Python notes

Mostly human syntax

Object-oriented (same as Java)

Versions 2 and 3 exist. Python 3 corrects some minor flaws in Python 2, which is more widely used as of now.

\_id of variables found with id()

Keywords and identifying them

Variables can hold objects of different types. Variables themselves don’t have a type. Find using type().

xxx.isidentifier()

keyword.iskeyword()

Now I begin to realise that this truly is a language. In order to do anything in it, I need fluency. To get fluency, I need to understand what the words/functions mean. It is said that you need to know 90% of the words in a language to have conversational fluency. That’s quite a bit, so that’s why I started the glossary below.

Seems like the page updated to elaborate on various data types: Numbers, Lists, Sets, Tuples, Dict

I wonder if I should stick with Python all the way. Hmm, well I’ll take a look at the edX curricula then.

It’s very useful for learning control flow. Understanding functions, scope, calling functions, decomposition, abstraction, modules. And now, comparison of iteration vs recursion.

**Contents from TechBeamers Python Basic Tutorials**

* [Python Keyword](http://www.techbeamers.com/python-keywords-identifiers-variables/)s, Identifiers and Variables
* [Python Statement](http://www.techbeamers.com/understand-python-statement-indentation/), Expression and Indentation
* [Python Comment](http://www.techbeamers.com/understand-python-comment-docstring/)
* [Python Data Types](http://www.techbeamers.com/python-data-types-learn-basic-advanced/)
* Numbers
* Lists []
* Sets {}
* Tuples ()
* Dictionaries {} also
* [Python Operators](http://www.techbeamers.com/python-operators-tutorial-beginners/)
* [Operator Precedence](http://www.techbeamers.com/python-operator-precedence-associativity/)
* [Python Namespace](http://www.techbeamers.com/python-namespace-scope/)
* [Python String](http://www.techbeamers.com/python-strings-functions-and-examples/)
* Python Class
* Python Inheritance
* [Python File I/O](http://www.techbeamers.com/python-file-handling-tutorial-beginners/)  
  (Stopped here because of FYP and workspace issues, started again in summer)
* [Python Copy File](http://www.techbeamers.com/python-copy-file/)
* [Python Exception](http://www.techbeamers.com/python-try-except-beginners/)
* [Python Try Except](http://www.techbeamers.com/use-try-except-python/)
* [Python Pickle](http://www.techbeamers.com/python-tutorial-using-pickle-for-serializing-python-objects/)
* [Python Time](http://www.techbeamers.com/python-time-functions-usage-examples/)
* [Python Random](http://www.techbeamers.com/using-python-random/)

