PYTHON

programming

* Stephen Shroud
* Christin McKiver
* Nilanjan Mhatre
* Dr. Dewan Ahmed

The paradigm of the language.

* Python is a high-level programming language
* It supports
  + **Imperative:** In Python, programs can be written as a set of commands one after the other, where each command changes the state of execution
  + **Procedural:** Python supports defining procedures or functions or subroutines, that contain a series of steps to be carried out. Any procedure can be called at any point during the program’s execution. Although, these subroutines won’t be executed if they are not called.
  + **Functional:** Functional programming is supported by Python, which is similar to procedural, but does include the feature of changing the state of execution or having a series of statements, but only rely on evaluation of expressions
  + **Object-Oriented:** Python allows creation of classes and encapsulation of methods and data inside a class. Other OOPS features include Inheritance, Polymorphism and abstraction

History of “PYTHON”

* Python was created by Guido Van Rossum in 1989 at CWI in Netherlands, when he published the code as version 0.9.0
* It was built to be an object-oriented programming language from the beginning
* But, it also supported procedural programming, i.e. unlike Java, it is not purely an object-oriented language
* Later, with contributions and patches from others like a Lisp hacker, Python reached its version 1.0 that included new features: lambda, map, filter, reduce
* There were many releases in the due course including the ‘Macintosh only’ 0.9.5 version
* Python 1.4 also acquired new features like the Modula-3 keyword arguments and support for complex numbers
* Python gained popularity because of its clean coding capability and ease of learning the language with its non-complex syntax
* In 2000, the Python core development team moved to BeOpen.com to form Python labs that resulted in the release of Python 2.0
* It had features of list comprehension, cycle-detecting garbage collector and Unicode
* Python 2.2 had a major change that combined the types written in C and classes in Python into one hierarchy
* Python 3.0 rectified certain issues in the language and made a few syntax changes
* Some of them are changing the print statement to print function, moving only ‘reduce’ out of the namespace into functools, using only Unicode for text strings, adding support for optional function annotations and also changing integer divisions (e.g. In 2.x, 5/2=2, while in 3.x 5/2=2.5 and 5//2=2)
* This broke the backwards compatibility, which is why tools must be used to convert programs written in Python2.x into Python 3.x
* All changes are now being operated on current 3.x series
* Python 3.6 had changes in the new UTF-mode

3. The elements of the language: reserved words, primitive data types, structured types

|  |  |  |  |
| --- | --- | --- | --- |
| **Summary of Python 3's built-in types** | | | |
| **Type** | **mutable** | **Description** | **Syntax example** |
| bool | immutable | [Boolean value](https://en.wikipedia.org/wiki/Boolean_value) | True False |
| bytearray | mutable | Sequence of [bytes](https://en.wikipedia.org/wiki/Byte) | bytearray(b'Some ASCII') bytearray(b"Some ASCII") bytearray([119, 105, 107, 105]) |
| bytes | immutable | Sequence of bytes | b'Some ASCII' b"Some ASCII" bytes([119, 105, 107, 105]) |
| complex | immutable | [Complex number](https://en.wikipedia.org/wiki/Complex_number) with real and imaginary parts | 3+2.7j |
| dict | mutable | [Associative array](https://en.wikipedia.org/wiki/Associative_array) (or dictionary) of key and value pairs; can contain mixed types (keys and values), keys must be a hashable type | {'key1': 1.0, 3: False} |
| ellipsis |  | An [ellipsis](https://en.wikipedia.org/wiki/Ellipsis_(programming_operator)) placeholder to be used as an index in [NumPy](https://en.wikipedia.org/wiki/NumPy) arrays | ... |
| float | immutable | [Floating point](https://en.wikipedia.org/wiki/Floating_point) number, system-defined precision | 3.1415927 |
| frozenset | immutable | Unordered [set](https://en.wikipedia.org/wiki/Set_(computer_science)), contains no duplicates; can contain mixed types, if hashable | frozenset([4.0, 'string', True]) |
| int | immutable | [Integer](https://en.wikipedia.org/wiki/Integer_(computer_science)) of unlimited magnitude[[75]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-pep0237-75) | 42 |
| list | mutable | [List](https://en.wikipedia.org/wiki/List_(computer_science)), can contain mixed types | [4.0, 'string', True] |
| set | mutable | Unordered [set](https://en.wikipedia.org/wiki/Set_(computer_science)), contains no duplicates; can contain mixed types, if hashable | {4.0, 'string', True} |
| str | [immutable](https://en.wikipedia.org/wiki/Immutable_object) | A [character string](https://en.wikipedia.org/wiki/Character_string): sequence of Unicode codepoints | 'Wikipedia' "Wikipedia" """Spanning multiple lines""" |
| tuple | immutable | Can contain mixed types | (4.0, 'string', True)But we can append elements using \_\_add\_\_ .  a = (4.0, 'string' , True).\_\_add\_\_(('hi' ,))  now a gives (4.0, 'string', True ,'hi') |

4. A description (in some form) of the syntax of the language.

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use [curly brackets](https://en.wikipedia.org/wiki/Curly_bracket_programming_language) to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than [C](https://en.wikipedia.org/wiki/C_(programming_language)) or [Pascal](https://en.wikipedia.org/wiki/Pascal_(programming_language)).

* Indentation- whitespace to delimit blocks
* Statements and control flow
  + The assignment statement (ex. = )
  + The if statement (executes a block code along with else and elif)
  + The for statement (iterates over object, capturing each element to a local variable for use by the attached block)
  + The try statement (allows exceptions to be caught and handled by except clauses)
  + The class statement (executes block code and attaches namespace to a class)
  + The def statement (defines function or method)
  + The with statement (encloses a code block within a context manager)
  + The pass statement (create an empty code block ; serves as NOP)
  + The assert statement (used during debugging to check for conditions that ought to apply)
  + The yield statement (returns a value from a generator function)
  + The import statement ( import modules whose functions or variables can be used in the current program) 4 ways to import
  + The print statement
* Expressions
  + +, - , \*, /
  + Matrix multiplication
  + = = (value comparison)
  + And, or, not (not symbols)
  + List comprehension
  + Lists vs tuples
  + String format operator %
  + String literals ( single or double quote marks, triple-quoted strings, raw string)
  + Array index and array slicing
* Methods
* Typing (duck typing and allows programmers to define their own types using classes) (see (3))
* Mathematics

5.The basic control abstractions of the language (loops, conditional controls, etc.)

Most Python implementations (including CPython) include a read–eval–print loop (REPL), permitting them to function as a command line interpreter for which the user enters statements sequentially and receives results immediately.

6.How the language handles abstraction (including functions, procedures, objects, modules, etc.)

7.An evaluation of the language's writability, readability, and reliability using the criteria discussed in chapter 1 and 2 of the textbook.

8. The major strengths and weaknesses of your language.

9.An overview of the programs that you included, and a discussion of what language features they highlight and how the language made the programs easy/hard to implement.