Glossary

Accumulator

An accumulator is a variable in a for-loop that stores information computed by the for-loop and will be still available when the for-loop is complete. For example, in the code

```
total = 0
for x in range (5):

     total = total + x
```

Application

An application is another name for a script.

Argument

An argument is an expression that occurs within the parentheses of a method call. The following call has two arguments: `x+y and y+w:`

```
min(x+y, y+w)
```

As a general rule, the number of arguments should equal the number of parameters that are called, except when using default arguments.

Assert statement

An assert statement is a statement in Python used to enforce an assertion. It has one of the following forms:

```
assert <bool>
assert <bool>, <message>
```

In both cases, Python evaluates the boolean expression. If the expression is True, nothing happens. If it is False, Python will raise an exception with the (optional) provided message.

Assignment statement

An assignment statement is a statement with the following form:

```
<variable> = <expression>
```

If the variable does not yet exist, the statement creates the variable and stores the given value inside it. If the variable does exist, then it replaces the old value with the one provided.

Attribute

An attribute is a variable belonging to either an object or a class. To reference a field, it must be preceded either by a variable containing the object name (for an object attribute) or the class name (for a class attribute). For example, to reference the attribute x inside of a Point object stored in the variable p, we use p.x.

Basic type

A basic type is any type that has a corresponding literal. Basic types are fully integrated into Python (they do not require an import statement to use) and are not mutable. In Python, the basic types are int, float, bool, complex, and str.

Bool

A bool or boolean is a basic type whose values are True and False.

Bug

A bug is an error in a program.

Call frame

A call frame is an area of memory created whenever a function is called. The call frame contains the following:

- The name of the function
- A program counter, which indicates the statement to execute next
- The function parameters
- Any local variables that have been created so far

The call frame changes over time as the function executes, adding, changing, or deleting variables.

Call stack

A call stack is the area of memory containing all active call frames. The call frames are arranged top to bottom with the most recent call on the bottom of the stack. A call frame may not be erased until all call frames below it are erased.

Cast

A cast is an operation that converts a value from one type to another. For example, the cast float(5+6) converts the value of the expression, which is 11, to float format.

A widening cast converts from less information to more information and can be done implicitly. A narrowing cast must be done explicitly because it may lose information or may be illegal. See narrower type for more information.

Class

In Python 3, class and type are synonyms. However, in this course, we will use the term class type to refer to a user-defined type that is not built into Python. The values of a class are called objects.

Command shell

The command shell is an OS-specific application that responds to text commands. On Windows, it is called the PowerShell. On MacOS (and most Linux systems), it is called the Terminal.

Comment

A comment is a line of code that is ignored by Python and only serves as a note to the programmer. Most comments start with a #. We sometimes refer to docstrings as comments. However, these are not actually ignored because they are used by the `help()` function.

Conditional expression

A conditional expression is an expression with the following form:

```
<exprl> if <bool> else <expr2>
```

To evaluate this expression, Python first evaluates the boolean expression `<bool>`. If the boolean is True, Python uses the value of `<expr1>` as the value of the conditional expression. Otherwise, Python uses the value of `<expr2>` as the value of the conditional expression.

Conditional statement

A conditional statement is a statement of the form

```
if <bool>:
        <statements>
```

Python executes this statement as follows: If the boolean expression is True, if executes the statements underneath. Otherwise, it skips over them.

An alternate form of a conditional is as follows:

```
if <bool>:
        <statements>

else:
        <statements>
```

This form is executed as follows: If the boolean expression is True, it executes the statements underneath the if. Otherwise it executes the statements underneath the else. There are additional forms of conditional statements that use the keyword elif.

Constructor call

A constructor call is a call to a constructor function. It has the form

```
<classname>(<arguments>)
```

For example, Point (1,2,3) is a constructor call for the class Point.

A constructor call is evaluated in three steps:

1. Create an instance of class classname
2. Execute the initializer `__init__(<arguments>)`
3. Return the folder name of the new instance

Debugging

Debugging is the act of searching for and removing bugs in a program.