**Detailed Contents**

* [Python Keyword](http://www.techbeamers.com/python-keywords-identifiers-variables/)s, Identifiers and Variables
* [Python Statement](http://www.techbeamers.com/understand-python-statement-indentation/), Expression and Indentation
  + Statements in Python
    - What is a Statement?
    - What is an Expression?
    - Simple Assignment Statement
    - Augmented Assignment Statement
  + Multi-line Statement in Python
    - Explicit line continuation
    - Implicit line continuation
  + Python Indentation
    - Ideal Indentation Size
    - Why is Indentation Important?
* [Python Comment](http://www.techbeamers.com/understand-python-comment-docstring/)
  + How to use single line comment in Python?
  + How to use multiline comments in Python?
    - Using hash mark for commenting
  + How to use docstrings in Python?
    - Using triple-quotation mark for docstring
    - docstring vs comment in Python
* [Python Data Types](http://www.techbeamers.com/python-data-types-learn-basic-advanced/)
  + Booleans
  + Numbers
  + Strings
  + Sets
  + Tuples
  + Dictionaries
  + Bytes
  + Lists
  + Tuples
  + Sets
  + Dictionaries
* Numbers
  + Types of Numbers in Python
  + Key Points to Remember
  + Type Conversion in Python
  + External Number Classes in Python – Decimal and Fractions
  + Python Mathematics
* Lists
  + Create a List in Python
    - Using [ ] to Create a List
    - With List() Method to Create a List
    - Using List Comprehension  
      Like Excel INDEX-MATCH
    - Creating a Multi-Dimensional List
    - List Extension
  + Access a List in Python (Indexing)
    - Using the Index Operator
    - Reverse Indexing
  + Slicing Python Lists
  + Iterate a List in Python
  + Add/Update Elements in a List
  + Remove/Delete Elements from a List
  + Searching Elements in a List
  + Sorting a List in Python
    - List’s Sort() Method
    - Built-in Sorted() Function
  + Essential List Methods
  + Built-in List Functions
* Sets
  + Instantiate
  + Modify
  + Delete Elements
  + Native Operations
    - Union
    - Intersection
    - Difference
    - Symmetric Difference
  + Membership Test
  + Frozen Set
* Tuples
  + Instantiate
  + Access
  + Modify/Delete
  + Membership/Traverse
  + Usage
    - Grouping Data
    - Tuple Assignment
    - Using Tuples in Functions as Return Values
    - Mixed Data Structures in the Form of Tuples
* Dictionaries
  + Creating/Updating Dictionaries
  + Delete/Remove Elements from a Dictionary
  + Different Attributes – Keys
  + Iteration Through a Dictionary
  + Dictionary Comprehension
* [Python Operators](http://www.techbeamers.com/python-operators-tutorial-beginners/)
  + What are built-in operators in Python?
  + Arithmetic operators
  + Comparison operators
  + Logical operators
  + Bitwise operators
  + Assignment operators
  + Identity operators
  + Membership operators
* [Operator Precedence](http://www.techbeamers.com/python-operator-precedence-associativity/)
  + Operator precedence in Python
  + Operator associativity in Python
  + Nonassociative operators
* [Python Namespace](http://www.techbeamers.com/python-namespace-scope/)
  + What Are Names and Namespaces In Python?
  + What Is Scope In Python?
  + Scope Resolution In Python – Examples
  + How To Correctly Import Modules In Python?
    - Import Specific Names From A Module
    - Import Just The Module Using Its Name
* [Python String](http://www.techbeamers.com/python-strings-functions-and-examples/)
  + Creating Python Strings.
  + Accessing Characters in Python Strings.
  + Modifying/Deleting Python Strings.
  + Python String Operators.
  + String Formatting operators.
    - Escape Characters.
    - Python Format Characters.
  + Unicode String.
  + Built-in String Functions for the following,
    - Python String Conversion.
    - Python Compare String.
    - Python String Padding.
    - Python String Replace.
* Python Class
  + What Is A Python Class?
  + How To Create A Class In Python?
    - 1. The “Class” Keyword
    - 2. The Instance Attributes
    - 3. The Class Attributes
    - 4. What Is Self?
    - 5. What Is \_\_init\_\_? Or What Is A Constructor In Python?
  + Python Class – A Complete Example
* Python Inheritance
  + Python Inheritance – Introduction
  + How To Use Inheritance In Python?
  + Python Inheritance – Example
  + The Super() Method
* [Python File I/O](http://www.techbeamers.com/python-file-handling-tutorial-beginners/)
  + Open a file in Python.
    - Python open() file method.
    - File open modes in Python.
    - The Python file object attributes.
      * Example: Python file attribute in action.
    - Python file encoding.
  + Close a file in Python.
    - Python close() file method.
      * Example-1: Basic close operation in Python.
      * Example-2: Using exception with close operation in Python.
      * Example-3: Close a file using ‘with’ clause in Python.
  + Python Write operation.
    - The write() file method.
      * Example: Writing to a file in Python.
  + Python Read operation.
    - Example: Read from a File in Python.
  + File read positions in Python.
    - Example: Setting offsets in Python.
  + Renaming and deleting files in Python.
    - The rename() method.
    - The remove() method.
  + Python file object methods.
* [Python Copy File](http://www.techbeamers.com/python-copy-file/)
  1. shutil
     + shutil copyfile() method
     + shutil copy() method
     + shutil copyfileobj() method
     + shutil copy2() method
  2. os
     + os popen method
     + os system() method
  3. threading Thread() method
  4. subprocess
     + subprocess call() method
     + subprocess check\_output() method
* [Python Exception](http://www.techbeamers.com/python-try-except-beginners/)
  + The difference between an error and the exception.
    - What is Error?
    - What is Exception?
  + How to handle an Exception?
    - Checklist to handle an exception in Python.
    - Python exception handling # example-1.
    - Exception handling# example-2.
  + How to define except Clause with No Exceptions?
    - Python exception handling # example-3.
  + How to use except Clause with Multiple Exceptions?
    - Python exception handling # example-4.
  + How to use “try-finally” clause in Python?
    - Python exception handling # example-5.
    - Exception handling# example-6.
  + How to raise an exception and provide arguments?
    - Python exception handling # example-7.
  + User-Defined Exceptions.
    - How to define a custom exception in Python?
    - Create an exception class in Python.
    - Python exception handling # example-8.
    - Example-8# Run with multiple tests data.
  + List of Python Built-in Exceptions.
* [Python Try Except](http://www.techbeamers.com/use-try-except-python/)
  + How to handle an arbitrary exception
  + Catch multiple exceptions in one except block
  + Handling multiple exceptions with one except block
  + Re-raising exceptions in Python
  + When to use the else clause
  + Make use of [finally clause]
  + Use the As keyword to catch specific exception types
  + Best practice for manually raising exceptions
  + How to skip through errors and continue execution
  + Most common exception errors in Python
  + Examples of most common exceptions in Python
* [Python Pickle](http://www.techbeamers.com/python-tutorial-using-pickle-for-serializing-python-objects/) – Serialisation and Deserialisation
  + What is Pickle and How to Use It?
  + What Data Type can Pickle Module Store?
  + How to Use Pickle Library for Serializing Python Objects?
    - Step-(1) Construct Pickle Data.
    - Step-(2) Saving Data as a Pickle File.
    - Step-(3) Loading Data From Pickle File.
* [Python Time](http://www.techbeamers.com/python-time-functions-usage-examples/)
  + Essential facts about time module.
    - What is EPOCH and how Python use it for time operations?
    - What is a tick in Python?
    - What is DST time standard?
  + Most useful time functions in Python.
    - time.time()
    - time.clock()
    - time.ctime()
    - time.sleep()
    - time.struct\_time class
    - time.strftime()
  + Work with Timezones in Python.
    - What is UTC?
    - What are GMT and UTC? List out their differences.
  + Check Timezone in Python.
    - 1- time.timezone property
    - 2- time.tzname property
  + Set Timezone in Python.
    - Use time.tzset() function
  + Quick wrap up.
* [Python Random](http://www.techbeamers.com/using-python-random/)
  + randrange function
  + randint function
  + choice function
  + shuffle function
  + sample function
  + random function
  + uniform function

