Python Fundamentals

Автори на книгата:

Борис Костов

Мартин Хърсовски

Python Fundamentals Course @ Atlas IT Academy

Съдържание

Предговор

Настоящата книга се използва като официален учебник в курса за начинаещи програмисти "**Python Fundamentals**" в **Atlas IT Academy**.

Python е **мощен език за програмиране**, който е **лесен за научаване** и **забавен за употреба**!

Книгата запознава читателя с **по-сложни понятия** от сферата на програмирането – **цикли**, **списъци**, **функции**, **речници** и **обработка на текст**.

Нейна основна цел е да представи програмирането като **концепция** и **начин на мислене**, и да изгради основите на **аналичното мислене** и **уменията за решаване на проблеми**.

Книгата включва **150 практически задачи**. Първоначално, задачите са придружени от подробни стъпки, които трябва да се следват. С натрупването на знания и опит, обаче, начинаещите програмисти започват да подхождат по-самостоятелно към решаването на проблемите.

Книгата е подходяща за всички **деца**, които вече са направили своите **първи стъпки** в света на програмирането и компютрите и желаят да продължат своето развитие.

Цикли

В тази глава ще се запознаем със специални **конструкции**, наречени **цикли**, които позволяват **повторение** на **поредица от команди**.

Защо използваме цикли?

В реалния свят често се налага да извършваме едно и също действие, докато не постигнем даден **краен резултат**.

Например, **четенето на книга**. За да прочетем една книга, трябва да прочетем всяка една нейна страница, т.е. трябва да повторим действието **прочитане на една страница**, докато имаме **непрочетени страници**.

В компютърните науки, конструкции, които позволяват **повтарянето на действие**, докато е изпълнено дадено **условие**, се наричат **цикли**.

Видове цикли

Голяма част от езиците за програмиране разполагат с **няколко конструкции**, които осъществяват **подобен тип поведение**.

Конструкциите, които ще използваме в езика **Python**, са **for цикъл** и **while цикъл**. Те са сходни до голяма степен. Налични са някои минимални разлики. Общото между тях е, че **повтарят дадено действие**, докато е изпълнено **дадено условие**.

Цикъл for

Нека напишем програма, която принтира **числата от 1 до 10**.

Принтиране на числата от 1 до 10

```
print(1)
print(2)
print(3)
print(4)
print(5)
print(6)
print(7)
print(8)
print(9)
print(10)
```

Както вече споменахме, циклите се използват, когато желаем да повторим дадено действие.

Конструкцията за **for цикъл** е **синтактично най-сложната**, но **най-лесната за употреба**, тъй като цикълът разполага с **брояч** и е видимо колко **итерации** ще извърши. **Итерация** наричаме едно **завъртане на цикъла**.

Принтиране на числата от 1 до 10

```
for counter in range(1, 11):
    print(counter)
```

Променливата counter се декларира директно във **for цикъла**. Тя служи за **брояч** и приема различна стойност на всяка **итерация на цикъла**.

Функцията **range()** задава **начало и край на интервала от стойности**.

Кодът, написан след символа **:** (**двуеточие**) се нарича **тяло на цикъла**.

Общ вид на for цикъла

```
for x in range(n, m):
    # do something
```

Извикването на **range(1, 11)** ще генерира числата от **1 до 10 включително**, без да включва числото **11**.

Принтиране на числата от 1 до 10

```
for counter in range(1, 11):
    print(counter)
```

Функцията **range(n, m, step)** задава с каква **стъпка** ще се осъществява броенето. В общия случай броим със **стъпка едно**: **1**, **2**, **3**, **4**, **5**, … Понякога, обаче, се налага да броим със **стъпка две**: **1**, **3**, **5**, **7**, … Или пък със **стъпка десет**: **1**, **11**, **21**, **31**, … В нашия случай, нека броим със **стъпка едно**.

Принтиране на числата от 1 до 10 със стъпка 2

```
for x in range(1, 11, 2):
    print(x)
```

Извикването на функцията **range(1, 11, 2)** ще генерира **нечетните числа от 1 до 10 включително**. С други думи, примерът ще принтира: **1**, **3**, **5**, **7**, **9**.

Примерна задача

Напишете програма, която принтира **четните числа** в **интервала [1 ... 10]**.

Подход:

Използваме **for цикъл**. Преминаваме през всяко едно от числата в **интервала [1 ... 10]**. Ако **текущото число** се **дели на 2 без остатък**, то трябва да бъде принтирано на конзолата.

Решение на задачата

```
for number in range(1, 11):
    if number % 2 == 0:
        print(number)
```

Цикъл while

Конструкцията за **while цикъл** доста наподобява **if конструкцията**.

Принтиране на числата от 1 до 10

```
counter = 1

while counter <= 10:
    print(counter)
    counter = counter + 1
```

Цикъл while изпълнява кода, поместен в **тялото на цикъла**, докато е изпълнено **дадено условие**. За да зададем условие в **цикъл while**, може да е необходимо да **декларираме променлива предварително**.

След като инструкциите, поместени в тялото на цикъла, биват изпълнени, **изпълнението на програмата се връща към проверка на условието counter <= 10**. Тялото на цикъла променя стойността на **брояча counter** – на всяка итерация стойността на **counter** се **увеличава с единица**.

Циклите while и **for** работят по аналогичен начин. Синтактично, обаче, те се изписват по различен начин. Каква е разликата между тях и кой цикъл кога да използваме?

Цикъл for използваме, **когато предварително знаем колко итерации ще бъдат извършени**.

Цикъл while използваме, **когато предварително не знаем колко итерации ще бъдат извършени**.

Нека илюстрираме с примери какво означава предварително да знаем броя **итерации**.

Трябва да прочетем книга, съдържаща **100 страници**:

Пример за for цикъл

```
for page in range(1, 101):
    # read one page
```

Трябва да играем игра, **докато не изпълним определена задача**:

Пример за while цикъл

```
taskCompleted = False

while taskCompleted == False:
    # draw gamefield
    # move player
    # update score
```

Във втория случай **не знаем предварително колко стъпки или колко време ще е нужно** на играча, за да изпълни поставената задача.

Примерна задача

Напишете програма, която пресмята **броя на цифрите на дадено число**.

Примери:

- **Числото 123 е съставено от 3 цифри**
- **Числото 34354 е съставено от 5 цифри**.

Използваме **while цикъл**, тъй като предварително **не знаем от колко цифри е съставено числото**.

Всъщност, нашата програма трябва да отговори точно на този въпрос.

Докато числото е **по-голямо от 0**, **на всяка итерация разделяме числото на 10** и **добавяме единица към брояча на цифрите**.

Пример за while цикъл

```
import math

n = int(input())
digitsCount = 0

while n > 0:
    n = math.trunc(n / 10)
    digitsCount += 1

print(digitsCount)
```

Безкраен цикъл

Циклите трябва да съдържат част, която променя условието по някакъв начин (освен ако не използваме ключовите думи **break** или **return**). В противен случай, ще се натъкнем на явлението **безкраен цикъл**.

Безкраен цикъл

```
counter = 1

while counter <= 10:
    print(counter)
```

Ако кодът не разполага с команда, която променя **counter**, той винаги ще бъде равен на **1** и условието **counter <= 10** ще е изпълнено, практически, докато не спрем изпълнението на програмата. **Този пример ще принтира цифрата 1 непрестанно**, **докато не спрем изпълнението на програмата**.

Ключовата дума break

Ключовата дума **break** се използва, когато желаем да прекратим изпълнението на даден цикъл.

Въпреки, че условието на цикъла е **true** (**константа**, непроменяща се стойност), можем да избегнем безкрайния цикъл, когато добавим **допълнителна проверка** и използваме ключовата дума **break**. Ако тази проверка се оцени до **true**, изпълнението на програмата ще достигне до командата **break**, която **ще прекрати изпълнението на цикъла**.

Принтиране на числата от 1 до 10

```
x = 1

while True:
    print(x)
    x += 1

    if x == 11:
        break
```

Ключовата дума continue

Ключовата дума **continue** се използва, когато желаем да **пропуснем** дадена итерация на цикъла.

Принтиране на всички четни числа в интервала [1 ... 100]

```
for x in range(1, 101):
    if x % 2 == 1:
        continue
    print(x)
```

В примера разполагаме с проверка, която проверява дали едно число е **нечетно**. Ако текущата стойност на брояча е **нечетно число**, изпълнението на програмата достига до ключовата дума **continue** и **изпълнението на програмата преминава към следващата итерация на цикъла**, т.е. пропуска принтирането на числото.

Цикли: Упражнения

Задача 1. Print Numbers

Напишете програма, която принтира **числата** в **интервала [1 ... 10]**.

Изход:

- Програмата принтира **числата** в **интервала [1 ... 10]**.

Следвайте стъпките:

1. Използвайте **for цикъл**.
2. Създайте брояч с име **counter**.
3. Използвайте функцията **range()**, задавайки **начална стойност 1** и **крайна стойност 11**.
4. На всяка итерация цикълът ще принтира стойността на брояча.
5. На всяка итерация броячът ще се увеличава със **стъпка 1**.

Ето как програмата трябва да работи:

Вход	Изход
	1 2 3 4 5 6 7 8 9 10

Задача 2. Print Even Numbers

Напишете програма, която принтира всички **четни числа** в следния **интервал**: **[1 ... 10]**.

Изход:

- Програмата принтира **четните числа в интервала [1 ... 10]**.

Следвайте стъпките:

1. Използвайте **for цикъл**.
2. Създайте брояч с име **counter**.
3. Използвайте функцията **range()**, задавайки **начална стойност 1** и **крайна стойност 11**.
4. На всяка итерация цикълът ще проверява стойността на брояча и ако тя е **четно число** ще я принтира.
5. На всяка итерация броячът ще се увеличава със **стъпка 1**.

Ето как програмата трябва да работи:

Вход	Изход
	2 4 6 8 10

Задача 3. Print Odd Numbers

Напишете програма, която принтира всички **нечетни числа** в следния **интервал**: **[1 ... 10]**.

Изход:

- Програмата принтира **нечетните числа в интервала [1 ... 10]**.

Следвайте стъпките:

1. Използвайте **for цикъл**.
2. Създайте брояч с име **counter**.
3. Използвайте функцията **range()**, задавайки **начална стойност 1** и **крайна стойност 11**.
4. На всяка итерация цикълът ще проверява стойността на брояча и ако тя е **нечетно число** ще я принтира.
5. На всяка итерация броячът ще се увеличава със **стъпка 1**.