Default argument

A default argument is an argument that is provided automatically if it is omitted in the function call. Default arguments are specified by writing the associated parameter in the form of an assignment statement. For example, the following function header has a default argument for y:

```
def foo(x,y=2):
```

In this case, the function call `foo(1)` is legal; the y parameter is given the default argument 2. In this course, we are primarily interested in object and class folder.

Dict

A dict or dictionary is a mutable type that associates keys with values. It is similar to a sequence, except that elements are accessed by the key, not the position. For example, in the dictionary

```
d = { 'alpha':1, 'beta':3 }
```

the value `d['alpha']` is 1, while the value `d['beta']` is 3.

While dictionaries are not sequences, they are iterable.

Docstring

A docstring is a string literal that begins and ends with three quotation marks. Document strings are used to write specifications and are displayed by the `help()` command.

Escape character

The escape character \ is used to write characters that cannot appear directly in a string. Here are a few (not exhaustive) examples:

New-Line Character	`\n`
Backslash Character	`\\`
Double-Quote Character	`\"`
Single-Quote Character	`\'`

Evaluate

Python evaluates an expression by turning it into a value.

Float

A float is a basic type whose values are scientific numbers like 3.46E-4.

For-loop

A for-loop is a statement of the form

```
for <variable> in <iterable>:
    <statements>
```

Python executes this statement as follows: It puts the first element of the iterable in the variable, and then executes the statements indented underneath. It repeats this process as long as there are still elements left in the iterable.

In a for-loop, the variable after the keyword for is called the loop-variable while the iterable is called the loop-iterable or loop-sequence.

Fruitful function

A function that terminates by executing a return statement, giving an expression whose value is to be returned. Fruitful functions (possibly) return a value other than None.

Function

A function is a set of instructions to be carried out via a function call. You can think of it like a recipe in a cookbook. We often separate functions into fruitful functions and procedures.

Function body

The function body consists of the lines of code specifying what is to be done when the function is called. The function body appears indented after the function header. We typically use the term function body to just refer to the actual Python code and exclude the function specification.

Function call

A function call is an expression that executes a function body. It has the following form:

```
<function-name>(<arguments>)
```

It is important to understand how a function call is executed. Its execution is performed in five steps:

- A call frame is added to the stack with the function name.
- The arguments are evaluated and assigned to the parameters.
- The function body is executed, one line at a time.
- The call frame is erased from the stack.
- The function value is returned if the function is fruitful.

If the function is a fruitful, it returns the value of the appropriate return statement. If it is a procedure, it returns None.

Function definition

The function definition is the code that tells Python what to do when a function is called. It consists of the function header and function body.

Function frame

A function frame is another term for a call frame.

Function header

The function header is the part of the definition that starts with the keyword def, followed by the function name and the parameters delimited by parentheses. The function body is immediately indented underneath.

Function name

The function name is the name used to call a function. It appears in the function header, immediately after the keyword def.

Function specification

The function specification defines what the function does. It is used to understand how to call the function. It must be clear, precise, and thorough, and it should mention all parameters, saying what they are used for. It should also include a precondition for each parameter.

In Python, function specifications are written with a docstring. They appear immediately after the function header, indented with the function body.

Function stub

A function stub is a function definition with a header and specification, but no body.

Global space

Global space is the region of memory that stores global variables.

Global variable

A global variable is a variable that is defined outside of a function or class definition. Typically, both function names and module names are global variables; these variables each contain the identifier of the folder that stores the function or module content in the heap.

Heap

The heap is the region of memory that stores all folders. Anything that is not a basic type must be represented as a folder in the heap.

Immutable

The adjective immutable means "incapable of being changed." It is typically used to refer to an attribute or a type.

Immutable attribute

An immutable attribute is a hidden attribute that has a getter but no setter. This implies that a user is not allowed to alter the value of this attribute. It is an important part of encapsulation.

Immutable type

An immutable type is any type that is not mutable. All of the basic types are immutable.

implementation

An implementation is a collection of Python code for a function, module, or class that satisfies a specification. This code may be changed at any time as long as it continues to satisfy the specification.

In the case of a function, the implementation is limited to the function body. In the case of a class, the implementation includes the bodies of all methods as well as any hidden attributes or methods. The implementation for a module is similar to that of a class.

