~/Documents/coding_project/coding_project/main.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

main.py

```
while(True):
    successes = []
    success_count = 0

    password = input("\nPlease enter a password: ")

    # Cross-check user input
    password = CreatePassword(password)
    password.passLength()
    password.passUpper()
    password.passDigit()
    password.passSpaces()
    password.passSpecSym()

   # Assign output value to successful input
    password.successPassLength()
    password.successPassUpper()
    password.successPassDigit()
    password.successPassSpaces()
    password.successPassSpecSym()

    # Assign variable names to function output
    success_length = password.successPassLength()
    success_upper = password.successPassUpper()
    success_digit = password.successPassDigit()
    success_spaces = password.successPassDigit()
    success_spec_sym = password.successPassSpecSym()

    # Append function variables to list 'successes'
    successes.append(success_length)
    successes.append(success_upper)
    successes.append(success_digit)
    successes.append(success_spaces)
    successes.append(success_spec_sym)

     # Add the value of each output variable in 'successes' to 'success_count'
    for successes in successes:
        if successes == 1:
            # If the variable == 1, success_count +=1
            success_count = success_count+1

    # If all variables == 1, success_count = 5: password accepted
    if success_count == 5:
        print("\nPassword accepted!")

        # password.password = variable + function
        password = password.password

        # Encrypt password with a salted hash function
        password = EncryptPass(password)
        password.encryptPass()

        hashed = password.encryptPass()

        # Included in test code only to make sure hash is working
        print(f"\nPassword hash: {hashed}")

        # Append successful password to dictionary
        user_1['password'] = hashed

        # Get saved username
        username = user_1.get('username')
        # Keep hash private
        password = password.password

        print("\nThank you for registering!")

        print(f"\tYour username: {username}")
        print(f"\tYour password: {password}")
```

Line 95, Column 27 Spaces: 4 Python

Tilix: lauxton@dravieu: ~/Documents/coding_project/coding_project

1: lauxton@dravieu: ~/Documents/coding_project/coding_project

```
Please enter a username: jk jk^
        - Username should not have any spaces
        - Username should not have any special characters

Please enter a username: jk^jk^
        - Username should not have any special characters

Please enter a username: Jenkins

Please enter a password: jk jk
        - Password needs to be between 8 and 16 characters long
        - Password should have at least one uppercase character
        - Password should have at least one number
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: jk kj jk jk
        - Password should have at least one uppercase character
        - Password should have at least one number
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: Jkj kj kj
        - Password should have at least one number
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: jk1 Jkj kj k
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: J1kkjkjkjkj
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: T4i2a1p8!

Password accepted!

Password hash: b'$2b$12$TMeud9Je2OEOjF5ixTZBmetYS7CEPb2MV.WDQ2o3Ycu/mo7q1RjOW'

Thank you for registering!
        Your username: Jenkins
        Your password: T4i2a1p8!

~~User Login~~

Please enter your username:
```

~/Documents/coding_project/coding_project/main.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

main.py

```python
# https://github.com/pyauth/pyotp
# !pip install pyotp
import pyotp

# coding_project/create_user_classes.py
from create_user_classes import CreateUsername, CreatePassword, EncryptPass
# coding_project/create_user_classes.py
from login_classes import VerifyPass, VerifyUser

# Create username and password

print('~~User Registration~~')

# Dictionary for username and password
user_1 = {}

# Create a username within certain parameters
while(True):
    successes = []
    success_count = 0

    username = input("\nPlease enter a username: ")

    # Cross-check user input
    username = CreateUsername(username)
    username.userLength()
    username.userSpaces()
    username.userSpecSym()

    # Assign output value to successful input
    username.successUserLength()
    username.successUserSpaces()
    username.successUserSpecSym()

    # Assign variable names to function output
    success_length = username.successUserLength()
    success_spaces = username.successUserSpaces()
    success_spec_sym = username.successUserSpecSym()

    # Append function variables to list 'successes'
    successes.append(success_length)
    successes.append(success_spaces)
    successes.append(success_spec_sym)

    # Add the value of each output variable in 'successes' to 'success_count'
    for successes in successes:
        if successes == 1:
            # If the variable == 1, success_count +=1
            success_count = success_count+1

    # If all variables == 1, success_count = 3: username accepted
    if success_count == 3:
        # Append username to dictionary
        user_1['username'] = username.username
        break

# Create a password within certain parameters
while(True):
    successes = []
    success_count = 0
```

Line 25, Column 40 Spaces: 4 Python

1 / 1 Tilix: lauxton@dravieu: ~/Documents/coding_project/coding_project

1: lauxton@dravieu: ~/Documents/coding_project/coding_project

```
lauxton@dravieu:~$ cd Documents/ASMIS/stelios_code
lauxton@dravieu:~/Documents/ASMIS/stelios_code$ source playground_env/bin/activate
(playground_env) lauxton@dravieu:~/Documents/ASMIS/stelios_code$ cd
(playground_env) lauxton@dravieu:~$ cd Documents/coding_project/coding_project
(playground_env) lauxton@dravieu:~/Documents/coding_project/coding_project$ python3 main.py
~~User Registration~~

Please enter a username: jk ^
        - Username needs to be between 5 and 16 characters long
        - Username should not have any spaces
        - Username should not have any special characters

Please enter a username: jk jk^
        - Username should not have any spaces
        - Username should not have any special characters

Please enter a username: jk^jk^
        - Username should not have any special characters

Please enter a username: Jenkins

Please enter a password:
```

