Python Programming 1

(CP1S02005E)

Białystok University of Technology
Faculty of Electrical Engineering

Industry Digitization, semester II
Academic year 2024/2025

Lecture no. 06 (09.04.2025)

Jarosław Forenc, PhD

Topics

- Test no. 1
  - solutions to example tasks

Python - test no 1 (task 1, version 1)

- Place your first and last name, student ID number, date, course code (CP1S02005E), class type (Specialist Workshop), and the name of the IDE (Visual Studio Code) at the beginning of the program's source code as a comment. [5 pts.]

```
# Author: John Smith
# Student ID: 123456
# Date: 09-04-2025
# Course Code: CP1S02005E
# Class Type: Specialist Workshop
# IDE: Visual Studio Code
```

Python - test no 1 (task 1, version 2)

- Place your first and last name, student ID number, date, course code (CP1S02005E), class type (Specialist Workshop), and the name of the IDE (Visual Studio Code) at the beginning of the program's source code as a comment. [5 pts.]

```
"""
Author: John Smith
Student ID: 123456
Date: 09-04-2025
Course Code: CP1S02005E
Class Type: Specialist Workshop
IDE: Visual Studio Code
"""
```

- instead of three quotation marks, you can use three apostrophes

Python - test no 1 (task 2, version 1)

- Calculate the perimeter of an ellipse L with parameters c and d. Use the appropriate constant for the number π. Read the values of parameters c and d from the keyboard. Display the result with an accuracy of 3 decimal places. [15 pts.]

$$L = \pi\left(1{,}5(c+d) - \sqrt{cd}\right)$$

```
import math
c = float(input("Enter the value of parameter c: "))
d = float(input("Enter the value of parameter d: "))
L = math.pi * (1.5 * (c + d) - math.sqrt(c * d))
print(f"The circumference of the ellipse is: {L:.3f}")
```

```
Enter the value of parameter c: 2.5
Enter the value of parameter d: 5
The circumference of the ellipse is: 24.236
```

Python - test no 1 (task 2, version 2)

- Calculate the perimeter of an ellipse L with parameters c and d. Use the appropriate constant for the number π. Read the values of parameters c and d from the keyboard. Display the result with an accuracy of 3 decimal places. [15 pts.]

$$L = \pi\left(1{,}5(c+d) - \sqrt{cd}\right)$$

```
import math
try:
    c = float(input("Enter the value of parameter c: "))
    d = float(input("Enter the value of parameter d: "))
except ValueError:
    print("Invalid input data!")
else:
    L = math.pi * (1.5 * (c + d) - math.sqrt(c * d))
    print(f"The circumference of the ellipse is: {L:.3f}")
```

Python - test no 1 (task 3)

- The cost of car insurance is a certain percentage of its value and depends on the driver's age. Drivers are divided into five groups (table).

Group	Age range	Percentage
0	18 to 25 years	8.25%
1	over 25 to 35 years	7.25%
2	over 35 to 45 years	6.25%
3	over 45 to 55 years	6.00%
4	over 55 years	6.50%

- Read the car's value and the driver's age. Based on the driver's age, determine and display the group they belong to. Then, calculate and display the cost of car insurance (as a real number). Display the cost with an accuracy of 2 decimal places and add the currency symbol. [15 pts.]

Python - test no 1 (task 3)

```
value = float(input("Enter the value of the car: "))
age = int(input("Enter the driver's age: "))

if 18 <= age <= 25:
    group = 0
    cost = 8.25 / 100 * value
elif 26 <= age <= 35:
    group = 1
    cost = 7.25 / 100 * value
elif 36 <= age <= 45:
    group = 2
    cost = 6.25 / 100 * value
elif 46 <= age <= 55:
    group = 3
    cost = 6.00 / 100 * value
else:
    group = 4
    cost = 6.50 / 100 * value

print(f"Group: {group}")
print(f"Insurance cost: {cost:.2f} EUR")
```

Python - test no 1 (task 3)

```
value = float(input("Enter the value of the car: "))
age = int(input("Enter the driver's age: "))
if 18 <= age <= 25:
    group = 0
    cost = 8.25 / 100 * value
elif 26 <= age <= 35:
    group = 1
    cost = 7.25 / 100 * value
elif 36 <= age <= 45:
    group = 2
    cost = 6.25 / 100 * value
elif 46 <= age <= 55:
    group = 3
    cost = 6.00 / 100 * value
else:
```

```
Enter the value of the car: 35000
Enter the driver's age: 23
Group: 0
Insurance cost: 2887.50 EUR
```

Python - test no 1 (task 4)

- Create a list containing n pseudo-random numbers ranging from 1 to 50. Display the list on the screen. Read the value of n from the keyboard. [6 pts.]
- Calculate and display the arithmetic mean of all numbers in the list. [4 pts.]
- Create two new lists. Copy numbers less than or equal to the mean into the first list and numbers greater than the mean into the second list. Display the contents of both lists on the screen. [10 pts.]

