1) C

2) B

3) D

4) B

5) A

6) D

7) C

8) Python sorts in ascii-betical (or uni-betical) order. Uppercase 'I' has a smaller ASCII value than lowercase 'a'

9)
```
def reverse_lookup (dictionary, value):
    # return with a list of all the keys in the dictionary that are
    # associated with the value (passed as second argument)
    answer = []
    for i in dictionary:
        if dictionary[i] == value:
            answer = answer +[i]
    return answer
```

10) 0, 8, 17

11)
```
def cap(data, big):
    for i in range(len(data)):
        for j in range(len(data[i])):
            if data[i][j] > big:
                data[i][j] = big
```

12)
```
def valid(password):
    if len(password) < 8:
        return False
    has_upper = False
    has_lower = False
    has_digit = False
    for c in password:
        if c.isupper():
```

```
            has_upper = True
          if c.islower():
            has_lower = True
          if c.isdigit():
            has_digit = True
          if has_upper and has_lower and has_digit:
            return True
        return False

###alternative solution below

def valid2(password):
    return len(password) >= 8 and \
    password.lower() != password and \
    password.upper() != password and \
    password.isalnum() and not password.isalpha()

def main():
    password = input("Type a password according to the rules:")
    while not valid(password):
        password = input("Type a password according to the rules:")
    password2 = input("Now type your password again:')
    while password2 != password:
        print("PASSWORDS DO NOT MATCH")
        password2 = input("Now type your password again:')
    print("SUCCESS")


13)
class Car():
      def __init__ (self, efficiency, capacity):
            self.__efficiency = efficiency
            self.__capacity = capacity
            self.__gas = 0

      def drive (self, miles):
            gallons_spent = miles / self.__efficiency
            updated_gas = self.__gas - gallons_spent
            if updated_gas < 0:
                  print ("You ran out of gas!")
                  updated_gas = 0
            self.__gas = updated_gas
```

```
    def add_gas (self, amount):
        updated_gas = self.__gas + amount
        if updated_gas > self.__capacity:
            print ("You added too much gas!")
            updated_gas = self.__capacity
        self.__gas = updated_gas


    def get_gas_level(self):
        return self.__gas
```

14)

```
def count_odd_length():
    input_file = open (argv[1], "r")
    odd = 0
    total = 0
    for line in input_file:
        total += 1
        if len(line) % 2 == 1:
            odd += 1
    print ("Total lines =", total)
    print ("Lines of ODD length =", odd)
    print ("% lines of EVEN length = ", 100 * (total - odd) / total )
```