~/Documents/coding_project/coding_project/main.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

main.py

```
    verify_password = verifyPass(password, hashed)
    verify_password.verifyPass()
    verify_password.verifyPassSuccess()

    verified_password = verify_password.verifyPassSuccess()
    successes.append(verified_password)

    # Add the value of each output variable in 'successes' to 'success_count'
    for successes in successes:
        # If the variable == 1, success_count +=1
        if successes == 1:
            success_count = success_count+1
    # If success_count = 1: password accepted
    if success_count == 1:
        break

# Two Factor Authorization
while(True):
    #Activates only if password is verified
    if success_count == 1:
        # OTP code generator
        totp = pyotp.TOTP('base32secret3232')
        print("\nYour OTP is: ", totp.now())

        # User input
        otp_code = input("What is your OTP code?\t")

        # verify the input code
        if totp.verify(otp_code)==True:
            # Simulated login
            print("\nSuccessful login")
            break
        else:
            print("\tUnsuccessful. Resending code.")
```

Line 85, Column 1 Spaces: 4 Python

Tilix: lauxton@dravieu: ~/Documents/coding_project/coding_project

1: lauxton@dravieu: ~/Documents/coding_project/coding_project

```
        - Password should have at least one uppercase character
        - Password should have at least one number
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: Jkj kj kj
        - Password should have at least one number
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: jk1 Jkj kj k
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: J1kkjkjkjkj
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: T4i2a1p8!

Password accepted!

Password hash: b'$2b$12$TMeud9Je2OEOjF5ixTZBmetYS7CEPb2MV.WDQ2o3Ycu/mo7q1RjOW'

Thank you for registering!
        Your username: Jenkins
        Your password: T4i2a1p8!

~~User Login~~

Please enter your username: jenkins
        Username is not correct.

Please enter your username: Jenkins

Please enter your password: kj kj kj
        Password does not match.

Please enter your password: T4i2a1p8!

Your OTP is:  736002
What is your OTP code?  738274
        Unsuccessful. Resending code.

Your OTP is:  480772
What is your OTP code?  480772

Successful login
(playground_env) lauxton@dravieu:~/Documents/coding_project/coding_project$
```

~/Documents/coding_project/coding_project/main.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

main.py

```
150     verified_username = verify_username.verifyUserSuccess()
151     successes.append(verified_username)
152
153     # Add the value of each output variable in 'successes' to 'success_count'
154     for successes in successes:
155         # If the variable == 1, success_count +=1
156         if successes == 1:
157             success_count = success_count+1
158     # If success_count = 1: username accepted
159     if success_count == 1:
160         break
161
162 # Verify an entered password
163 while(True):
164     successes = []
165     success_count = 0
166
167     # User input
168     password = input("\nPlease enter your password: ")
169
170     #Change string to bytes for comparison
171     password = password.encode('ASCII')
172     hashed = user_1.get('password')
173
174     # Verify password with verify function
175     verify_password = VerifyPass(password, hashed)
176     verify_password.verifyPass()
177     verify_password.verifyPassSuccess()
178
179     verified_password = verify_password.verifyPassSuccess()
180     successes.append(verified_password)
181
182     # Add the value of each output variable in 'successes' to 'success_count'
183     for successes in successes:
184         # If the variable == 1, success_count +=1
185         if successes == 1:
186             success_count = success_count+1
187     # If success_count = 1: password accepted
188     if success_count == 1:
189         break
190
191 # Two Factor Authorization
192 while(True):
193     #Activates only if password is verified
194     if success_count == 1:
195         # OTP code generator
196         totp = pyotp.TOTP('base32secret3232')
197         print("\nYour OTP is: ", totp.now())
198
199         # User input
200         otp_code = input("What is your OTP code?\t")
201
202         # verify the input code
```

Line 85, Column 1 Spaces: 4 Python

Tilix: lauxton@dravieu: ~/Documents/coding_project/coding_project

1: lauxton@dravieu: ~/Documents/coding_project/coding_project

```
        - Password should have at least one uppercase character
        - Password should have at least one number
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: jk jk jk jk
        - Password should have at least one uppercase character
        - Password should have at least one number
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: Jkj kj kj
        - Password should have at least one number
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: kjkjJ 1 kjkj
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: kjkjkjJ1kj
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: T4i2a1p8!

Password accepted!

Password hash: b'$2b$12$AtgTQDqzjxMLHyzWywuhTOlBPJspZc8HUzgfyE1cZn/UFJ4JLySJy'

Thank you for registering!
        Your username: Jenkins
        Your password: T4i2a1p8!

~~User Login~~

Please enter your username: jenkins
        Username is not correct.

Please enter your username: Jenkins

Please enter your password: T4i2a1p9!
        Password does not match.

Please enter your password: T4i2a1p8!

Your OTP is:  572053
What is your OTP code?
```

