Python for Data Analysis
BU.510.615
Naser Nikandish

Optional Practice Problems

These problems are prepared to help students practice the basics of Python. They are optional and designed to reinforce core concepts, enhance coding skills, and boost confidence in working with the language.

Numbers

1. Print the product of numbers 7 and 8.
2. Divide 10 by 3 and print the remainder.
3. Check if the number 5 is even or odd and print the result.
4. Compute and print the square root of 16.
5. Given the radius of a circle is 5 units, compute and print its area.
6. Given the base and height of a triangle are 10 and 5 units respectively, compute and print its area.

Strings

7. Print the length of the string `"Hello, BARM!"`.
8. Convert the string `"Business"` to uppercase and print the result.
9. Convert the string `"PYTHON"` to lowercase and print the result.
10. Check if the string `"12345"` is a digit and print the result.
11. Find the first occurrence of the substring `"python"` in the string `"I love python programming in Python for Data Analysis course"` and print the index.
12. Replace all occurrences of letter `"a"` with `"@"` in the string `"banana"` and print the result.
13. Check if the string `"radar"` is a palindrome and print the result.

14. Reverse the string `"python"` and print the result.
15. Extract and print the uppercase of the domain from the following email address. **Note:** the ideal output is `"JHU.EDU"`. Email address: `"naser.nikandish@jhu.edu"`.
16. (Review python functions first) Given the string `"World of Python Java C Rubi Julia"`, split it into words and print the longest word.
17. Given a string `"Python for Data Analysis, Data Analytics, Business Analytics"`, write a Python function that:
    (a) Converts the string to lowercase.
    (b) Removes any duplicate words while preserving the original order.
    (c) Returns the final string.

Lists

18. Print the length of the list `[1, 2, 3, 4, 5]`.
19. Add the number 6 to the end of the list `[1, 2, 3, 4, 5]` and print the updated list.
20. Insert the number 0 at the third position in the list `[1, 2, 3, 4, 5]` and print the updated list.
21. Remove the last element from the list `[1, 2, 3, 4, 5]` and print the updated list.
22. From the list `[3, 1, 4, 1, 5, 9, 2, 6]`, print the maximum and minimum value.
23. Find the index of the first occurrence of number 3 in the list `[1, 2, 3, 4, 3, 5]` and print it.
24. Count the number of occurrences of number 3 in the list `[1, 2, 3, 4, 3, 5]` and print it.
25. Sort the list `[3, 1, 4, 1, 5, 9, 2, 6]` in ascending order and print it.
26. Reverse the list `[1, 2, 3, 4, 5]` and print it.
27. From the list `[3, 1, 4, 1, 5, 9, 2, 6]`, print the indexes of the maximum and minimum values.

28. Remove all duplicates from the list `[1, 2, 2, 3, 4, 4, 5]` and print the updated list.
29. Create a list of all numbers from 1 to 1000 that:
    (a) Are divisible by both 3 and 5.
    (b) Have an even number of digits.

    Write your solution using a list comprehension.

Tuples

30. Given the tuple `(1, 2, 3, 4, 5)`, print its third element.
31. Convert the tuple `(1, 2, 3, 4, 5)` to a list and print it.
32. Convert the list `[1, 2, 3, 4, 5]` to a tuple and print it.
33. Check if the number 3 exists in the tuple `(1, 2, 3, 4, 5)` and print the result.
34. Count the number of occurrences of number 2 in the tuple `(1, 2, 3, 2, 4, 2, 5)` and print it.
35. Find the index of the first occurrence of number 3 in the tuple `(1, 2, 3, 4, 5)` and print it.
36. Given the tuple `(a, b, c, d)`, unpack its elements into variables and print them.
37. Merge the tuples `(1, 2, 3)` and `(4, 5, 6)` and print the result.
38. (Review functions part first please) Write a function that accepts an arbitrary number of arguments using `*args` and returns:
    (a) The sum of the numbers.
    (b) The number of arguments provided.
    (c) A tuple containing only the unique arguments.

