**Python** 5-8 Jan 2021

*(Learning from Radix Systems’ Python course, accompanying notes to course manual)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Topic** |  |  | | |
| Python2, Python3 | * Python3 has less features than Python2 (e.g. methods/functions) * Python3 is not backward compatible |  | | |
| Alternative Python implementations | * IronPython * Python running on .NET * Able to use .NET classes |  | | |
| Pypi | * Repository of software for the Python programming language * *snmp library*: network management |  | | |
| IDEs | * Spyder (looks like RStudio) |  | | |
| Python interactive shell/mode | Useful for   * Checking out features of a language without writing a program * Documentation (e.g. help(math))   Can check   * What’s inside namespace (e.g. dir()) |  | | |
| Python style guide | * Standard recommendation * Links: <https://www.python.org/dev/peps/pep-0008/>, <https://google.github.io/styleguide/pyguide.html> * PyLint – source-code, bug and quality checker for Python programs | pylint test.py | | |
| Object | * Python uses object pointer-reference   + Example: multiple variables can point to the same object * Assigning reference   + Example: * No primitive types in Python, *all* are objects   + 🙁 Slower then to directly reference   + Some implementation of Python have primitive type (e.g. Cython) * Many objects sequence property |  | | |
| Expression vs Statement | * Expressions will return values * Statements (e.g. assignments) |  | | |
| Iterator vs Iterable | Iterator   * Has next() * E.g. open(‘file.txt’) can go down the file line by line   Iterable   * Can generate an iterator from it, using **iter(x)** * Has no next() | # Example: TextIOWrapper is a iterator  dir(open(‘file.txt’)) | | |
| Namespace |  |  | | |
| Strings | Properties   * Immutable * Sequences of unicode characters * Not bytes (like in C) |  | | |
| Numbers | Integers   * No limit on how large/small the number can be * Can put base (default base 10)   Float   * Limit on the number of digits, will truncate   Binary number   * k = 0**b**101, k is int 5   Hexanumber   * k = 0**x**1234, k is int 4660   Octonumber   * k = 0**o**755, k is int 493 |  | | |
| List | * A sequence of objects * Closer to a vector/dynamic array * Can contain any object * Using \_\_add\_\_ operator (i.e. plus+ sign) is memory inefficient   + Creates a new list   + Different from append(), extend() |  | | |
| Generator functions | * Those with keyword yield in them * Allows declaration of a function that behaves like an iterator class   Following from <https://stackoverflow.com/questions/102535/what-can-you-use-python-generator-functions-for>:   * Think of generators as returning multiple items, as if returning a list * But instead of returning items all at once, they return them one-by-one * Generator function is paused until next item is requested * Good for:   + Calculating large sets of results (esp. when calculation involves loops)   + When you don’t necessary need all the results   + When you don’t want to allocate memory for all results at the same time |  | | |
| Operator overloading | * Assigning a different meaning to an operator * Operators used in Python: <https://docs.python.org/3/library/operator.html#mapping-operators-to-functions> * Override function/method, use operator to test changes * Note: you cannot define your own operators 🙁 | **Operation** | **Syntax** (operator) | **Function** |
| Addition | a **+** b | add(a, b) |
| Concatenation | seq1 + seq2 | concat(seq1, seq2) |
| Division | a \ b | truediv(a, b) |
| Floor division | a [\\](file:///\\) b | floordiv(a, b) |
| Exponentiation | a \*\* b | pow(a, b) |
| Left shift | a << b | lshift(a, b) |
| Matrix multiplication | a @ b | matmul(a, b) |
| Right shift | a >> b | rshift(a, b) |
| Modulo | a % b | mod(a, b) |
| Ordering | a < b | lt(a, b) |
| Ordering | a <= b | le(a, b) |
| Difference | a != b | ne(a, b) |
| Equality | a == b | eq(a, b) |
| Bitwise Exclusive Or | a ^ b | xor(a, b) |
| Method Overloading | * No overloading of methods in Python 🙁 |  |  |  |
| String formatting | Helpful links   * <https://www.python.org/dev/peps/pep-0498/> * [**https://mkaz.blog/code/python-string-format-cookbook/**](https://mkaz.blog/code/python-string-format-cookbook/) | **Format** | **Description** | |
| **{:.2f}** | Float (2 d.p.) | |
| **{:+.2f}** | Float (2 d.p. with sign) | |
| **{:.2f}** | Float (no d.p.) | |
| **{:0>2d}** | Pad number with zeroes on left (s.t. integer width 2) | |
| **{:x<4d}** | Pad number with x’s on right (s.t. integer with 4) | |
| **{:.2%}** | Format percentage | |
| **{:.2e}** | Exponent notation (e.g. 1.00e+09) | |
| Creating custom objects | 1. Creates an empty object in memory    * alice = BankAcct(‘Alice’, 120) 2. Empty object calls initialisation method, passes values to argument    * init(self, ‘Alice’, 120) 3. Object is initialised with values 4. Once object created, fields can be added    * alice.addr = ‘Tampines’ |  | | |
| Inheritance | * Inheriting classes   1. Need not have any constructor (can be omitted)   2. Can call base class constructor in its constructor      + Method 1: ParentClass.\_\_init\_\_(...)      + Method 2: super().\_\_init\_\_(…) |  | | |
|  |  |  | | |
| Unicode | * A character set * A list of characters with unique numbers * **(Character -> number)** * Cannot use unicode character directly in a text file | **Example**: in Unicode character set, number for  A  is  41 | | |
| UTF-8 | * Unicode Transformation Format * An encoding * An algorithm that translates a list of numbers to binary so that it can be stored on disk * **(Number -> binary)** * Encoding for unicode | **Example**: UTF-8 translates  1 2 3 4  to  00000001 00000010 00000011 00000100 | | |
| Unicode and UTF-8 | * <https://web.archive.org/web/20190710063815/http://www.polylab.dk/utf8-vs-unicode.html>   Flow   1. Application reads binary from disk 2. Application knows data represents Unicode string encoded with UTF-8 3. Convert binary data to numbers (use UTF-8 to decode) 4. Translate each number to a corresponding character (use Unicode character set) |  | | |
| Pylint | * Coding style guides: <https://www.python.org/dev/peps/pep-0008/>, <https://google.github.io/styleguide/pyguide.html> * Follows Python style guide * Standard recommendation | pylint hello.py | | |
| vi | * Tab-key means 4 spaces * :set expandtab * Tab-key means \t character * :setnoexpandtab |  | | |
| krita | Profession free and open-source painting program |  | | |