~/Documents/coding_project/coding_project/main.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

main.py

```
125         print(f"\tYour username: {username}")
126         print(f"\tYour password: {password}")
127         break
128
129
130 # Login using username, password, and Two Factor Authorization
131
132 print('\n~~User Login~~')
133
134 # Verify an entered username
135 while(True):
136     successes = []
137     success_count = 0
138
139     # User input
140     username = input("\nPlease enter your username: ")
141     username = username
142     # Cross-check variable
143     crosscheck_name = user_1.get('username')
144
145     # Verify username with verify functions
146     verify_username = VerifyUser(username, crosscheck_name)
147     verify_username.verifyUser()
148     verify_username.verifyUserSuccess()
149
150     verified_username = verify_username.verifyUserSuccess()
151     successes.append(verified_username)
152
153     # Add the value of each output variable in 'successes' to 'success_count'
154     for successes in successes:
155         # If the variable == 1, success_count +=1
156         if successes == 1:
157             success_count = success_count+1
158     # If success_count = 1: username accepted
159     if success_count == 1:
160         break
161
162 # Verify an entered password
163 while(True):
164     successes = []
165     success_count = 0
166
167     # User input
168     password = input("\nPlease enter your password: ")
169
170     #Change string to bytes for comparison
171     password = password.encode('ASCII')
172     hashed = user_1.get('password')
173
174     # Verify password with verify function
175     verify_password = VerifyPass(password, hashed)
176     verify_password.verifyPass()
177     verify_password.verifyPassSuccess()
```

Line 85, Column 1 Spaces: 4 Python

Tilix: lauxton@dravieu: ~/Documents/coding_project/coding_project

1: lauxton@dravieu: ~/Documents/coding_project/coding_project

```
        - Username should not have any special characters

Please enter a username: Jenkins

Please enter a password: jk jk
        - Password needs to be between 8 and 16 characters long
        - Password should have at least one uppercase character
        - Password should have at least one number
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: jk kj jk jk
        - Password should have at least one uppercase character
        - Password should have at least one number
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: Jkj kj kj
        - Password should have at least one number
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: jk1 Jkj kj k
        - Password should not have any spaces
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: J1kkjkjkjkj
        - Password should have at least one of the following symbols:
          ! @ # $ % & *

Please enter a password: T4i2a1p8!

Password accepted!

Password hash: b'$2b$12$TMeud9Je2OEOjF5ixTZBmetYS7CEPb2MV.WDQ2o3Ycu/mo7q1RjOW'

Thank you for registering!
        Your username: Jenkins
        Your password: T4i2a1p8!

~~User Login~~

Please enter your username: jenkins
        Username is not correct.

Please enter your username: Jenkins

Please enter your password:
```

~/Documents/coding_project/coding_project/main.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

main.py

```python
# https://github.com/pyauth/pyotp
# !pip install pyotp
import pyotp

# coding_project/create_user_classes.py
from create_user_classes import CreateUsername, CreatePassword, EncryptPass
# coding_project/create_user_classes.py
from login_classes import VerifyPass, VerifyUser

# Create username and password

print('~~User Registration~~')

# Dictionary for username and password
user_1 = {}

# Create a username within certain parameters
while(True):
    successes = []
    success_count = 0

    username = input("\nPlease enter a username: ")

    # Cross-check user input
    username = CreateUsername(username)
    username.userLength()
    username.userSpaces()
    username.userSpecSym()

    # Assign output value to successful input
    username.successUserLength()
    username.successUserSpaces()
    username.successUserSpecSym()

    # Assign variable names to function output
    success_length = username.successUserLength()
    success_spaces = username.successUserSpaces()
    success_spec_sym = username.successUserSpecSym()

    # Append function variables to list 'successes'
    successes.append(success_length)
    successes.append(success_spaces)
    successes.append(success_spec_sym)

    # Add the value of each output variable in 'successes' to 'success_count'
    for successes in successes:
        if successes == 1:
            # If the variable == 1, success_count +=1
            success_count = success_count+1

    # If all variables == 1, success_count = 3: username accepted
    if success_count == 3:
        # Append username to dictionary
```

Line 21, Column 5 Spaces: 4 Python

Tilix: lauxton@dravieu: ~/Documents/coding_project/coding_project

1: lauxton@dravieu: ~/Documents/coding_project/coding_project

```
lauxton@dravieu:~$ cd Documents/ASMIS/stelios_code
lauxton@dravieu:~/Documents/ASMIS/stelios_code$ source playground_env/bin/activate
(playground_env) lauxton@dravieu:~/Documents/ASMIS/stelios_code$ cd
(playground_env) lauxton@dravieu:~$ cd Documents/coding_project/coding_project
(playground_env) lauxton@dravieu:~/Documents/coding_project/coding_project$ python3 main.py
~~User Registration~~

Please enter a username: Jenkins

Please enter a password: T4i2a1p8!

Password accepted!

Password hash: b'$2b$12$SJahLY8ajo8/kXnMNJ9e0Oe4j/rDHpj6kV.ErsnajBUuPPHdzK1oe'

Thank you for registering!
        Your username: Jenkins
        Your password: T4i2a1p8!

~~User Login~~

Please enter your username: Jenkins

Please enter your password: T4i2a1p8!

Your OTP is:  285616
What is your OTP code?  285616
        Unsuccessful. Resending code.

Your OTP is:  423086
What is your OTP code?  423086

Successful login
(playground_env) lauxton@dravieu:~/Documents/coding_project/coding_project$
```

create_user_testcode.py

```python
class CreateUsername():
    '''Create a username within a set of parameters'''
    def __init__(self, username):
        '''Initialize username '''
        self.username = username


    def userLength(self):
        '''Check the length of the username'''
        if len(self.username)>5 and len(self.username)<=16:
            print("Success!")
        else:
            print("\t- Username needs to be between 5 and 16 characters long")

username = CreateUsername('Jenkins')
username.userLength()
```

```
Success!
[Finished in 12ms]
```

