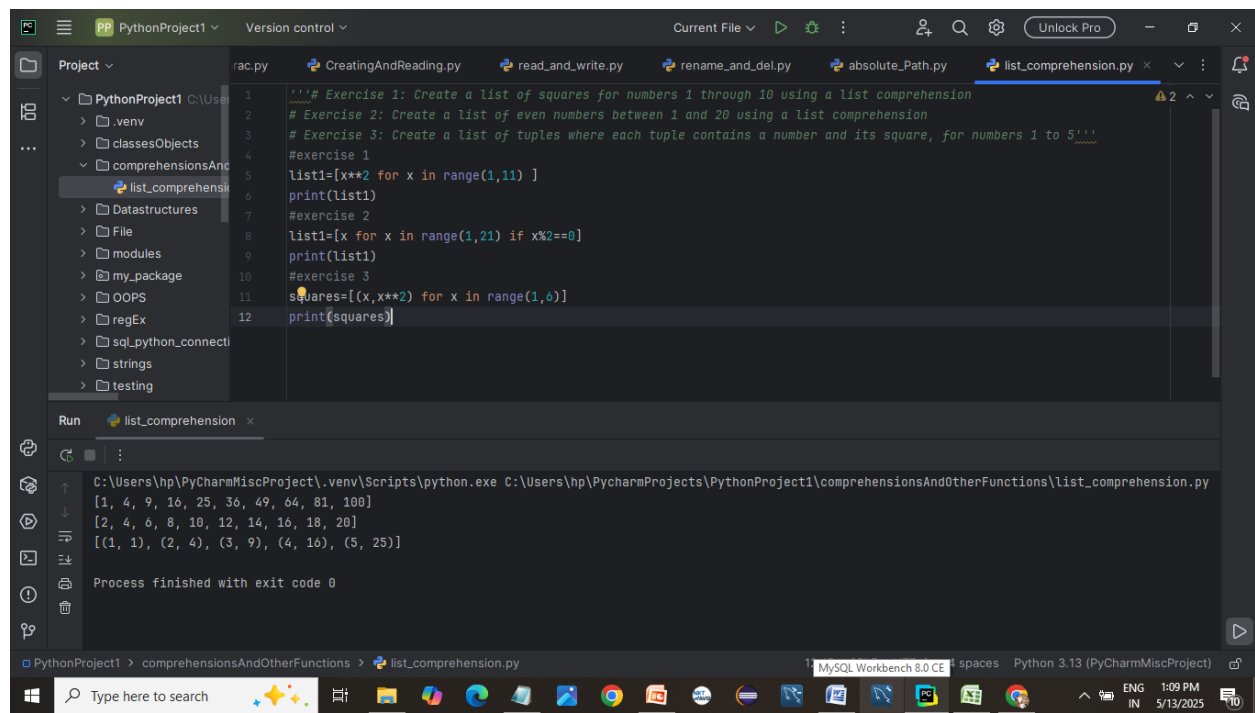


Hands-on ComprehensionsAndOtherFunctions

```
'''# 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



```
PythonProject1
rac.py
CreatingAndReading.py
read_and_write.py
rename_and_del.py
absolute_Path.py
list_comprehension.py

PythonProject1 C:\Users\hp\PyCharmProjects\PythonProject1
venv
classesObjects
comprehensionsAndOtherFunctions
list_comprehension.py
DataStructures
File
modules
my_package
OOPS
regEx
sql_python_connect
strings
testing

