ISSN: 2181-3906 2024

International scientific journal
«MODERN SCIENCE AND RESEARCH»
VOLUME 3 / ISSUE 5 / UIF:8.2 / MODERNSCIENCE.UZ

ENHANCING BUSINESS FINANCIAL CALCULATIONS WITH PYTHON: A PRACTICAL APPROACH

Anvarjonov Bunyodbek Baxodirovich
Senior teacher of TMS Institute.
@bunyodbek.anvarjonov@mail.ru

https://doi.org/10.5281/zenodo.11520851

Abstract. *In the realm of business mathematics, Python can be an invaluable tool for solving various financial problems efficiently and accurately. This document presents a series of Python code examples that address common business math scenarios, including simple interest calculation, compound interest calculation, break-even analysis, present value of future cash flows, and depreciation using the straight-line method.*

Each problem is addressed with a corresponding Python function, complete with example usage to demonstrate practical applications. These examples illustrate how Python can streamline financial calculations, making complex business math problems more approachable and manageable.

Keywords: *Business Mathematics, Python Programming, Financial Calculations, Simple Interest, Compound Interest, Break-Even Analysis, Present Value, Future Cash Flows, Depreciation, Straight-Line Method, Financial Planning, Investment Analysis, Asset Management, Tax Planning, Economic Decision-Making.*

УЛУЧШЕНИЕ БИЗНЕС-ФИНАНСОВЫХ РАСЧЕТОВ С ПОМОЩЬЮ PYTHON: ПРАКТИЧЕСКИЙ ПОДХОД

Аннотация. *В области бизнес-математики Python может быть бесценным инструментом для эффективного и точного решения различных финансовых задач. В этом документе представлена серия примеров кода Python, которые рассматривают распространенные математические сценарии бизнеса, включая расчет простых процентов, расчет сложных процентов, анализ безубыточности, текущую стоимость будущих денежных потоков и амортизацию с использованием линейного метода.*

Каждая проблема решается с помощью соответствующей функции Python, дополненной примерами использования для демонстрации практического применения. Эти примеры иллюстрируют, как Python может упростить финансовые расчеты, делая сложные математические бизнес-задачи более доступными и управляемыми.

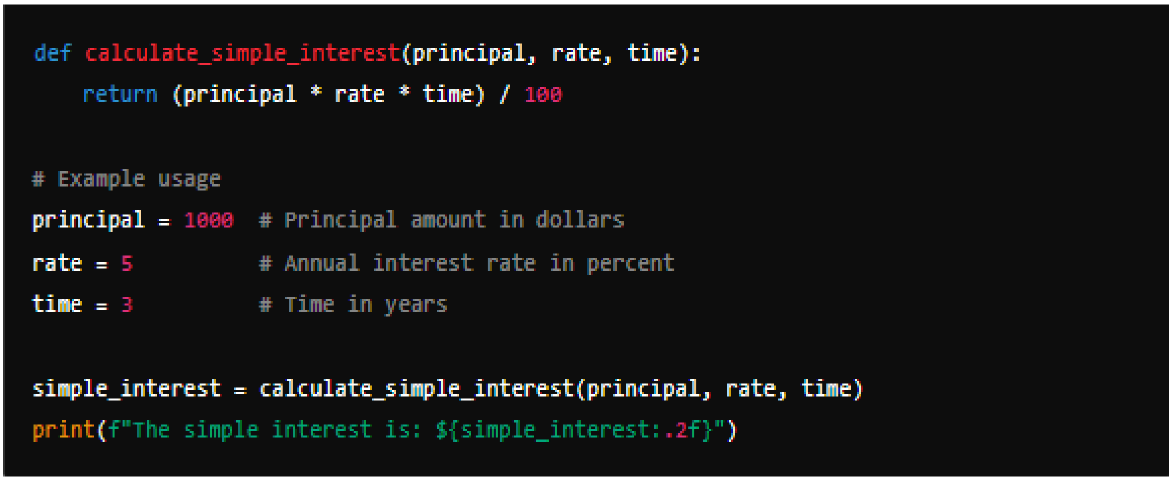
Ключевые слова: *бизнес-математика, программирование на Python, финансовые расчеты, простые проценты, сложные проценты, анализ безубыточности, текущая стоимость, будущие денежные потоки, амортизация, прямолинейный метод, финансовое планирование, инвестиционный анализ, управление активами, налоговое планирование, экономическое решение. -Изготовление.*