Line 16, Column 22 Spaces: 4 Python

create_user_testcode.py

```python
class CreateUsername():
    '''Create a username within a set of parameters'''
    def __init__(self, username):
        '''Initialize username '''
        self.username = username


    def userLength(self):
        '''Check the length of the username'''
        if len(self.username)>5 and len(self.username)<=16:
            print("Success!")
        else:
            print("\t- Username needs to be between 5 and 16 characters long")

username = CreateUsername('sj %')
username.userLength()
```

```
    - Username needs to be between 5 and 16 characters long
[Finished in 13ms]
```

Line 15, Column 32 Spaces: 4 Python

create_user_testcode.py

```python
class CreateUsername():
    '''Create a username within a set of parameters'''
    def __init__(self, username):
        '''Initialize username '''
        self.username = username


    def userSpaces(self):
        '''Check if username has unauthorized spaces'''
        text = self.username
        count = 0
        for char in text:
            if char == ' ':
                count = count+1
        if count == 0:
            print("Success!")
        else:
            print("\t- Username should not have any spaces")

username = CreateUsername('Jenkins')
username.userSpaces()
```

```
Success!
[Finished in 11ms]
```

Line 20, Column 35; Build finished Spaces: 4 Python

create_user_testcode.py

```python
class CreateUsername():
    '''Create a username within a set of parameters'''
    def __init__(self, username):
        '''Initialize username '''
        self.username = username


    def userSpaces(self):
        '''Check if username has unauthorized spaces'''
        text = self.username
        count = 0
        for char in text:
            if char == ' ':
                count = count+1
        if count == 0:
            print("Success!")
        else:
            print("\t- Username should not have any spaces")

username = CreateUsername('sj %')
username.userSpaces()
```

```
    - Username should not have any spaces
[Finished in 12ms]
```

Line 20, Column 32; Build finished

Spaces: 4 Python

create_user_testcode.py

```python
class CreateUsername():
    '''Create a username within a set of parameters'''
    def __init__(self, username):
        '''Initialize username '''
        self.username = username


    def userSpecSym(self):
        '''Check if username has unauthorized special symbols'''
        specialsymb = (
            '!', '@', '#', '$', '&', '*','?',"'",'"', "/", "\",'_', '-', '{', '}',
            '[',']','+','=','^','%', '.',',',':',';','~','`','<','>',
        )
        if not any(char in specialsymb for char in self.username):
            print("Success!")
        else:
            print("\t- Username should not have any special characters")

username = CreateUsername('Jenkins')
username.userSpecSym()
```

```
Success!
[Finished in 12ms]
```

Line 20, Column 21; Build finished

Spaces: 4 Python

create_user_testcode.py

```python
class CreateUsername():
    '''Create a username within a set of parameters'''
    def __init__(self, username):
        '''Initialize username '''
        self.username = username


    def userSpecSym(self):
        '''Check if username has unauthorized special symbols'''
        specialsymb = (
            '!', '@', '#', '$', '&', '*','?',"'",'"', "/", "\"",' ', '-', '{', '}',
            '[',']','+','=','^','%', '.',',',':',';','~','`','<','>',
        )
        if not any(char in specialsymb for char in self.username):
            print("Success!")
        else:
            print("\t- Username should not have any special characters")

username = CreateUsername('sj %')
username.userSpecSym()
```

```
    - Username should not have any special characters
[Finished in 10ms]
```

Line 20, Column 21; Build finished Spaces: 4 Python

create_user_testcode.py

```python
class CreateUsername():
    '''Create a username within a set of parameters'''
    def __init__(self, username):
        '''Initialize username '''
        self.username = username


    def successUserLength(self):
        '''Return value if username length is within range'''
        if len(self.username)>5 and len(self.username)<=16:
            return(int(1))
        else:
            return(int(0))

username = CreateUsername('Jenkins')
username.successUserLength()

print(username.successUserLength())
```

```
1
[Finished in 14ms]
```

Line 15, Column 35; Build finished

Spaces: 4 Python

create_user_testcode.py

```python
class CreateUsername():
    '''Create a username within a set of parameters'''
    def __init__(self, username):
        '''Initialize username '''
        self.username = username


    def successUserLength(self):
        '''Return value if username length is within range'''
        if len(self.username)>5 and len(self.username)<=16:
            return(int(1))
        else:
            return(int(0))

username = CreateUsername('sj %')
username.successUserLength()

print(username.successUserLength())
```

```
0
[Finished in 13ms]
```