Ето как програмата трябва да работи:

Вход	Изход
	1 3 5 7 9

Задача 4. Print Numbers 1 to N

Напишете програма, която принтира **числата в интервала [1 ... N]**, където **N** е **произволно число** и **N ≥ 1**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **числата в интервала [1 ... N]**.

Следвайте стъпките:

1. Използвайте **for цикъл**.
2. Създайте брояч с име **counter**.
3. Използвайте функцията **range()**, задавайки **начална стойност 1** и **крайна стойност N**.
4. На всяка итерация цикълът ще принтира стойността на брояча.
5. На всяка итерация броячът ще се увеличава със **стъпка 1**.

Ето как програмата трябва да работи:

Вход	Изход
2	1 2
Вход	**Изход**
4	1 2 3 4
Вход	**Изход**
5	1 2 3 4 5

Задача 5. Print Numbers N to 1

Напишете програма, която принтира **числата в интервала [N ... 1]**, където **N** е **произволно число** и **N ≥ 1**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **числата в интервала [N ... 1]**.

Следвайте стъпките:

1. Използвайте **for цикъл**.
2. Създайте брояч с име **counter**.
3. Използвайте функцията **range()**, задавайки **начална стойност N** и **крайна стойност 1**.
4. На всяка итерация цикълът ще принтира стойността на брояча.
5. На всяка итерация броячът ще намалява със **стъпка -1**.

Ето как програмата трябва да работи:

Вход	Изход
2	2 1
Вход	**Изход**
4	4 3 2 1
Вход	**Изход**
5	5 4 3 2 1

Задача 6. Print Numbers N to M

Напишете програма, която принтира **числата в интервала [N ... M]**, където **N** и **M** са **произволни числа** и **M ≥ N**.

Вход:

- Въвеждаме **стойността на N**.
- Въвеждаме **стойността на M**.

Изход:

- Програмата принтира **числата в интервала [N ... M]**.

Следвайте стъпките:

1. Използвайте **for цикъл**.
2. Създайте брояч с име **counter**.
3. Използвайте функцията **range()**, задавайки **начална стойност N** и **крайна стойност M**.
4. На всяка итерация цикълът ще принтира стойността на брояча.
5. На всяка итерация броячът ще се увеличава със **стъпка 1**.

Ето как програмата трябва да работи:

Вход	Изход
2 4	2 3 4
Вход	**Изход**
4 6	4 5 6
Вход	**Изход**
2 6	2 3 4 5 6

Задача 7. Sum Numbers in Interval

Напишете програма, която принтира **сумата** на всички **числа в интервала [N ... M]**, където **N** и **M** са **произволни числа** и **N ≤ M**.

Вход:

- Въвеждаме **стойността на N**.
- Въвеждаме **стойността на M**.

Изход:

- Програмата принтира **сумата на числата**.

Ето как програмата трябва да работи:

Вход	Изход
2 3	5

Вход	Изход
1 5	15

Вход	Изход
1 10	55

Вход	Изход
1 100	5050

Задача 8. Multiply Numbers in Interval

Напишете програма, която принтира **произведението** на всички **числа в интервала [N ... M]**, където **N** и **M** са **произволни числа** и **N ≤ M**.

Вход:

- Въвеждаме **стойността на N**.
- Въвеждаме **стойността на M**.

Изход:

- Програмата принтира **произведението на числата**.

Ето как програмата трябва да работи:

Вход	Изход
2 3	6
Вход	**Изход**
1 5	120
Вход	**Изход**
1 10	3628800
Вход	**Изход**
1 20	2432902008176640000

Задача 9. Sum Even Numbers in Interval

Напишете програма, която принтира **сумата** на всички **четни числа в интервала [N ... M]**, където **N** и **M** са **произволни числа** и **N ≤ M**.

Вход:

- Въвеждаме **стойността на N**.
- Въвеждаме **стойността на M**.

Изход:

- Програмата принтира **сумата на четните числа**.

Ето как програмата трябва да работи:

Вход	Изход
2 5	6

Вход	Изход
1 8	20

Вход	Изход
1 10	30

Вход	Изход
1 20	110

Задача 10. Sum Odd Numbers in Interval

Напишете програма, която принтира **сумата** на всички **нечетни числа в интервала [N ... M]**, където **N** и **M** са **произволни числа** и **N ≤ M**.

Вход:

- Въвеждаме **стойността на N**.
- Въвеждаме **стойността на M**.

Изход:

- Програмата принтира **сумата на нечетните числа**.

Ето как програмата трябва да работи:

Вход	Изход
2 5	8

Вход	Изход
1 8	16

Вход	Изход
1 10	25

Вход	Изход
1 20	100

Задача 11. Multiplication Table

Напишете програма, която принтира **част от таблицата за умножение** за **числото N**, където **N** е **произволно число в интервала [1 ... 10]**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **част от таблицата за умножение**.

Ето как програмата трябва да работи:

Вход	Изход
5	5 x 0 = 0 5 x 1 = 5 5 x 2 = 10 5 x 3 = 15 5 x 4 = 20 5 x 5 = 25 5 x 6 = 30 5 x 7 = 35 5 x 8 = 40 5 x 9 = 45 5 x 10 = 50
Вход	**Изход**
8	8 x 0 = 0 8 x 1 = 8 8 x 2 = 16 8 x 3 = 24 8 x 4 = 32 8 x 5 = 40 8 x 6 = 48 8 x 7 = 56 8 x 8 = 64 8 x 9 = 72 8 x 10 = 80

Задача 12. Raise to Power

Напишете програма, която повдига **числото N на M-та степен**: N^M, където **N** и **M** са **произволни числа** и **N > 1**, и **M > 1**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **стойността на N^M**.

Ето как програмата трябва да работи:

Вход	Изход
2 5	32
Вход	**Изход**
8 6	262144
Вход	**Изход**
3 4	81
Вход	**Изход**
4 4	256

Задача 13. Digits Count

Напишете програма, която принтира **броя на цифрите на числото N**, където **N** е **произволно число**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **броя на цифрите на числото N**.

Следвайте стъпките:

1. Създайте променлива, която се казва **digits_count** и съхранява **стойността 0**.
2. Използвайте **while цикъл**.
3. На всяка итерация ще се изрязва най-дясната цифра на числото **N** и стойността на променливата **digits_count ще се увеличава с 1**.
4. Цикълът ще продължи своето изпълнение, докато стойността на **числото N** е **по-голяма** от **числото 0**.

Ето как програмата трябва да работи:

Вход	Изход
2	1
Вход	**Изход**
4545	4
Вход	**Изход**
123456789	9

Задача 14. Digits Sum

Напишете програма, която принтира **сумата на цифрите на числото N**, където **N** е **произволно число**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **сумата на цифрите на числото N**.

Следвайте стъпките:

1. Създайте променлива, която се казва **digits_sum** и съхранява **стойността 0**.
2. Използвайте **while цикъл**.
3. На всяка итерация ще се изрязва най-дясната цифра на числото **N** и нейната стойност ще се добавя към променливата **digits_sum**.
4. Цикълът ще продължи своето изпълнение, докато стойността на **числото N** е **по-голяма** от **числото 0**.

Ето как програмата трябва да работи:

Вход	Изход
2	2
Вход	**Изход**
4545	18
Вход	**Изход**
123456789	45

Задача 15. Digits Product

Напишете програма, която принтира **произведението на цифрите на числото N**, където **N** е **произволно число**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **произведението на цифрите на числото N**.

Ето как програмата трябва да работи:

Вход	Изход
2	2
Вход	**Изход**
45	20
Вход	**Изход**
1234	24

Задача 16. Factorial

Напишете програма, която пресмята **факториел** на **числото N**, където **N** е **произволно число** и **N > 1**.

Формула: **N! = N * (N – 1) * (N – 2) * (N – 3) * ... * 2 * 1**

Примери:

- **5! = 5 * 4 * 3 * 2 * 1**
- **7! = 7 * 6 * 5 * 4 * 3 * 2 * 1**
- **10! = 10 * 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1**

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **факториел на числото N**.

Ето как програмата трябва да работи:

Вход	Изход
5	120
Вход	**Изход**
6	720
Вход	**Изход**
10	3628800
Вход	**Изход**
4	24

Задача 17. Sum Calculation

Напишете програма, която спрямо **произволно число N** намира **стойността на израза**:

$$S = 1 + 1 / 2 + 1 / 3 + 1 / 4 + ... + 1 / N$$

Програмата трябва да принтира **стойността на S**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **стойността на израза**, спрямо **числото N**.

Ето как програмата трябва да работи:

Вход	Изход
50	4.499205338329423
Вход	**Изход**
100	5.187377517639621
Вход	**Изход**
120	6.368868287353396

Задача 18. Product Calculation

Напишете програма, която спрямо **произволно число N** намира **стойността на израза**:

$$P = 1 * 1 / 2 * 1 / 3 * 1 / 4 * ... * 1 / N$$

Програмата трябва да принтира **стойността на P**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **стойността на израза**, спрямо **числото N**.

Ето как програмата трябва да работи:

Вход	Изход
50	4.499205338329423
Вход	**Изход**
100	5.187377517639621
Вход	**Изход**
120	6.368868287353396

Задача 19. Prime Number

Напишете програма, която проверява дали **числото N** е **просто число**.

Просто число се нарича **естествено число**, което се **дели само и единствено на себе си и на числото 1**.

Примери:

- **Числото 5 е просто число**, тъй като негови **единствени делители** са **числата 1** и **5**.
- **Числото 24 не е просто число**, тъй като негови **делители** са **числата 1**, **2**, **3**, **4**, **6**, **12** и **24**.

Ето как програмата трябва да работи:

Вход	Изход
`6`	`False`
Вход	**Изход**
`7`	`True`
Вход	**Изход**
`1`	`False`
Вход	**Изход**
`11`	`True`

Задача 20. Prime Numbers in Interval

Напишете програма, която принтира **всички прости числа** в **интервала [N ... M]**, където **N** и **M** са **произволни числа** и **N ≤ M**.

Ето как програмата трябва да работи:

Вход	Изход
1 10	2 3 5 7
Вход	**Изход**
1 20	2 3 5 7 11 13 17 19
Вход	**Изход**
0 2	2

Задача 21. Perfect Number

Напишете програма, която проверява дали **числото N** е **съвършено число**, **N** е **произволно число**.

Съвършено число се нарича **естествено число**, което е **точна сума** от своите **по-малки делители**.

Пример:

- **Числото 6 е съвършено число**, тъй като е **равно** на **сумата от своите делители**: **1 + 2 + 3 = 6**.

