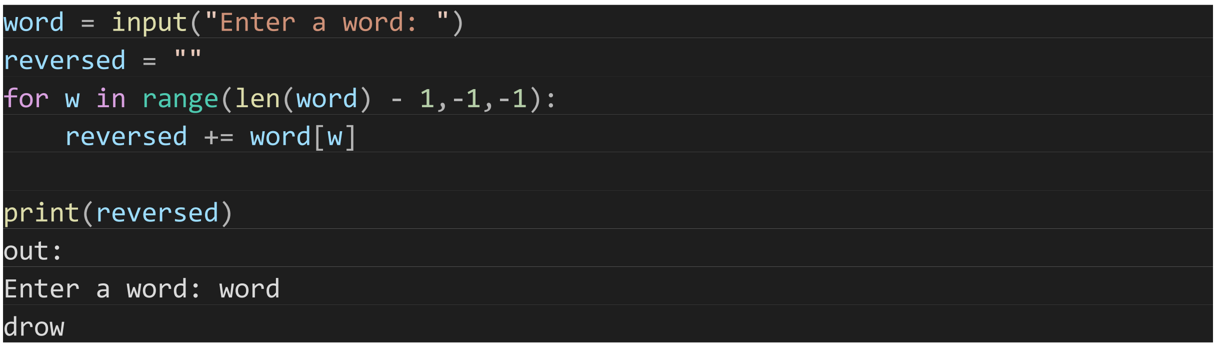
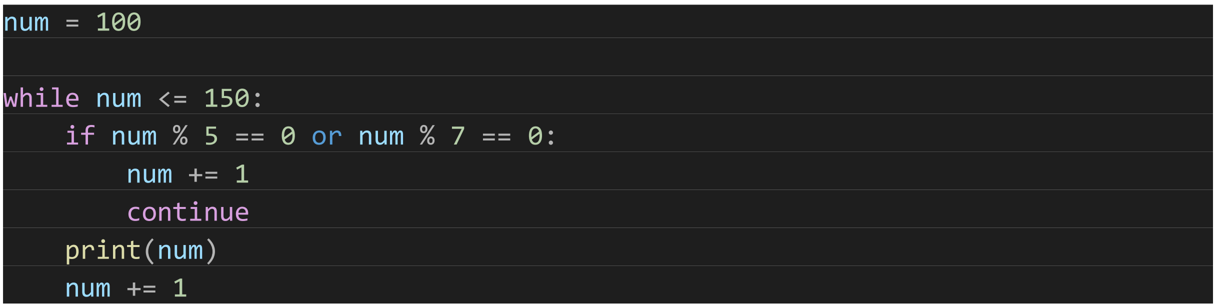
EIT, 306387, Gabriel Ćwiek, MAPT 2, lab 10, https://github.com/Astaree/MAPT_2

Ex 0:



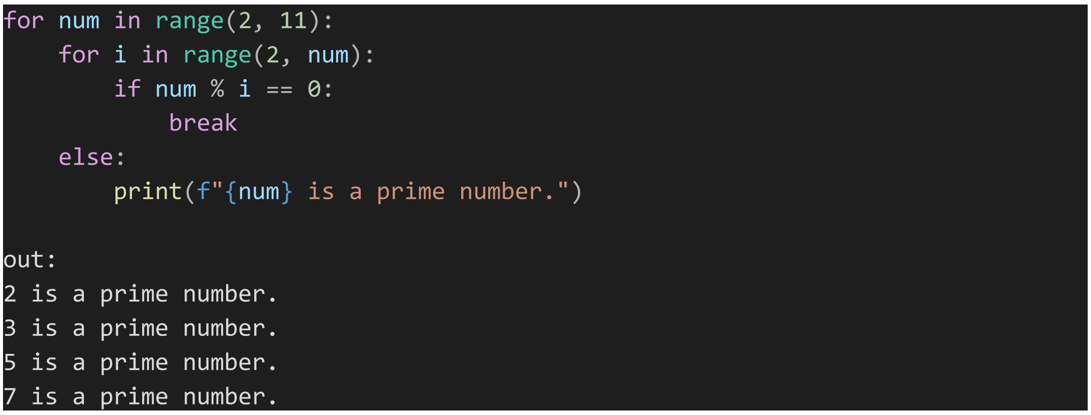
```
word = input("Enter a word: ")
reversed = ""
for w in range(len(word) - 1,-1,-1):
    reversed += word[w]

print(reversed)
out:
Enter a word: word
drow
```