**Glossary**

* [Python Keyword](http://www.techbeamers.com/python-keywords-identifiers-variables/)s, Identifiers and Variables
  + help> keywords
  + keyword.iskeyword()
  + <str>.isidentifier()
  + id()
  + type()
* [Python Statement](http://www.techbeamers.com/understand-python-statement-indentation/), Expression and Indentation
  + Assignment
  + Augmented assignment
  + Multi-line (\)
  + Implicit line continuation with ()/[]/{}
* [Python Comment](http://www.techbeamers.com/understand-python-comment-docstring/)
  + #
  + ‘‘‘\_’’’ Docstrings
* [Python Data Types](http://www.techbeamers.com/python-data-types-learn-basic-advanced/)
* Numbers (Math Library)
  + isinstance(a, b)
  + decimals.Decimal()
  + fractions.Fraction()
  + abs()
  + ceil()
  + cmp(a, b)
  + exp()
  + floor()
  + log()
  + log10()
  + max(x1, x2, …)
  + min(x1, x2, …)
  + modf()
  + mod()
  + pow(x, y)
  + round(x [,n])
  + sqrt()
  + pi
  + e
* Lists
  + L1 = […] ⬄ list()
  + len() – Size of list/string
  + List Comprehension: theList = [expression(iter) **for** iter **in** oldList **if** filter(iter)]
  + Multi-Dim List: two\_dim\_list = [ [0]\*3 ] \*3
  + list.extend()
  + list.append()
  + list[0] = value at index 0
  + list[-1] – Reverse indexing
  + list[START:STOP:STEP] – List Slicing
  + Iteration
    - ‘for’ statement
    - enumerate()
    - range(len())
  + insert()
  + Slice assignment: list[1:x] where x is length of new array
  + del
  + remove() for a given element
  + pop() for stacks (First In Last Out, FILO model)
  + ‘in’ operator
  + list.index()
  + min(list) & max(list)
  + list.sort() for ascending, list.sort(reverse=True) for descending
  + sorted() – Returns a sorted copy of list
  + count()
  + copy()
* Sets
  + set() – Static elements vs {} which creates a dict
  + add()
  + update()
  + discard()
  + remove() – Same as above except will throw KeyError if target isn’t part of set
  + pop() – Random because no indexes are used
  + clear()
  + union() OR ‘A | B’
  + intersection() OR ‘ A & B’
  + difference() OR ‘A – B’
  + symmetric\_difference() OR ‘A ^ B’
  + frozenset()
  + copy()
  + isdisjoint()
  + issubset()
  + issuperset()
* Tuples
  + tuple()
* Dictionaries
  + dict()
  + popitem()
* [Python Operators](http://www.techbeamers.com/python-operators-tutorial-beginners/)
  + Arithmetic
  + Comparison
  + Logic
  + Bitwise
  + Assignment
  + Identity – *is*
  + Membership – *in*
* [Operator Precedence](http://www.techbeamers.com/python-operator-precedence-associativity/)
* [Python Namespace](http://www.techbeamers.com/python-namespace-scope/)
* [Python String](http://www.techbeamers.com/python-strings-functions-and-examples/)
  + Concatenate (+)
  + Repetition (\*)
  + Slicing and Range Slicing: [] and [ : ]
  + Escape Characters
  + Format Characters
  + String Comparison
  + String Conversion
  + Find a String
  + Replace a String
* Python Class
  + class XYZ()  
    def \_init\_(self, x1, x2, x3…)
* Python Inheritance
  + class ABC(XYZ):  
    super().\_init\_(self, x1, x2, x3…)
* [Python File I/O](http://www.techbeamers.com/python-file-handling-tutorial-beginners/)
  + open()
  + close()
  + write()
  + File Read Positions
    - tell()
    - seek()
  + Rename and Delete
    - os.rename
    - os.delete
* [Python Copy File](http://www.techbeamers.com/python-copy-file/)
  + shutil
    - shutil copyfile() method
    - shutil copy() method
    - shutil copyfileobj() method
    - shutil copy2() method
  + os
    - os popen method
    - os system() method
  + threading Thread() method
  + subprocess
    - subprocess call() method
    - subprocess check\_output() method
* [Python Exception](http://www.techbeamers.com/python-try-except-beginners/)
  + try-except-else block
  + try:
  + except:
  + else:
* [Python Try Except](http://www.techbeamers.com/use-try-except-python/)
  + ex:
  + raise
  + finally:
  + as keyword
  + assert
* [Python Pickle](http://www.techbeamers.com/python-tutorial-using-pickle-for-serializing-python-objects/)
  + dump()
  + load()
* [Python Time](http://www.techbeamers.com/python-time-functions-usage-examples/)
  + time.time() – Wall clock
  + time.clock() – Processor time, deprecated in 3.3
  + time.perf\_counter() – Measures short duration with highest precision, factors in process sleep time.
  + time.process\_time() – Outputs sum of system and user CPU time of current process. Ignores sleep time.
  + time.ctime() – Seconds since EPOCH. Length of return string is 24 characters
  + time.sleep() – Halts process for specified number of seconds
  + time.struct\_time class – Only data structure for **time** module.
  + time.strftime() – Converts tuple or struct\_time to string and slices it.
  + localtime()
  + asctime()
  + time.timezone
  + time.tzname
* [Python Random](http://www.techbeamers.com/using-python-random/)
  + randrange(Stop)
  + randrange(Start, Stop[, Step])
  + randint(Low, High)
  + choice(Seq)
  + shuffle(List)
  + sample(Collection, Random List Length)
  + random() – Generates random floats
  + uniform(Lower, Upper)

Creating Identifiers:

1. Sequence of letters in uppercase/lowercase, digits or underscore
2. Digits can’t begin an identifier name, otherwise syntax error
3. Keywords are reserved and cannot be used
4. Special characters are forbidden
5. Max limit of 79 characters

Variables:

1. Must initialise, but don’t need to declare
2. E.g. test = 10: Can create a new variable, and creates an association of variable with a newly created object to refer the value.
3. When the expression changes, a new object is associated with the value and the old value sent to garbage.
4. For optimisation, Python builds a cache to reuse some of the immutable objects like small int or str.
5. An object is a region of memory to hold the actual object values, a type designator and a reference counter that determines if the object can be reclaimed.
6. Objects have a type. Variables can hold objects of different types.

Statements: Assignments and Expressions

Expressions are statements that contain a logical sequence of numbers, strings, obj and operators

Expressions may be typed directly into IDLE

eval() used to read strings as numbers

Simple Assignment Statements

variable = expression

LHS ⬄ RHS

1. RHS is a simple expression  
   Different variables with same values are assigned different addresses  
   Except when *interning* is used to save memory. This is when the variables are <20 characters and contain no whitespaces, or are integers with range [-5, +255]
2. RHS is a current Python variable  
   New variable is like another name for the old one. This is shown by their identical addresses.
3. RHS is an operation  
   Type depends on outcome of operation. Usually float or int.

Augmented Assignments: Combining operators as shortcuts for changing variables

Multi-line Statements: Implicit (\) and Explicit (Lists)

Indentation: Used to mark blocks of code like braces {} in other languages

Comments

Keep comments at same indent level as the code it refers to

Documentation strings are used to add quick notes. They are regular strings except that they can spread to multiple lines, so they are executable and will be garbage collected if not labelled.