Ето как програмата трябва да работи:

Вход	Изход
6	True
Вход	**Изход**
7	False
Вход	**Изход**
28	True

Задача 22. Perfect Numbers in Interval

Напишете програма, която принтира **всички съвършени числа в интервала [N ... M]**, където **N** и **M** са **произволни числа** и **N ≤ M**.

Ето как програмата трябва да работи:

Вход	Изход
1 10	6

Вход	Изход
1 100	6 28

Вход	Изход
1 1000	6 28 496

Задача 23. Fibonacci Sequence

Напишете програма, която принтира **първите N числа** от **редицата на Фибоначи**.

Редицата на Фибоначи започва със следните числа:

0, 1, 1, 2, 3, 5, 8, ...

В сила е **закономерност**, която гласи, че **всеки следващ елемент** в редицата е **сума на предишните два елемента**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **първите N числа на редицата**.

Ето как програмата трябва да работи:

Вход	Изход
5	0, 1, 1, 2, 3
Вход	**Изход**
6	0, 1, 1, 2, 3, 5
Вход	**Изход**
9	0, 1, 1, 2, 3, 5, 8, 13, 21

Задача 24. Reverse Digits

Напишете програма, която **обръща цифрите на числото N**, където **N** е **произволно число** и **N ≥ 10**.

Примери:

- **Числото 123** има следния **запис**: **1 * 100 + 2 * 10 + 3 * 1**.
- Числото, което трябва да получим е **321**.
- **Числото 321** има следния запис: **3 * 100 + 2 * 10 + 1 * 1**.

Ето как програмата трябва да работи:

Вход	Изход
2456	6542
Вход	**Изход**
8345	5438
Вход	**Изход**
36	63
Вход	**Изход**
5	5
Вход	**Изход**
123456789	987654321

Задача 25. Square Pattern

Напишете програма, която принтира **фигура**, спрямо въведено **число N**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **фигурата**, спрямо въведеното **число N**.

Ето как програмата трябва да работи:

Вход	Изход
3	*** *** ***
Вход	**Изход**
5	***** ***** ***** ***** *****
Вход	**Изход**
10	********** ********** ********** ********** ********** ********** ********** ********** ********** **********

Задача 26. Frame Pattern

Напишете програма, която принтира **фигура**, спрямо въведено **число N**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **фигурата**, спрямо въведеното **число N**.

Ето как програмата трябва да работи:

Вход	Изход
3	*** * * ***
Вход	**Изход**
5	***** * * * * * * *****
Вход	**Изход**
10	********** * * * * * * * * * * * * * * * * **********

Задача 27. Dollars Pyramid

Напишете програма, която принтира **фигура**, спрямо въведено **число N**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **фигурата**, спрямо въведеното **число N**.

Ето как програмата трябва да работи:

Вход	Изход
3	$ $$ $$$
Вход	**Изход**
5	$ $$ $$$ $$$$ $$$$$
Вход	**Изход**
10	$ $$ $$$ $$$$ $$$$$ $$$$$$ $$$$$$$ $$$$$$$$ $$$$$$$$$ $$$$$$$$$$

Задача 28. Christmas Tree

Напишете програма, която принтира **фигура**, спрямо въведено **число N**.

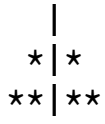
Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **фигурата**, спрямо въведеното **число N**.

Ето как програмата трябва да работи:

Вход	Изход
3	
Вход	**Изход**
5	
Вход	**Изход**
10	

Задача 29. Strong Number

Напишете програма, която проверява дали **числото N** е **силно число**.

Силно число се нарича **естествено число**, което е **равно на сумата от факториелите на неговите цифри**.

Пример:

- **Числото 145 е силно число**, тъй като в сила **равенството**:
  145 = 1! + 4! + 5! = 1 + 24 + 120 = 145

Ето как програмата трябва да работи:

Вход	Изход
120	False
Вход	**Изход**
145	True
Вход	**Изход**
2	True
Вход	**Изход**
112	False
Вход	**Изход**
40585	True

Задача 30. Strong Numbers in Interval

Напишете програма, която принтира **всички силни числа** в **интервала [N ... M]**, където **N** и **M** са **произволни числа** и **N ≤ M**.

Ето как програмата трябва да работи:

Вход	Изход
1 10	1 2
Вход	**Изход**
1 200	1 2 145
Вход	**Изход**
1 100000	1 2 145 40585

Списъци

В настоящата глава ще научим какво представлява **списъка**. Накратко, списъците са променливи, които съхраняват **множество от стойности**.

Защо използваме списъци?

Понякога се налага да обработваме **множество данни** – например, оценките на група курсисти в даден курс. Записвайки данните в променливи, ще трябва да съхраняваме по една променлива за всеки курсист. Операциите по намиране на най-висока оценка или намиране на средна оценка биха били трудни за реализация, тъй като бихме боравили с **N** на брой променливи, където **N** е броят курсисти.

Създаване на N на брой променливи

```
grade1 = 5.50
grade2 = 4.50
grade3 = 6
# …
gradeN = 5.50
```

За щастие, езиците за програмиране ни дават възможност да запишем **множество от стойности в една променлива**.

Създаване на списък

Следният код създава списък с име **grades** и записва **5 реални числа**, които репрезентират оценки на курсисти.

Списък с име grades, който съхранява 5 реални числа

```
grades = [5, 4, 6, 3, 4]
```

Списъкът **grades** съхранява **5 стойности**. Всяка стойност в списъка се нарича **елемент**. С други думи, това е списък, съставен от **5 елемента**.

Списъците могат да съхраняват и текстови стойности:

Списък с име names, който съхранява имена

```
names = ['George', 'Ben', 'Tom']
```

Както е видно от примерите, елементите на списъка се изреждат със **запетая** и се обграждат в **квадратни скоби** – **[ ]**.

Имената на списъците обикновено са **съществителни имена** в множествено число, тъй като списъците съхраняват множество от стойности: **students_grades**, **people_names**, **numbers**, **elements** и други.

Индексиране на елементите на списък

Нека използваме следния списък:

Списък с име numbers, който съхранява числата от 1 до 5

```
numbers = [1, 2, 3, 4, 5]
```

Можем да си представим, че в паметта на компютъра елементите на списъка изглеждат като **поредица от клетки**. Всеки **елемент** се достъпва по **номер**. Този номер се нарича **индекс**.

В програмирането, **броенето започва от 0**. Поради тази причина, ако трябва да номерираме елементите на списък, съставен от **5 елемента**, то тяхната номерация ще се осъществи с цифрите **0**, **1**, **2**, **3**, **4**. Първият елемент ще е на **позиция 0** (**индекс 0**), а последният – на **позиция 4** (**индекс 4**).

Индексиране на елементите на списък

```
numbers = [1, 2, 3, 4, 5]

numbers[0] = 5 # numbers: [5, 2, 3, 4, 5]
numbers[2] = 7 # numbers: [5, 2, 7, 4, 5]
numbers[4] = 9 # numbers: [5, 2, 7, 4, 9]

print(numbers[1]) # 2
print(numbers[3]) # 4
print(numbers[4]) # 9
```

В променливите можем да **записваме стойности**. Можем, също така, да **извлечем стойностите**, които сме записали в тях. По същият начин можем да работим с елементите на списъка, достъпвайки ги по **индекс** – можем да **присвоим стойност** на даден **елемент** на дадена **позиция** или пък да **прочетем стойността**, която сме присвоили.

Обхождане на елементите на списък

Преминаването през всеки елемент на списък се нарича **обхождане на елементите на списък**. За целта използваме **for цикъл**. Ще използваме брояча на **for цикъла**, за да индексираме елементите.

Обхождане на елементите на списъка numbers

```
numbers = [1, 2, 3, 4, 5]

for index in range(0, len(numbers)):
    print(numbers[index])
```

В горепосочения пример, **броячът** на **for цикъла** се казва **index**, тъй като той служи за **индексатор** на **елементите на списъка**, броенето започва от **0**. Крайната стойност, която броячът ще достигне, е **len(numbers) - 1**, тъй като при списък с **5 елемента**, индексите са **0**, **1**, **2**, **3**, **4**. На всяка итерация индексът се увеличава с **единица**.

Полезни функции

Всеки списък представлява **поредица от елементи от даден тип**.

При работа със списъци често се налага употребата на **стандартни операции** като:

- **Добавяне на елемент в края на списъка**;
- **Изтриване на елемент в края на списъка**;
- **Добавяне на елемент в началото на списъка**;
- **Изтриване на елемент в началото на списъка**;
- **Търсене на позицията на даден елемент в списъка**;
- ...

В тази секция ще разгледаме някои от **основните функционалности**, които са достъпни в езика **Python**, при работа със списъци:

- **Функцията append()**
- **Функцията pop()**
- **Функцията insert()**
- **Функцията index()**

Функцията append()

Функцията append() позволява добавянето на елемент в края на списъка. Функцията приема един аргумент – елементът, който ще бъде добавен в края на списъка.

Функцията append()

```
numbers = [1, 2, 3]

numbers.append(4)
print(numbers) # [1, 2, 3, 4]

numbers.append(5)
print(numbers) # [1, 2, 3, 4, 5]

fruits = ['Apple', 'Banana']

fruits.append('Strawberry')
print(fruits) # ['Apple', 'Banana', 'Strawberry']
```

Функцията pop()

Функцията pop() позволява изтриването на елемент в края на списъка.

Функцията pop()

```
numbers = [1, 2, 3, 4, 5, 6, 7]

numbers.pop(0)
print(numbers) # [1, 2, 3, 4, 5, 6]

numbers.pop()
print(numbers) # [1, 2, 3, 4, 5]

fruits = ['Apple', 'Banana', 'Strawberry', 'Orange']

fruits.pop()
print(fruits) # ['Apple', 'Banana', 'Strawberry']
```

Функцията може да приеме аргумент, указващ индекса, на елемента, който ще бъде изтрит.

Функцията pop()

```
numbers = [1, 2, 3, 4, 5]

numbers.pop(0)
print(numbers) # [2, 3, 4, 5]

numbers.pop(0)
print(numbers) # [3, 4, 5]

fruits = ['Apple', 'Banana', 'Orange']

fruits.pop(0)
print(fruits) # ['Banana', 'Orange']
```

Функцията insert()

Функцията insert() позволява добавянето на елемент в списъка. Функцията приема два аргумента – индексът, на който ще бъде добавен елемента и самия елемент, който ще бъде добавен.

