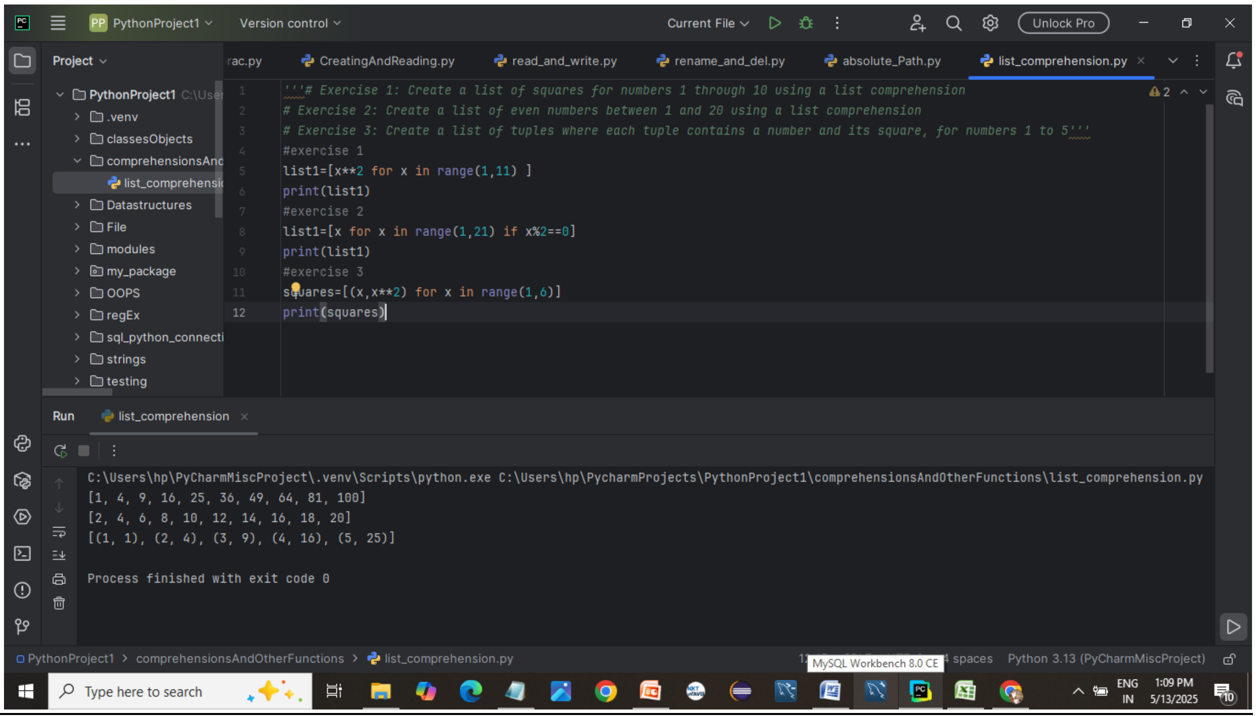
Hands-on ComprehensionsAndOtherFunctions

```
'''# Exercise 1: Create a list of squares for numbers 1 through 10 using a
list comprehension
# Exercise 2: Create a list of even numbers between 1 and 20 using a list
comprehension
# Exercise 3: Create a list of tuples where each tuple contains a number and
its square, for numbers 1 to 5'''
#exercise 1
list1=[x**2 for x in range(1,11) ]
print(list1)
#exercise 2
list1=[x for x in range(1,21) if x%2==0]
print(list1)
#exercise 3
squares=[(x,x**2) for x in range(1,6)]
print(squares)
```