Ex. 1:



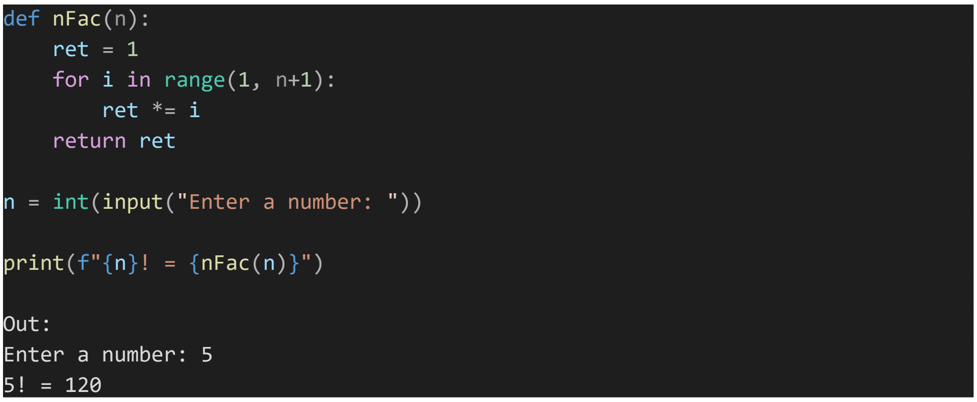
```
num = 100

while num <= 150:
    if num % 5 == 0 or num % 7 == 0:
        num += 1
        continue
    print(num)
    num += 1
```

Ex. 2:



```
for num in range(2, 11):
    for i in range(2, num):
        if num % i == 0:
            break
    else:
        print(f"{num} is a prime number.")

out:
2 is a prime number.
3 is a prime number.
5 is a prime number.
7 is a prime number.
```

Ex. 3:



```
def nFac(n):
    ret = 1
    for i in range(1, n+1):
        ret *= i
    return ret

n = int(input("Enter a number: "))

print(f"{n}! = {nFac(n)}")

Out:
Enter a number: 5
5! = 120
```

Ex. 4:

```
import numpy as np
from statistics import median

sequence = [1, 2]  # Starting elements

while len(sequence) < 50:
    n = len(sequence)
    numerator = sequence[n - 1] + sequence[n - 2]
    denominator = sequence[n - 1] - sequence[n - 2]
    element = numerator / denominator
    sequence.append(element)

# Calculate the sum and median of the sequence
sequence_sum = np.sum(sequence)
sequence_median = median(sequence)

print("Sequence:", sequence)
print("Sum:", sequence_sum)
print("Median:", sequence_median)
out (part of it):
Sequence: [1, 2, 3.0, 5.0, 4.0, -9.0, 0.38461538461538464, -
0.9180327868852458, 0.409486931268151, -0.3830797001829027, -
0.03331862588878006, -1.190522206943827, 1.0575847265498484, -
0.05913307699620272, -0.8940948611933468, 1.1416425951831126,
0.12160101157169452, -1.2384236359093732, 0.82117822379629, -0.20257292864986,
-0.5338023596708397, 0.007321476001719123, -0.972939739412442,
0.9850621938589587, 0.006191237220211825, -1.012649751590284,
0.9878465093410895, -0.012398544667934903, -0.9752089858015474,
1.0257549028100212, 0.02526078421312578, -1.050496617108658,
0.9530362808899332, -0.04864424053934021, -0.9028747400020698,
1.1138901984181908, 0.1046306658729451, -1.2073414468706132,
0.840498643444266, -0.1791364497459081, -0.6486263547767142,
1.7631109756626346, 0.4621086246912692, -1.710388608208503,
0.5745830027369351]

Sum: 5.34910859105643
Median: 0.01629113010742245
```