Useful for the maintenance stage of software (said to be 90% vs 10% spent on dev)

Data Types

1. Booleans
2. Numbers – Int (unlimited max length), Long, (elim by int in Python 3), Float, Complex  
   Automatically upcast in above order  
   0b, 0o, 0x for different bases
3. Strings – All Unicode in Python 3. Substrings also can be extracted using arrays
4. Bytes
5. Lists – Can contain items of different data types. Entries can be added inside and extracted (sliced). May be nested in each other.   
   Mutable means that their content can be altered without changing the identity of the list.  
   Actually distinct from arrays, which can only contain items of the same data type. This can make data storage lighter since there is no need to remember the data type of each individual item.  
   **NumPy** arrays are even more efficient at reading and writing, especially for multi-dim arrays. So they are used heavily in data science.  
   List operations can be called individually where they dynamically generate data, or as methods where they mutate the list object itself.
6. Tuples – Similar to lists, but are immutable. They cannot be edited later on, but their elements can. They have some advantages over lists, such as being more lightweight.  
   Can be returned by a function.
7. Sets – Like lists but arithmetic can be carried out on them. The elements of a set are immutable, while an immutable set is known as a frozen set.
8. Dictionaries – Lists in which each data entry is stored with its key. They can then be called by their key instead of their data. These are mutable.  
   I guess it’s kind of like a dictionary, in that each entry (word) has a longer meaning that is contained within its word.  
   Powerful because you don’t need to keep track of the data itself, but only the locations it’s transferred between.

<https://www.datacamp.com/community/tutorials/data-structures-python#array>

Lists can be further subdivided into structures that resemble their real-world models:

* Linear  
  Data items are ordered in (linear) sequence. One element is linked to only one other, so all the data items can be traversed in a single run.
  + Stacks – Last-In-First-Out (LIFO), just like a stack of plates in a cabinet.  
    In computer science, this concept is used for evaluating expressions and syntax parsing, scheduling algortihms/routines, etc.  
    Uses pop()
  + Queues – First-In-First-Out (FIFO). The first person to queue up is the first to be served. Not efficient to implement with lists because append() and pop() are not fast and cost memory movement. Moreover, there is a shift in element positions.
* Non-linear  
  Data items are not ordered sequentially. One element may be linked to several elements. All the data items might not be traversed in a single run.
  + Graphs – Network consisting of nodes/vertices which might have connections in between.
    - Edge – Line on the path connecting two nodes
    - Directed – Line with a particular direction/flow
    - Undirected – Line without flow

Used to find existence of path, or shortest path between two nodes. Weights/costs can also be assigned to different paths.

* + Trees – Hierarchical organisation of data
    - Root/Parent at top
    - Branch
    - Leaf
    - Children – Nodes below parent
    - Siblings – Nodes with same parent

Numbers

/ - Returns float quotient in Python 3

// - Returns int quotient

% - Returns remainder

Both are returned in divmod(a, b) 🡪 (//, %)

Decimals and Fractions libraries may be imported to handle values with greater precision. For WYSIWYG, put the value between quote marks.

Lists

List Comprehension is very powerful for manipulating the data within lists in certain manners, for certain lists, that satisfy certain conditions. Lists can even be selectively combined in this manner.

theList = [expression(iter) **for** iter **in** oldList **if** filter(iter)]

Multi-Dim lists are possible by specifying initial value for each element i.e. L2 = [0]\*3  
L3 = [ [0]\*3 ] \* 3, so each element becomes a sublist, vs separate objects. Editing values is difficult because other rows are affected by the change, hence list comprehension is needed to select for position within lists and sublists.

Reverse indexing is possible by inputting negative values, starting from -1 at the last term (which is technically more human-readable)

‘for-in’ loops are used to traverse lists and related data types.

Sets

Mutable objects that don’t use indexing and hence lack order. This means the elements can’t be changed via slicing or index access.

Their primary goal seems to be for a more direct comparison of groups of data, focusing on the values without having to worry about indexes. This is similar to the difference between permutations and combinations: *picking* something you have to take the order into account, but when *choosing* something then the only thing that matters is the presence of an element.

A list can be converted to a set.

Duplicates are automatically excluded.

Catch and print errors with a ‘try-except’ statement that stores the error in a variable and then prints it.

Tuples

Immutable to direct change but can hold modifiable elements. They can also be assigned with new data entirely. So the elements can’t be swapped out, but can only be treated as a whole. Likewise, deletion only works on the tuple as a whole.

May contain data of mixed numbers, mixed types and nested tuples.

Lists and sets can be converted to tuples.

Tuple assignment is possible by assigning several variables in a tuple to several values at the same time. This may be when the immutability of the tuple becomes useful.

Dictionaries

Like a list but keys are stored alongside values in the format {KEY:VALUE, …}. (But this is NOT a set).

Duplicate keys can’t exist. If a new value is input with an already-existing key, then the old value will be replaced.

Keys should derive from the immutable data types – Strings, numbers or tuples.

Dict comprehension can be used to create or read dictionaries.

(Can use original index positions as the key via list.index(x):x, where x is the entry of the list and traverse via ‘for-in’ loop. Constant value can be added to the key to offset it.)

Operators

1. Arithmetic
2. Comparison
3. Logical – and, or, not. Logical tests are compressed into these operators.
4. Bitwise
5. Assignment
6. Identity – To compare the memory addresses of two variables
7. Membership – Check if a value is a member of another Python object. Much easier than C which requires iterative checking. Can test against a dictionary only for the key.

Precedence and Associativity

Namespaces – Implemented in the form of dictionaries.

Available in C/C++ and Java.

If a variable is used both outside and within a function, then the value will change based on the level. This is because each function creates its own variable within the local scope, and the value shown based on this local context. However, if the variable is declared to be global then it will retain the latest value assigned to it in that state.

Difference between global variables and local variables. When importing overlapping modules, be aware that common functions will be overwritten.

* Import all names from a module
* Import specific names from a module – Use ‘from-import’ statement
* Import just the module using its name

Strings and their operators

Mainly slicing and escape characters, ‘\’ and format characters ‘%’ are used for formatting purposes.

Regular strings are stored as 8-bit ASCII while Unicode strings are stored as 16-bit ASCII.

String conversions alter the string output when printed, but the data remains the same.

String comparisons return Boolean values.

Find and replace strings

Classes

Define your class with the ‘class’ keyword. ‘self’ is implicitly the first argument for the ‘\_init\_’ method.