1. Simple Interest Calculation

Simple interest is calculated with the formula:

$$\text{Simple Interest} = \frac{P \times R \times T}{100}$$

where P is the principal amount, R is the rate of interest per year, and T is the time in years.



```
def calculate_simple_interest(principal, rate, time):
    return (principal * rate * time) / 100

# Example usage
principal = 1000  # Principal amount in dollars
rate = 5          # Annual interest rate in percent
time = 3          # Time in years

simple_interest = calculate_simple_interest(principal, rate, time)
print(f"The simple interest is: ${simple_interest:.2f}")
```

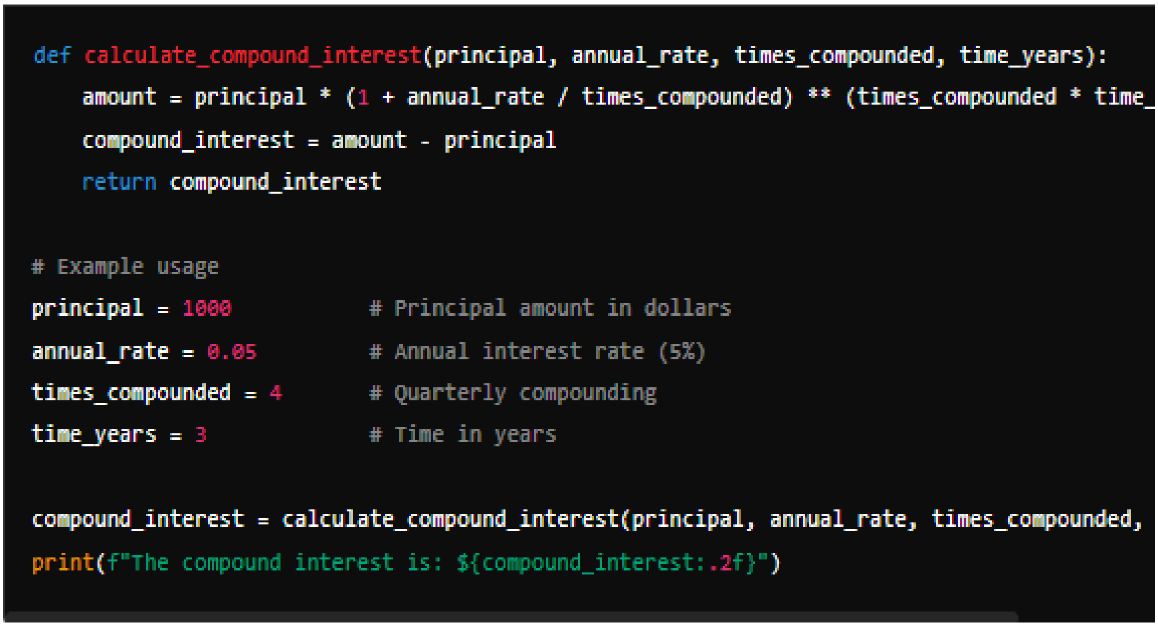
2. Compound Interest Calculation

Compound interest is calculated with the formula:

$$A = P\left(1 + \frac{r}{n}\right)^{nt}$$

where P is the principal amount, r is the annual interest rate (decimal), n is the number of times interest is compounded per year, and t is the time the money is invested for in years.

The compound interest is $A - P$.