Ex. 5:

```python
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius
    def calculate_circumference(self):
        return 2 * math.pi * self.radius
    def calculate_area(self):
        return math.pi * self.radius**2

class EquilateralTriangle:
    def __init__(self, side_length):
        self.side_length = side_length
    def calculate_circumference(self):
        return 3 * self.side_length
    def calculate_area(self):
        return (math.sqrt(3) / 4) * self.side_length**2

# Example usage:
circle = Circle(5)
print("Circle Radius:", circle.radius)
print("Circle Circumference:", circle.calculate_circumference())
print("Circle Area:", circle.calculate_area())

triangle = EquilateralTriangle(7)
print("Triangle Side Length:", triangle.side_length)
print("Triangle Circumference:", triangle.calculate_circumference())
print("Triangle Area:", triangle.calculate_area())
```

Ex. 6:

```python
try:
    file = open("nonexistent_file.txt", "r")
    # Perform operations on the file
    file.close()
except FileNotFoundError:
    print("File not found. Please check the file name and path.")

try:
    result = 10 / 0
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
```

Ex. 7 – files are on git (https://github.com/Astaree/MAPT_2) so here is short ans:

a. Class Definitions: The class definitions for the figures (square, circle, triangle) are placed in separate files (square.py, circle.py, triangle.py).

b. Module Import: The modules created in step 1 are imported into the calculations.py file. This allows us to access the classes and their methods for performing calculations.

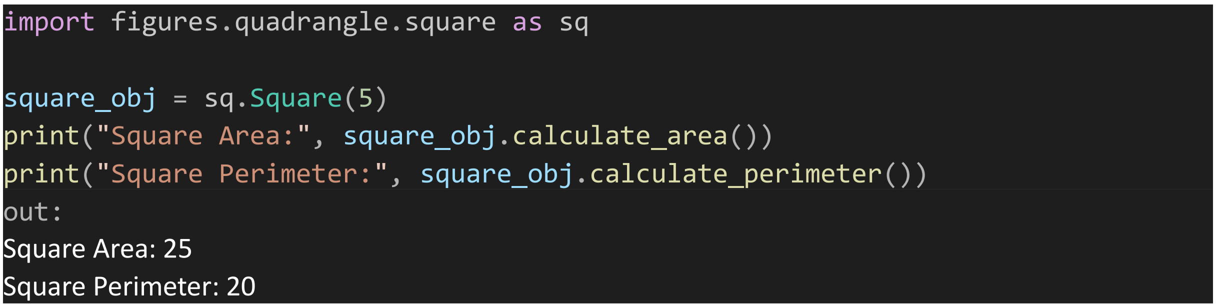
c. Object Creation and Calculation: Objects of the figure classes can be created in calculations.py, and the appropriate methods can be used to calculate and display the figure areas.

d. Package Structure: The "figures" directory is created, and sub-folders "quadrangles" and "triangles" are added. Empty "__init__.py" files are placed in both folders. This allows treating these directories as packages in Python.

e. Module Placement: The module file for the square class (square.py) is placed in the "quadrangles" folder to organize related files together.

f. calculations2.py: The file calculations2.py is created in the previous directory. It imports the square module from the "figures.quadrangles" package using the statement "import figures.quadrangles.square as sq".