    Demonstrate this function by calling it with at least 5 arguments.

Dictionaries

39. Create a dictionary with the following key-value pairs: `'apple': 1, 'banana': 2, 'cherry': 3`.

40. Access the value of key 'banana' from the dictionary `{'apple': 1, 'banana': 2, 'cherry': 3}`.

41. Update the value of key `'apple'` to 4 in the dictionary `{'apple': 1, 'banana': 2, 'cherry': 3}` and print the updated dictionary.

42. Add a new key-value pair `'date': 4` to the dictionary `{'apple': 1, 'banana': 2, 'cherry': 3}` and print the updated dictionary.

43. Remove the key-value pair with key `'banana'` from the dictionary `{'apple': 1, 'banana': 2, 'cherry': 3}` and print the updated dictionary.

44. Check if the key 'apple' exists in the dictionary `{'apple': 1, 'banana': 2, 'cherry': 3}` and print the result.

45. Print all the keys of the dictionary `{'apple': 1, 'banana': 2, 'cherry': 3}`.

46. Print all the values of the dictionary `{'apple': 1, 'banana': 2, 'cherry': 3}`.

47. Merge the dictionaries `{'apple': 1, 'banana': 2}` and `{'cherry': 3, 'date': 4}` and print the result.

48. Given the dictionary `grades = {"John": 82, "Anna": 90, "George": 67, "Marry": 92}`, print the names of students who have a grade above 85.

49. Merge the dictionaries `dict1 = {"a": 1, "b": 2}` and `dict2 = {"b": 3, "c": 4}`. If there's a key overlap, the values should be combined in a list.

50. Given the dictionary `values = {"x": 12, "y": 25, "z": 5}`, print keys that have the maximum and minimum values.

51. Convert the list of tuples `data = [("a", 1), ("b", 2), ("a", 3)]` into a dictionary. If a key appears more than once, combine the values in a list.

52. Given the string `s = "aabbccddeeffggghhiijj"`, count the occurrences of each character and store the results in a dictionary. Then, print the three most common characters and their counts.

53. Given the dictionary `words = {"apple": 5, "banana": 2, "cherry": 8}`, print the word that appears the most times.

54. Using the dictionary `conversion = {"USD": 1, "EUR": 0.85, "JPY": 110}`, create a function that converts a given amount from USD to the specified currency and print the result.

55. From the dictionary `colors = {"red": "#FF0000", "green": "#008000", "blue": "#0000FF"}`, print all colors whose hexadecimal code contains `"00"`.

56. Given the dictionary `votes = {"Alice": 3, "Bob": 4, "Charlie": 2}`, add a vote for `"Bob"` and remove a vote from `"Alice"`, then print the updated dictionary.

57. Consider a nested dictionary `school = {"John": {"Math": 90, "History": 85}, "Anna": {"Math": 80, "History": 92}}`. Calculate the average grade for each student across all subjects and print it.

58. (Challenging) Given a dictionary that stores the name and marks of students as follows:

    ```
    students = {"Alice": 85, "Bob": 92, "Clara": 87, "Dave": 75, "Eva": 90}
    ```

    Write a Python program to:

    (a) Find the student with the highest marks.

    (b) Add a new student `"Frank"` with marks 88.

    (c) Increase the marks of each student by 5%.

Sets

59. Create a set with the elements 1, 2, 3, 4, 5 and print it.

60. Add the number 6 to the set 1, 2, 3, 4, 5 and print the updated set.

61. Remove the number 3 from the set 1, 2, 3, 4, 5 and print the updated set.

62. Check if the number 4 exists in the set 1, 2, 3, 4, 5 and print the result.

63. Print the union of the sets 1, 2, 3 and 3, 4, 5.

64. Print the intersection of the sets 1, 2, 3 and 3, 4, 5.

65. Check if the set 1, 2, 3 is a subset of the set 1, 2, 3, 4, 5 and print the result.
66. Check if the set 1, 2, 3, 4, 5 is a superset of the set 1, 2, 3 and print the result.
67. Print the symmetric difference between the sets 1, 2, 3, 4 and 3, 4, 5, 6.
68. Given two sets:

    ```
    set1 = {1, 2, 3, 4, 5, 6}
    set2 = {4, 5, 6, 7, 8, 9}
    ```

    Write a function to perform the following:

    (a) Find the union of both sets.
    (b) Find the intersection of both sets.
    (c) Find the symmetric difference of both sets.

