Python Language Reference Coverage

This is a 1:1 mapping of the features listed in https://docs.python.org/3/reference/ and their support in TorchScript. The categorizations are as follows:

Section	Status	Note
1. Introduction	Not Relevant	
1.1. Alternate Implementations	Not Relevant	
1.2. Notation	Not Relevant	
2. Lexical analysis	Not Relevant	
2.1. Line structure	Not Relevant	
2.1.1. Logical lines	Not Relevant	
2.1.2. Physical lines	Supported	
2.1.3. Comments	Supported	
2.1.4. Encoding declarations	Not Supported	TorchScript explicitly don't support unicode
2.1.5. Explicit line joining	Supported	
2.1.6. Implicit line joining	Supported	
2.1.7. Blank lines	Supported	
2.1.8. Indentation	Supported	
2.1.9. Whitespace between tokens	Not Relevant	
2.2. Other tokens	Not Relevant	
2.3. Identifiers and keywords	Supported	
2.3.1. Keywords	Supported	
2.3.2. Reserved classes of identifiers	Supported	
2.4. Literals	Not Relevant	
2.4.1. String and Bytes literals	Supported	
2.4.2. String literal concatenation	Supported	
2.4.3. Formatted string literals	Partially Supported	
2.4.4. Numeric literals	Supported	
2.4.5. Integer literals	Supported	
2.4.6. Floating point literals	Supported	
2.4.7. Imaginary literals	Not Supported	
2.5. Operators	Partially Supported	Not supported: `<<`, `>>`, `:=`
2.6. Delimiters	Partially Supported	Not supported: `**=`, `<<=`, `>>=`, `%=`, `^=`, `@=`, `&=`, `//=`, `%` operator for some types (e.g. `str`)
3. Data model	Not Relevant	
3.1. Objects, values and types	Not Relevant	
3.2. The standard type hierarchy	Partially Supported	Not supported: NotImplemented, Ellipsis, numbers.Complex, bytes, byte arrays, sets, frozen sets, generators, coroutines, async generators, modules, I/O objects, internal objects, slice objects (though slicing is supported), classmethod
3.3. Special method names	Supported	
3.3.1. Basic customization	Partially Supported	Not supported: `__new__` , `__del__` , `__bytes__` , `__format__` , `__hash__` ,
3.3.2. Customizing attribute access	Not Supported	
3.3.2.1. Customizing module attribute access	Not Supported	
3.3.2.2. Implementing Descriptors	Not Supported	
3.3.2.3. Invoking Descriptors	Not Supported	
3.3.2.4. __slots__	Not Supported	
3.3.2.4.1. Notes on using __slots__	Not Supported	
3.3.3. Customizing class creation	Not Supported	
3.3.3.1. Metaclasses	Not Supported	
3.3.3.2. Resolving MRO entries	Not Supported	`super()` is not supported
3.3.3.3. Determining the appropriate metaclass	Not relevant	
3.3.3.4. Preparing the class namespace	Not relevant	