**Class creation**

class XXX:  
 def \_init\_(self, att1, att2)  
 self.att1 = att1  
 self.att2 = att2  
 XXX.count += 1

def Info(self)  
 print(“Attribute 1:”, self.att1)  
 print(“Attribute 2:”, self.att2)

**Instantiation**

b1 = XXX(“”, “”)

b2 = XXX(“”, ””)

**Check values**

b1.Info()

b2.Info()

print(“XXX Count:”, XXX.count())

Inheritance

A class can copy

class ParentClass:  
 Parent class attributes  
 Parent class methods

class ChildClass(ParentClass):  
 Child class attributes  
 Child class methods

super().\_init\_

Python File I/O (aka File Handling)

* Open files
* Close files
* Read files
* Write to files
* File read positions

If you open a file, then close it. If exceptions occur, then the program will stop and the file will remain open. Use a try-finally loop to close the file even with exceptions.

The WITH clause automatically closes files after they’ve been opened.

//Somehow, it’s really amazing that I can write and read a text (log) file from IDLE. It makes me wonder what other files I could manipulate like that. Moreover, it’s not me doing it directly – it’s the code I’ve written doing that, which means it can be automated.

The read function may read the entire file, or read up till the desired size. Following which it will read the rest of the file. This size becomes known as the *offset*.

Copy File

Imported from shutil module  
SameFileError: Both source and destination files are the same.

shutil copyfile() method:  
copyfile(source\_file, destination\_file)  
Copies only if write permissions are granted, else IOError  
Will overwrite destination files of same name  
Cannot copy to directories  
Uses copyfileobj()

shutil copy() method  
copyfile(source\_file, [destination\_file or dest\_dir])  
Like “cp” command in Unix  
Can copy to directories, creating a new file with same name (basename) as source file.  
Vs copy():

* Also sets permission bits
* Copyfile() uses copyfileobj(), but copy() uses copyfile() and copymode()
* **Slower** than copy() due to additional task of preserving permissions

shutil copyfileobj() method  
copyfileobj(self.stream, target, buffer\_size)  
Lower-level method  
Additional optional argument: Buffer size (default of 16KB)  
If the target (destination) is a file object, it has to be closed explicitly afterwards.

shutil copy2() method  
copy2(\_\_file\_\_, 'testfile')  
Functions like the copy() method, but also gets access and modification times. These are added in the metadata while copying the data.

Also compares copymode() and copystat()

os

os popen method  
Read as ‘p-open’.  
os.popen(command[, mode[, bufsize]])  
Creates a pipe to/from the command, and then returns an open file object that connects to a pipe

os system() method  
os.system('copy 1.txt.py 2.txt.py')  
Most common way of running any system command

threading Thread() method  
Thread(target=shutil.copy, args=[src, dst]).start()  
src and dst are the two files – Source and Destination  
For copying a file in an asynchronous method

subprocess

subprocess call() method  
status = subprocess.call(cmd, shell=True), with prior declaration of cmd  
Provides a simple interface to work with child processes.  
Enables launching of subprocesses, attachment to input/output/error pipes, and retrieval of return values  
Aims to replace legacy modules and functions like os.

subprocess check\_output() method  
status = subprocess.check\_output(['copy', src, dst], shell=True)  
Runs an external command/program and captures its output  
Also supports pipes

A **pipe** is a technique that unidirectionally transfers information between two program processes.   
https://whatis.techtarget.com/definition/pipe

Exception

Error – Usually internal to the program, and may arise from issues in syntax.   
Occurs at compile time

Exception – Usually external; An issue encountered during the runtime  
Occurs at runtime  
Escalated from an error situation that the script can’t cope with  
When a script raises an exception, a corresponding object is created. The exception is handled immediately, but if this cannot be done then the program terminates and prints a traceback to the error and its location.

The try-except-else block is used to handle exceptions, with the main code placed in the try sub-block.

raise [Exception [, args [, traceback]]]

Rough flowchart:

1. Try-block: Code works well and no printing of exception.
2. Except-block: Code doesn’t work and error thrown
3. No-exceptions clause: Possible but not useful
4. Multiple exceptions: The matching one is thrown out of several options, all with the same action
5. Try-finally: The catchall option that will be executed no matter what.
6. Try-finally: Incompatible with ‘except’ or ‘else’ clauses
7. Nesting: Try-finally inside try-except 🡪 Finally and except clauses printed in the end
8. Raise an Exception: Throwing an exception without actually having to trigger one
9. User-defined Exceptions: Create a custom exception class from base (parent) class
10. List of Built-In Exceptions

Try-Except

Allows common, predicted problems from code to be identified. Most errors and exceptions should be identifiable, reserving only the fringe cases for the ‘unknown’ category.  
Also used for controlling errors in loops and I/O.

How to handle an arbitrary exception – Prints any exception

Catch multiple exceptions in one except block

Handling multiple exceptions with one except block

Re-raising exceptions in Python

When to use the else clause

Make use of [finally clause]

Use the As keyword to catch specific exception types

Best practice for manually raising exceptions – Raise the most specific exception for your problem. Avoid generic exceptions because then other specific exceptions will be caught.

How to skip through errors and continue execution

Most common exception errors in Python

* IOError – It occurs on errors like a file fails to open.
* ImportError – If a python module can’t be loaded or located.
* ValueError – It occurs if a function gets an argument of right type but an inappropriate value.
* KeyboardInterrupt – It gets hit when the user enters the interrupt key (i.e. Control-C or Del key)
* EOFError – It gets raised if the input functions (input()/raw\_input()) hit an end-of-file condition (EOF) but without reading any data.

Examples of most common exceptions in Python

Another notable source of issues is the need for updated data. The functions may return data that becomes stale by the time it’s used.

Pickle

Serialisation of objects – Process of transforming data structures or objects into a transmissible format (a stream of bytes). They can then be transmitted by offloading to a file or memory cache, or transmitted over a network connection. The object can then be reconstructed in the same or different environment.

Aka marshalling, vs deserialization/unmarshalling

The library used for these processes is called ‘Pickle’. (Probably to ‘preserve’ the objects for future use, just like storing vegetables to extend their shelf life.)

