programmin with mosh

Python 3 Cheat Sheet

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. 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.

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

Variables

```
a = 1        # integer
b = 1.1      # float
c = 1 + 2j   # complex number (a + bi)
d = "a"      # string
e = True     # boolean (True / False)
```

Strings

```
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.upper()
x.lower()
x.title()
x.strip()
x.find("p")
x.replace("a", "b")
"a" in x
```

Type Conversion

```
int(x)
float(x)
bool(x)
string(x)
```

Falsy Values

```
0
""
[]
```

Conditional Statements

```
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

```
for n in range(1, 10):
    print(n)

while n < 10:
    print(n)
    n += 1
```

Functions

```
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

```

```
12
13  multiply(1, 2, 3, 4)
14
15  # Variable number of keyword arguments
16  def save_user(**user):
17      ...
18
19
20  save_user(id=1, name="Mosh")
```

Lists

```
01  # Creating lists
02  letters = ["a", "b", "c"]
03  matrix = [[0, 1], [1, 2]]
04  zeros = [0] * 5
05  combined = zeros + letters
06  numbers = list(range(20))
07
08  # Accessing items
09  letters = ["a", "b", "c", "d"]
10  letters[0]  # "a"
11  letters[-1] # "d"
12
13  # Slicing lists
14  letters[0:3]   # "a", "b", "c"
15  letters[:3]    # "a", "b", "c"
16  letters[0:]    # "a", "b", "c", "d"
17  letters[:]     # "a", "b", "c", "d"
18  letters[::2]   # "a", "c"
19  letters[::-1]  # "d", "c", "b", "a"
20
21  # Unpacking
22  first, second, *other = letters
23
24  # Looping over lists
25  for letter in letters:
26      ...
27
28  for index, letter in enumerate(letters):
29      ...
30
31  # Adding items
32  letters.append("e")
33  letters.insert(0, "-")
34
35  # Removing items
36  letters.pop()
37  letters.pop(0)
38  letters.remove("b")
39  del letters[0:3]
40
41  # Finding items
42  if "f" in letters:
43      letters.index("f")
44
45  # Sorting lists
46  letters.sort()
47  letters.sort(reverse=True)
48
49  # Custom sorting
50  items = [
51      ("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,

# 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

```
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

```
from array import array

numbers = array("i", [1, 2, 3])
```

Sets

```
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

```
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

```
values = (x * 2 for x in range(10000))
len(values)  # Error
for x in values:
```

Unpacking Operator

```
first = [1, 2, 3]
second = [4, 5, 6]
combined = [*first, "a", *second]

first = {"x": 1}
second = {"y": 2}
combined = {**first, **second}
```

Exceptions

```
# 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

```
# 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"
```

```
13
14      def __init__(self, x, y):
15          self.x = x
16
17  # Instance vs class methods
18  class Point:
19      def draw(self):
20          …
21
22      @classmethod
23      def zero(cls):
24          return cls(0, 0)
25
26
27  # Magic methods
28  __str__()
29  __eq__()
30  __cmp__()
31  ...
32
33  # Private members
34  class Point:
35      def __init__(self, x):
36          self.__x = x
37
38
39  # Properties
40  class Point:
41      def __init__(self, x):
42          self.__x = x
43
44      @property
45      def x(self):
46          return self.__x
47
48      @property.setter:
49      def x.setter(self, value):
50          self.__x = value
51
52  # Inheritance
53  class FileStream(Stream):
54      def open(self):
55           super().open()
56           …
57
58  # Multiple inheritance
59  class FlyingFish(Flyer, Swimmer):
60      …
61
62  # Abstract base classes
63  from abc import ABC, abstractmethod
64
65  class Stream(ABC):
66      @abstractmethod
67      def read(self):
68          pass
69
70  # Named tuples
71  from collections import namedtuple
72
73  Point = namedtuple("Point", ["x", "y"])
74  point = Point(x=1, y=2)
```