Python Indentation

Python indentation is a fundamental concept since improper indentation will result in an IndentationError and prevent the code from compiling.

Python indentation is the process of inserting white space before a statement in a specific code block. Stated differently, every statement that has an identical space on the right side is a part of the same code block.

For example:

The lines *print('Welcome to Mangaldai College')* and *print('CSIT Department')* are two separate code blocks. The two blocks of code in our example if-statement are both indented four spaces. The final *print('Thank you for visiting us')* is not indented, so it does not belong to the else block.

```
site = 'CSIT'
if site == 'CSIT' :
   print('Welcome to Mangaldai College ')
else:
   print('CSIT Department')
print('Thank you for visiting us')
```

Python List, Tuple, Set, Dictionary

List: Similar to dynamically sized arrays declared in other languages (such as vector in C++ and ArrayList in Java), Python lists are similar to those. To put it simply, a list is an assortment of items separated by commas and enclosed in [].

For Example:

```
Var = ["Welcome", "to", "CSIT"]
print(Var)
```

To create a list in Python, simply enclose the sequence in square brackets []. A list can create a list without the need for an internal function, unlike Sets.

Tuple: Similar to a list, a tuple is a collection of Python objects. A tuple is made up of an arbitrary sequence of values that are indexed by whole numbers.

"Commas" are used syntactically to separate values in a tuple. It is more common to define a tuple by closing the value sequence in parenthesis, though it is not required. This facilitates a better understanding of the Python tuples.

For example:

```
Tuple1 = ('Mangaldai', 'College')
print(Tuple1)
```

Set: An unordered, mutable, iterable collection of data types with no duplicate elements is called a set in Python. Even though a set may contain a variety of elements, the order in which they appear is not specified. Using a set instead of a list has the main benefit of having a highly efficient way to determine whether a given element is included in the set.

Sets can be created by using the built-in set() function with an iterable object or a sequence by placing the sequence inside curly braces, separated by a 'comma'.

For Example:

```
CSIT = set("Mangaldai College")
print(CSIT)
```

Dictionary: Python dictionaries, as opposed to other data types that only contain a single value as an element, are collections of keys and values that are used to store data values, much like a map.

For Example:

```
Dict = {1: 'CSIT', 2: 'Mangaldai', 3: 'College'}
print(Dict)
```

Exception Handling

Python errors come in two flavors: syntax errors and exceptions. Program errors are issues that cause the program to halt its execution. However, exceptions are raised when internal events take place that alter the program's normal flow.

Various Python exception types include:

When a problem arises while a program is being executed in Python, a number of built-in exceptions can be raised. The following are a few of the most typical Python exception types:

SyntaxError: This exception is raised when the interpreter encounters a syntax error in the code, such as a misspelled keyword, a missing colon, or an unbalanced parenthesis.

TypeError: This exception is raised when an operation or function is applied to an object of the wrong type, such as adding a string to an integer.

NameError: This exception is raised when a variable or function name is not found in the current scope.

IndexError: This exception is raised when an index is out of range for a list, tuple, or other sequence types.

KeyError: This exception is raised when a key is not found in a dictionary.

ValueError: This exception is raised when a function or method is called with an invalid argument or input, such as trying to convert a string to an integer when the string does not represent a valid integer.

AttributeError: This exception is raised when an attribute or method is not found on an object, such as trying to access a non-existent attribute of a class instance.
IOError: This exception is raised when an I/O operation, such as reading or writing a file, fails due to an input/output error.

ZeroDivisionError: This exception is raised when an attempt is made to divide a number by zero.

ImportError: This exception is raised when an import statement fails to find or load a module.

For Example:

```
x = 5
y = "hello"
try:
   z = x + y
except TypeError:
    print("Error: cannot add an int and a str")
```