    Return each of these results as a tuple in the order: (union, intersection, symmetric difference).

69. **Pizza Order Function**:
    - Write a function called `calculate_total` that takes in the number of pizzas ordered and returns the total cost. Assume each pizza costs $10.
    - Now, call your function with 5 pizzas.
70. **Greetings**:
    - Create a function called `greet` that takes in a name as an argument and prints out ”Hello, [name]!”.
    - Call your function with the name ”Alex”.
71. **List Sum**:
    - Write a function called `sum_of_list` that takes a list of numbers and returns their sum.
    - Call your function with the list [2, 4, 6, 8, 10].
72. **Shopaholic’s Discount**:
    - Create a function `apply_discount` that takes the original price of an item and applies a 10% discount, then returns the discounted price.
    - Call your function with a price of $50.

73. **Birthday Party**:
    - Write a function called `party_planner` that takes the number of guests and returns the number of pizzas to order, assuming each guest will eat 2 slices and each pizza has 8 slices.
    - Call your function with 16 guests.
74. **Favorite Colors**:
    - Write a function called `favorite_color` that takes in a name and a list of names and their favorite colors. The function should return the favorite color of the given name.
    - Call your function with the name "Alice" and the list [('Bob', 'Blue'), ('Alice', 'Red'), ('Eve', 'Green')].
75. **Adding Tax**:
    - Create a function `add_tax` that takes in a price and adds a 5% tax to it, then returns the total price.
    - Call your function with a price of $20.
76. **List of Squares**:
    - Write a function `squared_numbers` that takes a number `n` and returns a list containing squares of numbers from 1 to `n`.
    - Call your function with `n = 5`.
77. **Friend's Birthday**:
    - Create a function `is_birthday` that takes in a date (in the format 'MM-DD') and checks if it matches your friend's birthday, returning True if it does and False otherwise.
    - Call your function with the date '04-15', assuming your friend's birthday is on April 15th.
78. **Weekly Groceries**:
    - Write a function `weekly_groceries` that takes in a list of items and adds "Milk" to the list if it's not already there.
    - Call your function with the list ['Eggs', 'Bread', 'Butter'].

79. **Greeting Function**:
    - Write a lambda function that takes a name as an argument and returns a greeting. For example, given ”Alex”, it should return ”Hello, Alex!”.
80. **Double the Number**:
    - Create a lambda function that doubles the given number. If you input 5, it should return 10.
81. **Check for Even Numbers**:
    - Design a lambda function to determine if a given number is even. Check it with 6
82. **Book Recommendations**:
    - Create a function `recommend_books` that takes in a dictionary where keys are genres and values are lists of book names. The function should return a random book from a given genre.
    - Call your function with a genre from the following dictionary: {’Mystery’: [’Book1’, ’Book2’], ’Romance’: [’Book3’, ’Book4’]}.
83. **Roommate Expenses**:
    - Write a function `split_expenses` that takes in a dictionary of expenses and returns how much each roommate owes. Assume there are two roommates.
    - Call your function with the following expenses: {’rent’: 1000, ’utilities’: 150, ’groceries’: 200}.
84. **Set Intersection**:
    - Create a function `common_elements` that takes in two lists and returns a set of elements that are common to both lists.
    - Call your function with the lists [1, 2, 3, 4, 5] and [4, 5, 6, 7, 8].