**Useful Commands**

|  |  |  |
| --- | --- | --- |
| **Python** |  | **Example** |
| help(...) |  | help(math) # modules  A = 2; help(A) # obj/var |
| **dir() \*\*\*** | List what objects are currently in my namespace   * Run in interactive shell * E.g. variables declared, imported modules |  |
| dir(...) | List attributes of an object   * Introspection | # attributes of a module (..., \_\_name\_\_, ..., log, log10, ..., sin, ...)  dir(math)  # ?  Dir(\_\_builtins\_\_ |
| print(..., ...) | Concat and print multiple strings/objects   * Default joining delimeter: single whitespace | # output “x is 16”  print(“x is”, x) |
| print(..., end= “...”) | Specify default character(s) to print at the end of string   * Default: newline (\n) |  |
| ord(…) | Get unicode number of a **character**   * Can be letters of any language | # Returns 97  ord('a') |
| 'string\_here'.encode('utf8')  'string\_here'.decode('utf8') | Encode and decode in UTF8 |  |
| hex(…) | Convert number to hexadecimal string | # Returns 0x61  hex(97) |
| 'x is {:.2f}'.format(x)  f'y is {y:.2f}' | String formatting   1. Use .format 2. Reference existing object directly in string, and use f-quote | # 2 decimal  'x is {:.2f}'**.format**(x)  # Faster version  **f**'p is {p:.2f}' |
| open('chn.txt', 'rb') | Open text file as binary file   * Without 'b', default reads as txt file | # Output b'...'  Open('test.txt', 'rb') |
| int(‘1ff’, base = 16) | Convert string to integer in given numeric base   * Any base from 2 to 36 * “Normal” math is base 10 (uses symbols 0,1,2,3,4,5,6,7,8,9) * Base 2 uses symbols 0,1 * Can convert hex string to integer | # Function for calculating base  result = 0  for i in len(s):  result += s[i]\*(b\*\*(len(s)-i) |
| if \_\_name\_\_ == ‘\_\_main\_\_’:  ... | Checking in program is run as a module |  |
| repr(...) | Returns a printable representation of object   * Magic method | # Example  # 1. Calls **\_\_str\_\_** method  # 2. If \_\_str\_\_ method not defined, returns **\_\_repr\_\_**(self)  **print**(some\_obj)  # Example  # Shows all the **\n,\t** which would otherwise been not seen  for line in f:  print(**repr**(line)) |
| @staticmethod  def getCount():  return ... | Static method   * Not often used * Usually use modules instead |  |
| while …:  …  else:  ... | Finally-else equivalent |  |
| n = None or 20+2 | Using or and and in assignment |  |
| for n in 10,20,30:  print(n) | For-loop without using range()   * Use a tuple |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **Unix** |  | **Example** |
| #!/usr/bin/python3 | Tell what interpreter to use to run script   * **To place as the first line in script** * Only in Linux (Windows uses file extension to run file) * Default interpreter (if no specified): bash shell | # Command to run program  ./hello.py # no need python3 hello.py |
| #!/usr/bin/python3  #encoding: utf8 | Specify encoding program uses |  |
| chmod +x hello.py | Make program executable   * In Linux |  |
| hexdump test.txt | Display content of binary file in dexadecimal |  |
| file test.py | See encoding of a file   * Python2 assumes that txt file is ASCII | # Returns test.py: Python script, ASCII text executable  file test.py |
| echo $LANG | View locale of system   * Only in Unix | # Returns en\_US.UTF-8  Echo $LANG |
| ls –l test.txt | Command to see filesize in bytes   * Fun-fact: 1 chinese character is 3 bytes |  |
|  |  |  |