Output

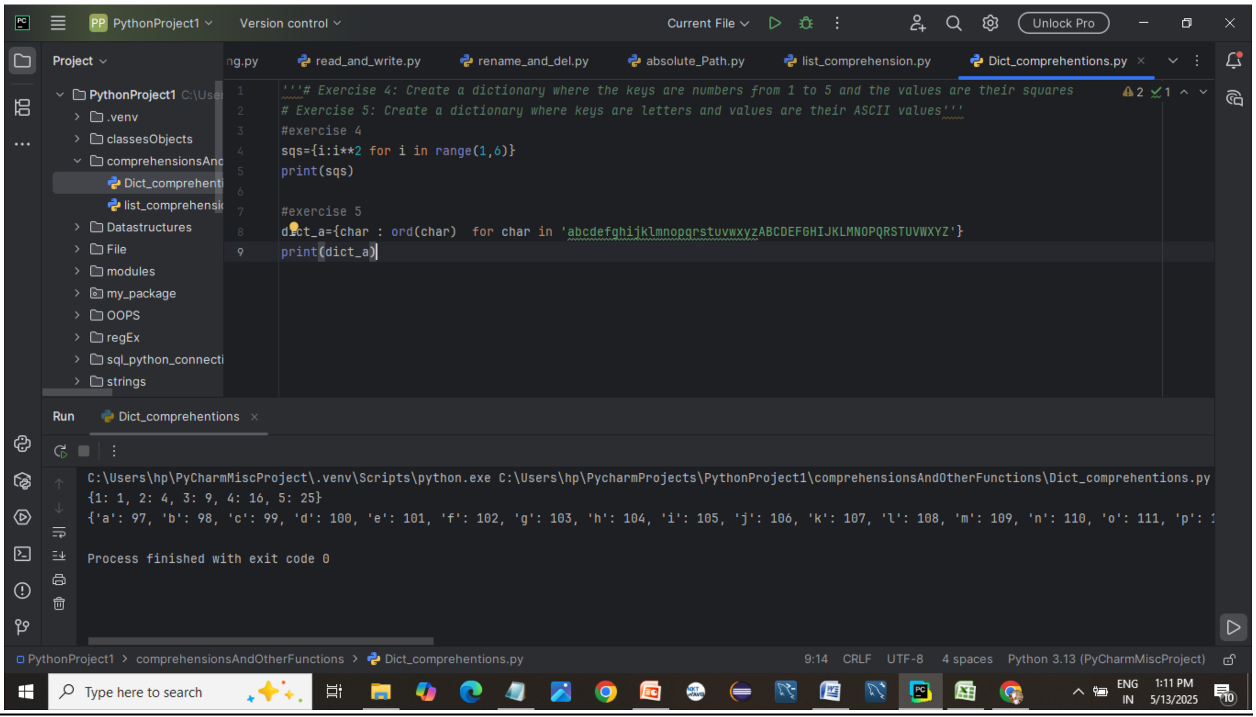


```
'''# Exercise 1: Create a list of squares for numbers 1 through 10 using a list comprehension
# Exercise 2: Create a list of even numbers between 1 and 20 using a list comprehension
# Exercise 3: Create a list of tuples where each tuple contains a number and its square, for numbers 1 to 5'''
#exercise 1
list1=[x**2 for x in range(1,11) ]
print(list1)
#exercise 2
list1=[x for x in range(1,21) if x%2==0]
print(list1)
#exercise 3
squares=[(x,x**2) for x in range(1,6)]
print(squares)
```

```
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PycharmProjects\PythonProject1\comprehensionsAndOtherFunctions\list_comprehension.py
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
[(1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]

Process finished with exit code 0
```

```python
'''# Exercise 4: Create a dictionary where the keys are numbers from 1 to 5 and the values are their squares
# Exercise 5: Create a dictionary where keys are letters and values are their ASCII values'''
#exercise 4
sqs={i:i**2 for i in range(1,6)}
print(sqs)

#exercise 5
dict_a={char : ord(char)  for char in 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'}
print(dict_a)
```

