# Password Generator

## Code:

```
 #!/usr/bin/env python3

import sys
import string
import random
import math
import argparse
def main():
    args = parse_args()

    if args is None:
        interactive()
    else:
        automatic(args)
def automatic(args):
    charset = make_charset(args.use_upper, args.use_lower, args.use_digits,
args.use_punctuation, args.use_space,
                           args.additional, args.blacklist)

    if not args.quiet:
        print("***** Password Generator - © Sakshi ByteCommander *****")
        print()
        print("Using this character set (excluding the arrows):")
        print("→{}←".format("".join(sorted(charset))))
        print("There may be at most {} occurrences of the same character per
password.".format(args.max_dupe)
              if args.max_dupe > 0 else "There are no duplicate character limits.")
        print()
        print("Generating {} password{} of length {}:".format(args.amount, "s" if args.amount >
1 else "", args.length))
        print()

    for _ in range(args.amount):
        password = generate_password(charset, args.length, args.max_dupe)
        print(password)


def interactive():
    print("***** Password Generator - © Sakshi ByteCommander *****")
    print()

    charset = ask_charset(True, True, True, True, True, "", "")
    length = ask_length(12)
    max_duplicate_chars = ask_max_duplicate_chars(0, len(charset), length)

    while True:
        password = generate_password(charset, length, max_duplicate_chars)
```

```
        print("And your new password is:")
        print("+-" + len(password) * "-" + "-+")
        print("| " + password + " |")
        print("+-" + len(password) * "-" + "-+")

        if not ask_yn("Generate another password with the same settings?", True):
            break

    print("Thank you for using this password generator. Have a nice day!")


def parse_args():
    if len(sys.argv) <= 1:
        # no arguments --> interactive mode
        return None

    parser = argparse.ArgumentParser(description="Highly customizable random password
generator",
                                     epilog="Run it without any arguments for interactive mode.")

    parser.add_argument("length", action="store", type=int,
                        help="length of the password to generate")
    parser.add_argument("-n", "--amount", action="store", type=int, dest="amount", default=1,
                        help="how many passwords to create")
    parser.add_argument("-m", "--max-duplicate-chars", action="store", dest="max_dupe",
default=0, metavar="LIMIT",
                        help="limits how often the same character may occur in a password at most")

    parser.add_argument("-q", "--quiet", action="store_true", dest="quiet",
                        help="print only one password per line, nothing else")

    p_charset = parser.add_argument_group("Character set specification")
    p_charset.add_argument("-u", "--uppercase", action="store_true", dest="use_upper",
                           help="include uppercase letters A-Z into the available character set")
    p_charset.add_argument("-l", "--lowercase", action="store_true", dest="use_lower",
                           help="include lowercase letters a-z into the available character set")
    p_charset.add_argument("-d", "--digits", action="store_true", dest="use_digits",
                           help="include digits 0-9 into the available character set")
    p_charset.add_argument("-p", "--punctuation", action="store_true",
dest="use_punctuation",
                           help="include punctuation into the available character
set".format(string.punctuation))
    p_charset.add_argument("-s", "--space", action="store_true", dest="use_space",
                           help="include the standard space into the available character set")

    p_charset.add_argument("-a", "--additional", action="store", default="", dest="additional",
                           help="additional characters to include into the available character set")
    p_charset.add_argument("-b", "--blacklist", action="store", default="", dest="blacklist",
                           help="characters to exclude from the available character set")

    args = parser.parse_args()
    if not any([args.use_upper, args.use_lower, args.use_digits, args.use_punctuation,
args.use_space,
                args.additional]):
        parser.error("You must enable at least one character class or add custom characters!")
    return args
```

```
def ask_yn(message, default):
    if not isinstance(message, str) or not isinstance(default, bool):
        raise TypeError

    msg = message + (" (Y/n) " if default else " (y/N) ")
    while True:
        answer = input(msg).lower().strip()
        if not answer:
            return default
        if answer in "yn":
            return answer == "y"
        print("Sorry, please do only enter [y] or [n] or leave it blank to accept the default. Try
again!")


