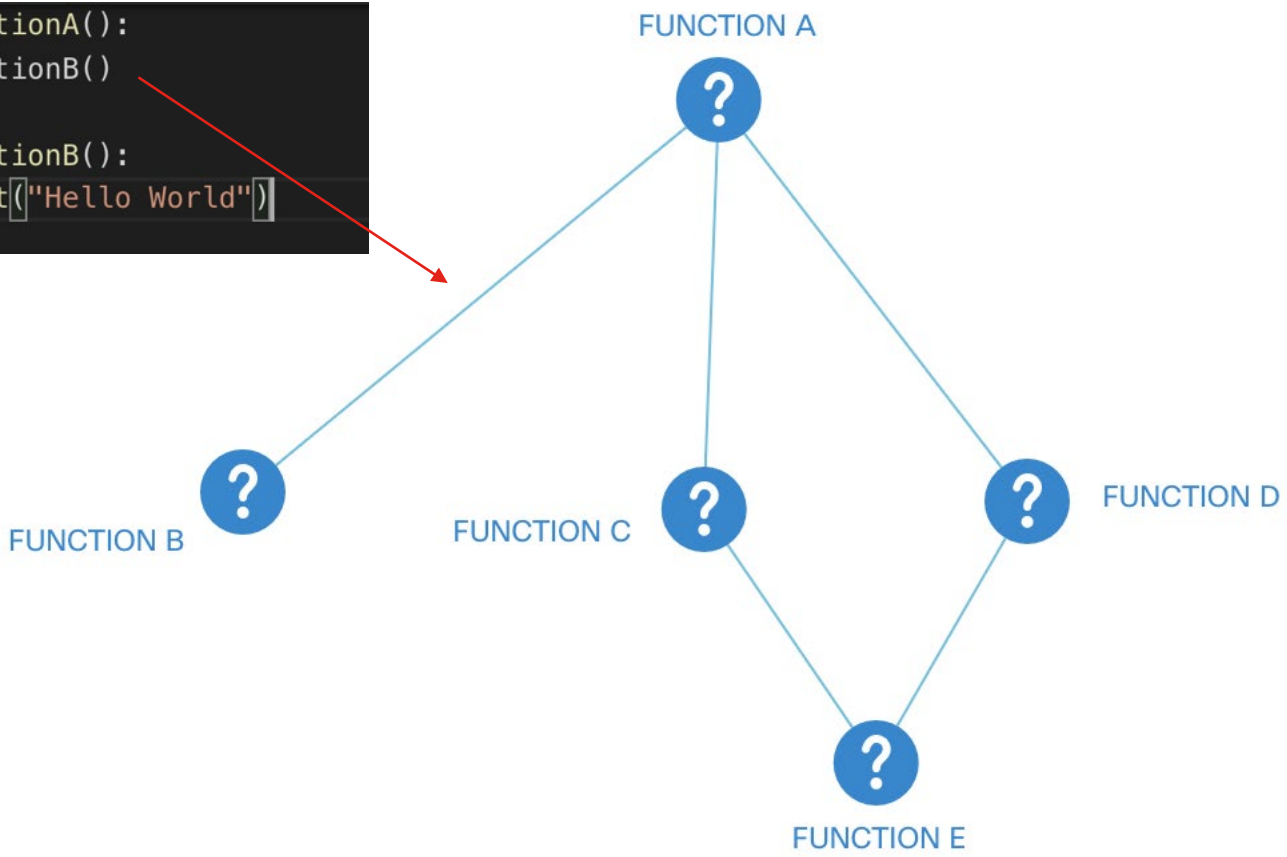
What is the flow of this program?

Troubleshooting

I have the line number for this error but how did I get here? I need some context



```
1   def functionA():
2       functionB()
3
4   def functionB():
5       print("Hello World")
```



NeXt UI Framework

NeXt UI toolkit is an HTML5/JavaScript based toolkit for network web application. It provides a network centric topology UI component featuring high performance and rich functionality. NeXt can display large complex network topologies, aggregated network nodes, traffic/path/tunnel/group visualizations and it includes different layout algorithms, map overlays, and preset user friendly interactions. NeXt can work together with DLUX to build ODL apps.