Total points: [20 pts.]

Python - test no 1 (task 4)

```
import random

n = int(input("Enter a number n: "))

numbers = [random.randint(1, 50) for _ in range(n)]

print("Pseudorandom numbers:")
print(numbers)

average = sum(numbers) / n
print(f"Arithmetic mean: {average:.2f}")

less = [num for num in numbers if num <= average]
greater = [num for num in numbers if num > average]

print("Numbers less than or equal to the mean:")
print(less)

print("Numbers greater than the mean:")
print(greater)
```

Python - test no 1 (task 4)

```
import random

n = int(input("Enter a number n: "))

numbers = [random.randint(1, 50) for _ in range(n)]

print("Pseudorandom numbers:")
print(numbers)

average = sum(numbers) / n
print(f"Arithmetic mean: {average:.2f}")

less = [num for num in numbers if num <= average]
```

```
Enter a number n: 10
Pseudorandom numbers:
[26, 46, 13, 18, 42, 28, 37, 3, 33, 17]
Arithmetic mean: 26.30
Numbers less than or equal to the mean:
[26, 13, 18, 3, 17]
Numbers greater than the mean:
[46, 42, 28, 37, 33]
```

Python - test no 1 (task 5)

- A group of people visited several cities in Europe during the summer. Define a dictionary where the keys are people's names, and the values are lists of cities they visited. One person could have visited multiple cities. [4 pts.]
- Define a second dictionary where the keys are city names, and the values are the distances of those cities from Bialystok. Use the following cities and distances: Berlin - 659 km, Paris - 1528 km, London - 1588 km, Rome - 1482 km, Prague - 692 km, Oslo - 1070 km. [4 pts.]
- For each person, display in a single line: the person's name, the visited cities, and the total distance from Bialystok to those cities. [10 pts.]
- Display the name of the person whose total travel distance is the greatest. [7 pts.]

Total points: [25 pts.]

Python - test no 1 (task 5)

```
trips = {
    "Paul": ["Berlin", "Paris", "Rome"],
    "Kate": ["London", "Prague"],
    "Betty": ["Oslo", "Berlin"],
    "John": ["Prague", "Rome", "Berlin", "London"],
}
distances = {

    "Berlin": 659,
    "Paris": 1528,
    "London": 1588,
    "Rome": 1482,
    "Prague": 692,
    "Oslo": 1070
}
```

Python - test no 1 (task 5)

```
max_person = ""
max_distance = 0

for person, cities in trips.items():
    text = person + ": "
    total = 0

    for city in cities:
        text += (city + ", ")
        total += distances[city]

    text += ("distance: " + str(total))
    print(text)

    if total > max_distance:
        max_distance = total
        max_person = person

print(f"Highest total distance: {max_person}")
```

Python - test no 1 (task 5)

```
max_person = ""
max_distance = 0

for person, cities in trips.items():
    text = person + ": "
    total = 0

    for city in cities:
        text += (city + ", ")
        total += distances[city]

    text += ("distance: " + str(total))
    print(text)

    if total > max_distance:
```

```
Paul: Berlin, Paris, Rome, distance: 3669
Kate: London, Prague, distance: 2280
Betty: Oslo, Berlin, distance: 1729
John: Prague, Rome, Berlin, London, distance: 4421
Highest total distance: John
```

Python - test no 1 (task 6)

- Three lists contain the names of people who participated in three rounds of a sports competition. The same individuals may appear on multiple lists. [6 pts.]
- Display an alphabetically sorted list of unique individuals who participated in at least one round. Names should not be repeated. [14 pts.]

Total points: [20 pts.]

Python - test no 1 (task 6, version 1)

```
group1 = ["Kate", "Peter", "John", "Ellen", "Martin"]
group2 = ["John", "Alex", "Paul", "Ellen"]
group3 = ["Martin", "John", "Grace", "Olivia"]

all_groups = set()

all_groups.update(group1)
all_groups.update(group2)
all_groups.update(group3)

all_groups = sorted(all_groups)

print("List of participants:")
for person in all_groups:
    print(person)
```

```
List of participants:
Alex
Ellen
Grace
John
Kate
Martin
Olivia
Paul
Peter
```

Python - test no 1 (task 6, version 2)

```
group1 = ["Kate", "Peter", "John", "Ellen", "Martin"]
group2 = ["John", "Alex", "Paul", "Ellen"]
group3 = ["Martin", "John", "Grace", "Olivia"]

all_groups = sorted(set(group1 + group2 + group3))

print("List of participants:")
for person in all_groups:
    print(person)
```

```
List of participants:
Alex
Ellen
Grace
John
Kate
Martin
Olivia
Paul
Peter
```

End of lecture no. 6

Thank you for your attention!