1. Generators

1. Generator for the first 10 even numbers:

```
def even_numbers():
    for i in range(2, 21, 2):
        yield i

for num in even_numbers():
    print(num)
```

2. Generator for the Fibonacci sequence:

```
def fibonacci():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

fib_gen = fibonacci()
for _ in range(15):
    print(next(fib_gen))
```

3. Generator for words with more than 5 characters:

```
def long_words(sentence):
    for word in sentence.split():
        if len(word) > 5:
            yield word

text = "Python generators are very useful for memory efficiency"
for word in long_words(text):
    print(word)
```

4. Generator for squares of numbers from 1 to n:

```
def square_numbers(n):
    for i in range(1, n + 1):
        yield i ** 2

n = int(input("Enter the value of n: "))
for square in square_numbers(n):
    print(square)
```

5. Generator to read a file line by line:

```
def read_file(file_path):
    with open(file_path, 'r') as file:
        for line in file:
            yield line.strip()

for i, line in enumerate(read_file("sample.txt"), start=1):
    print(line)
    if i == 3:
        break
```

---

2. Decorators

1. Execution time decorator:

```
import time
def timing_decorator(func):
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        print(f"Execution time: {end_time - start_time:.4f} seconds")
        return result
    return wrapper

@timing_decorator
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

print(factorial(10))
```

2. Logging decorator:

```
def logging_decorator(func):
    def wrapper(*args, **kwargs):
        print(f"Arguments: {args}, {kwargs}")
        result = func(*args, **kwargs)
        print(f"Return Value: {result}")
        return result
    return wrapper
@logging_decorator
def multiply(a, b):
    return a * b

print(multiply(3, 5))
```

3. Authentication decorator:

```
def authenticated_user(func):
    def wrapper(username):
        if username == "admin":
            return func(username)
        else:
            print("Access denied!")
    return wrapper

@authenticated_user
def secure_function(username):
    print(f"Welcome, {username}!")

secure_function("admin")
secure_function("guest")
```

4. Uppercase result decorator:

```
def uppercase_decorator(func):
    def wrapper():
        return func().upper()
    return wrapper

@uppercase_decorator
def greeting():
    return "hello world"

print(greeting())
```

5. Custom message decorator:

```
def message_decorator(func):
    def wrapper(*args, **kwargs):
        print("Starting the function...")
        result = func(*args, **kwargs)
        print("Function execution completed!")
        return result
    return wrapper

@message_decorator
def calculate_sum(numbers):
    return sum(numbers)

print(calculate_sum([1, 2, 3, 4, 5]))
```

3. Regex

1. Validate email address:

```
import re

def is_valid_email(email):
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return bool(re.match(pattern, email))

print(is_valid_email("example@example.com"))
print(is_valid_email("invalid-email"))
```

2. Extract phone numbers:

```
import re

text = "Contact us at 123-456-7890 or 987-654-3210"
pattern = r'\d{3}-\d{3}-\d{4}'
phone_numbers = re.findall(pattern, text)
print(phone_numbers)
```

3. Find and replace words:

```
import re

text = "Python is great. Python is dynamic."
replaced_text = re.sub(r'Python', 'Django', text)
print(replaced_text)
```

4. Extract dates:

```
import re

text = "The events are scheduled for 01-01-2025 and 15-03-2025."
pattern = r'\b\d{2}-\d{2}-\d{4}\b'
dates = re.findall(pattern, text)
print(dates)
```

5. Check alphanumeric string:

```
import re

def is_alphanumeric(string):
    pattern = r'^[a-zA-Z0-9]+$'
    return bool(re.match(pattern, string))

print(is_alphanumeric("Python123"))
print(is_alphanumeric("Python@123"))
```