<u>Output)</u>

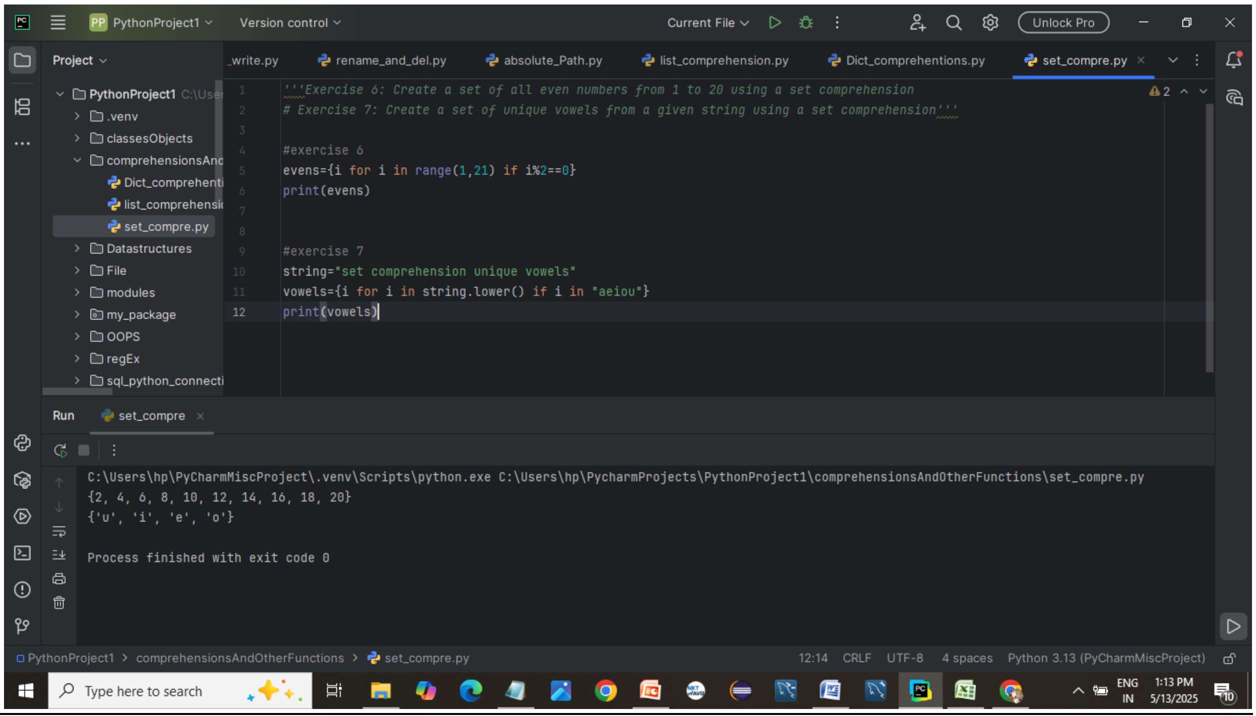


```python
'''# Exercise 4: Create a dictionary where the keys are numbers from 1 to 5 and the values are their squares
# Exercise 5: Create a dictionary where keys are letters and values are their ASCII values'''
#exercise 4
sqs={i:i**2 for i in range(1,6)}
print(sqs)

#exercise 5
dict_a={char : ord(char)  for char in 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'}
print(dict_a)
```

```
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PycharmProjects\PythonProject1\comprehensionsAndOtherFunctions\Dict_comprehentions.py
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
{'a': 97, 'b': 98, 'c': 99, 'd': 100, 'e': 101, 'f': 102, 'g': 103, 'h': 104, 'i': 105, 'j': 106, 'k': 107, 'l': 108, 'm': 109, 'n': 110, 'o': 111, 'p': 1

Process finished with exit code 0
```

```python
'''Exercise 6: Create a set of all even numbers from 1 to 20 using a set
comprehension
# Exercise 7: Create a set of unique vowels from a given string using a set
comprehension'''

#exercise 6
evens={i for i in range(1,21) if i%2==0}
print(evens)


#exercise 7
string="set comprehension unique vowels"
vowels={i for i in string.lower() if i in "aeiou"}
print(vowels)
```

Output)