Line 15, Column 32; Build finished

Spaces: 4 Python

create_user_testcode.py

```python
class CreateUsername():
    '''Create a username within a set of parameters'''
    def __init__(self, username):
        '''Initialize username '''
        self.username = username


    def successUserSpaces(self):
        '''Return value if username has no spaces'''
        text = self.username
        count = 0
        for char in text:
            if char == ' ':
                count = count+1
        if count == 0:
            return 1
        else:
            return 0

username = CreateUsername('Jenkins')
username.successUserSpaces()

print(username.successUserSpaces())
```

```
1
[Finished in 13ms]
```

Line 20, Column 35; Build finished

Spaces: 4 Python

create_user_testcode.py

```python
class CreateUsername():
    '''Create a username within a set of parameters'''
    def __init__(self, username):
        '''Initialize username '''
        self.username = username


    def successUserSpaces(self):
        '''Return value if username has no spaces'''
        text = self.username
        count = 0
        for char in text:
            if char == ' ':
                count = count+1
        if count == 0:
            return 1
        else:
            return 0

username = CreateUsername('sj %')
username.successUserSpaces()

print(username.successUserSpaces())
```

```
0
[Finished in 11ms]
```

Line 20, Column 32; Build finished

Spaces: 4 Python

create_user_testcode.py

```python
class CreateUsername():
    '''Create a username within a set of parameters'''
    def __init__(self, username):
        '''Initialize username '''
        self.username = username


    def successUserSpecSym(self):
        '''Return value if username has no special symbols'''
        specialsymb = (
            '!', '@', '#', '$', '&', '*','?',"'",'"', "/", "\",' ', '-', '{', '}',
            '[',']','+','=','^','%', '.',',',':',';','~','`','<','>',
        )
        if not any(char in specialsymb for char in self.username):
            return 1
        else:
            return 0

username = CreateUsername('Jenkins')
username.successUserSpecSym()

print(username.successUserSpecSym())
```

```
1
[Finished in 11ms]
```

Line 19, Column 35; Build finished

Spaces: 4 Python

create_user_testcode.py

```python
class CreateUsername():
    '''Create a username within a set of parameters'''
    def __init__(self, username):
        '''Initialize username '''
        self.username = username


    def successUserSpecSym(self):
        '''Return value if username has no special symbols'''
        specialsymb = (
            '!', '@', '#', '$', '&', '*','?',"'",'"', "/", "\"",'_', '-', '{', '}',
            '[',']','+','=','^','%', '.',',',':',';','~','`','<','>',
        )
        if not any(char in specialsymb for char in self.username):
            return 1
        else:
            return 0

username = CreateUsername('sj %')
username.successUserSpecSym()

print(username.successUserSpecSym())
```

```
0
[Finished in 11ms]
```

Line 19, Column 32; Build finished Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def passLength(self):
        '''Check length of password'''
        if len(self.password)>8 and len(self.password)<=16:
            print("Success!")
        else:
            print("\t- Password needs to be between 8 and 16 characters long")

password = CreatePassword('T4i2a1p8!')
password.passLength()
```

```
Success!
[Finished in 15ms]
```

Line 15, Column 22 Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def passLength(self):
        '''Check length of password'''
        if len(self.password)>8 and len(self.password)<=16:
            print("Success!")
        else:
            print("\t- Password needs to be between 8 and 16 characters long")

password = CreatePassword('jk ^')
password.passLength()
```

```
    - Password needs to be between 8 and 16 characters long
[Finished in 11ms]
```

Line 14, Column 32; Build finished Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def passUpper(self):
        '''Check if password has uppercase character(s)'''
        if not any(char.isupper() for char in self.password):
            print('\t- Password should have at least one uppercase character')
        else:
            print("Success!")

password = CreatePassword('T4i2a1p8!')
password.passUpper()
```

```
Success!
[Finished in 12ms]
```

Line 15, Column 19; Build finished

Spaces: 4

Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def passUpper(self):
        '''Check if password has uppercase character(s)'''
        if not any(char.isupper() for char in self.password):
            print('\t- Password should have at least one uppercase character')
        else:
            print("Success!")

password = CreatePassword('jk ^')
password.passUpper()
```

```
    - Password should have at least one uppercase character
[Finished in 11ms]
```

Line 15, Column 19; Build finished Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def passDigit(self):
        '''Check if password has digit(s)'''
        if not any(char.isdigit() for char in self.password):
            print('\t- Password should have at least one number')
        else:
            print("Success!")

password = CreatePassword('T4i2a1p8!')
password.passDigit()
```

```
Success!
[Finished in 12ms]
```

Line 14, Column 37; Build finished

Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def passDigit(self):
        '''Check if password has digit(s)'''
        if not any(char.isdigit() for char in self.password):
            print('\t- Password should have at least one number')
        else:
            print("Success!")

password = CreatePassword('jk ^')
password.passDigit()
```

```
    - Password should have at least one number
[Finished in 13ms]
```

Line 14, Column 32; Build finished

Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def passSpaces(self):
        '''Check if password has no spaces'''
        text = self.password
        count = 0
        for char in text:
            if char == ' ':
                count = count+1
        if count == 0:
            print("Success!")
        else:
            print("\t- Password should not have any spaces")

password = CreatePassword('T4i2a1p8!')
password.passSpaces()
```

```
Success!
[Finished in 13ms]
```

Line 19, Column 37; Build finished Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def passSpaces(self):
        '''Check if password has no spaces'''
        text = self.password
        count = 0
        for char in text:
            if char == ' ':
                count = count+1
        if count == 0:
            print("Success!")
        else:
            print("\t- Password should not have any spaces")

password = CreatePassword('jk ^')
password.passSpaces()
```

```
    - Password should not have any spaces
[Finished in 12ms]
```