First, create a dictionary that stores the metadata of the object (website in example). Then save the data as a Pickle file after importing the library. This data can be loaded later on by constructing a new object based on the scanned Pickle object, effectively reconstructing the first object.

Time

EPOCH, DST, tick

Wall-clock/Time elapsed/Stopwatch

Random

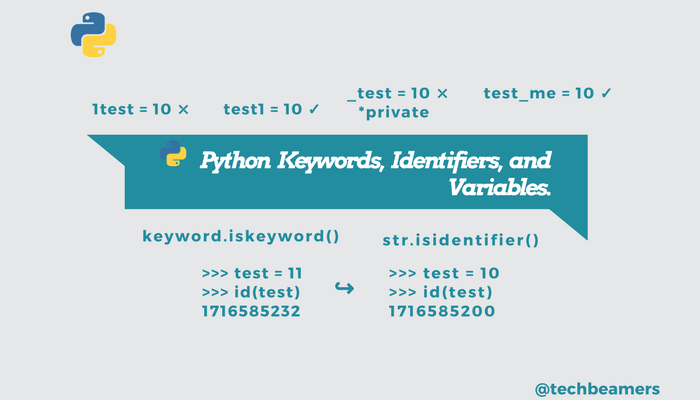
Uses Mersenne Twister algorithm, which is a pseudo-random number generator

Fisher-Yates algorithm for shuffle()

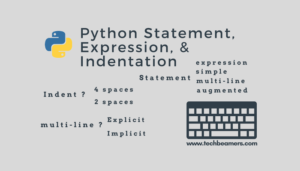
‘for-in’ loop should use range().

Define functions using the *def* method.

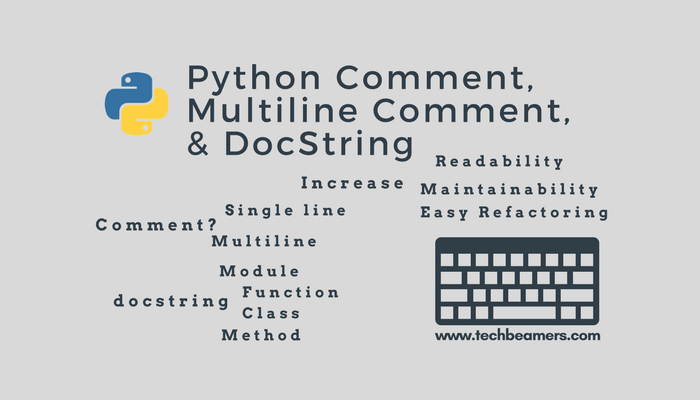
Keywords



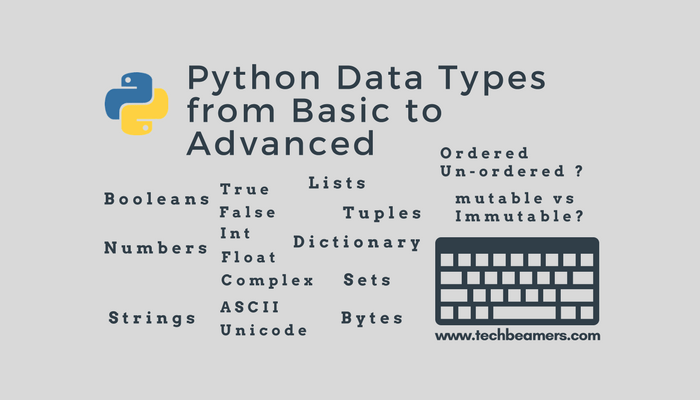
Statements



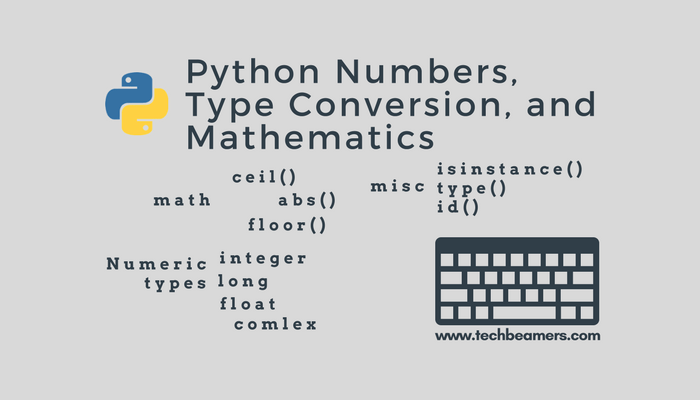
Comments



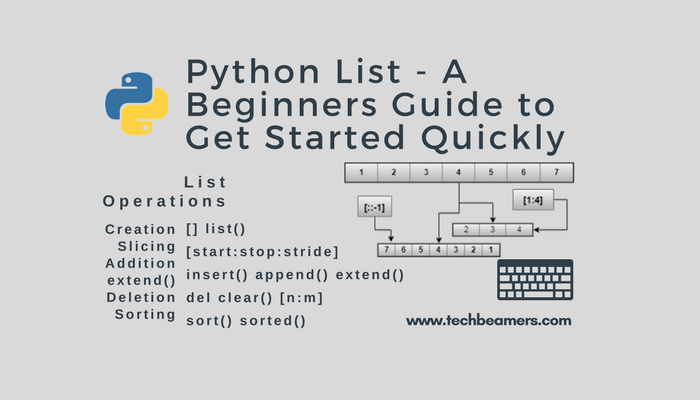
[Python Data Types](http://www.techbeamers.com/python-data-types-learn-basic-advanced/)