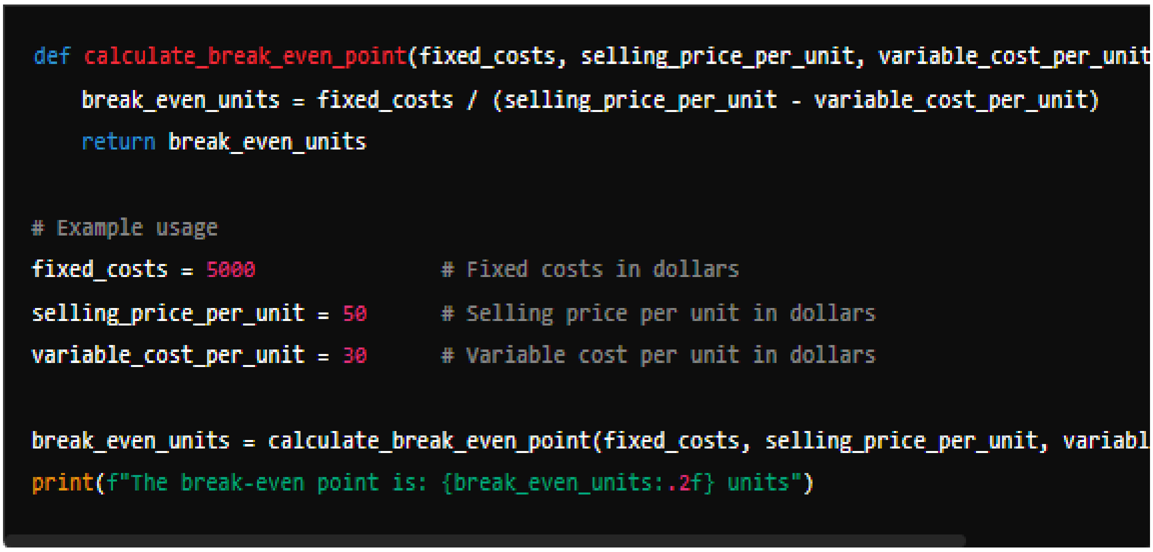


```
def calculate_compound_interest(principal, annual_rate, times_compounded, time_years):
    amount = principal * (1 + annual_rate / times_compounded) ** (times_compounded * time_
    compound_interest = amount - principal
    return compound_interest

# Example usage
principal = 1000          # Principal amount in dollars
annual_rate = 0.05        # Annual interest rate (5%)
times_compounded = 4      # Quarterly compounding
time_years = 3            # Time in years

compound_interest = calculate_compound_interest(principal, annual_rate, times_compounded,
print(f"The compound interest is: ${compound_interest:.2f}")
```

3. Break-Even Analysis

Break-even point in units is calculated with the formula:

$$\text{Break-Even Point} = \frac{\text{Fixed Costs}}{\text{Selling Price per Unit} - \text{Variable Cost per Unit}}$$



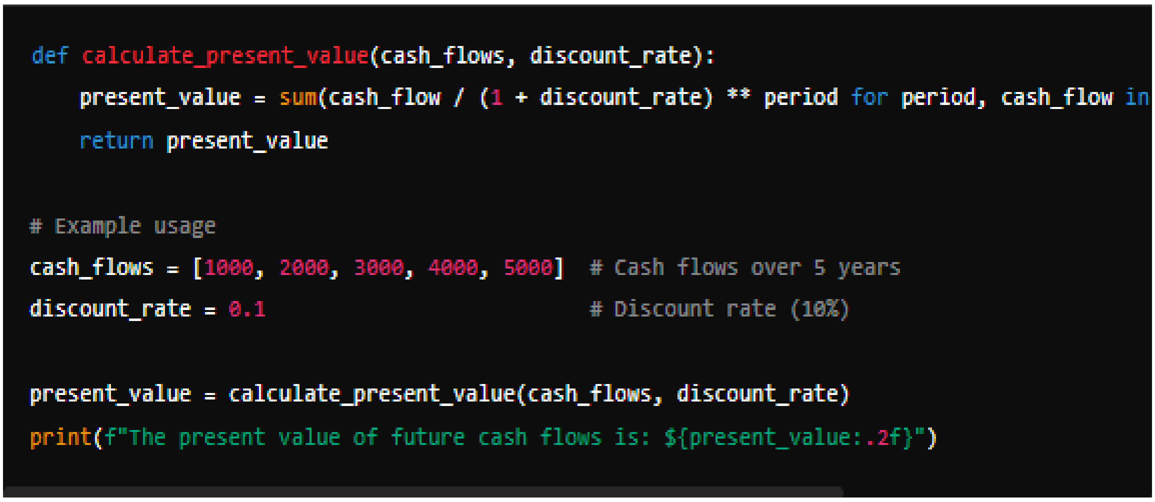
```
def calculate_break_even_point(fixed_costs, selling_price_per_unit, variable_cost_per_unit
    break_even_units = fixed_costs / (selling_price_per_unit - variable_cost_per_unit)
    return break_even_units

# Example usage
fixed_costs = 5000                # Fixed costs in dollars
selling_price_per_unit = 50       # Selling price per unit in dollars
variable_cost_per_unit = 30       # Variable cost per unit in dollars

break_even_units = calculate_break_even_point(fixed_costs, selling_price_per_unit, variabl
print(f"The break-even point is: {break_even_units:.2f} units")
```