Line 19, Column 32; Build finished Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def passSpecSym(self):
        '''Check if password has special symbol(s)'''
        specialsymb = ('!', '@', '#', '$', '&', '*')
        if not any(char in specialsymb for char in self.password):
            print(
                '\t- Password should have at least one of the following symbols:'
                '\n\t  ! @ # $ % & *'
            )
        else:
            print("Success!")

password = CreatePassword('T4i2a1p8!')
password.passSpecSym()
```

```
Success!
[Finished in 13ms]
```

Line 18, Column 37; Build finished Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def passSpecSym(self):
        '''Check if password has special symbol(s)'''
        specialsymb = ('!', '@', '#', '$', '&', '*')
        if not any(char in specialsymb for char in self.password):
            print(
                '\t- Password should have at least one of the following symbols:'
                '\n\t  ! @ # $ % & *'
            )
        else:
            print("Success!")

password = CreatePassword('jk ^')
password.passSpecSym()
```

```
    - Password should have at least one of the following symbols:
      ! @ # $ % & *
[Finished in 12ms]
```

Line 18, Column 32; Build finished Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def successPassLength(self):
        '''Return value if password length accepted'''
        if len(self.password)>8 and len(self.password)<=16:
            return 1
        else:
            return 0

password = CreatePassword('T4i2a1p8!')
password.successPassLength()

print(password.successPassLength())
```

```
1
[Finished in 12ms]
```

Line 14, Column 37; Build finished Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def successPassLength(self):
        '''Return value if password length accepted'''
        if len(self.password)>8 and len(self.password)<=16:
            return 1
        else:
            return 0

password = CreatePassword('jk ^')
password.successPassLength()

print(password.successPassLength())
```

```
0
[Finished in 13ms]
```

Line 14, Column 32; Build finished

Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def successPassUpper(self):
        '''Return value if password has uppercase character(s)'''
        if not any(char.isupper() for char in self.password):
            return 0
        else:
            return 1

password = CreatePassword('T4i2a1p8!')
password.successPassUpper()

print(password.successPassUpper())
```

```
1
[Finished in 12ms]
```

Line 14, Column 37; Build finished Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def successPassUpper(self):
        '''Return value if password has uppercase character(s)'''
        if not any(char.isupper() for char in self.password):
            return 0
        else:
            return 1

password = CreatePassword('jk ^')
password.successPassUpper()

print(password.successPassUpper())
```

```
0
[Finished in 10ms]
```

Line 14, Column 32; Build finished

Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def successPassDigit(self):
        '''Return value if password has digit(s)'''
        if not any(char.isdigit() for char in self.password):
            return 0
        else:
            return 1

password = CreatePassword('T4i2a1p8!')
password.successPassDigit()

print(password.successPassDigit())
```

```
1
[Finished in 12ms]
```

Line 14, Column 37; Build finished Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def successPassDigit(self):
        '''Return value if password has digit(s)'''
        if not any(char.isdigit() for char in self.password):
            return 0
        else:
            return 1

password = CreatePassword('jk ^')
password.successPassDigit()

print(password.successPassDigit())
```

```
0
[Finished in 12ms]
```

Line 14, Column 32; Build finished

Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def successPassSpaces(self):
        '''Return value if password has no spaces'''
        text = self.password
        count = 0
        for char in text:
            if char == ' ':
                count = count+1
        if count == 0:
            return 1
        else:
            return 0

password = CreatePassword('T4i2a1p8!')
password.successPassSpaces()

print(password.successPassSpaces())
```

```
1
[Finished in 10ms]
```

Line 19, Column 37; Build finished Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def successPassSpaces(self):
        '''Return value if password has no spaces'''
        text = self.password
        count = 0
        for char in text:
            if char == ' ':
                count = count+1
        if count == 0:
            return 1
        else:
            return 0

password = CreatePassword('jk ^')
password.successPassSpaces()

print(password.successPassSpaces())
```

```
0
[Finished in 12ms]
```

Line 19, Column 32; Build finished

Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def successPassSpecSym(self):
        '''Return value if password has special symbol(s)'''
        specialsymb = ('!', '@', '#', '$', '&', '*')
        if not any(char in specialsymb for char in self.password):
            return 0
        else:
            return 1

password = CreatePassword('T4i2a1p8!')
password.successPassSpecSym()

print(password.successPassSpecSym())
```

```
1
[Finished in 13ms]
```

Line 15, Column 37 Spaces: 4 Python

create_password_testcode.py

```python
class CreatePassword():
    '''Create a password within a set of parameters'''
    def __init__(self, password):
        '''Initialize password'''
        self.password = password

    def successPassSpecSym(self):
        '''Return value if password has special symbol(s)'''
        specialsymb = ('!', '@', '#', '$', '&', '*')
        if not any(char in specialsymb for char in self.password):
            return 0
        else:
            return 1

password = CreatePassword('jk ^')
password.successPassSpecSym()

print(password.successPassSpecSym())
```

```
0
[Finished in 11ms]
```

Line 15, Column 32; Build finished Spaces: 4 Python

create_password_testcode.py