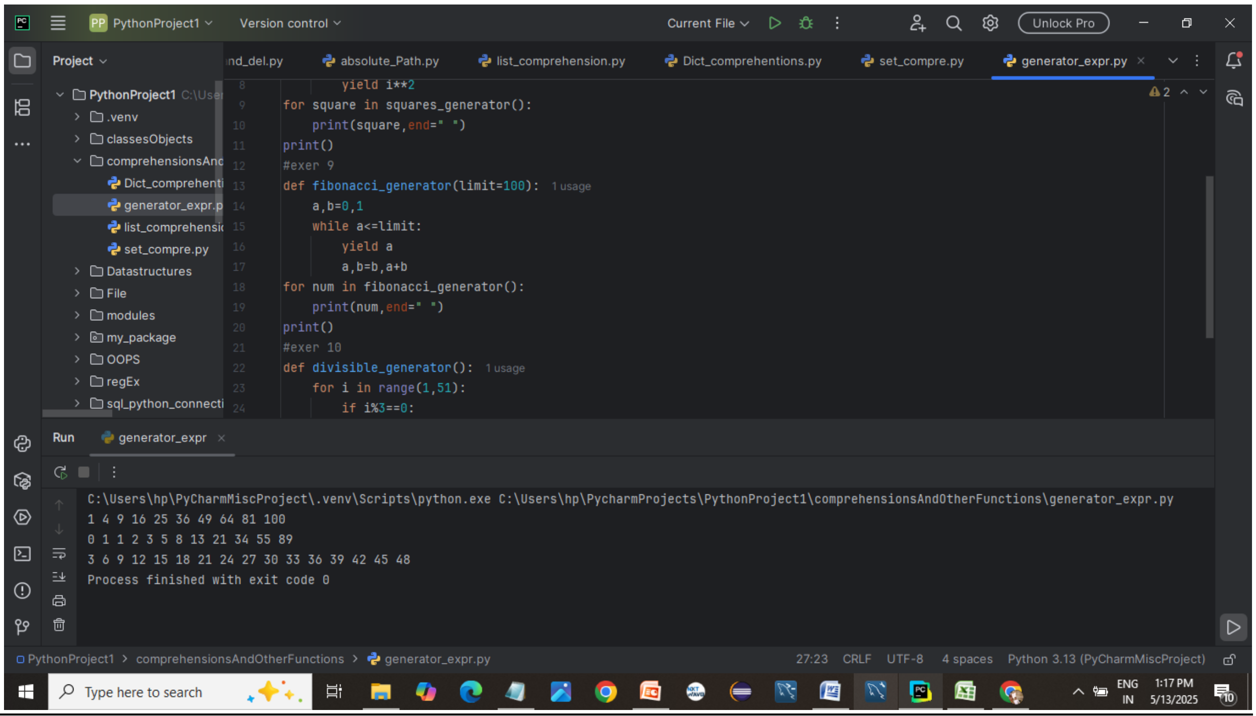


```python
'''Exercise 6: Create a set of all even numbers from 1 to 20 using a set comprehension
# Exercise 7: Create a set of unique vowels from a given string using a set comprehension'''

#exercise 6
evens={i for i in range(1,21) if i%2==0}
print(evens)


#exercise 7
string="set comprehension unique vowels"
vowels={i for i in string.lower() if i in "aeiou"}
print(vowels)
```

```
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PycharmProjects\PythonProject1\comprehensionsAndOtherFunctions\set_compre.py
{2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
{'u', 'i', 'e', 'o'}

Process finished with exit code 0
```

```python
'''# Exercise 8: Create a generator that generates squares for numbers 1 to 10
# Exercise 9: Use a generator to generate Fibonacci numbers up to 100
# Exercise 10: Write a generator that yields numbers that are divisible by 3
from 1 to 50'''
#exer 8
def squares_generator():

    for i in range(1,11):
        yield i**2
for square in squares_generator():
    print(square)
#exer 9
def fibonacci_generator(limit=100):
    a,b=0,1
    while a<=limit:
        yield a
        a,b=b,a+b
for num in fibonacci_generator():
    print(num)
#exer 10
def divisible_generator():
    for i in range(1,51):
        if i%3==0:
            yield i
for num in divisible_generator():
    print(num)
```