4. Present Value of Future Cash Flows

The present value (PV) of future cash flows is calculated with the formula:

$$PV = \sum_{t=1}^{n} \frac{C_t}{(1+r)^t}$$

where C_t is the cash flow at time t, r is the discount rate, and n is the number of periods.

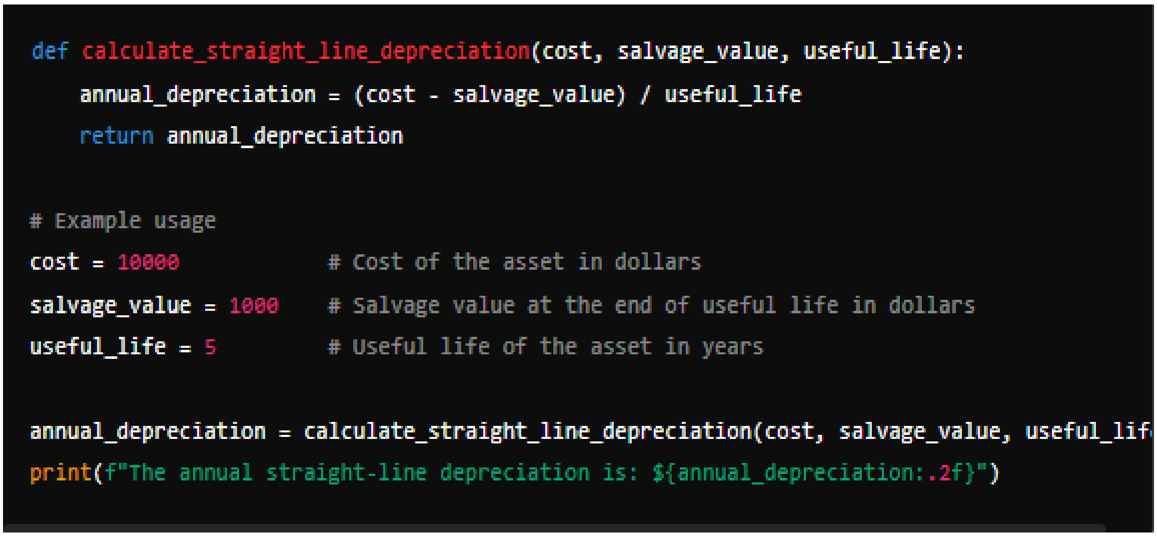


```
def calculate_present_value(cash_flows, discount_rate):
    present_value = sum(cash_flow / (1 + discount_rate) ** period for period, cash_flow in
    return present_value

# Example usage
cash_flows = [1000, 2000, 3000, 4000, 5000]  # Cash flows over 5 years
discount_rate = 0.1                          # Discount rate (10%)

present_value = calculate_present_value(cash_flows, discount_rate)
print(f"The present value of future cash flows is: ${present_value:.2f}")
```

5. Depreciation Using Straight-Line Method

Straight-line depreciation is calculated with the formula:

$$\text{Annual Depreciation Expense} = \frac{\text{Cost} - \text{Salvage Value}}{\text{Useful Life}}$$