Homepage : https://wiki.opendaylight.org/view/NeXt:Main

UI Toolkit Quicklook : https://www.youtube.com/watch?v=gBsUDu8aucs

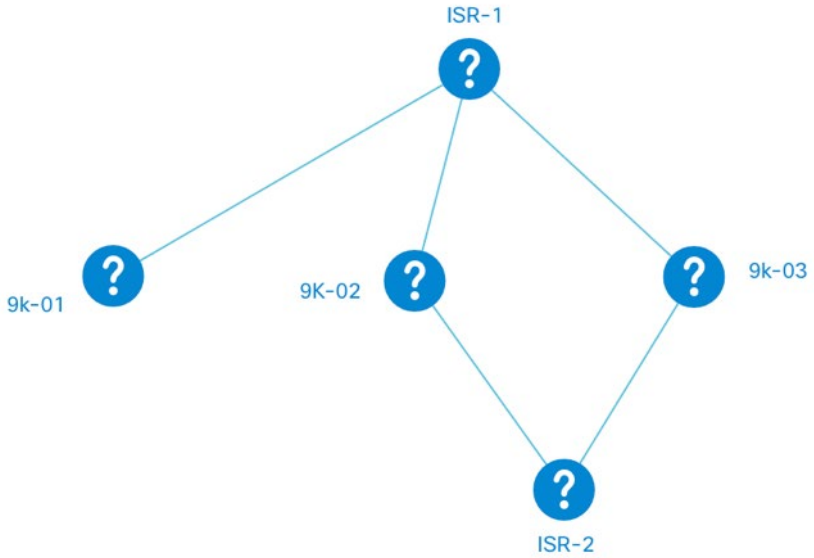
Current version : 0.9

Key Features

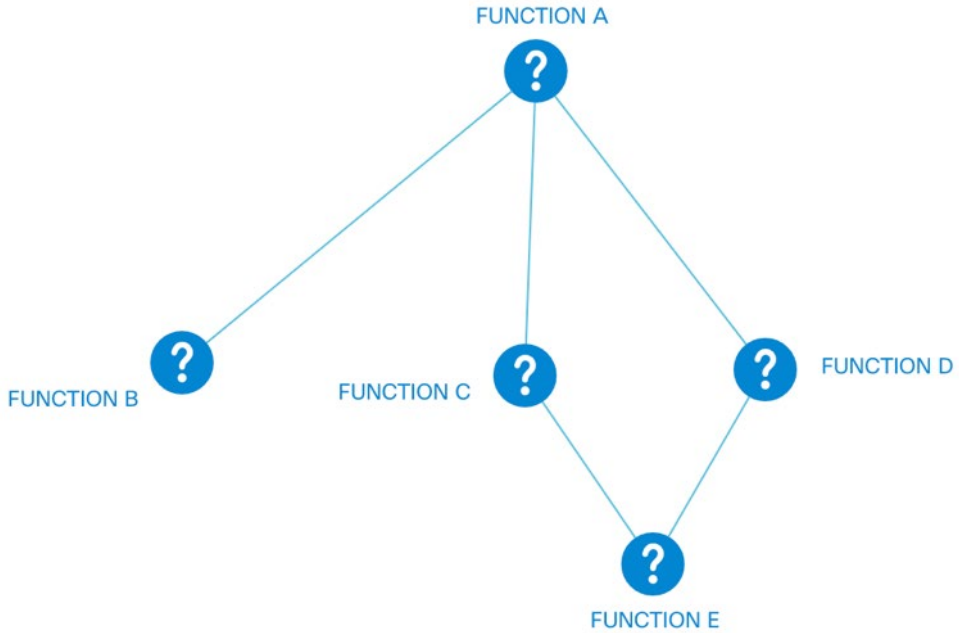
- Large complex network topologies
- Aggregated network nodes
- Traffic/path/tunnel/group visualizations
- Different layout algorithms
- Map overlays
- Preset user-friendly interactions