<u>Output)</u>

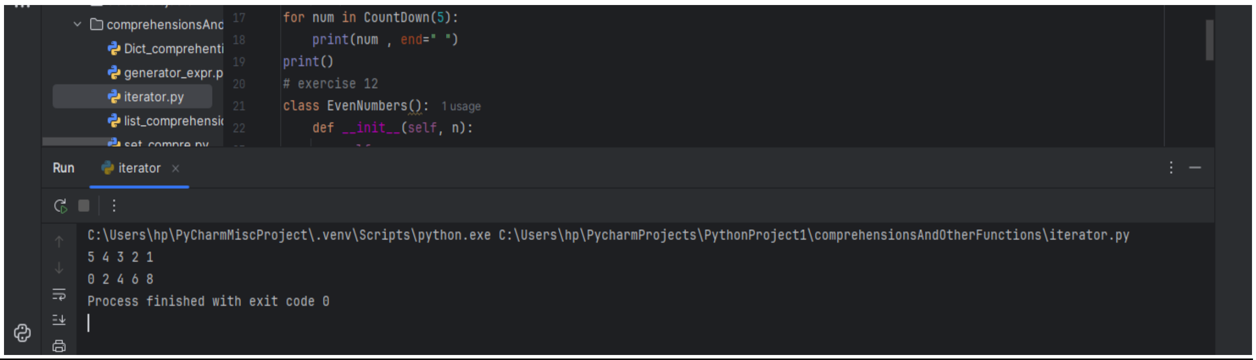


```python
        yield i**2
for square in squares_generator():
    print(square,end=" ")
print()
#exer 9
def fibonacci_generator(limit=100):  1 usage
    a,b=0,1
    while a<=limit:
        yield a
        a,b=b,a+b
for num in fibonacci_generator():
    print(num,end=" ")
print()
#exer 10
def divisible_generator():  1 usage
    for i in range(1,51):
        if i%3==0:
```

```
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PycharmProjects\PythonProject1\comprehensionsAndOtherFunctions\generator_expr.py
1 4 9 16 25 36 49 64 81 100
0 1 1 2 3 5 8 13 21 34 55 89
3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48
Process finished with exit code 0
```

```python
'''Exercise 11: Create a class `CountDown` that takes an integer `n` and
iterates down from n to 1
# Exercise 12: Implement an iterator that yields the first `n` even
numbers'''
# exer11
class CountDown():
    def __init__(self, n):
        self.n = n
    def __iter__(self):
        self.current = self.n
        return self
    def __next__(self):
        if self.current < 1:
            raise StopIteration
        else:
            val = self.current
            self.current -= 1
            return val
for num in CountDown(5):
    print(num , end=" ")
print()
# exercise 12
class EvenNumbers():
    def __init__(self, n):
        self.n = n
    def __iter__(self):
        self.current = 0
        self.count = 0
        return self
    def __next__(self):
        if self.count >= self.n:
            raise StopIteration
        val = self.current
        self.current += 2
        self.count += 1
        return val
for val in EvenNumbers(5):
    print(val,end=" ")
```

Output)