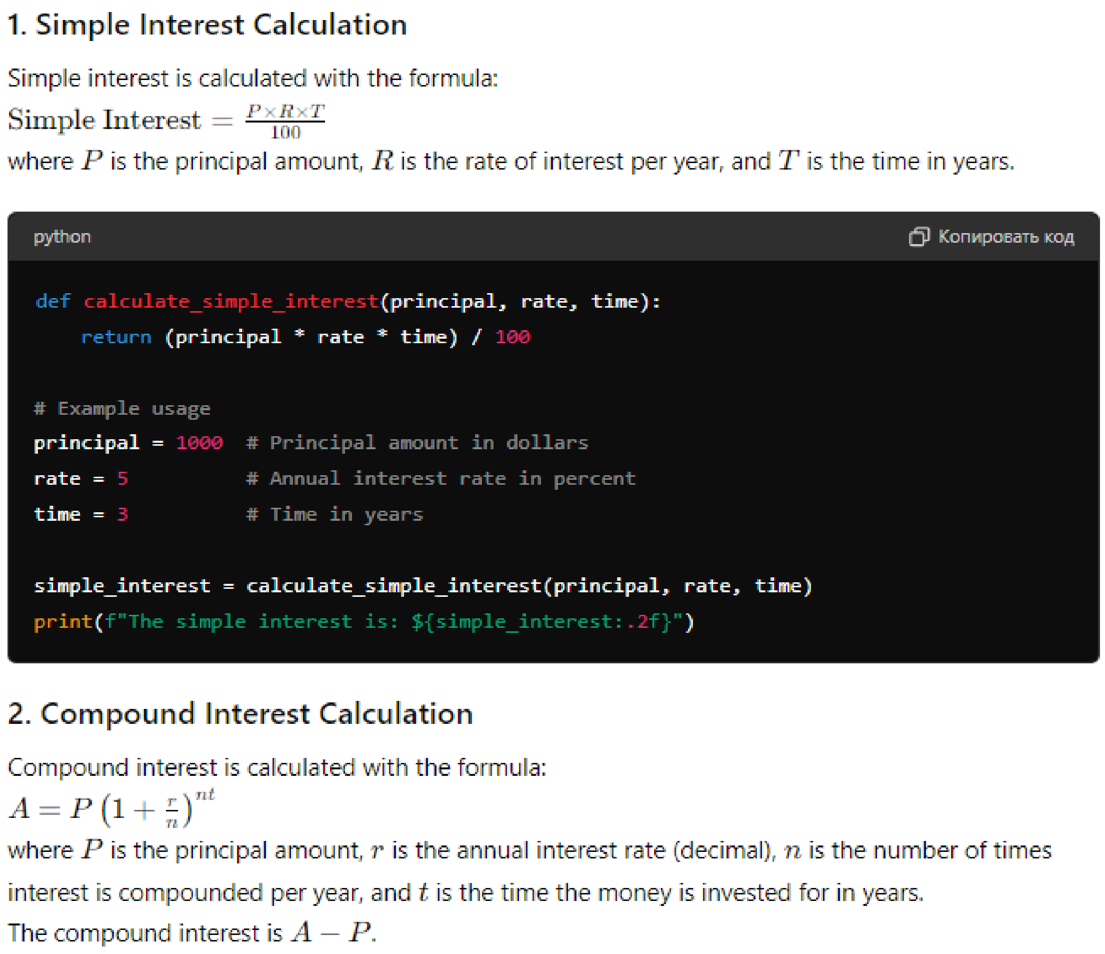


```
def calculate_straight_line_depreciation(cost, salvage_value, useful_life):
    annual_depreciation = (cost - salvage_value) / useful_life
    return annual_depreciation

# Example usage
cost = 10000            # Cost of the asset in dollars
salvage_value = 1000    # Salvage value at the end of useful life in dollars
useful_life = 5         # Useful life of the asset in years

annual_depreciation = calculate_straight_line_depreciation(cost, salvage_value, useful_lif
print(f"The annual straight-line depreciation is: ${annual_depreciation:.2f}")
```

These are basic examples of business math problems and their implementations in Python. Feel free to modify the parameters and use the functions as needed for different scenarios.

REFERENCES

1. Brigham, E. F., & Ehrhardt, M. C. (2019). *Financial Management: Theory & Practice*. Cengage Learning.
2. Bodie, Z., Kane, A., & Marcus, A. J. (2017). *Investments*. McGraw-Hill Education.
3. Ross, S. A., Westerfield, R. W., & Jordan, B. D. (2018). *Fundamentals of Corporate Finance*. McGraw-Hill Education.
4. Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *Journal of Finance*, 25(2), 383-417.
5. Modigliani, F., & Miller, M. H. (1958). The Cost of Capital, Corporation Finance and the Theory of Investment. *American Economic Review*, 48(3), 261-297.
6. Mallayev O., Anvarjonov B., Aziz M. Cache Problems in Parallel Computational Processes //Annals of the Romanian Society for Cell Biology. – 2021. – С. 8924-8934.
7. Bunyodbek A. Solving examples of the distance between two straight lines in Python //Innovations in exact science. – 2024. – Т. 1. – №. 3. – С. 1-7.



