Python Programming

Strings processing

Mariusz Dzieńkowski

Institute of Computer Science
Lublin University of Technology
m.dzienkowski@pollub.pl

Strings and characters

- A string is a sequence of characters and can include text and numbers.
- String values must be enclosed in matching single quotes(') or double quotes(").
- Python treats characters and strings the same way.
- Python does not have a data type for characters.

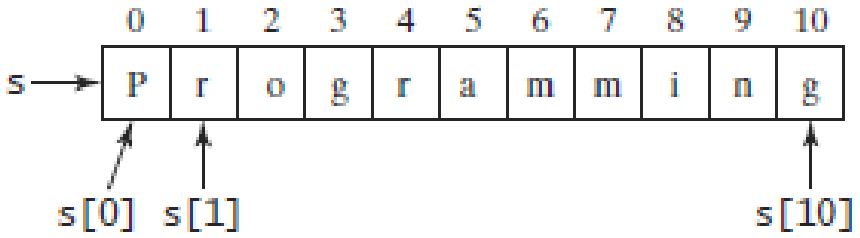
  A single-character string represents a character.

e.g.

```
letter = 'A'                    # Same as letter = "A"
numChar = '5'                   # Same as numChar = "5"
message = "Good morning"        # Same as message= 'Good morning'
```

Functions for strings

- `len(s)` – returns the number of the characters in a string
- `max(s)` , `min(s)` – to return the largest or smallest character in a string
- Index operator [ ] – a character in the string can be accessed through the index operator using the syntax: `s[index]`; the range of indexes - from `0` to `len(str)-1`



Functions for strings

- Strings are immutable, it is not possible to change their contents, e.g. `s[2] = 'A'`
- The slicing operator `[start : end]` – returns a slice of the string using the syntax `s[start : end]`, e.g.

```
>>> s = "Welcome"
>>> s[1 : 5]
>>> 'elco'
```

`s[1 : 4]` – returns a substring from index 1 to index 3

- The starting index or ending index may be omitted.
- In this case, be default the starting index is 0 and the ending index is the last index.

Functions for strings

- The concatenation (+) operator - to join or concatenate two strings, e.g.

```
>>> s1 = ”Welcome”
>>> s2 = ”Python”
>>> s3 = s1 + ” to ” + s2
>>> s3
’Welcome to Python’
```

- The repetition (*) operator - to concatenate the same string multiple times

```
>>> s4 = 3 * ”Welcome”
>>> s4
’WelcomeWelcomeWelcome’
```

Problem 1

Vowel removing – a program that removes all the vowels from a string

e.g.

input → ”alphabet”

output → ”lphbt”

- Guidelines:
  - Write a function that has one string parameter and returns string without vowels

Problem 1 - solution

- Source code

```
def remove_vowels(s):
    vowels = "aeiouAEIOU"
    s_sans_vowels = ""
    for x in s:
        if x not in vowels:
            s_sans_vowels += x
    return s_sans_vowels

if remove_vowels("compsci") == "cmpsc":
    print("True")
if remove_vowels("aAbEefIijOopUus") == "bfjps":
    print("True")
```

Problem 2

Cleaning up strings – a program that strips punctuation from a string

e.g.

input → '^a#b@c\d/e\\'

output → 'abcde'

- Guidelines:
  - Remember that strings are immutable, so we cannot change the string with the punctuation – it is necessary to traverse the original string and create a new string, omitting any punctuation
  - Punctuations: "!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~"

Problem 2 - solution

- Source code

```
punctuation = "!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~"

def remove_punctuation(s):
    s_sans_punct = ""
    for letter in s:
        if letter not in punctuation:
            s_sans_punct += letter
    return s_sans_punct
```

Problem 3

Checking Palindromes – a program that checks whether a string is a palindrome (if it reads the same forward and backward)

