# Introduction to programming exam

31-March-2025 (A)

## Evaluation: Python Expressions and Behavior

Evaluate each of the following Python expressions. **Write the value or result** that each one produces. Assume no syntax errors.

|  |  |
| --- | --- |
| **1.**  for i in range(3, 6):  print(i \* i, end="-") | **6.**  "yes" if len("abc") == 3 and 5 > 2 else "no" |
| **2.** "universe"[::2]  Suggestion:  Remember range with 3 parameters | **7.**  x = [1, 2]  x.extend([2, 4])  x |
| **3.** len(set("mississippi")) | **8.** "sun" not in "sunset" |
| **4.**[0, 1, 2] \* 2 + [3] | **9.** "Go" \* (2 + 1) |
| **5.**(1, 2, (3, 4))[2][1] | **10.** len(["a", ("b", "c"), "d"]) |

## Exercise: Validating User Input Within a Range (with Parameters)

## Write a Python function named get\_word\_with\_length(min\_len, max\_len=10) that repeatedly asks the user to enter a word until they provide one whose length is within the specified range. The function should:

## Use a while loop to prompt the user until the word length is within the range [min\_len, max\_len], inclusive.

## Use the built-in input() function to read user input.

## Return the valid word once it is received.

## You may assume the user always enters a single word (no spaces).

## Exercise: List Square Numbers Up to N

You are asked to write two functions:

1. is\_square(n): This function takes an integer n and returns True if n is a perfect cube, otherwise returns False.
2. squares\_up\_to(n) This function takes a positive integer n and returns a list of all perfect cubes from 1 up to and including n. You must use the is\_square() function inside squares\_up\_to().

Example:

squares\_up\_to(16) # Output: [1, 4, 9, 16]

Reminder:

* A perfect square is greater than 1 and equal to some integer raised to the power of 3.

## Exercize Convert Number written in English to Integers (Up to 100)

Write a function called words\_to\_int(words) that converts a number written in English words into its corresponding integer value.

* You may assume the input is valid and in lowercase.
* The input will represent a number between 1 and 100, inclusive.
* The function must return the correct integer value.

Examples:

words\_to\_int("one") # Output: 1

words\_to\_int("twenty") # Output: 20

words\_to\_int("forty two") # Output: 42

words\_to\_int("ninety nine") # Output: 99

words\_to\_int("one hundred") # Output: 100

#### Suggestions

* Use a dictionary to map words like "one", "two", …, "ninety", "hundred" to their corresponding numeric values.
* Handle different formats:
  + A single word like "seven" or "thirty"
  + Two words like "forty two"
  + A special case like "one hundred"
* Combine values: If the number has two parts like "eighty five", compute it as 80 + 5.

## Exercise: Return Maximum and Minimum as a Tuple

Write a function named earliest\_and\_latest(dates) that takes a non-empty list of date strings (in the format "YYYY-MM-DD") and returns a tuple containing:

* the earliest date as the first element
* the latest date as the second element

You must not use built-in functions like max(), min(), or sorted().

Use a for loop and manual comparison.

### Suggestion

The "YYYY-MM-DD" format follows the ISO 8601 standard, which means that lexicographic (alphabetical) order corresponds to chronological order.

This allows you to compare dates directly as strings using < and >.

**Example:**

earliest\_and\_latest(["2023-04-15", "2021-09-01", "2022-12-30"])

# Output: ("2021-09-01", "2023-04-15")

earliest\_and\_latest(["2000-01-01", "2000-01-01"])

# Output: ("2000-01-01", "2000-01-01")

## Exercize: Python Library (Data Analysis)

The following Python code uses the pandas library to create a table of student data.

import pandas as pd

import numpy as np

# Define the data

data = {

"ID": [101, 102, 103, 104, 105, 106],

"Name": ["Alice", "Bob", "Charlie", "Diana", "Eva", "Frank"],

"Department": ["HR", "Finance", np.nan, "IT", "Finance", "HR"],

"Score": [90, np.nan, 85, 78, 88, np.nan]

}

# Create the DataFrame

df = pd.DataFrame(data)

# Display the DataFrame

print(df.head())

Questions:

* How many rows does the table contain?
* How many columns does the table contain?
* What is the Department of the employee named “Diana”?
* What value is missing (NaN) for the employee with ID 102?
* Write exactly what is printed by the last line (print(df.head())), reproducing the table as it would appear on the screen.
* Write the code to remove all rows where the Department is missing.
* Write the code to replace missing scores (Score) with the value 0.