Функцията insert()

```
numbers = [3, 4, 5]

numbers.insert(0, 2)
print(numbers) # [2, 3, 4, 5]

numbers.insert(0, 1)
print(numbers) # [1, 2, 3, 4, 5]

fruits = ['Apple', 'Orange']

fruits.insert(0, 'Banana')
print(fruits) # ['Banana', 'Apple', 'Orange']
```

Функцията index()

Функцията index() позволява търсенето на даден елемент в списъка. Функцията приема един аргумент – елементът, който търсим. Функцията връща позицията (индекса) на елемента в списъка.

Функцията index()

```
numbers = [1, 2, 3, 4, 5]

index = numbers.index(1)
print(index) # 0

index = numbers.index(4)
print(index) # 3

fruits = ['Apple', 'Orange']

index = fruits.index('Orange')
print(index) # 1
```

Дължина на списък

Всеки списък в **Python** пази информация за своята **дължина**.

Стойността на дължината (броят на елементи на списъка) е достъпна, чрез **функцията len()**.

Функцията len()

```
numbers = [1, 2, 3, 4, 5]

elements_count = len(numbers)
print(elements_count) # 5

fruits = ['Apple', 'Orange']

elements_count = len(fruits)
print(elements_count) # 2
```

Списъци: Упражнения

Задача 1. Practice Indexes

Напишете програма, която **създава списък** с елементи **числата от 1 до 5**.

Използвайки списъка, променете следните елементи:

- Присвоете **стойността 5** на елемента с **индекс 0**.
- Присвоете **стойността 9** на елемента с **индекс 2**.
- Присвоете **стойността 4** на елемента с **индекс 1**.
- Присвоете **стойността 6** на елемента с **индекс 3**.
- Присвоете **стойността 7** на елемента с **индекс 4**.

Изход:

- Програмата принтира **елементите на списъка**.

Ето как програмата трябва да работи:

Вход	Изход
	5 4 9 6 7

Задача 2. Print List

Напишете програма, която **създава списък** с елементи **числата от 1 до 5** и го принтира на конзолата.

Изход:

- Програмата принтира **елементите на списъка**.

Следвайте стъпките:

1. Създайте променлива, която се казва **numbers** и съхранява **списък**, съставен от **5 цели числа** – числата **1**, **2**, **3**, **4**, **5**.
2. Принтирайте стойността на променливата **numbers**.

Ето как програмата трябва да работи:

Вход	Изход
	`[1, 2, 3, 4, 5]`

Задача 3. Print List Elements

Напишете програма, която **създава списък** с елементи **числата от 1 до 5** и принтира неговите елементи.

Изход:

- Програмата принтира **елементите на списъка**.

Следвайте стъпките:

1. Създайте променлива, която се казва **numbers** и съхранява **списък**, съставен от **5 цели числа** – числата **1**, **2**, **3**, **4**, **5**.
2. Използвайте **for цикъл**.
3. Създайте брояч с име **index** и **начална стойност 0**.
4. Индексът ще се изменя от **0** до **дължината на списъка минус 1**.
5. На всяка итерация индексът ще се увеличава с **1**.

Ето как програмата трябва да работи:

Вход	Изход
	1 2 3 4 5

Задача 4. Print Even Numbers

Напишете програма, която създава следния **списък**:

[3, 4, 5, 8, 6, 1, 2, 5, 4, 3, 7, 6, 5, 4]

Програмата трябва да принтира всички **четни числа в списъка**.

Изход:

- Програмата принтира всички **четни числа в списъка**.

Следвайте стъпките:

1. Създайте променлива, която се казва **numbers** и съхранява **списъка**.
2. Използвайте **for цикъл**.
3. Създайте брояч с име **index** и **начална стойност 0**.
4. Индексът ще се изменя от **0** до **дължината на списъка минус 1**.
5. На всяка итерация индексът ще се увеличава с **1**.
6. На всяка итерация в цикъла ще се проверява дали текущият елемент в списъка е **четно число**.

Ето как програмата трябва да работи:

Вход	Изход
	4 8 6 2 4 6 4

Задача 5. Print Odd Numbers

Напишете програма, която създава следния **списък**:

[3, **4**, **5**, **8**, **6**, **1**, **2**, **5**, **4**, **3**, **7**, **6**, **5**, **4]**

Програмата трябва да принтира всички **нечетни числа в списъка**.

Изход:

- Програмата принтира всички **нечетни числа в списъка**.

Следвайте стъпките:

1. Създайте променлива, която се казва **numbers** и съхранява **списъка**.
2. Използвайте **for цикъл**.
3. Създайте брояч с име **index** и **начална стойност 0**.
4. Индексът ще се изменя от **0** до **дължината на списъка минус 1**.
5. На всяка итерация индексът ще се увеличава с **1**.
6. На всяка итерация в цикъла ще се проверява дали текущият елемент в списъка е **нечетно число**.

Ето как програмата трябва да работи:

Вход	Изход
	3 **5** **1** **5** **3** **7** **5**

Задача 6. Contains Element

Напишете програма, която създава следния **списък**:

[3, 6, 4, 5, 2, 8, 6, 1, 2, 5, 3, 4, 3, 7, 6, 5, 4, 8, 7]

Програмата трябва да проверява дали **даден елемент X** се среща **в списъка**.

Изход:

- Програмата принтира дали **даден елемент X** се среща **в списъка**.

Следвайте стъпките:

1. Създайте променлива, която се казва **numbers** и съхранява **списъка**.
2. Използвайте **for цикъл**.
3. Създайте брояч с име **index** и **начална стойност 0**.
4. Индексът ще се изменя от **0** до **дължината на списъка минус 1**.
5. На всяка итерация индексът ще се увеличава с **1**.
6. На всяка итерация ще се проверява дали текущият елемент е равен на елемента **X**, който търсим. Ако сме намерили елемент, равен на **X**, можем да прекратим търсенето с **ключовата дума break**.

Ето как програмата трябва да работи:

Вход	Изход
0	False
Вход	**Изход**
1	True
Вход	**Изход**
7	True
Вход	**Изход**
9	False

Задача 7. Elements Sum

Напишете програма, която създава следния **списък**:

[3, 6, 4, 5, 2, 8, 6, 1, 2, 5, 3, 4, 3, 7, 6, 5, 4, 8, 7]

Програмата трябва да намери **сумата на елементите на списъка**.

Изход:

- Програмата принтира **сумата на елементите на списъка**.

Следвайте стъпките:

1. Създайте променлива, която се казва **numbers** и съхранява **списъка**.
2. Създайте променлива, която се казва **sum** и съхранява **числото 0**.
3. Използвайте **for цикъл**.
4. Създайте брояч с име **index** и **начална стойност 0**.
5. Индексът ще се изменя от **0** до **дължината на списъка минус 1**.
6. На всяка итерация индексът ще се увеличава с **1**.
7. На всяка итерация към сумата ще се добавя стойността на текущия елемент.

Ето как програмата трябва да работи:

Вход	Изход
	89

Задача 8. Elements Product

Напишете програма, която създава следния **списък**:

[3, **6**, **4**, **5**, **2**, **8**, **6**, **1**, **2**, **5**, **3]**

Програмата трябва да намери **произведението на елементите на списъка**.

Изход:

- Програмата принтира **произведението на елементите на списъка**.

Ето как програмата трябва да работи:

Вход	Изход
	`1036800`

Задача 9. Occurrences Count

Напишете програма, която създава следния **списък**:

[3, 6, 4, 5, 2, 8, 6, 1, 2, 5, 3, 4, 3, 7, 6, 5, 4, 8, 7]

Програмата трябва да **пресмята** колко пъти **даден елемент X се среща в списъка**.

Изход:

- Програмата принтира колко пъти **даден елемент X се среща в списъка**.

Ето как програмата трябва да работи:

Вход	Изход
0	0
Вход	**Изход**
3	3
Вход	**Изход**
5	3
Вход	**Изход**
7	2

Задача 10. Equal Elements

Напишете програма, която проверявали **дали два списъка съдържат еднакви елементи**.

Вход:

- Въвеждаме **елементите на първия списък**.
- Въвеждаме **елементите на втория списък**.

Изход:

- Програмата принтира **булева стойност**, която указва дали списъците съдържат **еднакви елементи**.

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 3]` `[1, 2, 3]`	`True`
Вход	**Изход**
`[1, 2, 4, 5, 6, 7, 8, 9]` `[1, 2, 3, 0, 6, 7, 8, 9]`	`False`
Вход	**Изход**
`[]` `[]`	`True`

Задача 11. Filter List Elements

Напишете програма, която създава следния **списък**:

[33, **16**, **24**, **455**, **62**, **48**, **61**, **13**, **22**, **54**, **33**, **412**, **33**, **74**, **56**, **45**, **43**, **28**, **171]**

Програмата трябва да принтира **елементите на списъка**, които отговарят на **следните условия**:

- **Елементът не е четно число**.
- **Елементът е число**, **чиято последна цифра е различна от 3**.

Изход:

- Програмата принтира **елементите на списъка**, които отговарят на **условията**.

Ето как програмата трябва да работи:

Вход	Изход
	455 61 45 171

Задача 12. Element Index

Напишете програма, която принтира **индекса на даден елемент в списък**.

Вход:

- Въвеждаме **елементите на списъка**.
- Въвеждаме **стойността на елемента**, чийто **индекс** търсим.

Изход:

- Програмата принтира **индекса на даден елемент в списък**.

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 3]` `2`	`1`
Вход	**Изход**
`[1, 2, 4, 5, 6, 7, 8, 9]` `3`	`-1`
Вход	**Изход**
`[8, 7, 6, 5, 4, 3, 2, 1]` `4`	`4`

Задача 13. Even Indexes

Напишете програма, която създава следния **списък**:

[33, **16**, **24**, **455**, **62**, **48**, **61**, **13**, **22**, **54**, **33**, **412**, **33**, **74**, **56**, **45**, **43**, **28**, **171]**

Програмата трябва да намери **сумата на елементите**, които се намират на **четен индекс** в **списъка**.

Изход:

- Програмата принтира **сумата на елементите**, които се намират на **четен индекс** в **списъка**.

Ето как програмата трябва да работи:

Вход	Изход
	538

Задача 14. Max List Element

Напишете програма, която **намира елементът с най-голяма стойност в даден списък**.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **елемента с най-голяма стойност**.