- Problem solution:
  - checking whether the first character in the string is the same as the last character
  - if so, then checking whether the second character is the same as the second-to-last character
  - continue proces until mismatch is found or all characters in the string are checked, exept for the middle character if the sting has an old number of characters
- Guidelines:
  - Prompt the user to enter a string
  - Report wheather the string is a palindrome

Problem 3 - solution

- Source code

```
def main():
    s = input("Enter a string: ").strip()

    if isPalindrome(s):
        print(s, "is a palindrome")
    else:
        print(s, "is not a palindrome")

def isPalindrome(s):
    low = 0
    high = len(s)-1

    while low < high:
        if s[low] != s[high]:
            return False
        low += 1
        high -= 1
    return True

main()
```

Testing strings

Function	*Description*
`isalnum(): bool`	Returns True if characters in this string are alphanumeric and there is at least one character.
`isalpha(): bool`	Returns True if characters in this string are alphabetic and there is at least one character.
`isdigit() :bool`	Returns True if this string contains only number characters.
`isidentifier() :bool`	Returns True if this string is a Python identifier.
`islower() :bool`	Returns True if all characters in this string are lowercase letters and there is at least one character.
`isupper() :bool`	Returns True if all characters in this string are upperrcase letters and there is at least one character.
`isspace() :bool`	Returns True if this string contains only whitespace characters.

Examples of using the string testing methods

```
>>> s = ”Welcome to python”
>>> s.isalnum()
>>> False
>>> ”Welcome”.isalpha()
>>> True
>>> ”2012”.isdigit()
>>> True
>>> ”first Number”.isidentifier()
>>> False
>>> s.islower()
>>> True
>>> s.isspace()
>>> False
```

Comparing strings

- Python compares strings by comparing their corresponding characters, and it does this by evaluating the characters' numeric codes.
- Comparison operators: ==, !=, >, >=, <, <=

  e.g.

```
>>> "green" == "glow"
False
>>> "green" != "glow"
True
>>> "ab" <= "abc"
True
```

Searching for substrings

Function	*Description*
`endswith(s1: str): bool`	Returns True if the string ends with the substring s1.
`startswith(s1: str): bool`	Returns True if the string starts with the substring s1.
`find(s1): int`	Returns the lowest index where s1 starts in this string, or -1 if s1 is not found in this string.
`rfind(s1): int`	Returns the highest index where s1 starts in this string, or -1 if s1 is not found in this string.
`count(substring): int`	Returns the number of non-overlapping occurrences of this substring.

Converting strings

Function	*Description*
`capitalize(): str`	Returns a copy of this string with only the first character capitalized.
`lower(): str`	Returns a copy of this string with all letters converted to lowercase.
`upper(): str`	Returns a copy of this string with all letters converted to uppercase.
`title(s1): str`	Returns a copy of this string with the first letter capitalized in each word.
`swapcase(old, new): str`	Returns a copy of this string in which lowercase letters are converted to uppercase and uppercase to lowercase.
`replace(old, new): str`	Returns a new string that replaces all the occurrences of the old string with a new string.

Examples of using the string converting methods

```
>>> s = ”welcome to python”
>>> s1 = s.capitalize()
>>> s1
’Welcome to python’
>>> s2 = s.title()
>>> s2
’Welcome To Python’
>>> s= ”New England”
>>> s3 = s.swapcase()
>>> s3
’nEW eNGLAND’
```

Stripping Whitespace Characters from a String

Function	*Description*
`lstrip(): str`	Returns a string with the leading whitespace characters removed.
`rstrip(): str`	Returns a string with the trailing whitespace characters removed.
`strip(): str`	Returns a string with the starting and trailing whitespace characters removed.

