Python Language & Syntax Cheat Sheet

Variable Assignment

```
integer = 1
string = "string"
unicode_string = u"unicode string"
mutli_line_string = """ multi-line
string
"""
tuple = (element1, element2, element3, ...
list = [ element1, element2, element3, ...
dictionary = { key1 : value1key2value2, ...
dictionary[key] = value
class_instance = ClassName(init_args)
```

Frequently Used Built-in Types

```
True                    False None
str         unicode int
float       list dict
```

Other than , and , these can also be used as functions to explicitly cast a value to that type

Functions

```
def function_name(arg1, arg2,
keyword1=val1, keyword2=val2, ...):
 <function body>
 return return_value
e.g.
def my_function(x, y, z=0):my_function(1, 2) → 3
 sum = x + y + zmy_function(1, 2, 3) → 6
 return summy_function(1, 2, y=4) → 7
```

Classes

```
class ClassName(SuperClass):
 class_variable = static_value
 def __init__(self, value1, <...>):
 self.instance_variable1 = value1
 self.instance_function()
 def instance_function(self, arg1, <...>):
 <function body>
 return return_value
e.g.
class MyClass(object):MyClass.offset → 1
 offset = 1
 def __init__(self, value):c = MyClass(2)
 self.value = valuec.value → 2
 def get_offset_value(self):c.get_offset_value() → 3
 return MyClass.offset +
 self.value
```

Imports

```
import module
from module import function, variable
```

Frequently Used String Manipulations

```
string1 + string1                    "str" + "ing" → "string"
"%s%s" % (string1, string2)          "%s%s" % ("s", "g") → "sg"
string.split("delim", limit)         "s/g".split("/") → ["s", "g"]
string.strip()                       "  string  ".strip() →
string.startswith("prefix")          "string"
substring in string                  "str".startswith("s") →
print string                         True "str" in "string" →
                                     True
```

List Comprehension

```
[value for value in list if condition]
e.g.
[x for x in [1,2,3,4,5,6,7,8,9] if x % 2 == 0] → [2,4,6,8]
```

Python is white-space dependent; code blocks are indented 4 sp

```
value = dictionary[key]
value = dictionary.get(key, default_value)
value = list[index]e.g. [5,6,7][2] → 7
                                        value = string[start:en
value = list[start:end]e.g. [1,2,3][1:2] → [2]
value = ClassName.class_variable
)value = class_instance.instance_variable
                                    ]value = class_instance.function(args)
, : }

value1 == value2"str" == "str" → True
value1 != value2"str" != "str" → False
value1 < value21 < 2 → True
value1 <= value22 <= 2 → True
value1 > value22 > 3 → False
value1 >= value23 >= 3 → True
value is [not] None
value in list1 in [2,3,4] → False
isinstance(class_instance, ClassName)


i = a + bi = a - b
i = a / bi = a * b
i = a % be.g.11 % 3 → 2
```

Comments

```
"""                          # Line Comment
  Multi-line comment
"""
```

Control Flow

```
if conditional                 i ==if 7:
  <body>                       "seven"print
elif conditional:              elif e.g. i == 8:
  <body>                             print"eight"
else:                          else
  <body>                             print str(i)
for in value list:                   for in [1, 2, 3, 4]:
  <body>                       if e.g. i == 2:      continu
  continue                     if i == 3:           e break
 break                           i          print
whileconditional:              while True:
  <body>                         e.g. "infinity"print
  continue
 break
```

Exceptions

```
try:                    try:
  <body>                              database.update()
raise Exception()       e.g.except          Exception as e:
except Exception        as e:          log.error(e.msg)
  <exception handling>                database.abort()
finally:                finally:
  <clean-up>                          database.commit()
```

File & Path Manipulation

```
import os # import the os module first
os.path.join(path_segment1, path_segment2, ...)
os.path.exists( )
os.listdir(directory_path)
os.remove(file_path)
os.mkdir(directory_path)
file = open(path, "rw")
file.read()
string.write("string")
```