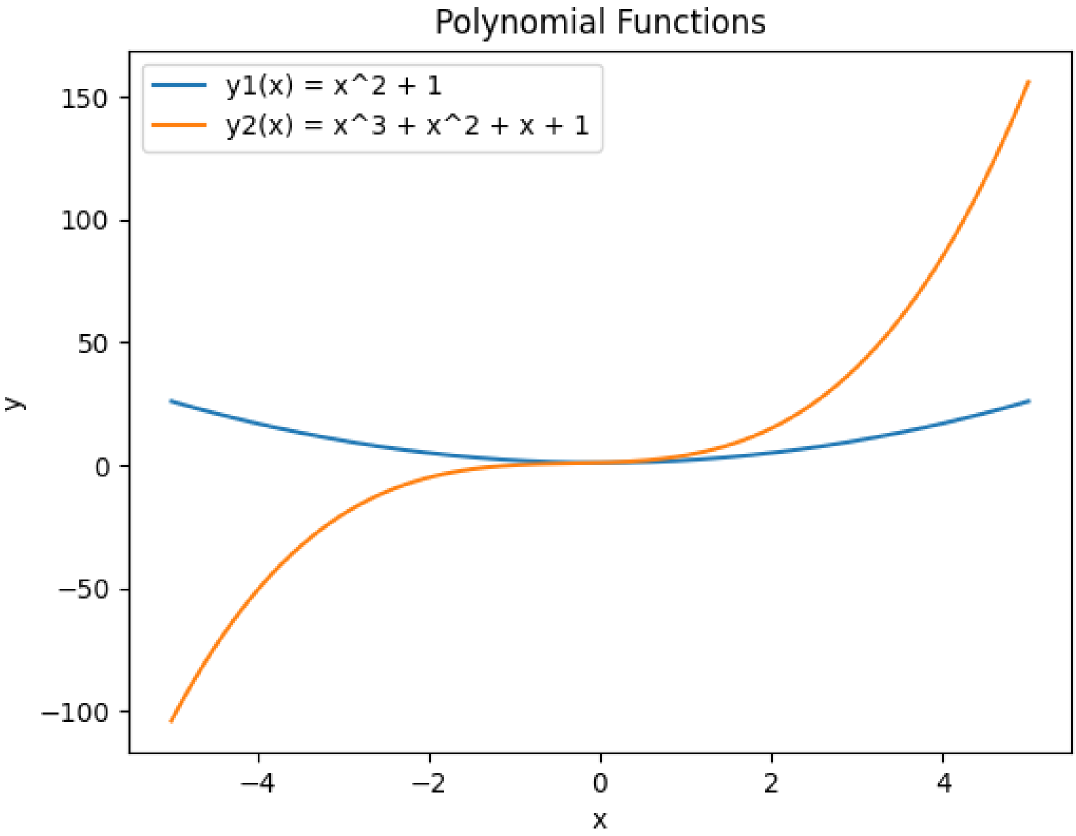
g. Conclusion: By organizing class definitions into separate files, creating packages and modules, and importing them appropriately, we can efficiently manage and reuse code related to different figures. This modular approach enhances code organization, reusability, and maintainability.



```
import figures.quadrangle.square as sq

square_obj = sq.Square(5)
print("Square Area:", square_obj.calculate_area())
print("Square Perimeter:", square_obj.calculate_perimeter())
out:
Square Area: 25
Square Perimeter: 20
```

Ex 8a – using list:

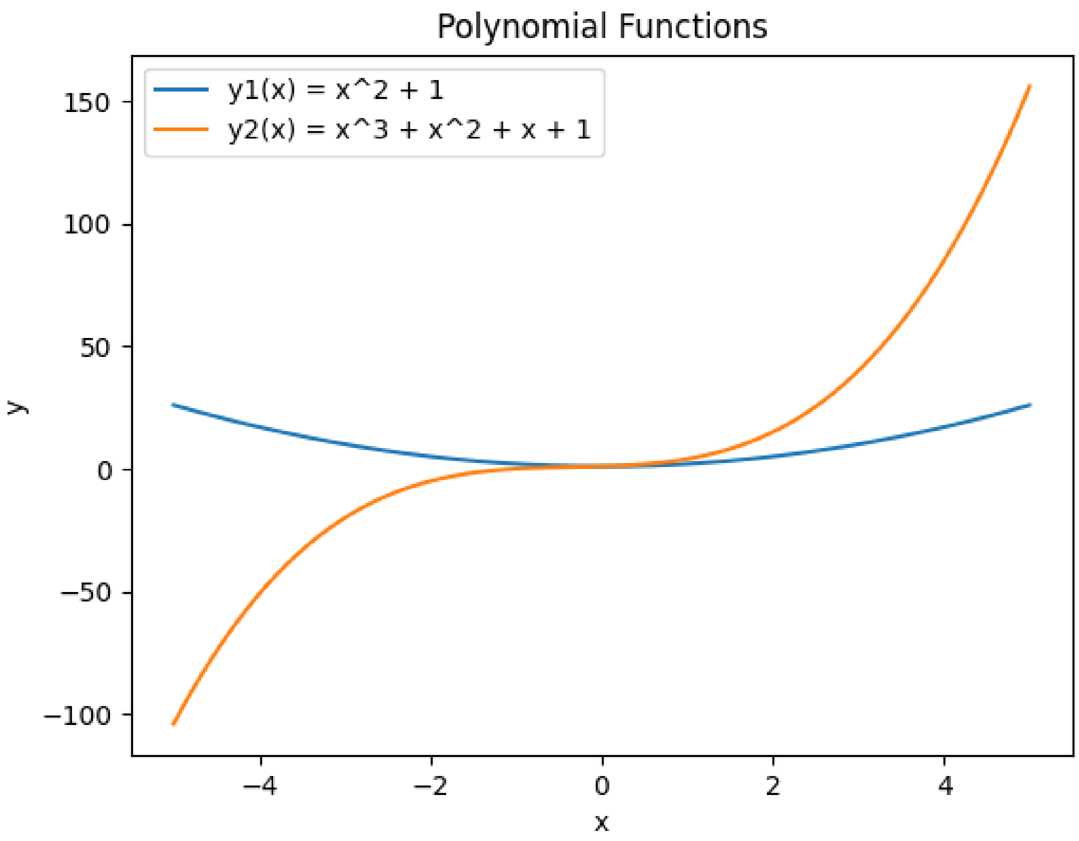
```
import matplotlib.pyplot as plt

# Calculate the range
x_range = range(-500, 501)

# Calculate x values based on the range
x = [xi * 0.01 for xi in x_range]

# Calculate y1 and y2 values for each x
y1 = [xi ** 2 + 1 for xi in x]
y2 = [xi ** 3 + xi ** 2 + xi + 1 for xi in x]

# Plotting
plt.plot(x, y1, label="y1(x) = x^2 + 1")
plt.plot(x, y2, label="y2(x) = x^3 + x^2 + x + 1")
plt.title("Polynomial Functions")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()
```