```python
import bcrypt

class EncryptPass():
    '''Encrypt the accepted password'''
    def __init__(self, password):
        '''Initialize password to hash'''
        self.password = password

    def encryptPass(self):
        '''Hash password with bcrypt'''
        b_password = self.password.encode('ASCII')
        # Generates a unique value with every hash
        salt = bcrypt.gensalt()
        # Generates the hash value
        hashed = bcrypt.hashpw(b_password, salt)
        return(hashed)

password = EncryptPass('jk ^')
password.encryptPass()

print(password.encryptPass())
```

```
b'$2b$12$H8jOMez3UVenOoFYS77wQOpDUyLKfk3NXT/6Vfo9psXxBCX32qIKK'
[Finished in 374ms]
```

Line 21, Column 30 Spaces: 4 Python

login_username_testcode.py

```python
class VerifyUser():
    '''Verify username for login'''
    def __init__(self, username, crosscheck_name):
        '''Initialize username and cross-check variables'''
        self.username = username
        self.crosscheck_name = crosscheck_name


    def verifyUser(self):
        '''Verify entered username against cross-check'''
        username = self.username
        cross_check = self.crosscheck_name

        if username == cross_check:
            print("Success!")
        else:
            print("\tUsername is not correct.")

username =  'Jenkins'
crosscheck_name = 'Jenkins'

verify_username = VerifyUser(username, crosscheck_name)
verify_username.verifyUser()
```

```
Success!
[Finished in 12ms]
```

Line 22, Column 55 Spaces: 4 Python

login_username_testcode.py

```python
class VerifyUser():
    '''Verify username for login'''
    def __init__(self, username, crosscheck_name):
        '''Initialize username and cross-check variables'''
        self.username = username
        self.crosscheck_name = crosscheck_name


    def verifyUser(self):
        '''Verify entered username against cross-check'''
        username = self.username
        cross_check = self.crosscheck_name

        if username == cross_check:
            print("Success!")
        else:
            print("\tUsername is not correct.")

username =  'Cairo'
crosscheck_name = 'Jenkins'

verify_username = VerifyUser(username, crosscheck_name)
verify_username.verifyUser()
```

```
    Username is not correct.
[Finished in 12ms]
```

Line 19, Column 19; Build finished Spaces: 4 Python

login_username_testcode.py

```python
class VerifyUser():
    '''Verify username for login'''
    def __init__(self, username, crosscheck_name):
        '''Initialize username and cross-check variables'''
        self.username = username
        self.crosscheck_name = crosscheck_name


    def verifyUserSuccess(self):
        '''Assign value to successful username'''
        password = self.username
        cross_check = self.crosscheck_name

        if username == cross_check:
            return 1
        else:
            return 0

username =  'Jenkins'
crosscheck_name = 'Jenkins'

verify_username = VerifyUser(username, crosscheck_name)
verify_username.verifyUserSuccess()

print(verify_username.verifyUserSuccess())
```

```
1
[Finished in 12ms]
```

Line 19, Column 21; Build finished

Spaces: 4 Python

login_username_testcode.py

```python
class VerifyUser():
    '''Verify username for login'''
    def __init__(self, username, crosscheck_name):
        '''Initialize username and cross-check variables'''
        self.username = username
        self.crosscheck_name = crosscheck_name


    def verifyUserSuccess(self):
        '''Assign value to successful username'''
        password = self.username
        cross_check = self.crosscheck_name

        if username == cross_check:
            return 1
        else:
            return 0

username =  'Cairo'
crosscheck_name = 'Jenkins'

verify_username = VerifyUser(username, crosscheck_name)
verify_username.verifyUserSuccess()

print(verify_username.verifyUserSuccess())
```

```
0
[Finished in 11ms]
```

Line 19, Column 19; Build finished

Spaces: 4 Python

login_password_testcode.py

```python
import bcrypt

class EncryptPass():
    '''Encrypt the accepted password'''
    def __init__(self, password):
        '''Initialize password to hash'''
        self.password = password

    def encryptPass(self):
        '''Hash password with bcrypt'''
        b_password = self.password.encode('ASCII')
        # Generates a unique value with every hash
        salt = bcrypt.gensalt()
        # Generates the hash value
        hashed = bcrypt.hashpw(b_password, salt)
        return(hashed)


class VerifyPass():
    '''Verify password for login'''
    def __init__(self, password, hashed):
        '''Initialize password and hashed password variables'''
        self.password = password
        self.hashed = hashed

    def verifyPass(self):
        '''Verify entered password against hashed password'''
        password = self.password
        hashed = self.hashed
        if bcrypt.checkpw(password, hashed):
            print("Success!")
        else:
            print("\tPassword does not match.")


password = EncryptPass('T4i2a1p8!')
hashed = password.encryptPass()

password = 'T4i2a1p8!'
password = password.encode('ASCII')

verify_password = VerifyPass(password, hashed)
verify_password.verifyPass()

```

```
Success!
[Finished in 373ms]
```

Line 18, Column 1 Spaces: 4 Python

login_password_testcode.py