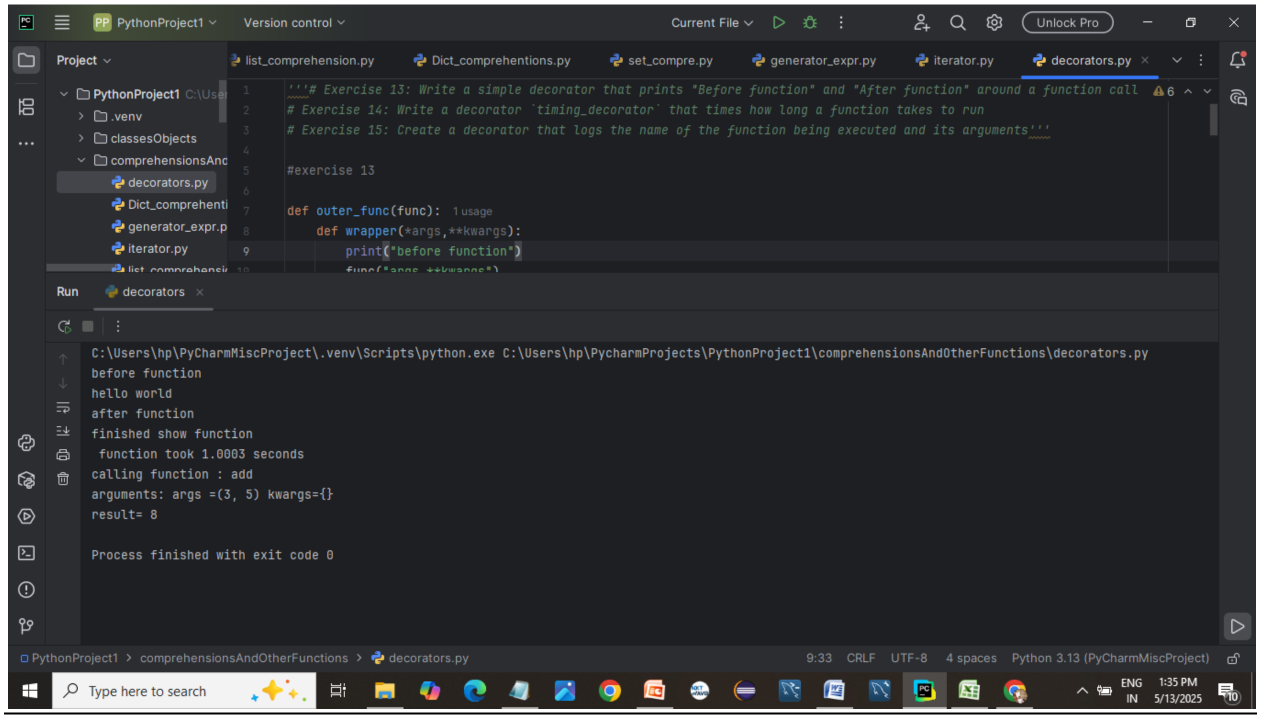


```python
for num in CountDown(5):
    print(num , end=" ")
print()
# exercise 12
class EvenNumbers():  1 usage
    def __init__(self, n):
```

```
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PycharmProjects\PythonProject1\comprehensionsAndOtherFunctions\iterator.py
5 4 3 2 1
0 2 4 6 8
Process finished with exit code 0
```

```python
'''# Exercise 13: Write a simple decorator that prints "Before function" and
"After function" around a function call
# Exercise 14: Write a decorator `timing_decorator` that times how long a
function takes to run
# Exercise 15: Create a decorator that logs the name of the function being
executed and its arguments'''

#exercise 13

def outer_func(func):
    def wrapper(*args,**kwargs):
        print("before function")
        func("args,**kwargs")
        print("after function")
    return wrapper
@outer_func
def display(*args,**kwargs):
    print("hello world")

display()

#exercise 14
import time

def timing_decorator(func):
    def wrapper(*args,**kwargs):
        start = time.time()
        func(*args,**kwargs)
        end=time.time()
        print(f" function took {end - start:.4f} seconds")
    return wrapper

@timing_decorator
def show():
    time.sleep(1)
    print("finished show function")

show()

#exercise 15
def logging_decorator(func):
    def wrapper(*args,**kwargs):
        print(f"calling function : {func.__name__}")
        print(f"arguments: args ={args} kwargs={kwargs}")
        return func(*args,**kwargs)
    return wrapper


@logging_decorator
def add(a,b):
    return a+b

result=add(3,5)
print("result=",result)
```