Ето как програмата трябва да работи:

Вход	Изход
`[2, 3, 4, 5, 1, 2, 3, 5]`	`5`
Вход	**Изход**
`[0]`	`0`
Вход	**Изход**
`[1, 2, 5, 4, 2, 3, 8]`	`8`
Вход	**Изход**
`[9, 9, 9, 9, 7, 5, 4, 3, 1]`	`9`

Задача 15. Min List Element

Напишете програма, която **намира елементът с най-малка стойност в даден списък**.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **елемента с най-малка стойност**.

Ето как програмата трябва да работи:

Вход	Изход
`[5, 4, 4, 2, 1, 8, 9, 5]`	`1`
Вход	**Изход**
`[0]`	`0`
Вход	**Изход**
`[1, 2, 1, 4, 4, 1, 2]`	`1`
Вход	**Изход**
`[96, 25, 19, 43, 74, 55, 64 ]`	`19`

Задача 16. Reverse Elements

Напишете програма, която **обръща елементите на даден списък**.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **елементите на списъка**, който се получава след като обърнем реда на елементите на дадения списък.

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 3, 4, 5]`	`[5, 4, 3, 2, 1]`
Вход	**Изход**
`[0]`	`[0]`
Вход	**Изход**
`[1, 1, 1]`	`[1, 1, 1]`
Вход	**Изход**
`[5, 4, 4, 2, 1, 8, 9, 5]`	`[5, 9, 8, 1, 2, 4, 4, 5]`

Задача 17. First & Last

Напишете програма, която, приемайки даден списък, проверява дали **първият елемент в списъка е равен на последния елемент**.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **булева стойност**, която указва дали **първият елемент в списъка е равен на последния елемент**.

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 3, 4]`	`False`
Вход	**Изход**
`[7, 2, 3, 7]`	`True`
Вход	**Изход**
`[6, 5, 3, 6, 4]`	`False`
Вход	**Изход**
`[1, 2, 3, 4, 5, 4, 3, 2, 1]`	`True`

Задача 18. List of Integers

Напишете програма, която **филтрира елементите на даден списък**.

Програмата трябва да принтира само онези **елементи на списъка**, които представляват **цели числа**.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира онези **елементи на списъка**, които представляват **цели числа**.

Ето как програмата трябва да работи:

Вход	Изход
`[1.2, 5.4, 3, 4.5, 3.4, 2.1]`	`[3]`
Вход	**Изход**
`[4, 5, 3.4, 2, 4.5, 4, 5.76]`	`[4, 5, 2, 4]`
Вход	**Изход**
`[1, 1, 1]`	`[1, 1, 1]`
Вход	**Изход**
`[5.5, 6.5, 3, 2, 5, 4.7]`	`[3, 2, 5]`

Задача 19. Numbers Rounding

Напишете програма, която **закръгля стойностите на елементите на даден списък**.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **елементите на списъка**, **след закръглянето**.

Ето как програмата трябва да работи:

Вход	Изход
`[1.6, 2.9, 3.1, 4.2]`	`[2, 3, 3, 4]`
Вход	**Изход**
`[0.6, 2, 3.3, 5.7, 6.7]`	`[0, 2, 3, 6, 7]`
Вход	**Изход**
`[6.1, 5, 0.3, 6.9, 4.4]`	`[6, 5, 0, 7, 4]`
Вход	**Изход**
`[1, 5, 6, 4, 3]`	`[1, 5, 6, 4, 3]`

Задача 20. Transform Elements

Напишете програма, която **променя елементите на даден списък**, добавяйки символа **#** (**диез**) **в началото и в края на текстовата репрезентация на елемента**.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **елементите на списъка**, **след промяната**.

Ето как програмата трябва да работи:

Вход	Изход
`[4]`	`#4#`
Вход	**Изход**

`[a, b, c]`	`#a#`
	`#b#`
	`#c#`

Вход	**Изход**
`[1, 1]`	`#1#` `#1#`
Вход	**Изход**

`[4, 5, 3, 1, 3, 4]`	`#4#`
	`#5#`
	`#3#`
	`#1#`
	`#3#`
	`#4#`

Задача 21. List Twins

Напишете програма, която **проверява дали дадени два списъка имат еднаква сума на техните елеметни**.

Вход:

- Въвеждаме **елементите на първия списък**.
- Въвеждаме **елементите на втория списък**.

Изход:

- Програмата принтира **булева стойност**, която указва дали **двата списъка имат еднаква сума на техните елементи**.

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 3, 4]` `[1, 3, 6]`	`True`
Вход	**Изход**
`[1, 2, 3, 7]` `[5, 8]`	`True`
Вход	**Изход**
`[6, 5, 3, 6, 4, 5, 6]` `[6, 5, 7, 4]`	`False`
Вход	**Изход**
`[1]` `[1]`	`True`

Задача 22. Number to List

Напишете програма, която **превръща дадено число** в **списък**.

Програмата трябва да **създаде списък**, чийто елементи представляват **цифрите** на **даденото число**.

Вход:

- Въвеждаме дадено число.

Изход:

- Програмата принтира **елементите на списъка**.

Ето как програмата трябва да работи:

Вход	Изход
345324	[3, 4, 5, 3, 2, 4]
Вход	**Изход**
14512838	[1, 4, 5, 1, 2, 8, 3, 8]
Вход	**Изход**
5	[5]
Вход	**Изход**
453134	[4, 5, 3, 1, 3, 4]

Задача 23. Remove Duplicates

Напишете програма, която **премахва повтарящите се елементи в даден списък**.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **елементите на списъка**, **след премахването на всички повтарящи се елементи**.

Ето как програмата трябва да работи:

Вход	Изход
`[2, 1, 2, 3, 1, 4, 5, 4]`	`[2, 1, 3, 4, 5]`
Вход	**Изход**
`[1, 1, 1, 2, 1, 2, 1]`	`[1, 2]`
Вход	**Изход**
`[1]`	`[1]`
Вход	**Изход**
`[1, 2, 3, 4, 5, 6]`	`[1, 2, 3, 4, 5, 6]`

Задача 24. Balancing Scales

Напишете програма, която **проверява дали сумата на лявата част и сумата на дясната част на даден списък са равни**.

Списъкът ще разполага с **нечетен брой елементи**. Елементът, намиращ се **точно по средата на списъка**, ще има **стойност 0** и ще служи за разделител на списъка, разделящ го на лява и дясна част.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира дали **сумата на лявата част и сумата на дясната част на списъка са равни**.

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 0, 2, 1]`	`True`
Вход	**Изход**
`[5, 4, 0, 2, 3]`	`False`
Вход	**Изход**
`[5, 0, 5]`	`True`
Вход	**Изход**
`[4, 4, 0, 3, 5]`	`True`

Задача 25. Palindrome

Напишете програма, която **проверява дали даден списък е палиндром**.

Палиндром наричаме **последователност** от **символи** или **числа**, която независимо в коя посока бива прочетена, носи един и същ смисъл.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **булева стойност**, която указва **дали списъкът е палиндром**.

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 0, 2, 1]`	`True`
Вход	**Изход**
`[1, 2, 3]`	`False`
Вход	**Изход**
`[1, 2, 3, 4, 3, 2, 1]`	`True`
Вход	**Изход**
`[1, 4, 3, 3, 4, 3]`	`False`

Задача 26. Digits Frequency

Напишете програма, която намира **броя срещания на всяка цифра** в **дадено число N**.

Вход:

- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **броя срещания на всяка цифра в N**.

Ето как програмата трябва да работи:

Вход	Изход
12546	0 - 0 1 - 1 2 - 1 3 - 0 4 - 1 5 - 1 6 - 1 7 - 0 8 - 0 9 - 0

Вход	Изход
9571239	0 - 0 1 - 1 2 - 1 3 - 1 4 - 0 5 - 1 6 - 0 7 - 1 8 - 0 9 - 2

Задача 27. Longest Word

Напишете програма, която намира **най-дългата дума** в **даден списък**.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **най-дългата дума в списъка**.

Ето как програмата трябва да работи:

Вход	Изход
`[a, aa, aaa, a]`	`aaa`
Вход	**Изход**
`[abcd, ab, a, abc]`	`abcd`
Вход	**Изход**
`[z, z, z, aa]`	`aa`
Вход	**Изход**
`[name, age, town, address]`	`address`

Задача 28. Increasing or Decreasing

Напишете програма, която **проверява дали елементите в даден списък образуват стриктно намаляваща или стриктно растяща редица**.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **символен низ**, който указва **дали елементите образуват стриктно намаляваща или стриктно растяща редица**.

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 3, 4]`	`increasing`
Вход	**Изход**
`[1, 1, 1]`	`neither`
Вход	**Изход**
`[4, 3, 2, 1]`	`decreasing`
Вход	**Изход**
`[1, 2, 1, 2, 1, 2]`	`neither`

Задача 29. Common Elements

Напишете програма, която намира **общите елементи на дадени два списъка**.

Вход:

- Въвеждаме **елементите на първия списък**.
- Въвеждаме **елементите на втория списък**.

Изход:

- Програмата принтира **общите елементи на двата списъка**.

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 3, 4]` `[8, 3, 6, 4]`	`3` `4`
Вход	**Изход**
`[1, 2, 3, 7]` `[6, 5, 4, 8]`	
Вход	**Изход**

`[6, 5, 3, 6, 4]`
`[6, 5, 7, 4]`
	`6`
	`5`
	`4`

Вход	**Изход**
`[1]` `[1]`	`1`

Задача 30. List Rotations

Напишете програма, която осъществява **завъртане на елементите** на **даден списък N на брой пъти**.

Вход:

- Въвеждаме **елементите на списъка**.
- Въвеждаме **стойността на N**.

Изход:

- Програмата принтира **елементите на списъка**, **след завъртането**.

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 3, 4, 5]` `1`	`[5, 1, 2, 3, 4]`
Вход	**Изход**
`[1, 2, 3, 4, 5]` `2`	`[4, 5, 1, 2, 3]`
Вход	**Изход**
`[1, 2, 3, 4, 5]` `3`	`[3, 4, 5, 1, 2]`
Вход	**Изход**
`[1, 2, 3, 4, 5]` `5`	`[1, 2, 3, 4, 5]`

Функции

В настоящата глава ще разгледаме какво представляват **функциите**. Ще разгледаме някои от **основните причини**, поради които трябва да използваме функции. Ще научим как функциите се създават и извикват. Накрая, ще спомена някои от **добрите практики** при **писането на качествен програмен код**.

Какво представляват функциите?

Изпълнението на една програма, всъщност, включва изпълнение на множество **подпрограми** (**функции**).

