# PYTHON

## Cheat Sheet

codewithmosh.com

 @moshhamedani

# Primitive Types



## Variables

```
a = 1 (integer)
b = 1.1 (float)
c = 1 + 2j (complex)
d = “a” (string)
e = True (boolean)
```

## 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.lower()
x.title()
x.strip()
x.find(“p”)
x.replace(“a”, “b”)
“a” in x
```

## Numer functions

```
round(x)
abs(x)
```

## Type conversion

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

## Falsy values

```
0
“”
None
```

# Control Flow



## Conditional statements

```
if x == 1:
    print(“a”)
elif x == 2:
    print(“b”)
else:
    print(“c”)
```

## Ternary operator

```
x = “a” if n > 1 else “b”
```

## Boolean operators

```
x and y (both should be true)
x or y  (at least one true)
not x   (inverses a boolean)
```

## Chaining comparison operators

```
if 18 <= age < 65:
```

## For loops

```
for n in range(1, 10):
    …
```

## While loops

```
while n > 10:
    …
```

## Equality operators

```
== (equal)
!= (not equal)
```

## Defining 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


multiply(1, 2, 3, 4)
```

## Variable number of keyword arguments

```
def save_user(**user):
    …


save_user(id=1, name="Mosh")
```

# Shortcuts

codewithmosh.com

## DEBUGGING

| | |
|---|---|
| Start Debugging | F5 |
| Step Over | F10 |
| Step Into | F11 |
| Step Out | Shift+F11 |
| Stop Debugging | Shift+F5 |

## CODING (Windows)

| | |
|---|---|
| End of line | End |
| Beginning of line | Home |
| End of file | Ctrl+End |
| Beginning of file | Ctrl+Home |
| Move line | Alt+Up/Down |
| Duplicate line | Shift+Alt+Down |
| Comment | Ctrl+/ |

## CODING (Mac)

| | |
|---|---|
| End of line | fn+Right |
| Beginning of line | fn+Left |
| End of file | fn+Up |
| Beginning of file | fn+Down |
| Move line | Alt+Up/Down |
| Duplicate line | Shift+Alt+Down |
| Comment | Cmd+/ |

# Lists



## 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])
```

## Zip function

```
list1 = [1, 2, 3]
list2 = [10, 20, 30]
combined = list(zip(list1, list2))
# [(1, 10), (2, 20)]
```

## Unpacking operator

```
list1 = [1, 2, 3]
list2 = [10, 20, 30]
combined = [*list1, “a”, *list2]
```

# Tuples, Sets, and Dictionaries



## Tuples

```
point = 1, 2, 3
point = (1, 2, 3)
point = (1,)
point = ()
point(0:2)
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}
```

## 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():
    ...
```

# Comprehensions



## List comprehensions

```
values = [x * 2 for x in range(5)]
values = [x * 2 for x in range(5) if x % 2 == 0]
```

## Set comprehensions

```
values = {x * 2 for x in range(5)}
```

## Dictionary comprehensions

```
values = {x: x * 2 for x in range(5)}
```

## Generator expressions

```
values = {x: x * 2 for x in range(500000)}
```

## 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:
   …
```

## 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)
```