https://github.com/opendaylight/next

```
var topologyData = {
    nodes: [{
        "id": 0,
        "x": 410,
        "y": 100,
        "name": "ISR-1",
    }, {
        "id": 1,
        "x": 410,
        "y": 280,
        "name": "ISR-2"
    }, {
        "id": 2,
        "x": 660,
        "y": 280,
        "name": "9K-02"
    }, {
        "id": 3,
        "x": 660,
        "y": 100,
        "name": "9k-01"
    }, {
        "id": 4,
        "x": 180,
        "y": 190,
        "name": "9k-03"
    }],
    links: [{
        "source": 1,
        "target": 2
    }, {
        "source": 4,
        "target": 1
    }, {
        "source": 2,
        "target": 0
    }, {
        "source": 3,
        "target": 0
    }, {
        "source": 0,
        "target": 4
    }]
};
```



```
var topologyData = {
    nodes: [{
        "id": 0,
        "x": 410,
        "y": 100,
        "name": "FUNCTION A",
    }, {
        "id": 1,
        "x": 410,
        "y": 280,
        "name": "FUNCTION E"
    }, {
        "id": 2,
        "x": 660,
        "y": 280,
        "name": "FUNCTION D"
    }, {
        "id": 3,
        "x": 660,
        "y": 100,
        "name": "FUNCTION B"
    }, {
        "id": 4,
        "x": 180,
        "y": 190,
        "name": "FUNCTION C"
    }],
    links: [{
        "source": 1,
        "target": 2
    }, {
        "source": 4,
        "target": 1
    }, {
        "source": 2,
        "target": 0
    }, {
        "source": 3,
        "target": 0
    }, {
        "source": 0,
        "target": 4
    }]
};
```



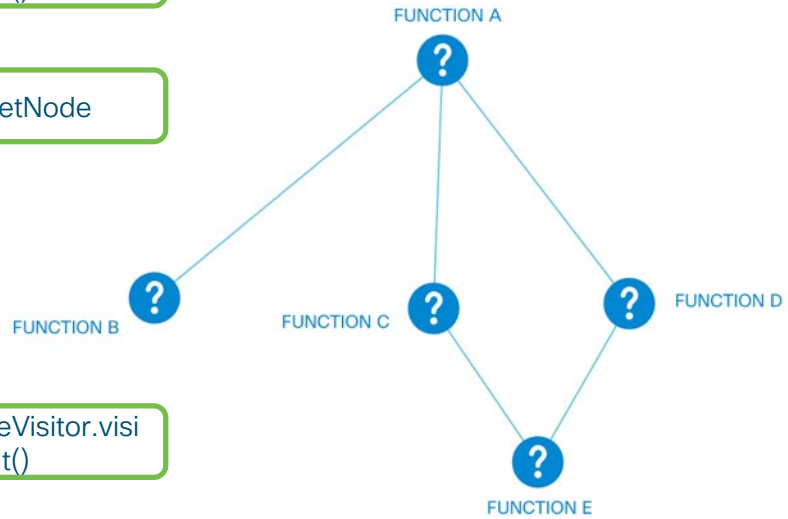
```
1   def functionA():
2       functionB()
3       functionC()
4       functionD()
5
6   def functionB():
7       print("Hello World")
8
9   def functionC():
10      functionE()
11
12  def functionD():
13      functionE()
14
15  def functionE():
16      pass
```