Пример за функция е функцията **print()**, която се използва за изписване на резултат на конзолата. Всъщност, функциите представляват **именувани парчета код**, които реализират дадена **функционалност**. Например, **print()** се използва винаги, когато желаем да изпишем текст на конзолата. Тази функция се използва само и единствено за тази функционалност и **няма странични ефекти**, т.е. не прави нищо друго, освен това, което казва, че прави.

Езиците за програмиране ни позволяват да пишем **функционалности**, да ги **именуваме по подходящ начин** и да ги **преизползваме многократно**.

Например, ако пишем игра, ще са ни необходими множество парчета малки функционалности - създаване на игрално поле, създаване и позициониране на герой, създаване и позициониране на врагове, логика, която реализира движението на героя, враговете и т.н. В такъв случай, нашият код може да изглежда по подобен начин:

Пример за употреба на функции

```
gamefield = create_gamefield(500, 500)

player = create_player()

while is_player_alive():
    play_game()
```

Програмата изглежда **кратка** и **описателна**. Този код е **четим**, дори от непрофесионалисти и от хора, които не са се занимавали с програмиране.

Цялата логика на програмата е **разделена** на малки блокове от код, всеки от които е отговорен за определена **логика** / **функционалност** / парче от програмата.

Освен това, всяка една от функционалностите има **едно единствено предназначение**. По този начин, в случай, че има проблем по време на изпълнение на програмата (**бъг**), този проблем би бил сравнително по-лесен за откриване.

Например, в парчето код **create_gamefield(500, 500)** се създава **2D игрално поле** с определени размери - **500 x 500**.

За да обясним понятието **функция**, трябва да обясним какво означава понятието **множество**. Според математиката, **множество е съвкупност от елементи**, които не се повтарят. Например, **множеството на естествените числа** (числата, с които броим). В случая, множеството има **безкрайно много елементи**, които **не се повтарят**.

Функция - това е **изображение** / **асоциация** на елемент от едно множество като / към елемент от друго множество. В математиката, функциите приемат **аргументи** и указват **закономерността**, чрез която се съставя **изображението**.

Нека имаме функцията: **f(x) = x + 1**.

Стойността **x** се нарича **параметър** на функцията, а **формулата x + 1** е закономерността, чрез която съставяме **изображението**.

За всяко **x функцията връща резултат**, който се **пресмята по формулата x + 1**. За всеки елемент **x** от някакво **множество от числа**, получаваме негово **изображение** - елемент от друго **множество от числа**.

f(2) = 2 + 1 = 3
f(5) = 5 + 1 = 6

По-лесен и достъпен начин, чрез който можем да опишем функциите, е като ги представим като **черни кутии**, които **приемат вход** и **връщат резултат**:

input → function → output

Защо бихме оприличили функциите на **черни кутии**? Защото в процеса на работа, ние се интересуваме от това **Какво прави функцията?,** а не от това **Как го прави?**.

Пример: Използваме функцията **math.trunc()**, интересувайки се от това какво функцията ще върне като резултат, а не от това как се осъществява пресмятането. Интересуваме се единствено от **входните параметри** - **input**, и **изходния резултат** - **output**.

Нека имаме функцията: **f(x) = x + 1**.

Подавайки конкретна стойност на функцията, тя връща друга стойност (**резултат**) спрямо формулата **x + 1**:

2 → function → 3
5 → function → 6
12 → function → 13

Компютърните езици използват много **концепции от математиката**, естествено, с известни **промени**. В езиците за програмиране, функциите представляват **именувани парчета код**. Аналогично, те могат да приемат стойност/и и да връщат резултат от своето изпълнение.

Създаване и извикване на функция

Нека дадем пример за функция, която събира две числа:

Пример за употреба на функции

```
def calculate_sum(a, b):
    sum = a + b
    return sum

sum = calculate_sum(5, 6)
print(sum)
```

Тази функция сумира две числа и **връща** сумата им. Налични са и функции, които не връщат стойности:

Пример за употреба на функции

```
def print_number(number):
    print(number)

print_number(5)
print_number(10)
```

Единственото действие, което това парче код ще извърши е да принтира числото, което сме подали на функцията. За да укажем, обаче, че желаем командите след символа **:** (**двуеточие**) да се изпълнят, трябва да **извикаме** функцията, чрез нейното **име**.

Извикването на функция се осъществява чрез изписване на нейното **име**, **последвано от кръгли скоби ()**.

Параметри на функция

Аналогично на функциите в математиката, функциите в **Python** могат да приемат **параметри**.

Нека имаме функцията: **f(x) = x * 2**

Пример за употреба на функции

```
def f(x):
    return x * 2

result = f(5)
print(result) # 10
```

При горепосочената функция **f(x)** - **x** е **параметър**. **Параметър** наричаме **шаблона**, чрез който **поставяме стойност**.

Аргумент наричаме **реалната стойност**, с която **заместваме параметъра**.

Функциите могат да приемат **множество параметри**.

Пример за употреба на функции

```
def sum(a, b, c):
    return a + b + c

result = sum(1, 1, 1)
print(result) # 3

result = sum(5, 10, 15)
print(result) # 30

result = sum(35, 40, 10)
print(result) # 85
```

Ключовата дума return

Ключовата дума **return** се използва, когато желаем да върнем стойност от дадена функция. Тя може да се изпише многократно, но се изпълнява **само веднъж**. В момента, в който се върне стойност от функция, тя **приключва своето изпълнение** и изпълнението на кода се връща точно там, където е била извикана функцията.

Пример за употреба на функции

```
def max(a, b):
    if (a > b):
        return a
    return b

result = max(5, 6)
print(result) # 6
```

Защо използваме функции?

Функциите представляват именувани парчета код. Програмирането е немислимо без тяхната употреба.

По-добра четимост на кода

Раздробяването на логиката на парчета функционалност допринася за лесното разбиране какво всъщност ще се случи при изпълнение на програмата.

Естествено, неправилното или подвеждащото именуване на функциите би довело до проблеми.

Функциите улесняват писането на код, тъй като в общия случай, програмистът се интересува от това **Какво се случва?**, а не от това **Как се случва?**.

Например, в задачите, които сме решавали до момента, ни е интересувало да изведем дадено съобщение на конзолата, но не ни е интересувало как точно работи функцията **print()**.

С други думи, **използваме нещо**, **което знаем**, **че работи**, **макар че не сме запознати в дълбочина с неговите детайли**, **не се интересуваме от това как точно работи**.

В съвременния свят можем да дадем множество такива примери - използваме дистанционното на телевизора без да знаем как всъщност работи, използваме **Wi-Fi** и **Интернет** без да сме напълно запознати как работят маршрутизаторите и мрежите.

По-добра структура на кода

Понякога програмите са съставени от **хиляди или дори милиони редове код**. Записването на целия код в един файл или **без ясна структура е немислимо**.

Този подход **би затруднил работата и ориентирането в кода**, в случай, че се изисква **редакция** или **добавяне на нова функционалност**. За редактирането на малко парче код, ще трябва да се сблъскаме с цялата програма и всички нейни **функционалности**.

Когато програмата е разделена на множество малки функционалности, нейната структура **позволява бързото и лесно добавяне или заменяне на елементи в кода**, с цел добавяне на **нова функционалност** или **отстраняване на бъгове**.

Избягване на повторения

Едно от най-важните неща, които получаваме при използването на функции, е **избягването на повторения на кода**.

Когато дадено парче код се изпълнява често, на различни места в нашата програма, е много удобно и лесно да го **преизползваме** чрез **функция**. Например, функцията **print()**.

Добри практики

Всяко парче код, което се повтаря, трябва да бъде изнесено във **функция**.

Функциите трябва да имат **описателни имена**, които отговарят на въпроса **Какво прави този код?** / **Какво прави тази функционалност?**.

Препоръчително е имената на функциите да съдържат **глагол**. Например, може да съдържат думите **get**, **calculate**, **count** и други.

Имената на променливите също трябва да бъдат **достатъчно описателни** и да отговарят на въпроса **Какво съхранява тази променлива?**.

Препоръчително е имената на променливите да бъдат съставени от **съществително име** или комбинация от **прилагателно и съществително име**.

Строго препоръчително е функциите да описват точно **едно действие** / **една функционалност**.

Вместо функция, която създава игрално поле, герой, врагове и т.н., по-скоро можем да създадем няколко по-малки функции за всяка една от функционалностите.

Всяка функция трябва да има **строго определена**, **конкретна задача** и **да върши точно това**, **което казва**, **че върши** (спрямо нейното име или нейната документация).

Функциите не трябва да водят до странични ефекти. Ако дадена функция създава игрално поле, то **не трябва помежду другото да прави още нещо**, **което не очакваме**. Ако функция сумира елементите на списък и връща сумата им, то не трябва като странично действие да принтира списъка на конзолата, например, тъй като не очакваме това.

Функции: Упражнения

Задача 1. Print Function

Напишете **функция**, която **принтира на конзолата**, използвайки вградената функция за принтиране.

Функцията би трябвало да се използва по следния начин:

Print Function

`my_print(2)`
 	
`my_print(5.5)`
 	
`my_print('Python')`
 	
`my_print('Hello World')`

Задача 2. Sum Function

Напишете **функция**, която **сумира две числа**, подадени като **параметри на функцията**.

Функцията би трябвало да се използва по следния начин:

Sum Function

`first_number = 5`
`second_number = 12`
 	
`sum = sum_numbers(first_number, second_number)`
`print(sum) # 17`

Ето как програмата трябва да работи:

Вход	Изход
5 12	17
Вход	**Изход**
2 8	10
Вход	**Изход**
7 7	14

Задача 3. Multiply Function

Напишете **функция**, която **умножава две числа**, подадени като **параметри на функцията**.

Функцията би трябвало да се използва по следния начин:

Multiply Function

`first_number = 5`
`second_number = 2`
 	
`product = multiply_numbers(first_number, second_number)`
`print(product) # 10`

Ето как програмата трябва да работи:

Вход	Изход
5 12	60
Вход	**Изход**
2 8	16
Вход	**Изход**
7 7	49

Задача 4. Square Area Function

Напишете **функция**, която намира **лицето на квадрат**, при подадена **дължина на негова страна** като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Square Area Function

`square_side = 5`
 	
`square_area = calculate_square_area(square_side)`
`print(square_area) # 25`

Ето как програмата трябва да работи:

Вход	Изход
12	144
Вход	**Изход**
8	64
Вход	**Изход**
7	49

Задача 5. Rectangle Perimeter Function

Напишете **функция**, която намира **периметъра на правоъгълник**, при подадени **дължини на две негови срещуположни страни** като **параметри на функцията**.