Import statement

An import statement is a statement with one of the following forms:

```
import <module>            # encapsulate contents in module folder
from <module> import item  # pull module element into global space
```

It is used to access the global variables, functions, and classes defined within a module. When we import a module, all module variables (and function and class names) are stored in a folder. Unless the from syntax is used, all content must be accessed using a variable with the same name as the module.

For example, if we use the statement

```
import math
```

then we write math.pi to access the variable pi in this module. On the other hand, if we write

```
from math import *
```

then all of the contents of the module math are dumped into global space. In that case, we only need to write pi to access the same variable.

Instance

An instance of a class is an object. The terms instance and object are synonyms. We typically use the word instance when we are referring to a collection of objects all of the same class.

Instantiate

The word instantiate means "to create an instance of." Evaluation of a class expression C(...) instantiates an instance of the class C.

Int

An int is a basic type whose values are integers.

Interactive shell

The interactive shell is a program that allows the user to type Python expressions and statements one at a time, evaluating them or executing them after each step. It is not to be confused with the command shell, which is a more general purpose tool. However, the interactive shell is often run within the command shell.

Interface

The interface is the information that another user needs to know to use a Python feature, such as a function, module, or class. The simplest definition for this is any information displayed by the `help()` function.

For a function, the interface is typically the specification and the function header. For a class, the interface is typically class specification as well as the list of all unhidden methods and their specifications. The interface for a module is similar to that of a class.

Is

The is operator works like == except that it compares folder names, not contents. The meaning of this operator is can never be overloaded.

Iterable

An iterable type is the type of any value that may be used in a for-loop. Examples include lists, strings, and dictionaries.

Keyword

A keyword is a special reserved word telling Python to do something. Examples include if, for, and def. Keywords may not be used as variable names.

List

A list is a mutable type representing a sequence of values. Lists are represented as a comma-separated sequence of expressions inside square brackets, like this

```
[1, 'Hello', True, 3.5]
```

Literal

A literal is a way to represent a value in Python. Literals are used to write expressions with any of the basic types. Here are several examples of literals:

int	`354`
float	`3.56`
complex	`3.5j`
bool	`True`
str	`"Hello World!"`
str	`'Hello World!'`

Local variable

A local variable is a variable that is created within the body of a function or method.

Loop condition

The loop condition is a boolean expression in the header of a while-loop.

Loop iterable

A loop iterable is the entity that produces values for a for-loop. Because most loop iterables are sequences, this term is often used interchangeably with a loop sequence.

Loop sequence

A loop sequence is the entity that produces values for a for-loop. The name is chosen because most of the time for-loops use a sequence. However, the term loop iterable is more accurate.

Loop variable

A loop variable is a variable that acquires the next element of the loop iterable through each iteration of a for-loop. In some cases, the term loop variable is extended to include any variable that appears in the loop condition of a while-loop.

Method

A method is a function that is contained within a class definition. Methods are able to access the attributes of the class, in addition to the parameters and local variables that any function can access.

Method call

A method call is a function call for a method. It is similar to a function call except that it has the following form:

```
<folder>.<method-name>(<arguments>)
```

where `<folder>` is an object in the case of an instance method and a class in the case of a class method.

A method call creates a call frame exactly like a function call except that the folder name of `<folder>` is assigned to self.

Module

A module is a file containing global variables, function definitions, class definitions, and other Python code. The file containing the module must be the same name as the module and must end in .py. A module is used by either importing it or running it as a script.

Module specification

A module specification is a description of the purpose of a module and how to use it. Module specifications are typically written using docstrings.

Mutable

The adjective mutable means "capable of being changed." It is typically used to refer to an attribute or a type.

Mutable attribute

A mutable attribute is an attribute that can be freely changed. Mutable attributes need not be encapsulated. But if the attribute is hidden, then it will need both a getter and a setter.

Mutable type

A mutable type is a type where the values are containers and it is possible to alter the contents of the container. An int is not mutable because it does not contain anything; it is just a number. A string is a container of characters but it cannot be changed. The types list and dictionary are examples of mutable types, as are most user-defined classes.

