

Reference Docs

The Python Standard Library: <https://docs.python.org/3/library/>
 Built-in functions: <https://docs.python.org/3/library/functions.html#dir>
 Datetime strftime Format Codes: <https://strftime.org/>

Dicts

```
# initializing
d1 = { 'k1': 'v1', 'k2': 'v2' }
d2 = dict(k3='v3', k4='v4')
# merging
d3 = d1|d2 # python3.9 or later
d3 = { **d1, **d2 }
# comprehension syntax
d4 = { i:'static_value' for i in some_list }
# iterating
for k,v in d1.items():
    print( f"Key {k} has a value of {v}")
```

Sets

```
s1 = {1,2,3}
s2 = {3,4,5}
s3 = {5,6,7}
s1.intersection(s2)
# {3}
s1.union(s2)
# {1, 2, 3, 4, 5}
s1.union(s2, s3)
# {1, 2, 3, 4, 5, 6, 7}
```

Operators

```
# ternary operators
x = 'foo' if 2+2==5 else 'bar' # x = 'bar'
```

Precedence of Python Operators

Operator	Meaning
()	Parentheses
**	Exponent
+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
	Bitwise OR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparisons, Identity, Membership operators
not	Logical NOT
and	Logical AND
or	Logical OR

print()

```
## 'sep' argument ##
print(1, 2, 3, sep="\n ")
# 1
# 2
# 3
## end argument ##
print( 'address', end='@')
print( 'domain[.]com')
# address@domain[.]com
## repeat printing ##
print( " Python " * 3)
# Python Python Python
```



By **eazykaye**
cheatography.com/eazykaye/

Published 10th November, 2023.
 Last updated 10th November, 2023.
 Page 2 of 3.

Sponsored by **Readable.com**
 Measure your website readability!
<https://readable.com>

Argparse

```
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('-c', '--count', type=int,
                    required=True)
parser.add_argument('-o', '--operator',
                    choices=['add', 'subtract'])
parser.add_argument('-n', '--number', dest='number',
                    alternate_name='number')
parser.add_argument('-v', '--verbose',
                    action='store_true', help='enable verbose
                    output')

args = parser.parse_args()
print(args.count)
print(args.alternate_name)
```

Classes

```
from dataclasses import dataclass

@dataclass
class Person():
    name: str
    age: int
    employed: bool
    def __repr__(self):
        return f"Name: {self.name}, Age:
        {self.age}, Employed: {self.employed}"

class Car:
    def __init__(self, make, model):
        self.make = make
        self.model = model

    def OtherFunction(self):
        pass
```

Strings

```
"AAABBBBB".count('C') # 4
"AAA BBC CCC".index('B') # 3
" strip both sides ".strip() # 'strip both sides'
" strip left side ".lstrip() # 'strip left side'
```

Strings (cont)

```
> " strip right side ".rstrip() # 'strip right side'
", ".join(['list', 'of', 'words', 'to', 'join']) # "list, of, words, to, join"
```

String Methods ref: <https://docs.python.org/3/library/stdtypes.html#string-methods>

F-String ref: <https://cheatography.com/eazykaye/cheat-sheets/python3-f-strings/>

Lists

```
# merge lists
list3 = list1 + list2

# basic list comprehension
# syntax: [exp for member in iterable (if conditional)]
[x for x in range(10) if x > 5] # [6, 7, 8, 9]
# nested list comprehension
[f"{x}{y}" for x in "ab" for y in "cd"] #
['ac', 'ad', 'bc', 'bd']

# slicing | syntax: List[ Start Index : End
Index(not inclusive) : Step ]
L1 = [0,1,2,3,4,5,6,7,8,9]
L1[4:7] # [4, 5, 6]
L1[:2] # [0, 2, 4, 6, 8]
L1[::-1] # [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

# unpacking
first, *middle, last = [0,1,2,3,4,5,6,7,8,9]
# first = 0
# middle = [1, 2, 3, 4, 5, 6, 7, 8]
# last = 9

# zip
a = [1, 2, 3]
b = [4, 5, 6]
z = zip(a, b)
list(z)
# [(1, 4), (2, 5), (3, 6)]
```



Maths

```
round(5.2) # 5
round(5.7) # 6
from math import floor, ceil
floor(5.2) # 5
ceil(5.2) # 6
abs(-5.2) # 5.2
min([1, 2, 3]) # 1
max([1, 2, 3]) # 3
sum([1, 2, 3]) # 6
```

Misc

```
#eval()
exp = "2 + 2"
eval(exp) # 4
exp = "+.join([str(n) for n in range(9)])"
print(exp) # '0+1+2 +3+ 4+5 +6+7+8'
eval(exp) # 36
all([ 2 > 1, 3 > 2, 4 > 5 ])
# False
any([ 2 > 1, 3 > 2, 4 > 5 ])
# True
```

Logging

```
import logging
from logging.handlers import RotatingFileHandler, TimedRotatingFileHandler
logger = logging.getLogger("logger_name")
logger.setLevel(logging.DEBUG)
fmtr = logging.Formatter('%(asctime)s - %(levelname)s - [line %(lineno)s] - %(message)s')
logfile = '/path/to/log/file'
fh = TimedRotatingFileHandler(logfile, when='midnight', interval=1, backupCount=30)
fh.setFormatter(fmtr)
```

Logging (cont)

```
> fh.setLevel(logging.DEBUG)
logger.addHandler(fh)
ch = logging.StreamHandler()
ch.setFormatter(fmtr)
ch.setLevel(logging.DEBUG)
logger.addHandler(ch)
```

Exception Handling

```
try:
    # code here
except Exception as e:
    logger.critical("Exception running that code!")
    logger.critical(e, exc_info=True)
else:
    #runs if no exceptions were raised
finally
    #always runs, good for closing resources and such
```

OS

```
import os, glob
# concatenate file paths
filepath = os.path.join('/tmp/', 'subdir', 'filename.txt') # '/tmp/subdir/filename.txt'
# read in files using context manager
with open(filepath, 'r') as f:
    file_contents = f.readlines()
```