<u>Output</u>

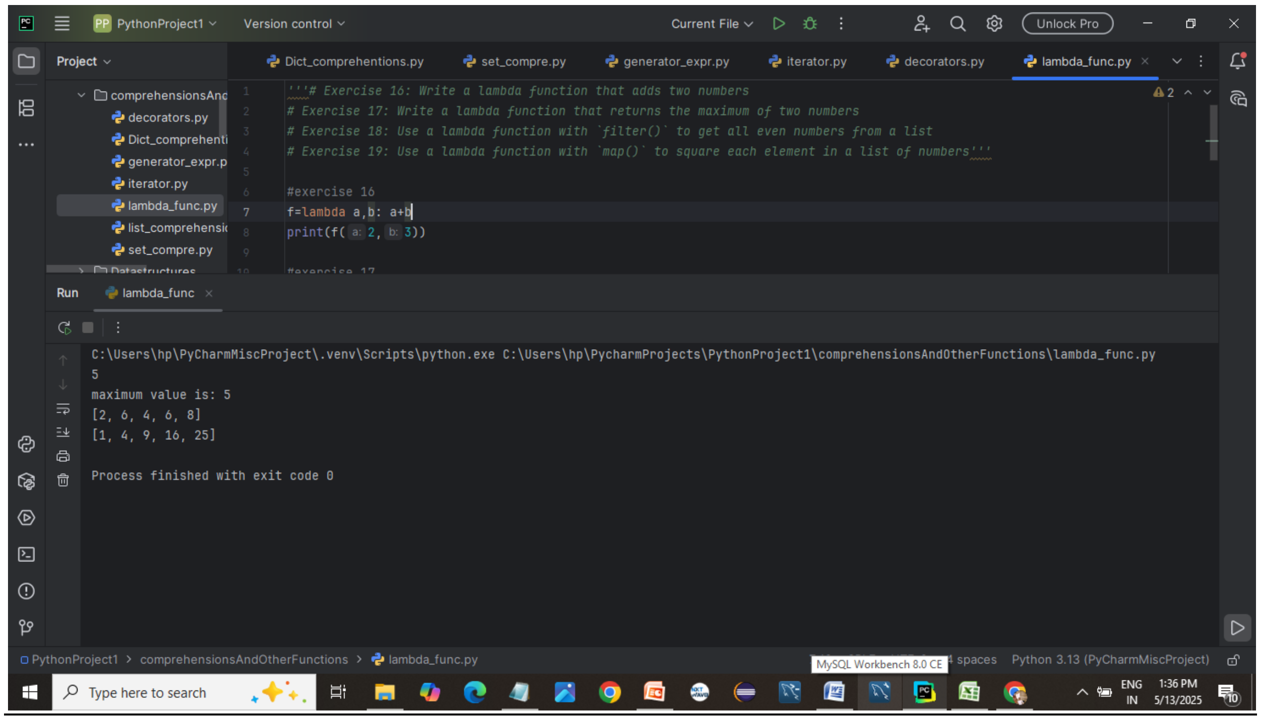


```python
'''# Exercise 13: Write a simple decorator that prints "Before function" and "After function" around a function call
# Exercise 14: Write a decorator `timing_decorator` that times how long a function takes to run
# Exercise 15: Create a decorator that logs the name of the function being executed and its arguments'''

#exercise 13

def outer_func(func):
    def wrapper(*args,**kwargs):
        print("before function")
```

```
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PycharmProjects\PythonProject1\comprehensionsAndOtherFunctions\decorators.py
before function
hello world
after function
finished show function
 function took 1.0003 seconds
calling function : add
arguments: args =(3, 5) kwargs={}
result= 8

Process finished with exit code 0
```

```python
'''# Exercise 16: Write a lambda function that adds two numbers
# Exercise 17: Write a lambda function that returns the maximum of two
numbers
# Exercise 18: Use a lambda function with `filter()` to get all even numbers
from a list
# Exercise 19: Use a lambda function with `map()` to square each element in a
list of numbers'''

#exercise 16
f=lambda a,b: a+b
print(f(2,3))

#exercise 17
f=lambda a,b: a if a>b else b
print("maximum value is:",f(4,5))

#exercise 18

l=[1,2,3,5,6,4,6,7,8]
evens=list(filter(lambda x: x%2==0,l))
print(evens)

#exercise 19
l=[1,2,3,4,5]
squares=list(map(lambda x: x**2,l))
print(squares)
```

Output)