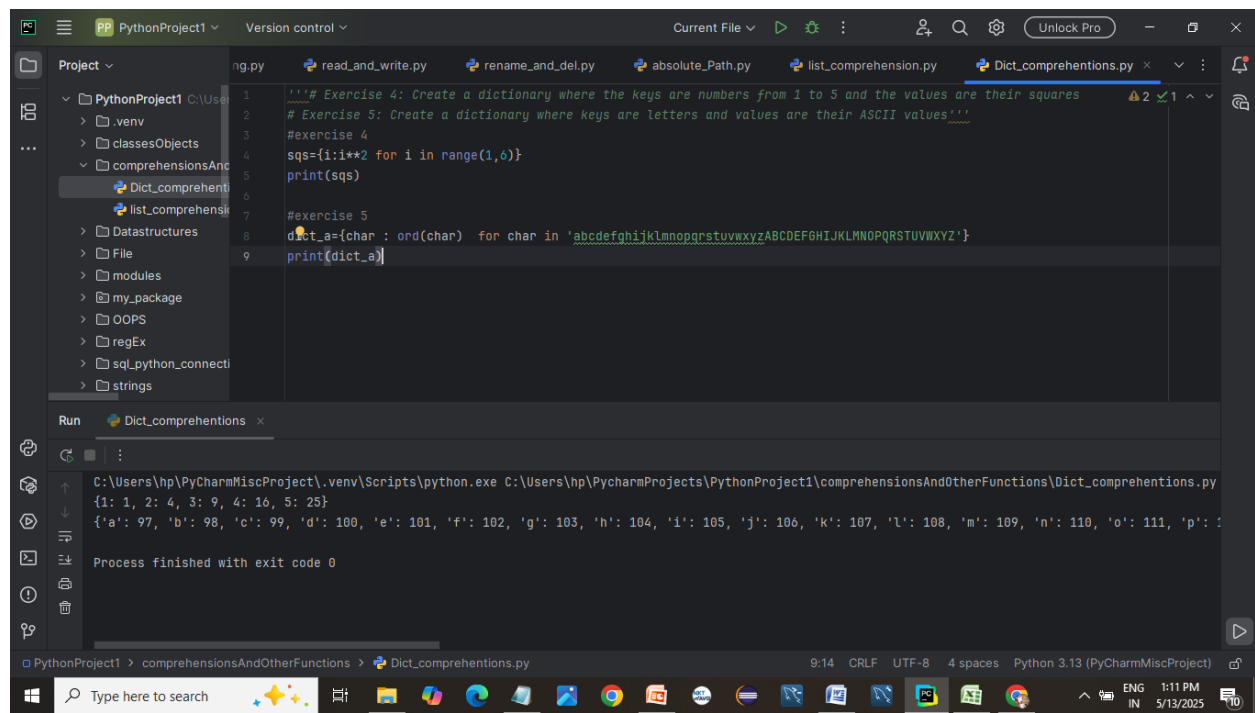
Run list_comprehension
C:\Users\hp\PyCharmMiscProject\venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\comprehensionsAndOtherFunctions\list_comprehension.py
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
[(1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]

Process finished with exit code 0
```

```
'''# 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)
```

Output)



```
PythonProject1
Version control
Current File
Unlock Pro

Project
PythonProject1
venv
classesObjects
comprehensionsAndOtherFunctions
Dict_comprehensions.py
list_comprehensions.py
Datastructures
File
modules
my_package
OOPS
regEx
sql_python_connection
strings

ng.py
read_and_write.py
rename_and_del.py
absolute_Path.py
list_comprehension.py
Dict_comprehensions.py

1 '''# Exercise 4: Create a dictionary where the keys are numbers from 1 to 5 and the values are their squares
2 # Exercise 5: Create a dictionary where keys are letters and values are their ASCII values'''
3 #exercise 4
4 sqs={i:i**2 for i in range(1,6)}
5 print(sqs)
6
7 #exercise 5
8 dict_a={char : ord(char) for char in 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'}
9 print(dict_a)

Run Dict_comprehensions.py
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\comprehensionsAndOtherFunctions\Dict_comprehensions.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': 112, 'q': 113, 'r': 114, 's': 115, 't': 116, 'u': 117, 'v': 118, 'w': 119, 'x': 120, 'y': 121, 'z': 122, 'A': 65, 'B': 66, 'C': 67, 'D': 68, 'E': 69, 'F': 70, 'G': 71, 'H': 72, 'I': 73, 'J': 74, 'K': 75, 'L': 76, 'M': 77, 'N': 78, 'O': 79, 'P': 80, 'Q': 81, 'R': 82, 'S': 83, 'T': 84, 'U': 85, 'V': 86, 'W': 87, 'X': 88, 'Y': 89, 'Z': 90}
Process finished with exit code 0

PythonProject1 > comprehensionsAndOtherFunctions > Dict_comprehensions.py
9:14 CRLF UTF-8 4 spaces Python 3.13 (PyCharmMiscProject)
```

```
'''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)

The screenshot displays the PyCharm IDE interface. The top toolbar shows the 'Run' button (a green play icon). The main editor window shows the Python code from the previous block. Below the editor, the 'Run' console is visible, showing the command executed: `C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\comprehensionsAndOtherFunctions\set_compre.py`. The output of the script is displayed in the console: `{2, 4, 6, 8, 10, 12, 14, 16, 18, 20}` and `{'u', 'i', 'e', 'o'}`. At the bottom, a status bar indicates the file is `set_compre.py`, the encoding is `UTF-8`, and the Python version is `Python 3.13 (PyCharmMiscProject)`.

```

'''# 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)

```

Output)

The screenshot shows the PyCharm IDE with the file `generator_expr.py` open. The Run console at the bottom displays the output of the program:

```

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

```

```

'''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)

```

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

```

```

'''# 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()