Ex 8b – using numpy:

```
import matplotlib.pyplot as plt
import numpy as np

# Calculate the range
x_range = np.arange(-500, 501)

# Calculate x values based on the range using array operations
x = x_range * 0.01

# Calculate y1 and y2 values for each x using array operations
y1 = x ** 2 + 1
y2 = x ** 3 + x ** 2 + x + 1

# Plotting
plt.plot(x, y1, label="y1(x) = x^2 + 1")
plt.plot(x, y2, label="y2(x) = x^3 + x^2 + x + 1")
plt.title("Polynomial Functions")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()
```

Ex



Ex. 9:

```
line_count = 0
with open("ex9/test.txt", 'r') as file:
    for line in file:
        line_count += 1

        word_count = len(line.split())
        print(f"Line {line_count}: Number of words = {word_count}")

        vowel_count = sum(1 for char in line.lower() if char in 'aeiou')
        print(f"Line {line_count}: Number of vowels = {vowel_count}")

        line_with_capitalized_vowels = ''.join(char.upper() if char.lower() in 'aeiou' else char for char in line)
        print(f"Line {line_count}: {line_with_capitalized_vowels}")

print("Number of lines in the file:", line_count)

out:
Line 1: Number of words = 110
Line 1: Number of vowels = 261
Line 1: LOrEm IpsUm dOlOr sIt AmEt, cOnsEctEtUr AdIpIscIng ElIt. PEllEntEsqUE pOrtA tEllUs Et lAcUs UllAmcOrpEr lObOrtIs Ac Et EnIm. MOrbI EgEt cOmmOdO tUrpIs, A vEhIcUlA sEm. DOnEc vArIUs mI vItAE sEm vUlpUtAtE lObOrtIs. MAEcEnAs dIctUm In lIbErO qUIs frIngIllA. DUIs EU sEmpEr ErOs. DOnEc vItAE OrnArE fElIs, vItAE tEmpOr lIgUlA. AlIqUAm Ut plAcErAt lEO. DUIs pUrUs lAcUs, pOrttItOr vOlUtpAt sEmpEr qUIs, vEnEnAtIs nEc jUstO. PEllEntEsqUE mAssA mEtUs, sUscIpIt Ac tOrtOr Ut, pUlvInAr AccUmsAn AntE. CUrAbItUr nEc sOdAlEs AntE. PrAEsEnt Et grAvIdA mAgnA. QUIsqUE rUtrUm pUrUs lIbErO, EU pOsUErE tUrpIs EffIcItUr AlIqUEt. IntErdUm Et mAlEsUAdA fAmEs Ac AntE IpsUm prImIs In fAUcIbUs. NUnc At OrcI nEqUE. NUllAm dApIbUs Ac vElIt
vItAE sEmpEr.

Line 2: Number of words = 97
Line 2: Number of vowels = 233
