In [ ]:
```
#file path for this file: /Users/vikshah/Desktop/Python Bootcamp/revisio
n1
```

In [ ]:
```
#https://programmingwithmosh.com/python/python-3-cheat-sheet/
```

In [182]:
```
#Variables

a = 1 #integer
b = 1.1 # float
c = "a" #string
d = True #boolean(True/False)

print(a,b,c,d)
```

```
1 1.1 a True
```

In [8]:
```
#Strings

x = "Python"

print(len(x))
print(x[0])
print(x[-1])
print(x[0:3])
```

```
6
P
n
Pyt
```

In [9]:
```
#Formatted strings
first = "John"
last = "Smith"
name = f"{first} {last}"
print(name)
```

```
John Smith
```

In [13]:
```
#Escape Sequences
#\" \' \\ \n
```

In [28]:
```
#string methods
x = "python"

# print(x.upper())
# print(x.lower())
# print(x.title())
# print(x.strip())
# print(x.find("p"))
print(x.replace("p","m"))
```

```
mython
```

In [183]:
```
#Type Conversion
x = 10
print(int(x))
print(float(x))
print(bool(x))
#print(string(x))
```

```
10
10.0
True
```

In [38]:
```
#Falsy Values
0
""
[]
```

Out[38]:
```
[]
```

In [50]:
```
#Conditional Statements
x = 5
if x == 1:
    print("a")
elif x == 3:
    print("b")
else:
    print("c")
```

```
c
```

In [56]:
```
#Chaining comparison operators

age = 50
if age <= 65:
    print("Major")
else:
    print("Senior")
```

```
Major
```

In [58]:
```
#Loops

for n in range(0,10):
    print(n)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [98]:
```
# For Loops
a = 2
for n in range(0,10,2):
    print(f"n: {n} -- {a}")

for n in range(0,10):
    print(n)
```

```
n: 0 -- 2
n: 2 -- 2
n: 4 -- 2
n: 6 -- 2
n: 8 -- 2
0
1
2
3
4
5
6
7
8
9
```

In [101]:
```
# While Loops
z = 0
while z <10:
    print(z)
    z = z + 1
```

```
0
1
2
3
4
5
6
7
8
9
```

In [131]:
```
# Functions
def increment(number, by=1):
    return number + by

increment(number, by=1)

#Keyword arguments
increment(3, by=1)
```

Out[131]:
```
4
```

In [130]:
```
#Variable number of arguments:
def multiply(*numbers):
    for number in numbers:
        print(number)

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

```
1
2
3
4
```

In [141]:
```
#Variable number of keyword arguments
def save_user(**user):

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

In [147]:
```
#Lists
#Creating Lists
letters = ["a","b","c"]
matrix = [0,1,2,3]
matrix1 = [[0,1],[2,3]]
zeros = [0]*5
combined = zeros + letters
numbers = list(range(20))

print(letters)
print(matrix)
print(matrix1)
print(zeros)
print(combined)
print(numbers)
```

```
['a', 'b', 'c']
[0, 1, 2, 3]
[[0, 1], [2, 3]]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 'a', 'b', 'c']
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [163]:
```
#Accessing items
letters = ["a","b","c","d"]
print(letters[0])
print(letters[-1])
```

```
a
d
```

In [35]:
```
#slicing lists
letters = ["a","b","c","d","e","f"]

print(letters[0:3])
print(letters[:3])
print(letters[0:])
print(letters[:])
print(letters[::2])
print(letters[::-1])
#print(letters[]) #invalid syntax
print("\n")
```

```
['a', 'b', 'c']
['a', 'b', 'c']
['a', 'b', 'c', 'd', 'e', 'f']
['a', 'b', 'c', 'd', 'e', 'f']
['a', 'c', 'e']
['f', 'e', 'd', 'c', 'b', 'a']
```

In [37]:
```
#Unpacking

letters = ["a","b","c","d","e","f"]

first,second,*other = letters

print(first)
print(second)
print(*other)
```

```
a
b
c d e f
```

In [64]:
```
#Looping over list

letters = ["a","b","c","d"]

for i in letters:
     print(i)
print("\n")

a = letters

print(len(a))
print("\n")

for a in range(0,len(a)):
    print(letters[a])

print("\n")

print(letters)
```

```
a
b
c
d


4


a
b
c
d


['a', 'b', 'c', 'd']
```

In [70]:
```
#Enumerate

letters = ["a","b","c","d"]

for index, letter in enumerate(letters):
    print(index, letter)
```

```
0 a
1 b
2 c
3 d
```

In [75]:
```
#Adding items
letters = ["a","b","c"]
letters.append("e")
print(letters)
print("\n")

letters.insert(0,"-")
print(letters)
letters.insert(2,">")
print(letters)
```

```
['a', 'b', 'c', 'e']


['-', 'a', 'b', 'c', 'e']
['-', 'a', '>', 'b', 'c', 'e']
```