1. Simple Interest Calculation

Simple interest is calculated with the formula:

$\text{Simple Interest} = \frac{P \times R \times T}{100}$

where P is the principal amount, R is the rate of interest per year, and T is the time in years.

python Копировать код

```python
def calculate_simple_interest(principal, rate, time):
    return (principal * rate * time) / 100

# Example usage
principal = 1000  # Principal amount in dollars
rate = 5          # Annual interest rate in percent
time = 3          # Time in years

simple_interest = calculate_simple_interest(principal, rate, time)
print(f"The simple interest is: ${simple_interest:.2f}")
```

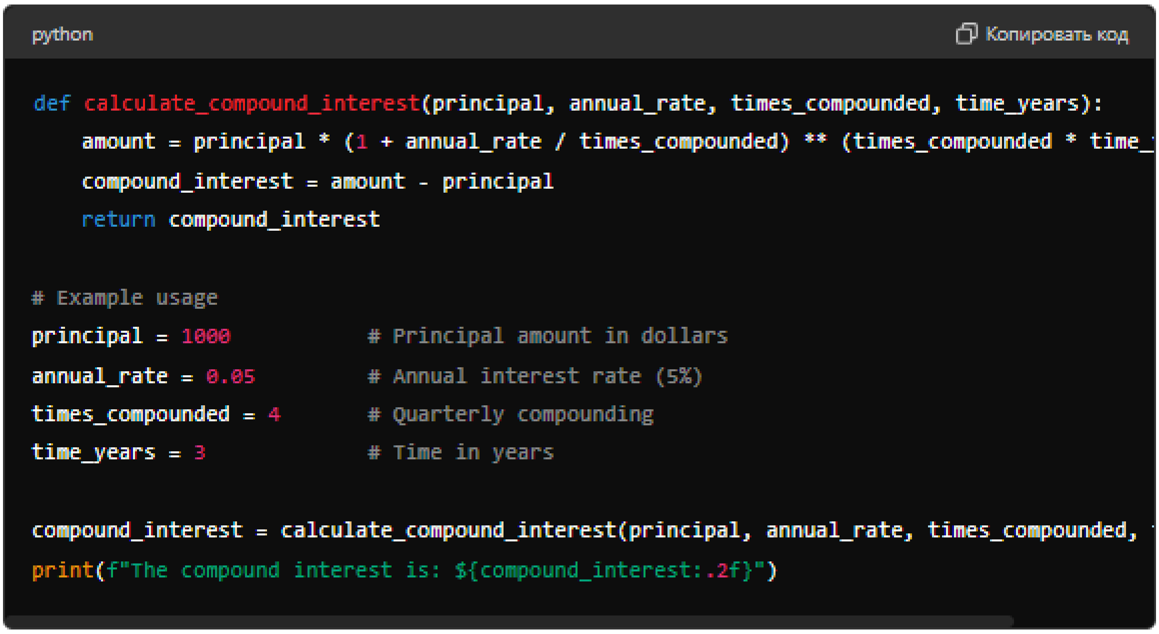
2. Compound Interest Calculation

Compound interest is calculated with the formula:

$A = P\left(1 + \frac{r}{n}\right)^{nt}$

where P is the principal amount, r is the annual interest rate (decimal), n is the number of times interest is compounded per year, and t is the time the money is invested for in years.

The compound interest is $A - P$.