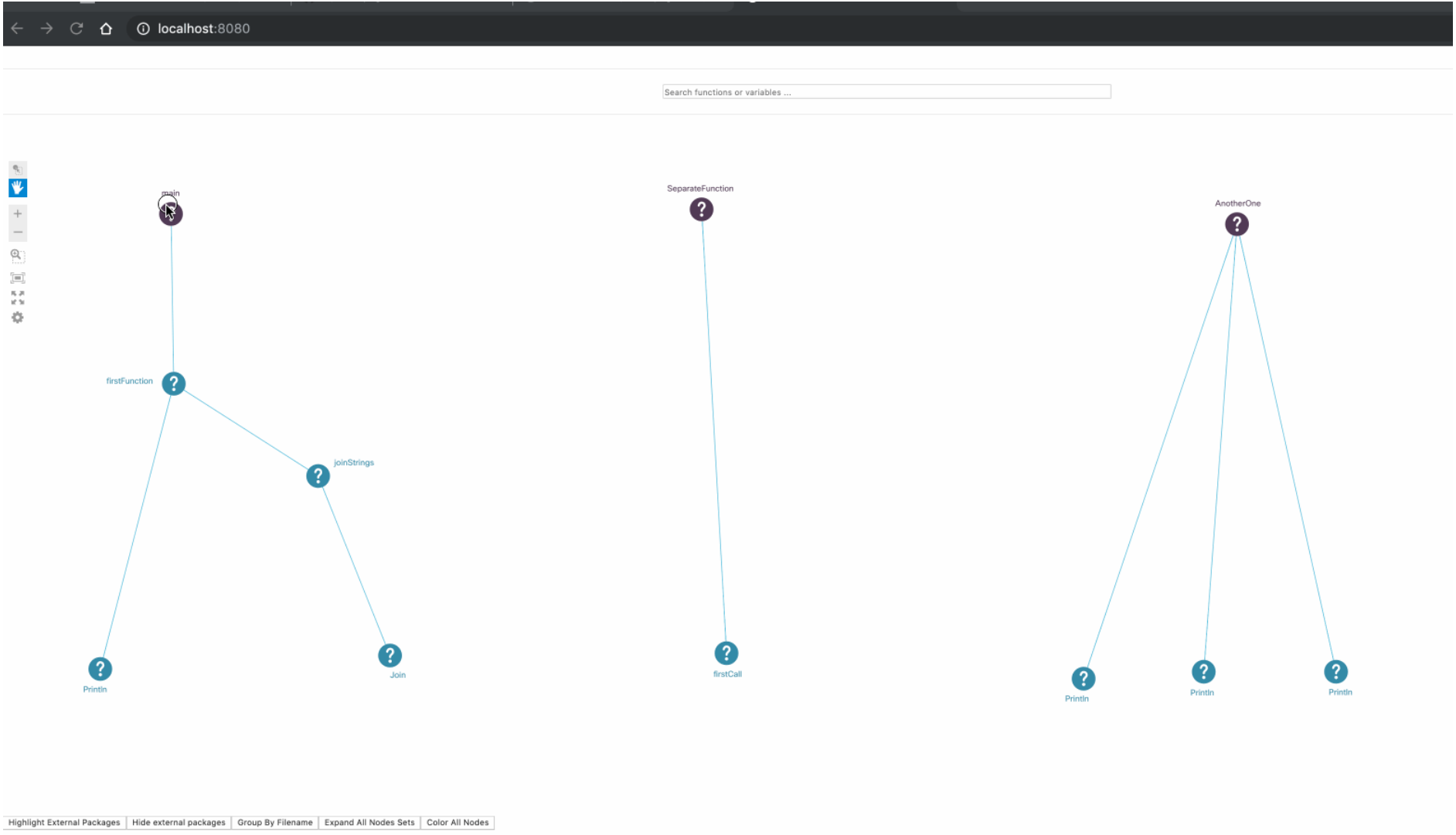
SourceNode

Ast.NodeVisitor.visit()

TargetNode

Ast.NodeVisitor.visit()




localhost:8080
Search functions or variables ...
main
firstFunction
joinStrings
Println
Join
SeparateFunction
firstCall
AnotherOne
Println
Println
Println
Highlight External Packages
Hide external packages
Group By Filename
Expand All Nodes Sets
Color All Nodes


unmarshalWithLeftovers
Unmarshal
Unmarshal
structToMap
Unmarshal
Marshal
example_patch
Int64
PatchCluster
NewClient
PatchUser
Int64
NewClient
example_delete
DeleteUser
example_get
Open
ReadAll
GetHealth
Unmarshal
NewClient
Equal
GetProviderClientConfig
GetRole
GetUsers
GetProviderClientConfigVsphereDatacenter
GetProviderClientConfigVsphereDatacenterClusters
Login
GetClusters
GetLivenessHealth
GetProviderClientConfigs
GetClusters
GetACIProfiles
GetLDAPSetup
example_add
Int64
Login
Bool
NewClient
Bool
AddClusterBasic
Int64
prettyPrint
main

References

- https://github.com/python/cpython
- https://devguide.python.org/compiler/
- https://docs.python.org/3/library/ast.html
- https://docs.python.org/3/reference/lexical_analysis.html
- https://www.asmeurer.com/brown-water-python/tokens.html
- https://data-flair.training/blogs/python-interpreter/
- https://usi-pl.github.io/cc-2019/notes/03.ast.html
- https://medium.com/@kamneemaran45/python-ast-5789a1b60300
- https://ruslanspivak.com/lsbasi-part7/
- https://www.mattlayman.com/blog/2018/decipher-python-ast/

References

- https://github.com/adamtornhill/code-maat
- https://mvdwoord.github.io/exploration/2017/08/18/ast_explore.html
- https://opensource.com/article/18/4/introduction-python-bytecode
- https://nedbatchelder.com/blog/201803/is_python_interpreted_or_compiled_yes.html
- https://www.digitalocean.com/community/tutorials/understanding-class-inheritance-in-python-3
- https://hackernoon.com/modifying-the-python-language-in-7-minutes-b94b0a99ce14
- https://tomassetti.me/guide-parsing-algorithms-terminology
- http://flint.cs.yale.edu/cs421/lectureNotes/c04.pdf
- https://repositories.lib.utexas.edu/bitstream/handle/2152/32311/SALLING-MASTERSREPORT-2015.pdf?sequence=1&isAllowed=y
- https://github.com/coetaur0/staticfg
- https://dev.to/btaskaya/lifecycle-of-a-python-code---cpythons-execution-model-85i

Learn more about the new DevNet Certifications and how you can prepare now!

