COMP10001: Foundations of Computing

Contents

Foundations of Computing

Summary

- Python is a programming language with objects, modules, threads, exceptions and automatic memory management. The benefits of pythons are that it is simple and easy, portable, extensible, build-in data structure and it is an open source.
- Python language is an interpreted language.
    - Python program runs directly from the source code.
    - does not need to be compiled before its run
    - it converts the source code that is written by the programmer into an intermediate language, which is again translated into machine language that has to be executed.
- runtime is slower than of compiled languages
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.
- dynamically typed :variable types don't need to be typecast when declared

PEP 8

- coding convention, a set of recommendation, about how write readable Python code

Spacing

- Use 4 spaces for indentation
- Don't use tab
- Line length must not exceed 79 characters.
- There should be no spaces after between function/method names and arguments (also for slices and indexing)
- Use blank lines between logically separate blocks of code (e.g. between functions and major sections associated with different comments).
- Avoid multiple statements on the same line

- Always start a new line after if, elif, else, while, for etc.

commenting

- Keep the indentation level of any comments the same as the line of code it is associated with.
- Make sure comments do not contradict your code.
- Do not state the obvious in comments.

Naming etc:

- Do not use l (lower-case "L"), I (upper-case "I") or O (upper-case "O") as variable names.
- Function names should be lowercase, with words separated by underscores as necessary to improve readability.
- Constants should be in all-caps.
- Never compare with Booleans (== True/False)

garbage collection

- Python deletes unneeded objects (built-in types or class instances) automatically to free the memory space
- Python's garbage collector runs during program execution and is triggered when an object's reference count reaches zero. An object's reference count changes as the number of aliases that point to it changes.
-

Variables and Types

Variables : Names which values are assigned to

- Syntax Rules for variable names
  - First character must be alphabetical or underscore
  - Following characters can be alphabetical, underscores or digits

Literals : Constant values (integers, strings)

Operators : Written in between arguments (Different operators are used for different purposes)

- Arithmetic Operators
- Comparison Operators
- Assignment Operators
- Logical Operators
- Membership Operators
- Identity Operators

Exponent : Raises its left argument to its right argument (**)

- Remember that exponents are left to right associative
- Left exponents will be performed before the right one
- Eg. **a**b**c = (a**b)**c**

Objects : Every value which python can compute is an object

Type : Each object has a Type

- Can Identify type through **type()** function

Numerical Computation

- Note that Boolean values True and false can be used interchangeably as integers in python:
  - **True** - 1
  - **False** - 0
- Dividing two integer type numbers
  - If the result of dividing two integer Type numbers consists of decimals places/ floating point numbers, python will floor (Only the base number is returned) the result.
  - Either of the numbers must be converted into a float type number before the division for the result to show the floating-point numbers.
  - Note that you cannot apply the **float( )** function to the entire division expression as it would return only the base number as in integer then turn that into a float with a .0
- Divisions involving 0 as the denominator will result in a *'ZeroDivisionError:'*.

Logical Expressions & Boolean values

- The bool type consists of True and False Values
- Represents the value of logical expressions
- Logical expressions test to see if a condition is True or False

Operators:

- Logical Operators: (from highest to lowest precedence)

```
not         -           Highest Precedence
and         -
or          -           Lowest Precedence
```

- Comparison Operators:

```
==          :           Equals to
<           :           Less than
<=          :           Less than or Equals to
>           :           More than
>=          :           More than or Equals to
!=          :           Not Equals to
```

- Identity Operator:

```
in
not in
```

Boolean False Value Objects:

```
None Type               :None
Boolean False           :           :False
Decimal zero            :           0
Float zero              :0.0
Empty String            :           ‘ ‘
Empty List              :[ ]
```

All other objects have a True Boolean value

Scope and Namespaces

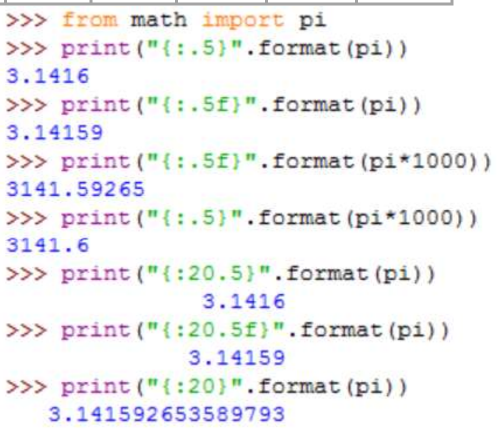
- Scope is the area of python code where a particular namespace is used
- A namespace is a mapping from names to objects
- When a function is called, a local namespace for the function is called
- **dir()** : is a function that lists the current namespace
- when python tries to find an option, it first looks in local namespace and then global namespace

Slicing/Indexing

- variable_name[Num1:Num2:Num3]
  - Num1 : Start index (inclusive)
  - Num2 : End index (exclusive)
  - Num3 : Optional, steps (use -1 to reverse order)

Zero offset	0	1	2	3	4	5	6	7	8	9	10	11	12
One offset	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- If start/end point not specified, assumed to start slicing at beginning/end
- Empty space count as index position
- Place Holders:
  - Strings : **%s**
  - Decimals : **%d**
  - Floats : **%f**
- Decimal places can be set using %0.2variable_name
- The decimal in front will correspond to the number of decimal places
- Field width(spaces in front or behind the variable inserted) can be set for decimal integers (%d) by placing a number in the middle of the % and ‘d’ corresponding to how many spaces
  - Positive number for spaces on the left
  - Negative number for spaces on the right



```
>>> from math import pi
>>> print("{:.5}".format(pi))
3.1416
>>> print("{:.5f}".format(pi))
3.14159
>>> print("{:.5f}".format(pi*1000))
3141.59265
>>> print("{:.5}".format(pi*1000))
3141.6
>>> print("{:20.5}".format(pi))
              3.1416
>>> print("{:20.5f}".format(pi))
             3.14159
>>> print("{:20}".format(pi))
   3.141592653589793
```