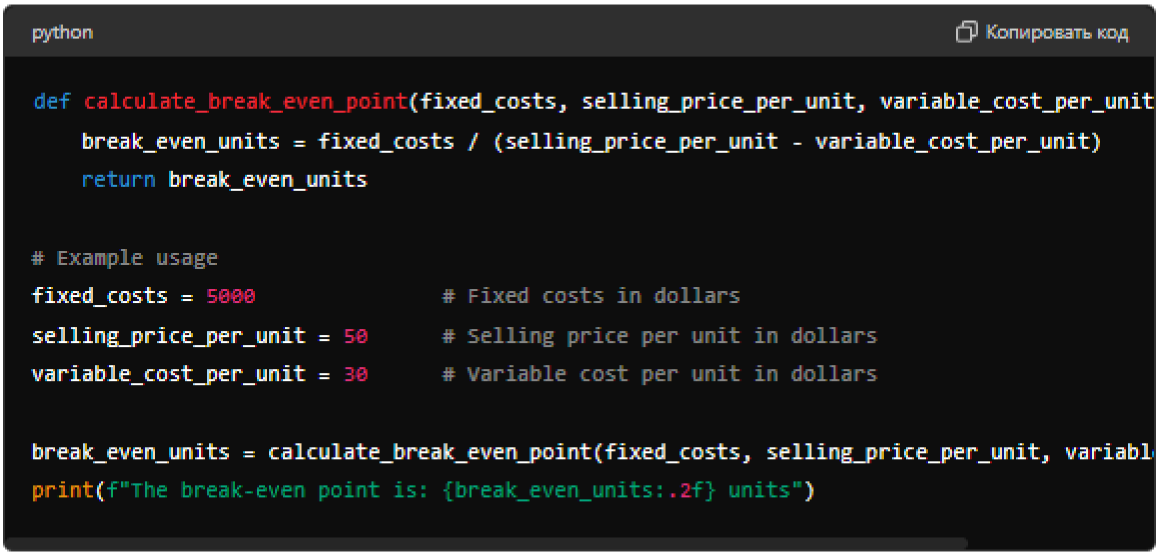


```python
def calculate_compound_interest(principal, annual_rate, times_compounded, time_years):
    amount = principal * (1 + annual_rate / times_compounded) ** (times_compounded * time_
    compound_interest = amount - principal
    return compound_interest

# Example usage
principal = 1000            # Principal amount in dollars
annual_rate = 0.05          # Annual interest rate (5%)
times_compounded = 4        # Quarterly compounding
time_years = 3              # Time in years

compound_interest = calculate_compound_interest(principal, annual_rate, times_compounded,
print(f"The compound interest is: ${compound_interest:.2f}")
```

3. Break-Even Analysis

Break-even point in units is calculated with the formula:

$$\text{Break-Even Point} = \frac{\text{Fixed Costs}}{\text{Selling Price per Unit} - \text{Variable Cost per Unit}}$$