```python
import bcrypt

class EncryptPass():
    '''Encrypt the accepted password'''
    def __init__(self, password):
        '''Initialize password to hash'''
        self.password = password

    def encryptPass(self):
        '''Hash password with bcrypt'''
        b_password = self.password.encode('ASCII')
        # Generates a unique value with every hash
        salt = bcrypt.gensalt()
        # Generates the hash value
        hashed = bcrypt.hashpw(b_password, salt)
        return(hashed)


class VerifyPass():
    '''Verify password for login'''
    def __init__(self, password, hashed):
        '''Initialize password and hashed password variables'''
        self.password = password
        self.hashed = hashed

    def verifyPass(self):
        '''Verify entered password against hashed password'''
        password = self.password
        hashed = self.hashed
        if bcrypt.checkpw(password, hashed):
            print("Success!")
        else:
            print("\tPassword does not match.")


password = EncryptPass('jk ^')
hashed = password.encryptPass()

password = 'T4i2a1p8!'
password = password.encode('ASCII')

verify_password = VerifyPass(password, hashed)
verify_password.verifyPass()

```

```
    Password does not match.
[Finished in 382ms]
```

Line 36, Column 29 Spaces: 4 Python

login_password_testcode.py

```
        self.password = password

    def encryptPass(self):
        '''Hash password with bcrypt'''
        b_password = self.password.encode('ASCII')
        # Generates a unique value with every hash
        salt = bcrypt.gensalt()
        # Generates the hash value
        hashed = bcrypt.hashpw(b_password, salt)
        return(hashed)


class VerifyPass():
    '''Verify password for login'''
    def __init__(self, password, hashed):
        '''Initialize password and hashed password variables'''
        self.password = password
        self.hashed = hashed


    def verifyPassSuccess(self):
        '''Assign value to successful password'''
        password = self.password
        hashed = self.hashed
        if bcrypt.checkpw(password, hashed):
            return 1
        else:
            return 0


password = EncryptPass('T4i2a1p8!')
hashed = password.encryptPass()

password = 'T4i2a1p8!'
password = password.encode('ASCII')

verify_password = VerifyPass(password, hashed)
verify_password.verifyPassSuccess()
print(verify_password.verifyPassSuccess())
```

```
1
[Finished in 556ms]
```

Line 37, Column 34; Build finished Spaces: 4 Python

login_password_testcode.py

```
        self.password = password

    def encryptPass(self):
        '''Hash password with bcrypt'''
        b_password = self.password.encode('ASCII')
        # Generates a unique value with every hash
        salt = bcrypt.gensalt()
        # Generates the hash value
        hashed = bcrypt.hashpw(b_password, salt)
        return(hashed)


class VerifyPass():
    '''Verify password for login'''
    def __init__(self, password, hashed):
        '''Initialize password and hashed password variables'''
        self.password = password
        self.hashed = hashed


    def verifyPassSuccess(self):
        '''Assign value to successful password'''
        password = self.password
        hashed = self.hashed
        if bcrypt.checkpw(password, hashed):
            return 1
        else:
            return 0


password = EncryptPass('jk ^')
hashed = password.encryptPass()

password = 'T4i2a1p8!'
password = password.encode('ASCII')

verify_password = VerifyPass(password, hashed)
verify_password.verifyPassSuccess()
print(verify_password.verifyPassSuccess())
```

```
0
[Finished in 553ms]
```

Line 37, Column 29; Build finished Spaces: 4 Python

login_password_testcode.py

```
1  import pyotp
2
3  # OTP code generator
4  totp = pyotp.TOTP('base32secret3232')
5  print("\nYour OTP is: ", totp.now())
6
7  # User input
8  otp_code = input("What is your OTP code?\t")
9
10 if totp.verify(otp_code)==True:
11         # Simulated login
12         print("\nSuccessful login")
13 else:
14     print("\nUnsuccessful login")
```

```
  File "/home/lauxton/Documents/ASMIS/stelios_code/co
  <module>
    import pyotp
ModuleNotFoundError: No module named 'pyotp'
[Finished in 64ms with exit code 1]
[cmd: ['python3', '-u', '/home/lauxton/Documents/ASMI
login_password_testcode.py']]
[dir: /home/lauxton/Documents/ASMIS/stelios_code/codi
[path: /home/lauxton/.local/bin:/usr/local/sbin:/usr/
games:/usr/local/games:/snap/bin:/snap/bin]
```

Line 12, Column 36

```
lauxton@dravieu:~$ cd Documents/ASMIS/stelios_code
lauxton@dravieu:~/Documents/ASMIS/stelios_code$ source playground_env/bi
n/activate
(playground_env) lauxton@dravieu:~/Documents/ASMIS/stelios_code$ cd codi
ng_project
(playground_env) lauxton@dravieu:~/Documents/ASMIS/stelios_code/coding_p
roject$ python3 login_password_testcode.py

Your OTP is:  436504
What is your OTP code?  436504

Successful login
(playground_env) lauxton@dravieu:~/Documents/ASMIS/stelios_code/coding_p
roject$ python3 login_password_testcode.py

Your OTP is:  436504
What is your OTP code?  679283

Unsuccessful login
(playground_env) lauxton@dravieu:~/Documents/ASMIS/stelios_code/coding_p
roject$ python3 login_password_testcode.py

Your OTP is:  436504
What is your OTP code?  436504

Unsuccessful login
(playground_env) lauxton@dravieu:~/Documents/ASMIS/stelios_code/coding_p
roject$
```