Narrower type

A narrower type is a type for which Python will perform a cast from automatically. In other words, Python will automatically cast from a narrower type to a wider type. This concept only applies to bool and the numeric types. The progression below shows the progression from narrowest type to the widest type:

```
bool -> int -> float -> complex
```

None

The value None actually represents the absence of a value. If Python attempts an attribute reference like None.b or a method call like None.m(...), it will raise an exception.

Object

An object is an instance of a class. Each object for a class can access the attributes and methods defined in and inherited by the class.

Object attribute

An object attribute is an attribute that is unique to a specific object and is therefore stored in the object folder.

Object class

The object class is a special class built into Python named object. It is the superest class of all. Any class that does not extend another class must extend the object class.

Object folder

The object folder is the folder in the heap that contains the attribute values unique to an object. We use this term in place of the term object when we want to exclude any attributes or methods that might be inherited.

Object identifier

The object identifier is the folder name for an object-folder. This identifier can be accessed with the `id()` function, but it is rarely useful.

Object representation

An object representation is a string that can uniquely identify an object. It is retrieved with the repr function.

Parameter

A parameter is a variable that is declared within the header of a function or method.

Pass statement

A pass statement is a statement that tells Python to do nothing. It is used to fill the body of a function stub.

Procedure

A procedure is a function that has no explicit return statement. A function call on a procedure always evaluates to None.

Return statement

A return statement is a statement that terminates the execution of the function body in which it occurs. If it is followed by an expression, then the function call returns the value of that expression. Otherwise, the function call returns None.

Scope

The scope of a variable is the set of statements in which it can be referenced. For a parameter, it is the corresponding function body. For a local variable, it is from the initial assignment statement until the variable is deleted, or the end of the function body.

Script

A script is a Python program that is meant to be run outside of interactive mode. In particular, scripts often have the following line of code to prevent them from being imported:

```
if __name__ == "__main__":
```

To run a script, type python <script> in the OS command shell. When a script is run, it will execute all of the code indented under the conditional above.

Sequence

A sequence is a value that is an ordered list of other values. The contents of a sequence can be accessed by bracket notation. For a sequence seq, seq[2] is the element in position 2. seq[1:5] is the slice (subsequence) from positions 1 to 5 (including 1, but not including 5).

Examples of sequences include lists, strings, and tuples.

Statement

A statement is a command of Python to do something. Python executes a statement.

Str

A str or string is a basic type whose values are text, or sequences of characters. String literals must be either in double or single quotes.

Test-case

A test-case is a collection of inputs, together with an expected output, used to test a single run of a program. It is often used in unit tests to test a function.

Testing

Testing is the act of analyzing a program, looking for bugs. Unlike debugging, it does not necessarily involve removing bugs.

Try-except statement

A try-except statement is a statement of the following form

```
try:
      <statements>

except:
      <statements>
```

Python attempts to execute all of the statements underneath try. If there is no error, then Python does nothing and skips over all the statements underneath except. However, if Python crashes while inside the try portion, it recovers and jumps over to the except. It then executes all the statements underneath there.

Tuple

A tuple is an immutable type representing a sequence of values. Tuples are represented as a comma-separated sequence of expressions inside parentheses, like this

```
(1, 'Hello', True, 3.5)
```

Type

A type is a set of values together with the operations on them. The type of a value can be determined via the `type()` function.

Typing

Typing is a special form of precondition that requires that a variable hold a value of a specific type. It is the most common form of function precondition in this course.

Unit test

A unit test is a collection of test cases used to test a unit of code, typically a function or method.

Variable

A variable is named with an associated value. We sometimes refer to it as a named box with the value in the box. In Python, there are four basic kinds of variable, determined by the memory region they fit in:

- Parameters
- Local variables
- Attributes

While-loop

A while-loop is a statement with the following form:

```
while <bool>:
      <statements>
```

Python executes this statement as follows: It first evaluates the boolean expression. If it is True, then it executes the statements indented underneath. It repeats this process as long as the boolean expression is True. This boolean expression is often referred to as the loop condition.

While-loops are difficult to use properly and are often combined with loop invariants.

Wider type

A wider type is a type for which Python will perform a cast from automatically. In other words, Python will automatically cast from a narrower type to a wider type. This concept only applies to bool and the numeric types. The progression below shows the progression from narrowest type to the widest type:

```
bool -> int -> float -> complex
```