- The characters: ' ', \t, \f, \r, \n are called the whitespace characters

```
>>> s = " Welcome to Python\t"
>>> s1 = s.lstrip()
>>> s1
'Welcome to Python\t'
```

Problem 4

Check password - app that checks whether a string is a valid password

- The password rules are as follows:
  - A passowrd must have at least eight characters
  - A password must consist of only letters and digits
  - A password must contain at least two digits
- Guidelines:
  - Prompt the user to enter a password
  - Display `valid password` if the rules are followed or `invalid password` otherwise

Problem 4 - solution

- Source code

```
def main():
    p = input("Enter a passowrd: ").strip()

    if checkPassword(p):
        print(p, "Valid password")
    else:
        print(p, "Invalid password")

def checkPassword(p):
    if len(p)<8:
        return False
    i = 0
    while i<len(p):
        if not p[i].isalpha() and not p[i].isdigit():
            return False
        i += 1
    i = d = 0
    while i<len(p):
        if p[i].isdigit():
            d += 1
        i += 1
    if d<2:
        return False
    return True
```

Problem 5

Conversion between number systems - app that converts a number from one number system to another

- Guidelines:
  - Define the **convert()** function for conversion:
    - Prompt the user to enter a base of a number
    - Check if a base is in the range 2 .. 36
    - Prompt the user to enter a number in appropriate number system
    - Convert an entering number and display it in binary, octal, decimal and hexadecimal system
  - Define the **`parse(n,b)`** function for checking if characters used in a number are appopriate; function should return `True` if the number is valid, otherwise `False`
    - Check if the number has prefix (’0x’, ’0o’, ’0b’) or not; use `startswith` function to search a substring (prefix) in a string (number)

Problem 5

- Here is a sample run of the program:

```
Enter a base: 2
Enter a number: 0b101

Binary form is  0b101
Octal form is  0o5
Decimal form is  5
Hexadecimal form is  0x5
```

```
Enter a base: 8
Enter a number: 0o17

Binary form is  0b1111
Octal form is  0o17
Decimal form is  15
Hexadecimal form is  0xf
```

```
Enter a base: 16
Enter a number: ff

Binary form is  0b11111111
Octal form is  0o377
Decimal form is  255
Hexadecimal form is  0xff
```

```
Enter a base: 4
Enter a number: 103

Binary form is  0b10011
Octal form is  0o23
Decimal form is  19
Hexadecimal form is  0x13
```

```
Enter a base: 8
Enter a number: 0o18
The number contains invalid characters!
```

Problem 5 - solution

- Source code

```
def convert():
    base = int(input("enter a base: "))
    if 1<base<=36:
        num = input("enter a number: ")
        if parse(num,base):
            num = int(num,base)
            print("\nbinary form is ", bin(num))
            print("octal form is ", oct(num))
            print("decimal form is ", int(num))
            print("hexadecimal form is ", hex(num))
        else:
            print("The number contains invalid characters!")
    else:
        print("The base is out of range <2 - 36>!")

def parse(n,b):
    s='0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    sb=s[:b]
    if n.startswith('0x') or n.startswith('0o') or n.startswith('0b'):
        n=n[2:len(n)]
    n=n.upper()
    for i in n:
        if i not in sb:
            return False
    return True

convert()
```