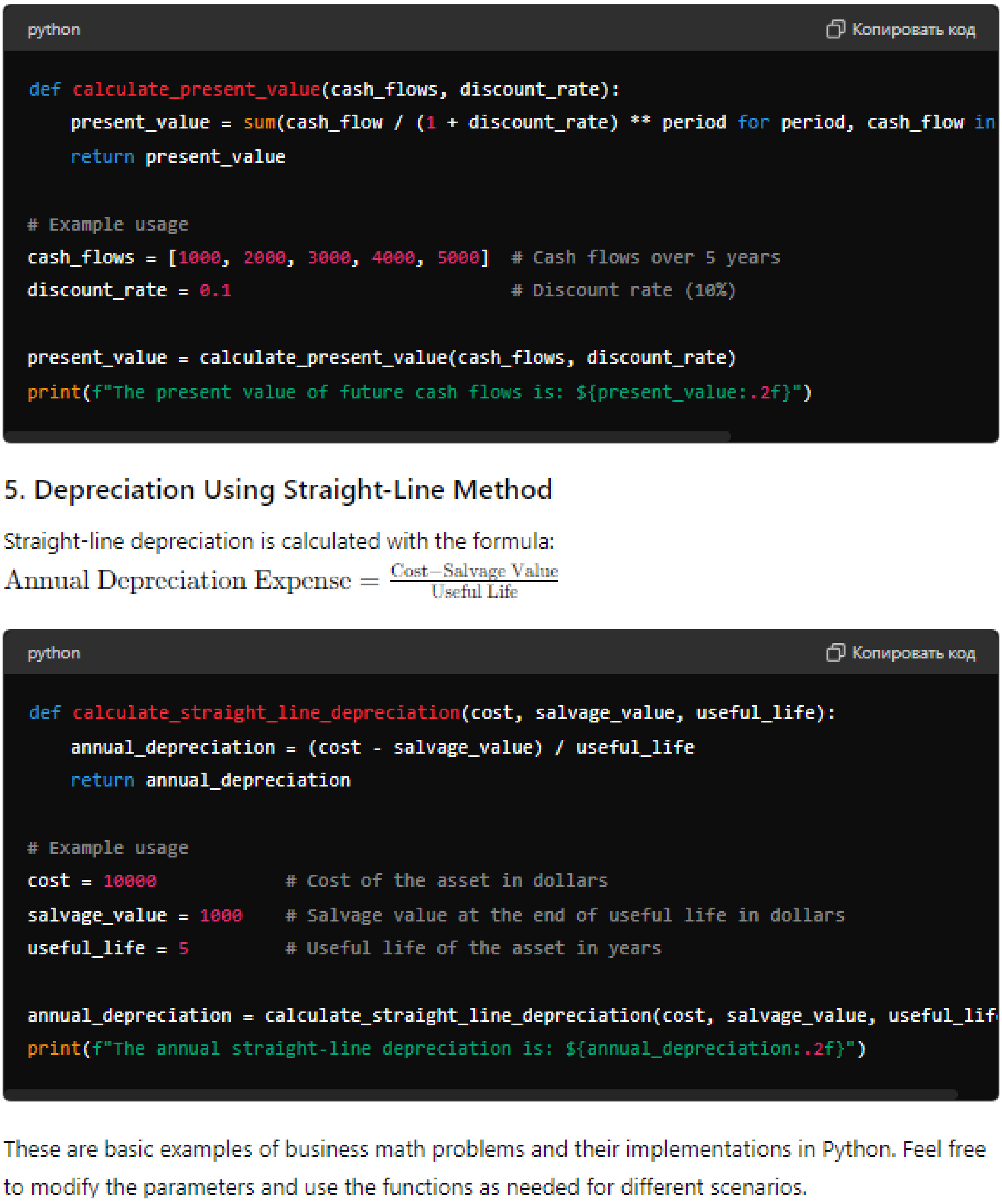
```python
def calculate_break_even_point(fixed_costs, selling_price_per_unit, variable_cost_per_unit
    break_even_units = fixed_costs / (selling_price_per_unit - variable_cost_per_unit)
    return break_even_units

# Example usage
fixed_costs = 5000              # Fixed costs in dollars
selling_price_per_unit = 50     # Selling price per unit in dollars
variable_cost_per_unit = 30     # Variable cost per unit in dollars

break_even_units = calculate_break_even_point(fixed_costs, selling_price_per_unit, variabl
print(f"The break-even point is: {break_even_units:.2f} units")
```

4. Present Value of Future Cash Flows

The present value (PV) of future cash flows is calculated with the formula:

$$PV = \sum_{t=1}^{n} \frac{C_t}{(1+r)^t}$$

where C_t is the cash flow at time t, r is the discount rate, and n is the number of periods.



```python
def calculate_present_value(cash_flows, discount_rate):
    present_value = sum(cash_flow / (1 + discount_rate) ** period for period, cash_flow in
    return present_value

# Example usage
cash_flows = [1000, 2000, 3000, 4000, 5000]  # Cash flows over 5 years
discount_rate = 0.1                          # Discount rate (10%)

present_value = calculate_present_value(cash_flows, discount_rate)
print(f"The present value of future cash flows is: ${present_value:.2f}")
```

5. Depreciation Using Straight-Line Method

Straight-line depreciation is calculated with the formula:

$$\text{Annual Depreciation Expense} = \frac{\text{Cost} - \text{Salvage Value}}{\text{Useful Life}}$$

```python
def calculate_straight_line_depreciation(cost, salvage_value, useful_life):
    annual_depreciation = (cost - salvage_value) / useful_life
    return annual_depreciation

# Example usage
cost = 10000            # Cost of the asset in dollars
salvage_value = 1000    # Salvage value at the end of useful life in dollars
useful_life = 5         # Useful life of the asset in years

annual_depreciation = calculate_straight_line_depreciation(cost, salvage_value, useful_lif
print(f"The annual straight-line depreciation is: ${annual_depreciation:.2f}")
```

These are basic examples of business math problems and their implementations in Python. Feel free to modify the parameters and use the functions as needed for different scenarios.