 October 22nd, 2018

[Comments](https://programmingwithmosh.com/python/python-3-cheat-sheet/#disqus_thread)

[**Python 3 Cheat Sheet**](https://programmingwithmosh.com/python/python-3-cheat-sheet/)

**Update (Nov 19 2018):**Added exceptions and classes.

I’ve created this Python 3 cheat sheet to help beginners remember Python language syntax. You can also download this cheat sheet as a beautiful PDF [here](https://programmingwithmosh.com/wp-content/uploads/2018/11/Python-3-Cheat-Sheet-v3.pdf).

**NOTE:** This cheat sheet is a work in progress and is not complete yet. I’ll be adding new stuff to it over the next few weeks. So, be sure to come back and get the latest version.



If you’re starting out with Python and are looking for a fun and comprehensive tutorial, check out my YouTube tutorials. I have two Python tutorials. If you have no or little programming experience, I suggest you check out my [Python tutorial for beginners](https://youtu.be/yE9v9rt6ziw). Otherwise, if you know the basics (eg variables, functions, conditional statements, loops) and are looking for a tutorial that gets straight to the point and doesn’t treat you like a beginner, check out my [Python tutorial for programmers](https://youtu.be/f79MRyMsjrQ).

If you enjoy this post, please spread the love by sharing this post with others.

**Variables**

|  |  |
| --- | --- |
| 1  2  3  4  5 | a = 1       # integer  b = 1.1     # float  c = 1 + 2j  # complex number (a + bi)  d = “a”     # string  e = True    # boolean (True / False) |

**Strings**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20 | x = “Python”  len(x)  x[0]  x[-1]  x[0:3]    # Formatted strings  name = f”{first} {last}”    # Escape sequences  \” \’ \\ \n    # String methods  x.upper()  x.lower()  x.title()  x.strip()  x.find(“p”)  x.replace(“a”, “b”)  “a” in x |

**Type Conversion**

|  |  |
| --- | --- |
| 1  2  3  4 | int(x)  float(x)  bool(x)  string(x) |

**Falsy Values**

|  |  |
| --- | --- |
| 1  2  3 | 0  “”  [] |

**Conditional Statements**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12 | if x == 1:      print(“a”)  elif x == 2:      print(“b”)  else:      print(“c”)    # Ternary operator  x = “a” if n > 1 else “b”    # Chaining comparison operators  if 18 <= age < 65: |

**Loops**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | for n in range(1, 10):      print(n)    while n < 10:      print(n)      n += 1 |

**Functions**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20 | def increment(number, by=1):      return number + by    # Keyword arguments  increment(2, by=1)    # Variable number of arguments  def multiply(\*numbers):      for number in numbers:          print number      multiply(1, 2, 3, 4)    # Variable number of keyword arguments  def save\_user(\*\*user):      ...      save\_user(id=1, name="Mosh") |

**Lists**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69 | # Creating lists  letters = ["a", "b", "c"]  matrix = [[0, 1], [1, 2]]  zeros = [0] \* 5  combined = zeros + letters  numbers = list(range(20))    # Accessing items  letters = ["a", "b", "c", "d"]  letters[0]  # "a"  letters[-1] # "d"    # Slicing lists  letters[0:3]   # "a", "b", "c"  letters[:3]    # "a", "b", "c"  letters[0:]    # "a", "b", "c", "d"  letters[:]     # "a", "b", "c", "d"  letters[::2]   # "a", "c"  letters[::-1]  # "d", "c", "b", "a"    # Unpacking  first, second, \*other = letters    # Looping over lists  for letter in letters:      ...    for index, letter in enumerate(letters):      ...    # Adding items  letters.append("e")  letters.insert(0, "-")    # Removing items  letters.pop()  letters.pop(0)  letters.remove("b")  del letters[0:3]    # Finding items  if "f" in letters:      letters.index("f")    # Sorting lists  letters.sort()  letters.sort(reverse=True)    # Custom sorting  items = [      ("Product1", 10),      ("Product2", 9),      ("Product3", 11)  ]    items.sort(key=lambda item: item[1])    # Map and filter  prices = list(map(lambda item: item[1], items))  expensive\_items = list(filter(lambda item: item[1] >= 10, items))    # List comprehensions  prices = [item[1] for item in items]  expensive\_items = [item for item in items if item[1] >= 10]    # Zip function  list1 = [1, 2, 3]  list2 = [10, 20, 30]  combined = list(zip(list1, list2))    # [(1, 10), (2, 20)] |

**Tuples**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10 | point = (1, 2, 3)  point(0:2)     # (1, 2)  x, y, z = point  if 10 in point:      ...    # Swapping variables  x = 10  y = 11  x, y = y, x |

**Arrays**

|  |  |
| --- | --- |
| 1  2  3 | from array import array    numbers = array("i", [1, 2, 3]) |

**Sets**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10 | first = {1, 2, 3, 4}  second = {1, 5}    first | second  # {1, 2, 3, 4, 5}  first & second  # {1}  first - second  # {2, 3, 4}  first ^ second  # {2, 3, 4, 5}    if 1 in first:      ... |

**Dictionaries**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12 | point = {"x": 1, "y": 2}  point = dict(x=1, y=2)  point["z"] = 3  if "a" in point:      ...  point.get("a", 0)   # 0  del point["x"]  for key, value in point.items():     ...    # Dictionary comprehensions  values = {x: x \* 2 for x in range(5)} |

**Generator Expressions**

|  |  |
| --- | --- |
| 1  2  3 | values = (x \* 2 for x in range(10000))  len(values)  # Error  for x in values: |

**Unpacking Operator**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | first = [1, 2, 3]  second = [4, 5, 6]  combined = [\*first, "a", \*second]    first = {"x": 1}  second = {"y": 2}  combined = {\*\*first, \*\*second} |

**Exceptions**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | # Handling Exceptions  try:    …  except (ValueError, ZeroDivisionError):    …  else:    # no exceptions raised  finally:    # cleanup code    # Raising exceptions  if x < 1:      raise ValueError(“…”)    # The with statement  with open(“file.txt”) as file:     … |

**Classes**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74 | # Creating classes  class Point:      def \_\_init\_\_(self, x, y):          self.x = x          self.y = y        def draw(self):          …    # Instance vs class attributes  class Point:      default\_color = “red”        def \_\_init\_\_(self, x, y):          self.x = x    # Instance vs class methods  class Point:      def draw(self):          …        @classmethod      def zero(cls):          return cls(0, 0)      # Magic methods  \_\_str\_\_()   \_\_eq\_\_()  \_\_cmp\_\_()  ...    # Private members  class Point:      def \_\_init\_\_(self, x):          self.\_\_x = x      # Properties  class Point:      def \_\_init\_\_(self, x):          self.\_\_x = x        @property      def x(self):          return self.\_\_x        @property.setter:      def x.setter(self, value):          self.\_\_x = value    # Inheritance  class FileStream(Stream):      def open(self):           super().open()           …    # Multiple inheritance  class FlyingFish(Flyer, Swimmer):      …    # Abstract base classes  from abc import ABC, abstractmethod    class Stream(ABC):      @abstractmethod      def read(self):          pass    # Named tuples  from collections import namedtuple    Point = namedtuple(“Point”, [“x”, “y”])  point = Point(x=1, y=2) |