Функцията би трябвало да се използва по следния начин:

Rectangle Perimeter Function

`a = 5`
`b = 6`
 	
`rectangle_perimeter = calculate_rectangle_perimeter(a, b)`
`print(rectangle_perimeter) # 22`

Ето как програмата трябва да работи:

Вход	Изход
12 13	50
Вход	**Изход**
8 7	30
Вход	**Изход**
3 9	24

Задача 6. Is Even Function

Напишете **функция**, която проверява дали **дадено число**, подадено като **параметър на функцията**, е **четно**.

Функцията би трябвало да се използва по следния начин:

Is Even Function

`number = 5`
 	
`is_number_even = is_even(number)`
`print(is_number_even) # False`

Ето как програмата трябва да работи:

Вход	Изход
12	True
Вход	**Изход**
1	False
Вход	**Изход**
4	True
Вход	**Изход**
3	False

Задача 7. Last Digit Function

Напишете **функция**, която **връща най-дясната цифра** на **дадено число**, подадено като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Last Digit Function

`number = 73`
 	
`last_digit = extract_last_digit(number)`
`print(last_digit) # 3`

Ето как програмата трябва да работи:

Вход	Изход
112	2
Вход	**Изход**
1949	9
Вход	**Изход**
45	5
Вход	**Изход**
7	7

Задача 8. Max Function

Напишете **функция**, която намира **по-голямото измежду две числа**, подадени като **параметри на функцията**.

Функцията би трябвало да се използва по следния начин:

Max Function

`first_number = 5`
`second_number = 2`
 	
`max_number = max(first_number, second_number)`
`print(max_number) # 5`

Ето как програмата трябва да работи:

Вход	Изход
5 12	12
Вход	**Изход**
2 8	8
Вход	**Изход**
7 7	7
Вход	**Изход**
7 2	7

Задача 9. Min Function

Напишете **функция**, която намира **по-малкото измежду две числа**, подадени като **параметри на функцията**.

Функцията би трябвало да се използва по следния начин:

Min Function

`first_number = 5`
`second_number = 2`
 	
`min_number = min(first_number, second_number)`
`print(min_number) # 2`

Ето как програмата трябва да работи:

Вход	Изход
5 12	5
Вход	**Изход**
2 8	2
Вход	**Изход**
7 7	7
Вход	**Изход**
7 2	2

Задача 10. Print List Function

Напишете **функция**, която **принтира елементите на списък**, **изредени със запетая**. **Списъкът** се подава като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Print List Function

`numbers = [3, 4, 5, 6, 7, 5, 4]`
 	
`print_list(numbers)`

Ето как програмата трябва да работи:

Вход	Изход
`[3, 4, 5, 6, 7, 5, 4]`	3 4 5 6 7 5 4
Вход	**Изход**
`[1, 2, 3, 4]`	1 2 3 4
Вход	**Изход**
`[]`	

Задача 11. Sum List Elements Function

Напишете **функция**, която **намира сумата на елементите на списък**. **Списъкът** се подава като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Sum List Elements Function

`numbers = [3, 4, 5, 6, 7, 5, 4]`
 	
`elements_sum = sum_elements(numbers)`
`print(elements_sum) # 34`

Ето как програмата трябва да работи:

Вход	Изход
`[3, 4, 5, 6, 7, 5, 4]`	`34`
Вход	**Изход**
`[1, 2, 3, 4]`	`10`
Вход	**Изход**
`[]`	`0`

Задача 12. Multiply List Elements Function

Напишете **функция**, която **намира произведението на елементите на списък**. **Списъкът** се подава като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Multiply List Elements Function

`numbers = [3, 4, 5, 6, 7, 5, 4]`
 	
`elements_product = multiply_elements(numbers)`
`print(elements_product) # 50400`

Ето как програмата трябва да работи:

Вход	Изход
`[3, 4, 5, 6, 7, 5, 4]`	`50400`
Вход	**Изход**
`[1, 2, 3, 4]`	`24`
Вход	**Изход**
`[]`	`1`

Задача 13. Max List Element Function

Напишете **функция**, която **намира елементът с най-голяма стойност в даден списък**. **Списъкът** се подава като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Max List Element Function

`numbers = [3, 4, 5, 6, 7, 5, 4]`
 	
`max_element = get_max_element(numbers)`
`print(max_element) # 7`

Ето как програмата трябва да работи:

Вход	Изход
`[3, 4, 5, 6, 7, 5, 4]`	`7`
Вход	**Изход**
`[1, 2, 3, 4]`	`4`
Вход	**Изход**
`[-1, -2]`	`-1`
Вход	**Изход**
`[1]`	`1`

Задача 14. Min List Element Function

Напишете **функция**, която **намира елементът с най-малка стойност в даден списък**. **Списъкът** се подава като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Min List Element Function

`numbers = [3, 4, 5, 6, 7, 5, 4]`
 	
`min_element = get_min_element(numbers)`
`print(min_element) # 3`

Ето как програмата трябва да работи:

Вход	Изход
`[3, 4, 5, 6, 7, 5, 4]`	`3`
Вход	**Изход**
`[1, 2, 3, 4]`	`1`
Вход	**Изход**
`[-1, -2]`	`-2`
Вход	**Изход**
`[1]`	`1`

Задача 15. Contains Element Function

Напишете **функция**, която **проверява дали дадено число се съдържа в даден списък**. **Списъкът** и **числото** се подават като **параметри на функцията**.

Функцията би трябвало да се използва по следния начин:

Contains Element Function

`numbers = [3, 4, 5, 6, 7, 5, 4]`
`n = 6`
 	
`in_list = contains(numbers, n)`
`print(in_list) # True`

Ето как програмата трябва да работи:

Вход	Изход
`[3, 4, 5, 6, 7, 5, 4]` `3`	`True`
Вход	**Изход**
`[1, 11, 22, 34, 19, 4]` `34`	`True`
Вход	**Изход**
`[1, 2, 5, 4, 3, 2, 1, 5, 6]` `9`	`False`
Вход	**Изход**
`[1, 4, 5, 6, 4, 2, 3, 6, 7]` `4`	`True`

Задача 16. Number Split

Напишете **функция**, която **разделя дадено число N на две части**, които представляват **два елемента**, **поместени в списък**. **Числото N** се подава като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Number Split Function

`number = 10`
 	
`list = split_number(number)`
`print(list) # [5, 5]`

Ето как програмата трябва да работи:

Вход	Изход
15	[7, 8]
Вход	**Изход**
4	[2, 2]
Вход	**Изход**
11	[5, 6]

Задача 17. Absolute Value Function

Напишете **функция**, която **връща абсолютната стойност** на **дадено число**, подадено като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Absolute Value Function

`number = 5`
 	
`absolute_value = get_absolute_value(number)`
`print(absolute_value) # 5`

Ето как програмата трябва да работи:

Вход	Изход
1	1
Вход	**Изход**
-5	5
Вход	**Изход**
5	5
Вход	**Изход**
0	0

Задача 18. Count Digits Function

Напишете **функция**, която **връща броя цифри** на **дадено число**, подадено като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Count Digits Function

`number = 17436`
 	
`digits_count = count_digits(number)`
`print(digits_count) # 5`

Ето как програмата трябва да работи:

Вход	Изход
1122	4
Вход	**Изход**
19	2
Вход	**Изход**
455124	6

Задача 19. Sum Digits Function

Напишете **функция**, която **връща сумата на цифрите** на **дадено число**, подадено като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Sum Digits Function

`number = 17436`
 	
`digits_sum = sum_digits(number)`
`print(digits_sum) # 21`

Ето как програмата трябва да работи:

Вход	Изход
123	6
Вход	**Изход**
2345	14
Вход	**Изход**
72567	27
Вход	**Изход**
17436	21

Задача 20. Repeat Symbol Function

Напишете **функция**, която **връща** текст, съставен от **даден символ**, **повторен N на брой пъти**. **Символът** и **числото N** се подават като **параметри на функцията**.

Функцията би трябвало да се използва по следния начин:

Repeat Symbol Function

`symbol = '*'`
`count = 5`
 	
`text = repeat_symbol(symbol, count)`
`print(text) # *****`

Ето как програмата трябва да работи:

Вход	Изход
/ 4	////
Вход	**Изход**
@ 8	@@@@@@@@
Вход	**Изход**
~ 5	~~~~~
Вход	**Изход**
# 2	##

Задача 21. Evenish or Oddish

Напишете **функция**, която проверява дали **дадено число N** е **evenish** или **oddish**. **Числото N** се подава като **параметър на функцията**.

Ако **сумата на цифрите** на **числото N** е **четно число**, то **N е evenish**.

Ако **сумата на цифрите** на **числото N** е **нечетно число**, то **N е oddish**.

Функцията би трябвало да се използва по следния начин:

Evenish or Oddish Function

`number = 4433`
 	
`type = get_type(number)`
`print(type) # evenish`
 	
`number = 43`
 	
`type = get_type(number)`
`print(type) # oddish`

Ето как програмата трябва да работи:

Вход	Изход
43	oddish
Вход	**Изход**
373	oddish
Вход	**Изход**
4433	evenish

Задача 22. Initcap Function

Напишете **функция**, която **променя думите** в **даден текст**. **Текстът** се подава като **параметър на функцията**.

Програмата трябва да промени **всяка дума в текста**, изписвайки я с **главна буква**.

Функцията би трябвало да се използва по следния начин:

Initcap Function

`text = 'Python is awesome!'`
 	
`result = initcap(text)`
`print(result) # Python Is Awesome!`

Ето как програмата трябва да работи:

Вход	Изход
`john doe`	`John Doe`
Вход	**Изход**
`conditional statements & loops`	`Conditional Statements & Loops`
Вход	**Изход**
`coding is fun!`	`Coding Is Fun!`

Задача 23. Uppercase Function

Напишете **функция**, която **променя думите** в **даден текст**. **Текстът** се подава като **параметър на функцията**.

Програмата трябва да промени **всяка дума в текста**, изписвайки я изцяло с **главни букви**.

Функцията би трябвало да се използва по следния начин:

Uppercase Function

`text = 'Python is awesome!'`
 	
`result = uppercase(text)`
`print(result) # PYTHON IS AWESOME!`

Ето как програмата трябва да работи:

Вход	Изход
John Doe	JOHN DOE
Вход	**Изход**
Conditional Statements & Loops	CONDITIONAL STATEMENTS & LOOPS
Вход	**Изход**
Coding is fun!	CODING IS FUN!

Задача 24. Lowercase Function

Напишете **функция**, която **променя думите** в **даден текст**. **Текстът** се подава като **параметър на функцията**.