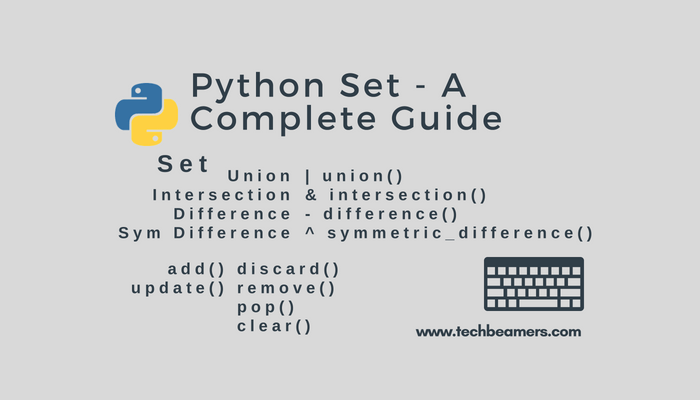
Numbers



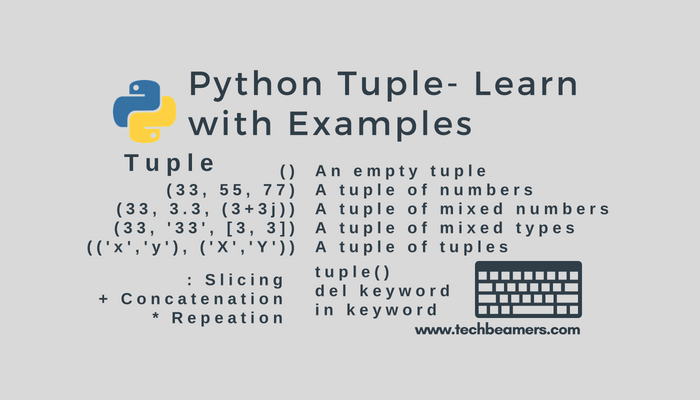
Lists



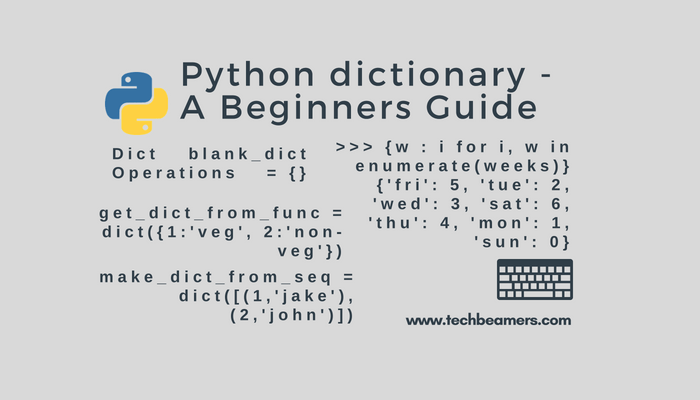
Sets



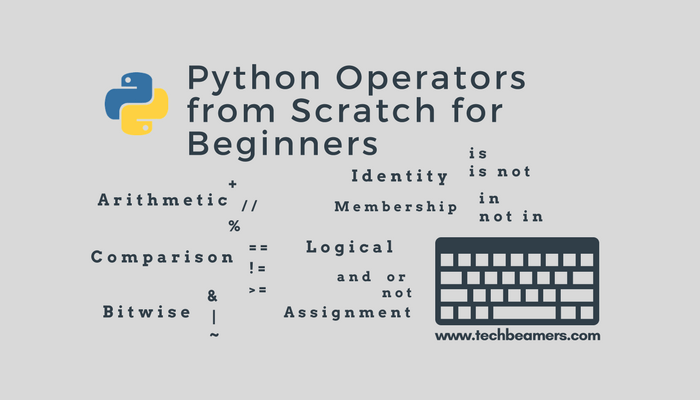
Tuples



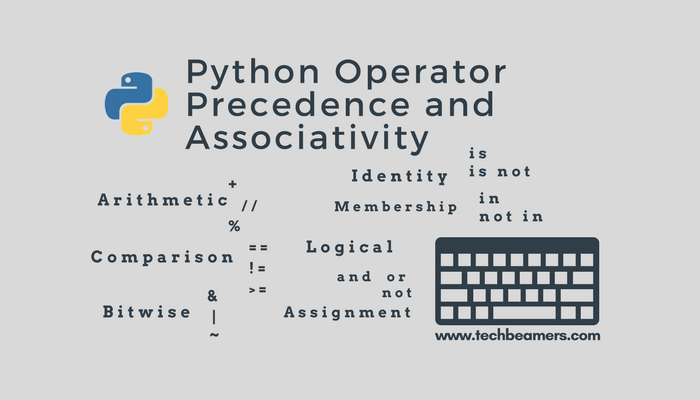
Dictionaries



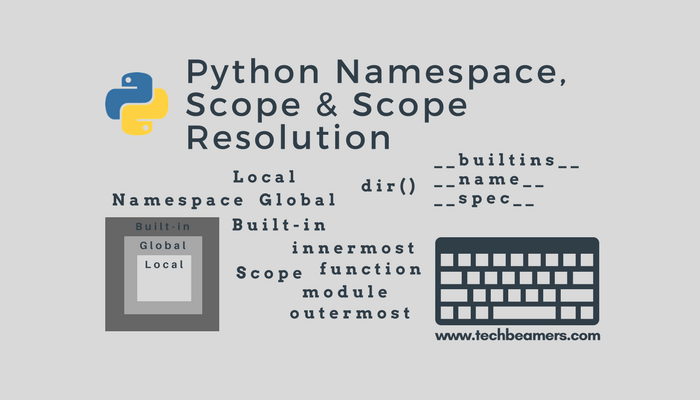
[Python Operators](http://www.techbeamers.com/python-operators-tutorial-beginners/)



[Operator Precedence](http://www.techbeamers.com/python-operator-precedence-associativity/)



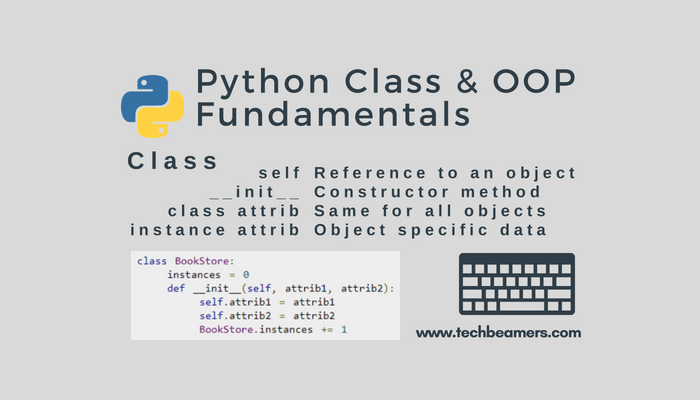
[Python Namespace](http://www.techbeamers.com/python-namespace-scope/)