	Associate Level	Specialist Level	Professional Level	Expert Level
Engineering				
Software				

Future Offering

Start Here | Upcoming Cisco DevNet Certifications

- Start at **Meet DevNet**

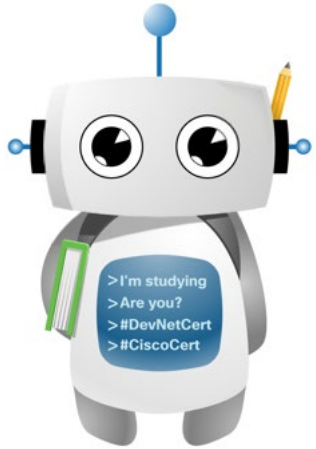
  DEVNET-2864: Getting ready for Cisco DevNet Certifications
  Offered daily at 9am, 1pm & 4pm at Meet DevNet

- Attend a **brownbag session**

  DEVNET-4099: DevNet Certifications: Bringing software practices & software skills to networking
  Offered daily 12:15-12:45 in the DevNet Zone Theater

- Visit the **Learning@Cisco** booth

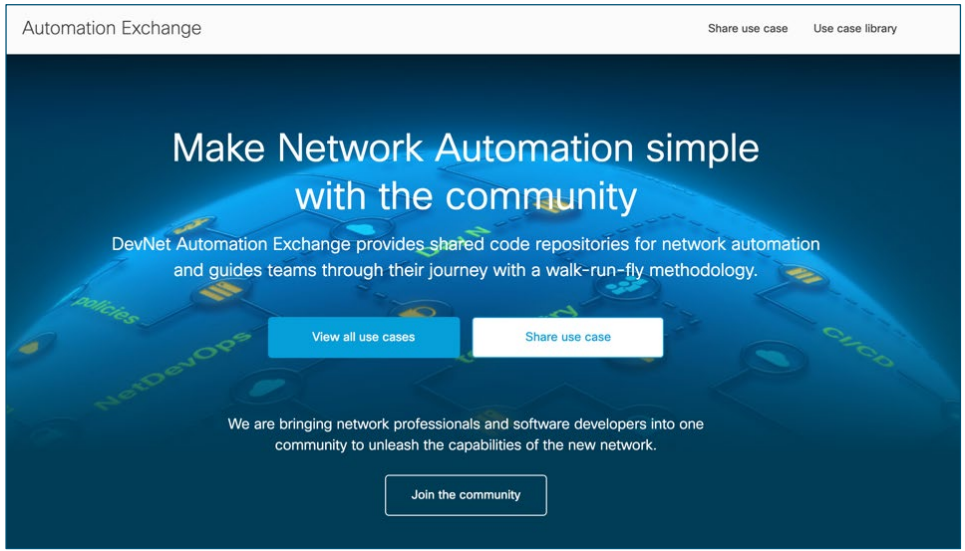
- Scan this code to **sign up** for the latest updates or go to http://cs.co/20eur02



cisco Live!

Find shared code repositories of use cases for network automation & more!

Don't miss our 5 **Automate Infrastructure demos** in the DevNet Zone!



Start at **Meet DevNet**

DEVNET-3010 [a-j] Learn how to make Network Automation Simple with the Community

Offered Monday 2pm & 5pm, Tuesday & Wednesday 10am, 2pm & 5pm, and Thursday 10am & 5pm at Meet DevNet

Scan this code or go to the URL to **learn more**

http://cs.co/20eur01

Complete your online session survey

- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (starting on Thursday) to receive your Cisco Live t-shirt.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Content Catalog on ciscolive.com/emea.

Cisco Live sessions will be available for viewing on demand after the event at ciscolive.com.

cisco Live!

Continue your education

Demos in the Cisco Showcase

Walk-In Labs

Meet the Engineer 1:1 meetings

Related sessions

Thank you

You make **possible**