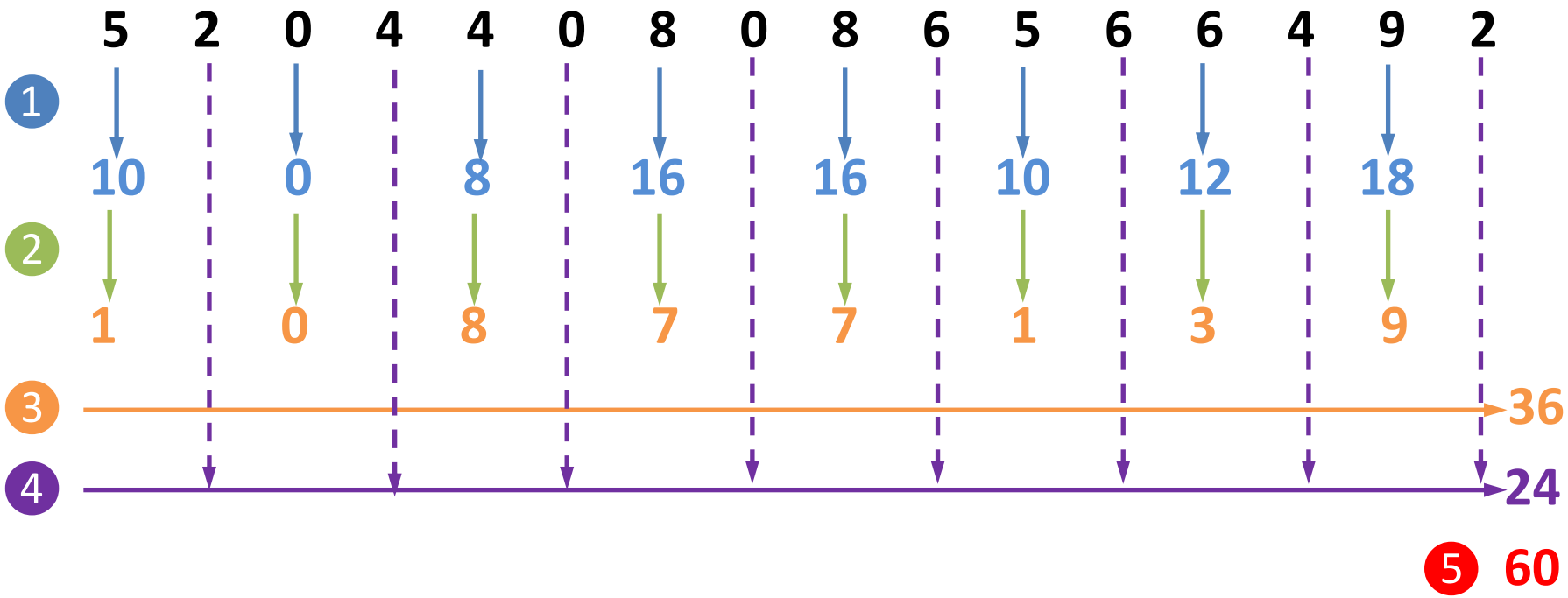
Problem 6

Credit Card Number Validation (The Luhn Algorithm) - app that checks whether a credit card number is a valid or invalid

- Description of the algorithm:
  - Step 1. Double every even digit from right to left.
  - Step 2. If doubling of a digit results in a two-digits number, add up the two digits to get a single-digit number.
  - Step 3. Add up all single-digit numbers from step 2.
  - Step 4. Add all digits in the even places from right to left in the card number.
  - Step 5. Sum the results from steps 3 and 4. If the sum is divisible by 10, the card number is valid, otherwise it is valid.

Problem 6

Credit Card Number Validation



Problem 6

Credit Card Number Validation

- Design your program to use the following functions:
  - Get the result from Step 3

    `sumOfDoubleEvenPlace(number)`
  - Return sum of odd place digits in number

    `sumOfOddPlace(number)`
  - Retrun true if the card number is valid

    `isValid(number)`
  - Check 3 example numbers:

    `main('4388576018402626')`

    `main('4388576018410707')`

    `main('5204408086566492')`

Problem 6 - solution

- Source code

```
def sumOfDoubleEvenPlace(number):
    sum=0
    for i in range(0,len(number),1):
        if i%2==0:
            x = int(number[i])*2
            if x//10==1:
                x = 1 + x%10
            sum += x
    return sum

def sumOfOddPlace(number):
    sum=0
    for i in range(0,len(number),1):
        if i%2!=0:
            sum += int(number[i])
    return sum
```

```
def isValid(number):
    sum = sumOfDoubleEvenPlace(number) + sumOfOddPlace(number)
    if sum%10==0:
        return True
    else:
        return False
```

Problem 6 - solution

- Source code

```
def main(number):
    #number = input("Enter a credit card number: ").strip()

    if isValid(number):
        print(number, "- valid number")
    else:
        print(number, "- invalid number")
```

```
main('4388576018402626')
main('4388576018410707')
main('5204408086566492')
```