Python3 Pro Tips Cheat Sheet

by eazykaye via cheatography.com/164268/cs/34572/

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(k 3='v3', k4='v4')
# merging
d3 = d1|d2 # python3.9 or later
d3 = { **d1, **d2 }
# compre hension syntax
d4 = { i:'sta tic _value' for i in some_list }
# iterating
for k,v in d1.ite ms():
        pri nt( f"Key {k} has a value of {v}")
```

Sets

```
s1 = {1,2,3}
s2 = {3,4,5}
s3 = {5,6,7}
s1.int ers ect ion(s2)
# {3}
s1.uni on(s2)
# {1, 2, 3, 4, 5}
s1.uni on(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( 'ad dress', end='@')
print( 'do mai n[.] com')
# addres s@d oma in[.]com
## repeat printing ##
print( " Pyt hon " * 3)
# Python Pyt hon Python
```

Cheatography

Python3 Pro Tips Cheat Sheet

by eazykaye via cheatography.com/164268/cs/34572/

Argparse

```
import argparse
parser = argpar se.A rg ume ntP arser()
parser.ad d_a rgu men t('-c', '--count', type=int,
requir ed= True)
parser.ad d_a rgu men t('-o', '--ope rator',
choice s=[ 'add', 'subtr act'])
parser.ad d_a rgu men t('-n', '--num ber', dest=' -
alt ern ate _va ria ble _name')
parser.ad d_a rgu men t('-v', '--ver bose',
action ='s tor e_t rue', help=' enable verbose
output')
args = parser.pa rse _args()
print( arg s.c ount)
print( arg s.a lte rna te_ var iab le_ name)
```

Classes

```
from dataclasses import dataclass
@dataclass
class Person():
        name: str
       age: int
        emp loyed: bool
       def __repr __( self):
                return f"Name: {self.n ame}, Age:
{self.a ge}, Employed: {self.e mp loy ed} "
class Car:
       def __init __( self, make, model):
                sel f.make = make
                sel f.model = model
       def OtherF unc tio n(s elf):
               pass
```

Strings

```
"AAABBCCCC".count('C') # 4
" AAA BBC CCC ".in dex ('B') # 3
" strip both sides ".st rip() # 'strip both sides'
" strip left side ".ls trip() # '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 compre hension
# syntax: [exp for member in iterable (if condit -
ional)]
[x for x in range(10) if x > 5] # [6, 7, 8, 9]
# nested list compre hension
[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 = " +".j oin ([s tr(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 loggin g.h andlers import Rotati ngF ile Han -
dler, TimedR ota tin gFi leH andler
logger = loggin g.g etL ogg er( " log ger _na me")
logger.se tLe vel (lo ggi ng.D EBUG)
fmtr = loggin g.F orm att er( '%( asc time)s - %
(leve lname)s - [line %(line no)s] - %(mess -
age)s')
logfile = '/path /to /lo g/file'
fh = TimedR ota tin gFi leH and ler (lo gfile,
when=' mid night', interv al=1, backup Cou nt=30)
fh.set For mat ter (fmtr)
```

Logging (cont)

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

Exception Handling

```
try:
       # code here
except Exception as e:
        log ger.cr iti cal ("Ex ception running
that code!")
        log ger.cr iti cal(e, exc_in fo= True)
else:
        #runs if no exceptions were raised
finally
        #always runs, good for closing resources
and such
```

OS

```
import os, glob
# concat enate file paths
filepath = os.pat h.j oin ('/ tmp/', 'subdir',
'filen ame.txt') # '/tmp/ sub dir /fi len ame.txt'
# read in files using context manager
with open(f ile path, 'r') as f:
        fil e_c ontents = f.read lines()
```