Section	Status	Note
3.3.3.5. Executing the class body	Not relevant	
3.3.3.6. Creating the class object	Not relevant	
3.3.3.7. Uses for metaclasses	Not relevant	
3.3.4. Customizing instance and subclass checks	Not Supported	
3.3.5. Emulating generic types	Not Supported	
3.3.6. Emulating callable objects	Supported	
3.3.7. Emulating container types	Partially Supported	Some magic methods not supported (e.g. `__iter__`)
3.3.8. Emulating numeric types	Partially Supported	Magic methods with swapped operands not supported (`__r*__`)
3.3.9. With Statement Context Managers	Not Supported	
3.3.10. Special method lookup	Not relevant	
3.4. Coroutines	Not Supported	
3.4.1. Awaitable Objects	Not Supported	
3.4.2. Coroutine Objects	Not Supported	
3.4.3. Asynchronous Iterators	Not Supported	
3.4.4. Asynchronous Context Managers	Not Supported	
4. Execution model	Not Relevant	
4.1. Structure of a program	Not Relevant	
4.2. Naming and binding	Not Relevant	Names are bound at compile time in TorchScript
4.2.1. Binding of names	Not Relevant	See `global` and `nonlocal` statements section
4.2.2. Resolution of names	Not Relevant	See `global` and `nonlocal` statements section
4.2.3. Builtins and restricted execution	Not Relevant	
4.2.4. Interaction with dynamic features	Not Supported	Python values cannot be captured
4.3. Exceptions	Partially Supported	See `try` and `raise` statement section
5. The import system	Not Relevant	
6. Expressions	Not Relevant	See expressions section
6.1. Arithmetic conversions	Supported	
6.2. Atoms	Not Relevant	
6.2.1. Identifiers (Names)	Supported	
6.2.2. Literals	Partially Supported	`bytesliteral`, `imagnumber` not supported
6.2.3. Parenthesized forms	Supported	
6.2.4. Displays for lists, sets and dictionaries	Partially Supported	Not supported: comprehension ifs, async iterators
6.2.5. List displays	Supported	
6.2.6. Set displays	Not Supported	
6.2.7. Dictionary displays	Supported	dict() constructor with kwargs doesn't work, dict comprehensions, dictionary unpacking
6.2.8. Generator expressions	Not Supported	
6.2.9. Yield expressions	Not Supported	
6.2.9.1. Generator-iterator methods	Not Supported	
6.2.9.2. Examples	Not Supported	
6.2.9.3. Asynchronous generator functions	Not Supported	
6.2.9.4. Asynchronous generator-iterator methods	Not Supported	
6.3. Primaries	Supported	
6.3.1. Attribute references	Supported	
6.3.2. Subscriptions	Supported	
6.3.3. Slicings	Partially Supported	Tuple slicing with stride is not supported
6.3.4. Calls	Partially Supported	Args unpack / kwargs unpack is not supported
6.4. Await expression	Not Supported	
6.5. The power operator	Supported	

Section	Status	Note
6.6. Unary arithmetic and bitwise operations	Partially Supported	Some bitwise operators are not implemented for primitive types (e.g. `~x` where `x` is an `int` is not currently supported)
6.7. Binary arithmetic operations	Partially Supported	See delimiters section
6.8. Shifting operations	Not Supported	
6.9. Binary bitwise operations	Supported	
6.10. Comparisons	Supported	
6.10.1. Value comparisons	Partially Supported	Dictionary equality checks are not currently supported
6.10.2. Membership test operations	Partially Supported	Not supported for TorchScript classes
6.10.3. Identity comparisons	Supported	
6.11. Boolean operations	Supported	
6.12. Conditional expressions	Supported	
6.13. Lambdas	Not Supported	
6.14. Expression lists	Partially Supported	Iterable unpacking not supported
6.15. Evaluation order	Supported	
6.16. Operator precedence	Supported	
7. Simple statements	Supported	
7.1. Expression statements	Supported	
7.2. Assignment statements	Supported	
7.2.1. Augmented assignment statements	Partially Supported	See delimiters section
7.2.2. Annotated assignment statements	Supported	
7.3. The assert statement	Partially Supported	Exception message is not customizable
7.4. The pass statement	Supported	
7.5. The del statement	Not Supported	
7.6. The return statement	Supported	Some other features of returning (e.g. behavior with try..finally) are unsupported
7.7. The yield statement	Not Supported	
7.8. The raise statement	Partially Supported	Exception message is not customizable
7.9. The break statement	Supported	Some other features of returning (e.g. behavior with try..finally) are unsupported
7.10. The continue statement	Supported	Some other features of returning (e.g. behavior with try..finally) are unsupported
7.11. The import statement	Not Supported	
7.11.1. Future statements	Not Supported	
7.12. The global statement	Not Supported	
7.13. The nonlocal statement	Not Supported	
8. Compound statements	Irrelevant	
8.1. The if statement	Supported	
8.2. The while statement	Partially Supported	while..else is not supported
8.3. The for statement	Partially Supported	for..else is not supported
8.4. The try statement	Not Supported	
8.5. The with statement	Partially Supported	`__exit__` is always called with `exc_type`, `exc_value`, and `traceback` set to None, even if an exception was raised, and `__exit__`'s return value is ignored.
8.6. Function definitions	Not Supported	
8.7. Class definitions	Not Supported	
8.8. Coroutines	Not Supported	
8.8.1. Coroutine function definition	Not Supported	
8.8.2. The async for statement	Not Supported	
8.8.3. The async with statement	Not Supported	
9. Top-level components	Not Relevant	
9.1. Complete Python programs	Not Relevant	
9.2. File input	Not Relevant	
9.3. Interactive input	Not Relevant	
9.4. Expression input	Not Relevant	