Sharif University of Technology
Department of Computer Engineering

Fundamentals of Programming

Python Language

Arman Malekzadeh
PhD Candidate in Artificial Intelligence

Table of contents

Combination of Loops and Conditional Logic

Exercise 1: Greatest Common Divisor

Exercise 1: Greatest Common Divisor

```
def gcd(a, b):
    result = 1
    for i in range(1, min(a, b) + 1):
        if a % i == 0 and b % i == 0:
            result = i
    return result
```

Exercise 1: Greatest Common Divisor

```
def gcd(a, b):
    result = 1
    for i in range(1, min(a, b) + 1):
        if a % i == 0 and b % i == 0:
            result = i
    return result
```

Tracing the code for gcd(20, 8) gives:

```
i = 1, result = 1
i = 2, result = 2
i = 3, result = 2
i = 4, result = 4
i = 5, result = 4
i = 6, result = 4
i = 7, result = 4
i = 8, result = 4
```

Euclidean Algorithm for Finding the Greatest Common Divisor

Lemma

The greatest common divisor of two numbers does not change if the larger number is replaced by its difference with the smaller number.

1. Divide the larger number by the smaller number.
2. Use remainder as new smaller number.
3. Repeat until remainder is zero.
4. The GCD is the last non-zero remainder.

Example: GCD(48,18)

Let's find out GCD(48,18) using Euclidean algorithm:
Start with:

$48 = 2 * 18 + 12$

Then use 18 as our new larger number and 12 as our new smaller number:

$18 = 1 * 12 + 6$

Continue until we get no remainder:

$12 = 2 * 6 + 0$ So, GCD(48,18) is 6.

Hence proved that Euclidean algorithm is an efficient method to find GCD of two numbers.

Exercise 1: Greatest Common Divisor

Exercise 1: Greatest Common Divisor

```
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a
```

Tracing the code for gcd(20, 8) gives:

```
a = 20, b = 8
a = 8, b = 4
a = 4, b = 0
```

Docstring in Python

- A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition.
- It is used to define what a function does, and how to use it.
- It is surrounded by triple quotes (single or double).

Docstring in Python: Example

```
def gcd(a, b):
    """Return the greatest common divisor of a and b.
    """
    while b != 0:
        a, b = b, a % b
    return a
```

Prerequisite: ASCII Codes

- ASCII stands for American Standard Code for Information Interchange.
- It is a numeric value given to different characters and symbols, for computers to store and manipulate.
- For example, the ASCII value of the letter 'A' is 65. ('B': 66, 'C':67, 'a': 97, 'b': 98, 'c': 99)

Exercise 2: Julius Caesar's Encryption

- Julius Caesar used a simple encryption method to send messages to his generals.
- Each letter in the original message was replaced by a letter some fixed number of positions down the alphabet.
- For example, with a shift of 3, A would be replaced by D, B would become E, and so on.
- The last three letters of the alphabet would be replaced by A, B, and C.

Solution to Exercise 2: Julius Caesar's Encryption

```
def encrypt(text, shift):
    result = ""
    for i, char in enumerate(text):
        if (char.isupper()):
            result += chr((ord(char) + shift - 65) % 26 + 65)
        else:
            result += chr((ord(char) + shift - 97) % 26 + 97)
    return result
```

Exercise 3: Counting Vowels

Exercise 3: Counting Vowels

```
def count_vowels(text):
    """Return the number of vowels in text.
    """
    count = 0
    for char in text:
        if char in "aeiouAEIOU":
            count += 1
    return count
```

Exercise 4: Sum of Digits using Recursive Functions

```
def sum_digits(n):
    """Return the sum of the digits of n.
    """
    if n < 10:
        return n
    else:
        all_but_last, last = n // 10, n % 10
        return sum_digits(all_but_last) + last
```

Tracing the code for `sum_digits(1729)` gives:

```
all_but_last = 172, last = 9
all_but_last = 17, last = 2
all_but_last = 1, last = 7
all_but_last = 0, last = 1
```

Exercise 5: Palindrome

```
def is_palindrome(text):
    """Return whether text is a palindrome.
    """
    if len(text) <= 1:
        return True
    else:
        return text[0] == text[-1] and is_palindrome(text[1:-1])
```

Tracing the code for `is_palindrome('noon')` gives:

```
text = 'noon'
text = 'oo'
text = ''
```

Exercise 6: Multiple Word Palindromes

- “go dog” is an example of a multiple word palindrome.
- Write a function that takes a string and determines whether it is a palindrome when spacing is ignored.
- The function should ignore spacing, capitalization and punctuation.

Solution to Exercise 6: Multiple Word Palindromes

```
    def is_multuple_palindrome(text):
        """Return whether text is a palindrome.
        """
        text = text.lower()
        text = text.replace(" ", "")
        text = text.replace(",", "")
        text = text.replace(".", "")
        return is_palindrome(text)
```

Exercise 7: Compute the Perimeter of a Polygon

- A polygon is a closed shape with straight sides.
- The perimeter of a polygon is the sum of the lengths of its sides.
- Write a function that takes a list of tuples, where each tuple contains the x and y coordinate of a vertex of a polygon in order, and returns the perimeter of the polygon.

Solution to Exercise 7: Compute the Perimeter of a Polygon

```
    def distance(p1, p2):
        """Return the distance between points p1 and p2.
        """
        return ((p1[0] - p2[0]) ** 2 + (p1[1] - p2[1]) ** 2) ** 0.5

    def perimeter(vertices):
        """Return the perimeter of a polygon.
        """
        result = 0
        for i, vertex in enumerate(vertices):
            result += distance(vertices[i], vertices[(i + 1) % len(vertices)])
        return result
```

Exercise 8: Binary Search

- Binary search is a search algorithm that finds the position of a target value within a sorted array.
- The algorithm compares the target value to the middle element of the array.
- If they are not equal, the half in which the target cannot lie is eliminated and the search continues on the remaining half, again taking the middle element to compare to the target value, and repeating this until the target value is found.

Solution to Exercise 8: Binary Search

```
def binary_search(array, target):
    """Return the index of target in array, or -1 if target is not in array
                                                    .
    """
    low, high = 0, len(array) - 1
    while low <= high:
        mid = (low + high) // 2
        if array[mid] == target:
            return mid
        elif array[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1
```

Tracing the Binary Search Algorithm

- Trace the binary search algorithm for the following array and target value:
  - `array = [1, 15, 32, 45, 78, 81, 93]`
  - `target = 81`

```
low = 0 (1), high = 6 (93), mid = 3 (45)
low = 4 (78), high = 6 (93), mid = 5 (81)
low = 6 (81), high = 6 (81), mid = 6 (81)
```

References

References I

[1] B Downey, A. (2015). Think Python: How to Think Like a Computer Scientist-2nd Edition.

[2] Deitel, H. M., & Deitel, P. J. (2004). C: How to program. Pearson Educacion.

Sharif University of Technology
Department of Computer Engineering

Arman Malekzadeh

Fundamentals of Programming

Python Language