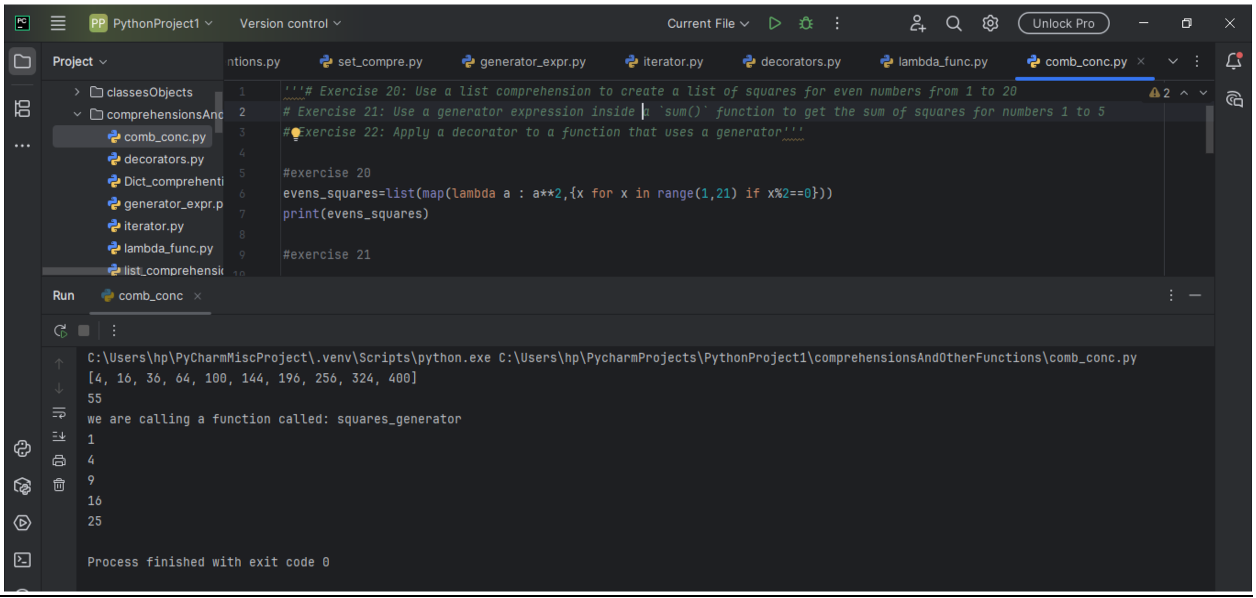


```python
'''# Exercise 16: Write a lambda function that adds two numbers
# Exercise 17: Write a lambda function that returns the maximum of two numbers
# Exercise 18: Use a lambda function with `filter()` to get all even numbers from a list
# Exercise 19: Use a lambda function with `map()` to square each element in a list of numbers'''

#exercise 16
f=lambda a,b: a+b
print(f( a: 2, b: 3))

#exercise 17
```

```
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PycharmProjects\PythonProject1\comprehensionsAndOtherFunctions\lambda_func.py
5
maximum value is: 5
[2, 6, 4, 6, 8]
[1, 4, 9, 16, 25]

Process finished with exit code 0
```

```python
'''# Exercise 20: Use a list comprehension to create a list of squares for
even numbers from 1 to 20
# Exercise 21: Use a generator expression inside a `sum()` function to get
the sum of squares for numbers 1 to 5
# Exercise 22: Apply a decorator to a function that uses a generator'''

#exercise 20
evens_squares=list(map(lambda a : a**2,{x for x in range(1,21) if x%2==0}))
print(evens_squares)

#exercise 21

total=sum(x**2 for x in range(1,6))
print(total)

#exercise 22

def outer_func(func):
    def wrapper(*args,**kwargs):
        print("we are calling a function called:",func.__name__)
        return func(*args,**kwargs)
    return wrapper

@outer_func
def squares_generator(n):
    for i in range(1,n+1):
        yield i**2

for square in squares_generator(5):
    print(square)
```

<u>Output)</u>



```python
'''# Exercise 20: Use a list comprehension to create a list of squares for even numbers from 1 to 20
# Exercise 21: Use a generator expression inside a `sum()` function to get the sum of squares for numbers 1 to 5
#Exercise 22: Apply a decorator to a function that uses a generator'''

#exercise 20
evens_squares=list(map(lambda a : a**2,{x for x in range(1,21) if x%2==0}))
print(evens_squares)

#exercise 21
```

```
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PycharmProjects\PythonProject1\comprehensionsAndOtherFunctions\comb_conc.py
[4, 16, 36, 64, 100, 144, 196, 256, 324, 400]
55
we are calling a function called: squares_generator
1
4
9
16
25

Process finished with exit code 0
```