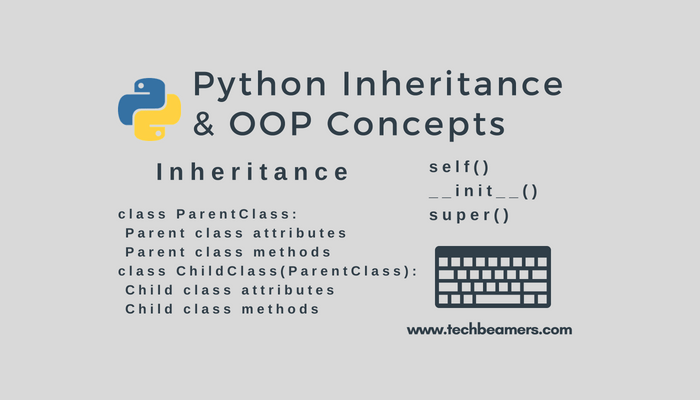
[Python String](http://www.techbeamers.com/python-strings-functions-and-examples/)

N/A

Python Class

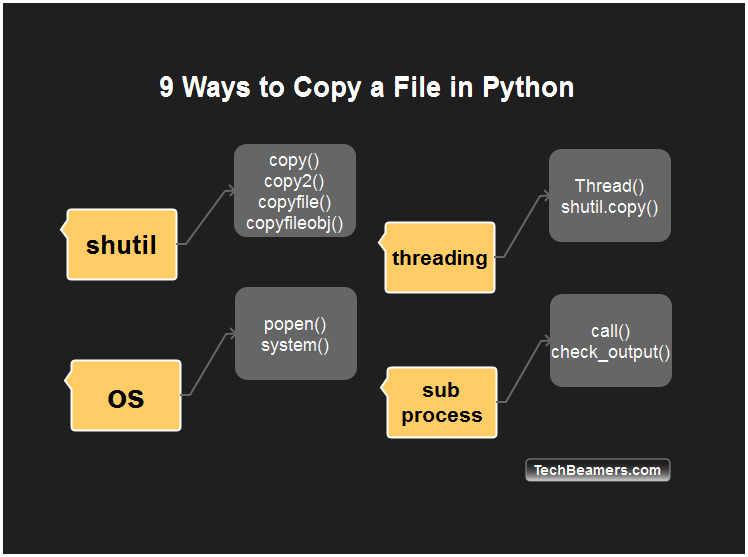


Python Inheritance

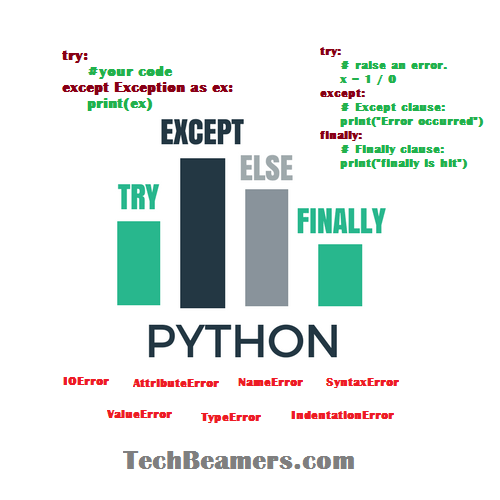


[Python File I/O](http://www.techbeamers.com/python-file-handling-tutorial-beginners/)  
N/A

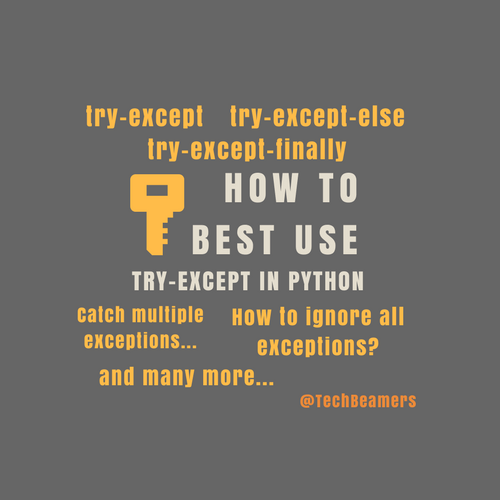
[Python Copy File](http://www.techbeamers.com/python-copy-file/)



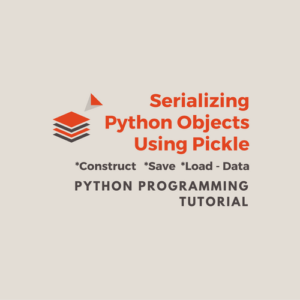
[Python Exception](http://www.techbeamers.com/python-try-except-beginners/)



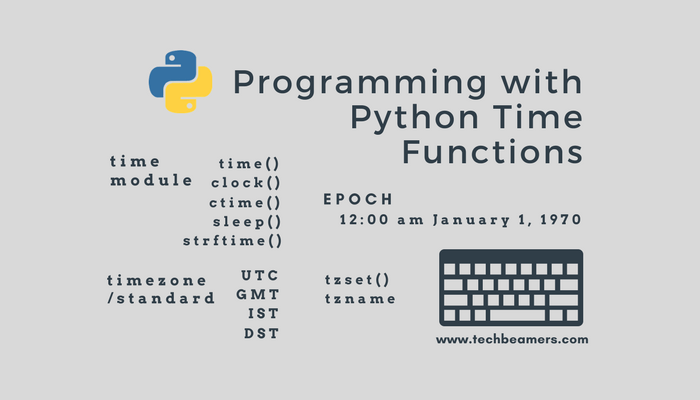
[Python Try Except](http://www.techbeamers.com/use-try-except-python/)



[Python Pickle](http://www.techbeamers.com/python-tutorial-using-pickle-for-serializing-python-objects/)



[Python Time](http://www.techbeamers.com/python-time-functions-usage-examples/)



[Python Random](http://www.techbeamers.com/using-python-random/)