def make_charset(use_upper, use_lower, use_digits, use_punctuation, use_space, additional,
blacklist):
    if not all(isinstance(x, bool) for x in [use_upper, use_lower, use_digits, use_punctuation,
use_space]) \
            or not all(isinstance(x, str) for x in [additional, blacklist]):
        raise TypeError

    return set(use_upper * string.ascii_uppercase +
               use_lower * string.ascii_lowercase +
               use_digits * string.digits +
               use_punctuation * string.punctuation +
               use_space * " " +
               additional) \
        .difference(set(blacklist))


def ask_charset(default_upper, default_lower, default_digits, default_punctuation,
default_space,
                default_additional, default_blacklist):
    if not all(isinstance(x, bool) for x in
               [default_upper, default_lower, default_digits, default_punctuation, default_space]) \
            or not all(isinstance(x, str) for x in [default_additional, default_blacklist]):
        raise TypeError

    default_chars = make_charset(default_upper, default_lower, default_digits,
default_punctuation, default_space,
                                 default_additional, default_blacklist)

    print("The default character set to generate passwords is this (not including the arrows):")
    print("→{}←".format("".join(sorted(default_chars))))
    if ask_yn("Do you want to change the character set?", False):

        return make_charset(
            ask_yn("Do you want to allow uppercase letters '{}'?".format(string.ascii_uppercase),
default_upper),
            ask_yn("Do you want to allow lowercase letters '{}'?".format(string.ascii_lowercase),
default_lower),
            ask_yn("Do you want to allow digits '{}'?".format(string.digits), default_digits),
```

```
            ask_yn("Do you want to allow punctuation '{}'?".format(string.punctuation),
default_punctuation),
            ask_yn("Do you want to allow space '{}'?".format(" "), default_space),
            input("Please enter additional characters you want to allow (if any): "),
            input("Please enter characters you want to blacklist (if any): "))
    else:
        return default_chars


def ask_length(default_length):
    if not isinstance(default_length, int):
        raise TypeError

    print("The default password length is {} characters.".format(default_length))
    while True:
        answer = input("Enter your desired length or leave it blank to use the default: ").strip()
        if answer:
            try:
                return int(answer)
            except ValueError:
                print("Sorry, please do only enter a number or leave it blank to accept the default.
Try again!")
        else:
            return default_length


def ask_max_duplicate_chars(default_mdc, charset_len, password_len):
    if not isinstance(default_mdc, int):
        raise TypeError

    minimum_mdc = math.ceil(password_len / charset_len)

    if default_mdc:
        print("The default maximum occurrences of a character is {}
times.".format(default_mdc))
    else:
        print("By default, there is no limit how often a character may appear in the password.")

    while True:
        print("Enter your desired maximum duplicate character limit or leave it blank to use the
default.")
        answer = input("A value of 0 means no limit: ").strip()
        if answer:
            try:
                mdc = int(answer)
                if mdc >= minimum_mdc:
                    return mdc
                else:
                    print("Sorry, this limit is too low for the given character set and password
length.\n"
                          "You need to allow at least {} duplicate characters. Try
again!".format(minimum_mdc))
            except ValueError:
                print("Sorry, please do only enter a number or leave it blank to accept the default.
Try again!")
        else:
```

```
        return default_mdc
def generate_password(charset, length, max_duplicate_chars):
    if not isinstance(charset, set) or not isinstance(length, int) or not
isinstance(max_duplicate_chars, int):
        raise TypeError

    my_charset = charset.copy()
    password = ""
    while len(password) < length:
        password += random.SystemRandom().choice(list(my_charset))
        if max_duplicate_chars:
            for c in set(password):
                if password.count(c) >= max_duplicate_chars:
                    my_charset.discard(c)
    return password
if __name__ == "__main__":
    main()
```

## Output:

```
***** Password Generator - © Sakshi ByteCommander *****

The default character set to generate passwords is this (not including the arrows):
→ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~←
Do you want to change the character set? (y/N) y
Do you want to allow uppercase letters 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'? (Y/n) Y
Do you want to allow lowercase letters 'abcdefghijklmnopqrstuvwxyz'? (Y/n) Y
Do you want to allow digits '0123456789'? (Y/n) Y
Do you want to allow punctuation '!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~'? (Y/n) Y
Do you want to allow space ' '? (Y/n) Y
Please enter additional characters you want to allow (if any):
Please enter characters you want to blacklist (if any): Xx
The default password length is 12 characters.
Enter your desired length or leave it blank to use the default: 14
By default, there is no limit how often a character may appear in the password.
Enter your desired maximum duplicate character limit or leave it blank to use the default.
A value of 0 means no limit: 1
And your new password is:
+----------------+
| ^d&)?OepFN(:E< |
+----------------+
Generate another password with the same settings? (Y/n) y
And your new password is:
+----------------+
| >KP/Tm@O-4GU,u |
+----------------+
Generate another password with the same settings? (Y/n) y
And your new password is:
+----------------+
| EaN{|R"^*Zf'dr |
+----------------+
Generate another password with the same settings? (Y/n) n
Thank you for using this password generator. Have a nice day!
```