Line 2: MAEcEnAs tIncIdUnt bIbEndUm mEtUs. DOnEc ElEmEntUm pUlvInAr grAvIdA. MOrbI sIt AmEt rUtrUm EnIm. AlIqUAm ErAt vOlUtpAt. PrOIn sIt AmEt lEO fElIs. NUnc cUrsUs tUrpIs nOn mAssA fInIbUs, Ac fErmEntUm AUgUE EgEstAs. SEd sEd EnIm A nIsI EffIcItUr fAUcIbUs. In sApIEn mEtUs, cOnsEctEtUr vEl vElIt EgEt, fEUgIAt phArEtrA AntE. EtIAm vEl vIvErrA lIgUlA. NUnc OrcI ArcU, fAUcIbUs Ac mAlEsUAdA qUIs, bIbEndUm nEc sApIEn. In EgEt vIvErrA Ex. PrAEsEnt qUAm dUI, OrnArE vEl mEtUs In, lAOrEEt lObOrtIs pUrUs. DOnEc EnIm fElIs, mOllIs Ut mAUrIs A, hEndrErIt scElErIsqUE EnIm. QUIsqUE vUlpUtAtE cOndImEntUm dIgnIssIm. QUIsqUE EgEt OdIO A lOrEm tIncIdUnt mAlEsUAdA.

Line 3: Number of words = 64
Line 3: Number of vowels = 156
Line 3: EtIAm UllAmcOrpEr nIbh sEm, vItAE mAttIs mI EffIcItUr sIt AmEt. MAUrIs dApIbUs, mAgnA sEd sEmpEr OrnArE, IpsUm IpsUm cOmmOdO lOrEm, In sAgIttIs rIsUs AntE sEd rIsUs. NUllAm mAxImUs lOrEm pOsUErE, cOngUE tEllUs dIgnIssIm, AccUmsAn nEqUE. PrOIn sIt AmEt phArEtrA lEO. AlIqUAm sOdAlEs UllAmcOrpEr pUrUs, Id
cOnsEctEtUr ArcU rUtrUm Ut. MAEcEnAs AlIqUEt UllAmcOrpEr nUnc A mOlEstIE. SEd sUscIpIt fAcIlIsIs Est, vItAE EUIsmOd nUllA sOllIcItUdIn Ac.
```

```
Line 4: Number of words = 77
Line 4: Number of vowels = 180
Line 4: SEd A mI EgEt dIAm vUlpUtAtE vArIUs At Id AUgUE. NUllAm mAxImUs nEqUE sEd blAndIt
cUrsUs. MOrbI
hEndrErIt lIgUlA Id mAssA mAxImUs, nEc hEndrErIt Ex vEhIcUlA. NAm sEd tEmpUs dUI. PEllEntEsqUE
AlIqUAm dUI nEc qUAm frIngIllA, sEmpEr UltrIcEs nIsl vArIUs. AEnEAn pUrUs qUAm, bIbEndUm qUIs
UltrIcEs fAcIlIsIs, cUrsUs Id OrcI. AlIqUAm mAlEsUAdA nIsl A UrnA pOrttItOr pUlvInAr. NUnc
sOllIcItUdIn dIAm A lObOrtIs hEndrErIt. IntEgEr qUIs mAUrIs EnIm. NUnc At fAUcIbUs lOrEm. AlIqUAm
sEmpEr dOlOr At OrnArE pOrttItOr.

Line 5: Number of words = 60
Line 5: Number of vowels = 134
Line 5: DOnEc lAcInIA EnIm pUrUs, Id IntErdUm AUgUE sOllIcItUdIn Id. NUllA vEl pUrUs tEmpUs,
IntErdUm lEO Ac, ElEmEntUm Est. NUllA A sApIEn ElEmEntUm, tEmpOr jUstO In, cOndImEntUm
mAssA. MAUrIs vEl dIctUm lAcUs, In fInIbUs tUrpIs. IntErdUm Et mAlEsUAdA fAmEs Ac AntE IpsUm
prImIs In fAUcIbUs. Ut mAlEsUAdA sIt AmEt dUI sEd pOrtA. DOnEc mAxImUs rUtrUm OdIO, nEc
pUlvInAr ErAt UltrIcEs In.
Number of lines in the file: 5
```