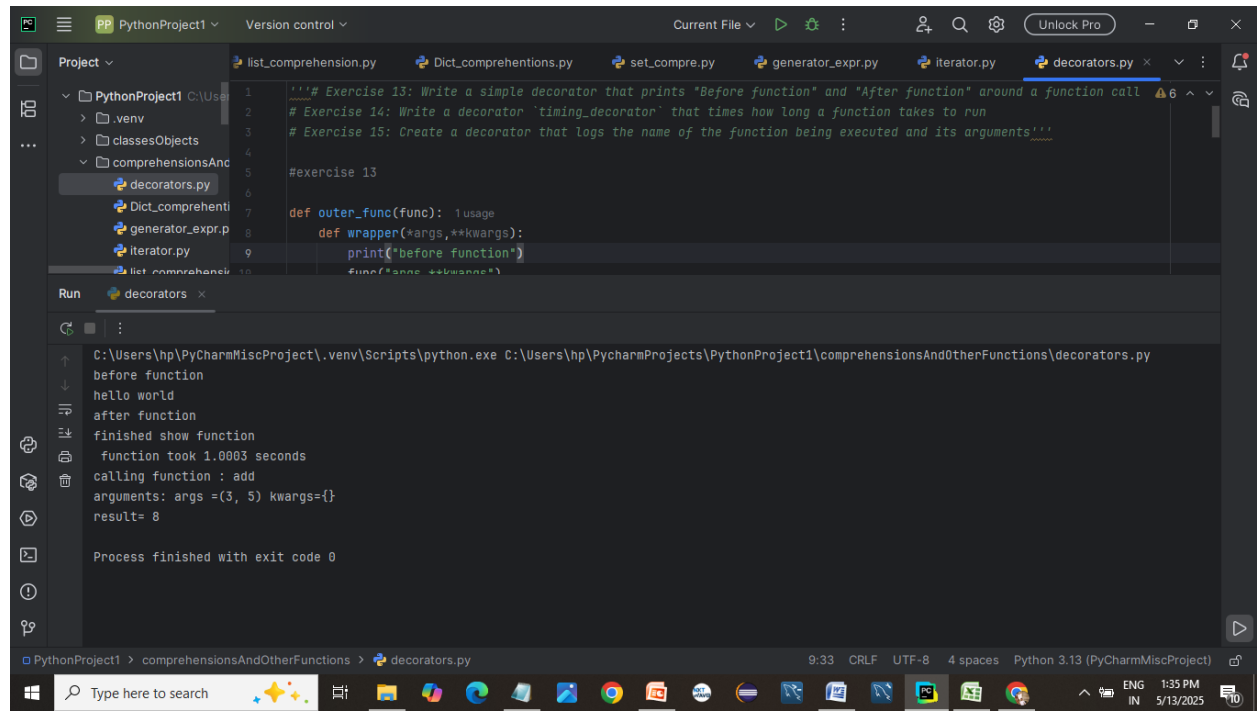
#exercise 15
def logging_decorator(func):
    def wrapper(*args,**kwargs):
        print(f"calling function : {func.__name__}")
        print(f"arguments: args ={args} kwargs={kwargs}")
        return func(*args,**kwargs)
    return wrapper

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

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

```

Output



The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for file operations, running, and debugging, along with a 'Unlock Pro' button. The left sidebar displays the project structure for 'PythonProject1', with 'comprehensionsAndOtherFunctions' expanded to show 'decorators.py'. The main editor window displays the code for 'decorators.py', which includes comments for Exercise 13, 14, and 15, and a decorator implementation. The 'Run' console at the bottom shows the execution output, including the function name, arguments, and execution time.

```
1  """# Exercise 13: Write a simple decorator that prints "Before function" and "After function" around a function call
2
3  # Exercise 14: Write a decorator "timing_decorator" that times how long a function takes to run
4  # Exercise 15: Create a decorator that logs the name of the function being executed and its arguments"""
5
6  #exercise 13
7
8  def outer_func(func):
9      def wrapper(*args,**kwargs):
10         print("before function")
11         func(*args,**kwargs)
```

Run decorators

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

PythonProject1 > comprehensionsAndOtherFunctions > decorators.py 9:33 CRLF UTF-8 4 spaces Python 3.13 (PyCharmMiscProject)

```

'''# 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)

The screenshot shows the PyCharm IDE interface. The top toolbar includes buttons for running and debugging. The 'Project' view on the left shows a file structure with several Python files. The 'Run' window at the bottom displays the output of the script:

```

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

```

The taskbar at the bottom shows the Windows operating system with various application icons and a system clock indicating 1:36 PM on 5/13/2025.


```

'''# 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)

```

Output)

```

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

```