85. **Grades Average**:

    - Write a function `compute_average` that takes in a dictionary where keys are student names and values are lists of grades. The function should return a dictionary with student names and their average grades.
    - Call your function with the dictionary {'John': [85, 90, 78], 'Jane': [90, 88, 80]}.

86. **Password Strength Checker** (challenging question with three different solutions provided):

    - Create a function called `password_strength` that takes in a password string and checks for the following:
        - At least 8 characters long
        - Contains both uppercase and lowercase characters
        - Has at least one digit
        - Has at least one special character (e.g. '!', '@', '#', '$', etc.)
    - The function should return a boolean value: True if the password meets all the criteria, and False otherwise.
    - Call your function with the password "P@ssw0rd123".

87. **Sort by Last Name** (Challenging question):

    - You have a list of names: `['Alice Brown', 'Charlie Adams', 'Bob Clark']`. Write a lambda function to sort this list by last name.

88. **Price Calculator**:

    - Create a lambda function that calculates the price after tax. The function should take in a price and a tax rate and return the total cost. For instance, if the input price is $100 and the tax rate is 0.05, it should return $105.

89. **Friend Age Filter**:

    - You're hosting a 21 and older party. Design a lambda function to filter out friends from a list who are under 21. Given a dictionary of names and ages, like `{'Alice': 22, 'Bob': 19, 'Charlie': 23}`, it should return a list of names who are 21 and over.

90. **Custom Statistical Mode**:

- Write a function `find_mode` that takes in a list of numbers and returns the mode (the number that appears most often). If there are multiple modes, return a set of them.
- Call your function with the list [1, 2, 3, 2, 4, 4, 5].

91. **Shopping Preferences**:
    - Create a function `find_preferred_shops` that takes in a dictionary where keys are person names and values are sets of their favorite shops. The function should return a dictionary of shops and the count of people who prefer them.
    - Call your function with the dictionary {'Alex': {'Walmart', 'Target'}, 'Jane': {'Walmart', 'Macy's'}}.
92. **Advanced Discounts**:
    - Write a function `bulk_discount` that takes in a list of prices. Apply a 10% discount if there are more than 5 items and a 20% discount if there are more than 10 items. Return the total price after applying the discount.
    - Call your function with the list of prices [10, 20, 30, 40, 50, 60, 70].
93. **Friendship Compatibility**:
    - Create a function `common_interests` that takes in two dictionaries where keys are person names and values are sets of their interests. The function should return a set of common interests between the two people.
    - Call your function with the dictionaries {'John': {'reading', 'hiking', 'cooking'}, 'Jane': {'cooking', 'traveling', 'reading'}}.
94. **Grade Categories**:
    - Write a function `categorize_grades` that takes in a list of grades and returns a dictionary with categories ('F', 'D', 'C', 'B', 'A') as keys and counts of grades in each category as values.
    - Call your function with the list of grades [85, 90, 78, 60, 72, 88, 95].

95. **Custom Exponentiation**:

    - Write a lambda function that takes in two numbers, `base` and `exponent`, and returns the result of raising the base to the power of the exponent. Don't use the ** operator.

96. **Employee Salary Increment**:

    - You have a list of employees and their salaries: `'Alice': 50000, 'Bob': 60000, 'Charlie': 55000`. Design a lambda function to give each employee a 10% raise and return the updated salary list.
    - Do you see anything weird in the outcome? Why is that the case? Can you explain it?

97. **Extract Domain**:

    - From a list of email addresses, use a lambda function to extract and return a list of domain names. For example, from the email "naser@gmail.com", the function should extract "gmail.com". Test your function with a few examples.

98. Given a list of tuples representing employees' names and their salaries:

```
employees = [("John", 50000),
             ("Jane", 75000),
             ("Doe", 65000),
             ("Alice", 60000)]
```

    Sort this list:

    (a) By the employees' names in alphabetical order.

    (b) By the employees' salaries in descending order.

    Write your solution using lambda functions.