In [83]:
```
#Removing items
letters = ["a","b","c","d","e","f","g","h"]

#by default it'll pop the last item in the list
letters.pop()
print(letters)

#it'll pop the first item in the list
letters.pop(0)
print(letters)

letters.remove("b")
print(letters)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g']
['b', 'c', 'd', 'e', 'f', 'g']
['c', 'd', 'e', 'f', 'g']
```

In [84]:
```
#Removing items
#Deleting items

letters = ["a","b","c","d","e","f","g","h"]
del letters[0:3]
print(letters)
```

```
['d', 'e', 'f', 'g', 'h']
```

In [89]:
```
#Finding items

letters = ["a","b","c","d","e","f","g","h"]

if "f" in letters:
    letters.index("f")
```

In [102]:
```
#Sorting lists
letters = ["d","h","c","a","e","f","g","b"]
print(letters)

letters.reverse()
print(letters)

letters.sort()
print(letters)

#In the below function
#it'll sort the list first and then reverse the list
letters.sort(reverse=True)
print(letters)
```

```
['d', 'h', 'c', 'a', 'e', 'f', 'g', 'b']
['b', 'g', 'f', 'e', 'a', 'c', 'h', 'd']
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
['h', 'g', 'f', 'e', 'd', 'c', 'b', 'a']
```

In [104]:
```
#Custom sorting
items = [
    ("Product1", 10),
    ("Product2", 9),
    ("Product3", 11)
]
items.sort(key=lambda item:item[1])
print(items)
```

```
[('Product2', 9), ('Product1', 10), ('Product3', 11)]
```

In [113]:
```
#Map and filter
items = [
    ("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, items))

print(prices)
print(expensive_items)
```

```
[9, 10, 11]
[('Product1', 10), ('Product3', 11)]
```

In [119]:
```
items = [
    ("Product1", 10),
    ("Product2", 9),
    ("Product3", 11)
]
items.sort(key=lambda item:item[1])

#List comprehensions
prices = [item[1] for item in items]
expensive_items = [item for item in items if item[1] >= 10]


print(prices)
print(expensive_items)
```

```
[9, 10, 11]
[('Product1', 10), ('Product3', 11)]
```

In [124]:
```
#Zip function

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

```
[(1, 10), (2, 20), (3, 30)]
```

In [143]:
```
# Tuples (very rarely used)

point = (1,2,3)
#point(0:2) #1,2
x,y,z = point
print(x,y,z)
```

```
1 2 3
```

In [148]:
```
# Arrays

from array import array
numbers = array("i",[1,2,3])
print(numbers)
print("\n")
print(numbers[0])
```

```
array('i', [1, 2, 3])


1
```

In [152]:
```
#Sets
#Read about how sets work

# first = {1,2,3,4}
# second = {1,5}

# first | second
# first & second
# first - second
# first ^ second

# if 1 in first:
```

In [176]:
```
#Dictionaries
#You can define keys and values couple of different ways
point = {"x":1, "y": 2}
point = dict(x=1, y=2)
point["z"] = 3
if "z" in point:
    print("correct")
else:
    print("incorrect")
#point.get("a",0) - not sure what this particular line of code doing
del point["x"] #it'll delete key x and it's value
for key, value in point.items():
    print(key,value)

#Dictionary comprehensions
values = {x:x*2 for x in range(5)}
print(values)
```

```
correct
y 2
z 3
{0: 0, 1: 2, 2: 4, 3: 6, 4: 8}
```

In [5]:
```
#Generator Expressions

values = (x * 2 for x in range(10))
#len(values) # Error
for x in values:
    print(x)
```

```
0
2
4
6
8
10
12
14
16
18
```

In [2]:
```
#Unpacking Operator

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

print(first)
print(second)
print(combined)
```

```
[1, 2, 3]
[4, 5, 6]
[1, 2, 3, 'a', 4, 5, 6]
```

In [13]:
```
#Unpacking Operator

a = {"x": 1}
b = {"y": 2}
ab = {**a, **b}
ab1 = {*a,*b} #why it's printing 'y' first?

print(a)
print(b)
print(ab)
print(ab1)
```

```
{'x': 1}
{'y': 2}
{'x': 1, 'y': 2}
{'y', 'x'}
```

In [ ]:
```
#Exceptions

#Handling Exceptions
try:
except #(ValueError, ZeroDivisionError):
else:
    #no exceptions raised
finally:
    #cleanup code
#Raising exceptions
if x < 1:
    raise ValueError("...")
#That with statement
with open("file.txt") as file:
```

In [ ]:
```
#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"

    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)
```