**Python.py answers**

**1.** **import math**

**class Circle:**

**def \_\_init\_\_(self, radius):**

**self.radius = radius**

**def getArea(self):**

**return math.pi \* self.radius \*\* 2**

**def getCircumference(self):**

**return 2 \* math.pi \* self.radius**

**# Example :**

**radius = 10**

**circle = Circle(radius)**

**print("Area of the circle:", circle.getArea())**

**print("Circumference of the circle:", circle.getCircumference())**

2.

def add\_product(product\_name, price, quantity):

"""

Create a dictionary object for the product with the given parameters.

Args:

product\_name (string): representing the name of the product.

price (float): representing the price of the product.

quantity (int): representing the initial quantity of the product in stock.

Returns:

dict: Dictionary object representing the product.

"""

product = {

'product\_name': product\_name,

'price': price,

'quantity': quantity

}

return product

def update\_price(product, new\_price):

"""

Update the price of the product in the dictionary.

Args:

product (dictionary): representing the product dictionary object.

new\_price (float): representing the updated price of the product.

Returns:

dict: Modified dictionary representing the product with updated price.

"""

product['price'] = new\_price

return product

def update\_quantity(product, quantity\_change):

"""

Update the quantity of the product in the dictionary.

Args:

product (dictionary): representing the product dictionary object.

quantity\_change (int): representing the change in quantity of the product

(positive for addition, negative for subtraction).

Returns:

dict: Modified dictionary representing the product with updated quantity.

"""

product['quantity'] += quantity\_change

return product

# Example usage:

product = add\_product("Laptop", 999.99, 10)

print("Initial product:", product)

product = update\_price(product, 899.99)

print("Product after price update:", product)

product = update\_quantity(product, -5)

print("Product after quantity update:", product)

3.

import datetime

class BankAccount:

def \_\_init\_\_(self, account\_number, customer\_name, balance=0):

self.account\_number = account\_number

self.customer\_name = customer\_name

self.balance = balance

self.date\_of\_opening = datetime.datetime.now()

def deposit(self, amount):

self.balance += amount

return amount

def withdraw(self, amount):

if self.balance >= amount:

self.balance <= amount

return amount

else:

return "Insufficient balance"

def check\_balance(self):

print("Current balance:", self.balance)

def customer\_details(self):

print("Customer Name:", self.customer\_name)

print("Account Number:", self.account\_number)

print("Date of Account Opening:", self.date\_of\_opening)

print("Balance:", self.balance)

# Create an instance of the BankAccount class

account = BankAccount("7770182980705", "abel watoka")

# Perform operations

print("Deposit Amount:", account.deposit(1000))

print("Withdrawn Amount:", account.withdraw(500))

print("Withdrawn Amount:", account.withdraw(700))

account.check\_balance()

print("Account Details:")

account.customer\_details()

4.

class Book:

def \_\_init\_\_(self, title, author, publication\_year):

self.title = title

self.author = author

self.publication\_year = publication\_year

self.borrowed = False

def borrow\_book(self):

self.borrowed = True

def return\_book(self):

self.borrowed = False

def display\_info(self):

print("Title:", self.title)

print("Author:", self.author)

print("Publication Year:", self.publication\_year)

print("Borrowed:", "Yes" if self.borrowed else "No")

class LibraryMember:

def \_\_init\_\_(self, name, member\_id):

self.name = name

self.member\_id = member\_id

def display\_info(self):

print("Name:", self.name)

print("Member ID:", self.member\_id)

5.

class Book:

def \_\_init\_\_(self, title, author, publication\_year):

self.title = title

self.author = author

self.publication\_year = publication\_year

self.borrowed = False

def borrow\_book(self):

self.borrowed = True

def return\_book(self):

self.borrowed = False

def display\_info(self):

print("Title:", self.title)

print("Author:", self.author)

print("Publication Year:", self.publication\_year)

print("Borrowed:", "Yes" if self.borrowed else "No")

class LibraryMember:

def \_\_init\_\_(self, member\_id, name):

self.member\_id = member\_id

self.name = name

self.borrowed\_books = []

def borrow\_book(self, book):

self.borrowed\_books.append(book)

book.borrow\_book()

def return\_book(self, book):

if book in self.borrowed\_books:

self.borrowed\_books.remove(book)

book.return\_book()

def display\_info(self):

print("Member ID:", self.member\_id)

print("Name:", self.name)

print("Borrowed Books:")

for book in self.borrowed\_books:

book.display\_info()

print()