Програмата трябва да промени **всяка дума в текста**, изписвайки я изцяло с **малки букви**.

Функцията би трябвало да се използва по следния начин:

Lowercase Function

`text = 'Python is awesome!'`
 	
`result = lowercase(text)`
`print(result) # python is awesome!`

Ето как програмата трябва да работи:

Вход	Изход
John Doe	john doe
Вход	**Изход**
Conditional Statements & Loops	conditional statements & loops
Вход	**Изход**
Coding is fun!	coding is fun!

Задача 25. Replace Character

Напишете **функция**, която **замества първото срещане на даден символ в даден текст с друг символ**. **Текстът** и **двата символа** се подават като **параметри на функцията**.

Функцията би трябвало да се използва по следния начин:

Replace Character Function

`text = 'Python is awesome!'`
`character_to_replace = 'p'`
`replacement = 't'`
 	
`result = replace(text, character_to_replace, replacement)`
`print(result) # Tython is awesome!`

Ето как програмата трябва да работи:

Вход	Изход

`Microsoft`
`s`
`t`
	`Microtoft`

Вход	**Изход**

`Help`
`p`
`l`
	`Hell`

Вход	**Изход**

`white`
`t`
`l`
	`while`

Задача 26. Count Symbol Function

Напишете **функция**, която **преброява колко пъти даден символ се среща в даден текст**. **Текстът** и **символът** се подават като **параметри на функцията**.

Функцията би трябвало да се използва по следния начин:

Count Symbol Function

`text = 'Python is awesome!'`
`symbol = 'o'`
 	
`count = count_symbol(text, symbol)`
`print(count) # 2`

Ето как програмата трябва да работи:

Вход	Изход
`Microsoft` `o`	`2`
Вход	**Изход**
`Coding is fun!` `!`	`1`
Вход	**Изход**
`JavaScript` `e`	`0`

Задача 27. Argument Type

Напишете **функция**, която връща **типа данни** на **подадения аргумент**. **Функцията приема един параметър**.

Функцията би трябвало да се използва по следния начин:

Argument Type Function

`type = get_type(5)`
`print(type) # <class 'int'>`
 	
`type = get_type(True)`
`print(type) # <class 'bool'>`

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 3, 4, 5]`	`<class 'list'>`
Вход	**Изход**
`5`	`<class 'int'>`
Вход	**Изход**
`JavaScript`	`<class 'str'>`
Вход	**Изход**
`False`	`<class 'bool'>`

Задача 28. List Filter Function

Напишете **функция**, която **филтрира елементите на списък**, спрямо **дадено число N**. **Списъкът** и **числото N** се подават като **параметри на функцията**.

Програмата трябва да принтира **стойностите на списък**, чийто елементи представляват само **онези елементи от оригиналния списък**, които са **по-големи или равни на числото N**.

Функцията би трябвало да се използва по следния начин:

List Filter Function

`numbers = [3, 4, 5, 6, 7, 5, 4]`
`list = filter(numbers, 5)`
 	
`for element in list:`
`    print(element)`

Ето как програмата трябва да работи:

Вход	Изход

`[1, 2, 3, 4, 5]`
`3`
	`3`
	`4`
	`5`

Вход	**Изход**

`[2, 3, 5, 4, 6, 5, 3, 5, 2]`
`4`
	`5`
	`4`
	`6`
	`5`
	`5`

Вход	**Изход**
`[1, 2]` `0`	`[1, 2]`

Задача 29. Character Position Function

Напишете **функция**, която **връща позицията**, **на която даден символ се намира в даден текст**. **Символът** и **текстът** се подават като **параметри на функцията**.

Функцията би трябвало да се използва по следния начин:

Character Position Function

`character = 'y'`
`text = 'Python is awesome!'`
 	
`index = character_position(character, text)`
`print(index) # 1`

Ето как програмата трябва да работи:

Вход	Изход
Python h	3
Вход	**Изход**
hello from the other side f	6
Вход	**Изход**
What is your name? b	-1

Задача 30. Sort Digits Function

Напишете **функция**, която преобразува **дадено число N**. **Числото N** се подава като **параметър на функцията**.

Функцията трябва да преобразува **числото N** така, че цифрите му да бъдат **сортирани** (подредени) **по големина**.

Функцията би трябвало да се използва по следния начин:

Sort Digits Function

`number = 9548456`
 	
`result = sort_digits(number)`
`print(result) # 4455689`

Ето как програмата трябва да работи:

Вход	Изход
43	34
Вход	**Изход**
2783234	2233478
Вход	**Изход**
12345	12345

Речници

В настоящата глава ще разгледаме какво представляват **речниците** в езика **Python**. Накратко, речниците ни позволяват да описваме **обекти** / **предмети от реалния свят**.

Защо използваме речници?

Повечето програмни езици разполагат с различни **структури от данни**, които могат да съхраняват **множество стойности**, с известни разлики във функционалността и бързодействието.

Използването на **речници** позволява съхранение на множество стойности в една променлива.

Речниците много приличат на списъците, тъй като и двете структури предоставят възможност за съхранение на множество елементи, които могат да се достъпят по **индекс**.

При речниците, обаче, **индексите не представляват числа**, както при списъците, а **символни низове** (**текст**).

Какво представляват речниците?

Речниците представляват структура от данни, която позволява съхранението на **двойки ключ-стойности**.

Всъщност, речниците в **Python** много приличат на речниците, които използваме в училище – например **английско-български речник**. Всяка дума на английски език (**ключ**) е преведена на български език (**стойност**).

Съхранението на данни във **формат ключ-стойност** позволява лесното описание на обекти от реалния свят.

В реалния свят, човекът е **обект**. Всеки човек има имена и възраст.

Речникът person

```
person = {
    'first_name': 'Ivan',
    'last_name': 'Ivanov',
    'age': 25
}
```

Използвайки речника **person**, можем да принтираме стойността на всяка негова характеристика.

Речникът person

```
print(person['first_name']) # Ivan
print(person['last_name']) # Ivanov
print(person['age']) # 25
```

Можем да си представяме речниците като **множество от стойности**, **които се разглеждат като едно цяло**, подобно на списъците.

Нека разгледаме как бихме дефинирали **речника кола**.

Речникът car

```
car = {
    'brand': 'Fiat',
    'model': '500',
    'color': 'black',
    'production_year': 2014,
    'horsepower': 200,
    'fuel_amount_liters': 45,
    'price': 25000,
    'weight_kg': 1600,
    'is_automatic': True
}

print(car['brand']) # Fiat
print(car['model']) # 500
print(car['color']) # black
print(car['production_year']) # 2014
print(car['horsepower']) # 200
print(car['fuel_amount_liters']) # 45
print(car['price']) # 25000
print(car['weight_kg']) # 1600
print(car['is_automatic']) # True
```

Ключове и стойности

Нека разгледаме **речника person**.

Речникът person

```
person = {
    'first_name': 'Ivan',
    'last_name': 'Ivanov',
    'age': 25,
    'is_male': True
}
```

В този пример, **person** е **речник**, а **first_name**, **last_name**, **age** и **is_male** са **ключове**. Всеки ключ съхранява **стойност**.

Нека имаме следния списък.

Списъкът numbers

```
numbers = [1, 2, 3, 4, 5]
```

Знаем, че елементите на списъка се достъпват чрез **индекс**. Индексът е **номер на елемент** – **число между 0 и дължината на списъка минус 1**.

Индексиране на елементи в списък

```
numbers = [1, 2, 3, 4, 5]

print(numbers[0]) # 1
print(numbers[1]) # 2
```

По същия начин можем да „индексираме“ **стойностите** на даден **речник**.

Речникът person

```
person = {
    'first_name': 'Ivan',
    'last_name': 'Ivanov',
    'age': 25
}

print(person['first_name']) # Ivan
print(person['last_name']) # Ivanov
print(person['age']) # 25
```

Речници: Упражнения

Задача 1. Human Dictionary

Напишете програма, която създава речника **human** (**човек**).

Речникът трябва да притежава следните **ключ-стойности**:

- **first_name**: **String**
- **last_name**: **String**
- **age**: **Number**

Пример:

Human Dictionary Пример

```
human = {
    'first_name': _________,
    'last_name': _________,
    'age': _________
}

print(human['first_name'])
print(human['last_name'])
print(human['age'])
```

Задача 2. Hero Dictionary

Напишете програма, която създава речника **hero** (**герой**).

Речникът трябва да притежава следните **ключ-стойности**:

- **name**: **String**
- **health**: **Number**
- **class**: **String**
- **level**: **Number**
- **power**: **Number**

Пример:

Hero Dictionary Пример

```
hero = {
    'name': _________,
    'health': _________,
    'class': _________,
    'level': _________,
    'power': _________
}

print(hero['name'])
print(hero['health'])
print(hero['class'])
print(hero['level'])
print(hero['power'])
```

Задача 3. Book Dictionary

Напишете програма, която създава речника **book** (**книга**).

Речникът трябва да притежава следните **ключ-стойности**:

- **title**: **String**
- **author**: **Object**
  - **first_name**: **String**
  - **last_name**: **String**
- **pages_count**: **Number**

Пример:

Book Dictionary Пример

```
book = {
    'title': _________,
    'author': {
        'first_name': _________,
        'last_name': _________,
    },
    'year': _________,
    'pages_count': _________
}

print(book['title'])
print(book['author']['first_name'])
print(book['author']['last_name'])
print(book['year'])
print(book['pages_count'])
```

Задача 4. Game Dictionary

Напишете програма, която създава речника **game** (**игра**).

Речникът трябва да притежава следните **ключ-стойности**:

- **name**: **String**
- **genres**: **List of Strings**
- **rating**: **Number**
- **price**: **Number**

Пример:

Game Object Пример

```
game = {
    'names': _________,
    'genres': [
        _________,
        _________,
        _________
    ],
    'rating': _________,
    'price': _________
}

print(game['names'])
print(game['genres'][0])
print(game['genres'][1])
print(game['genres'][2])
print(game['rating'])
print(game['price'])
```

Задача 5. Triangle Dictionary

Напишете програма, която създава речника **triangle** (**триъгълник**).

Речникът трябва да притежава следните **ключ-стойности**:

- **side_a**: **Number**
- **side_b**: **Number**
- **side_c**: **Number**
- **height_a**: **Number**

Създайте функциите **get_perimeter()** и **get_area()**, които **приемат речника като аргумент** и връщат съответно неговите **периметър** и **лице**.

Пример:

Triangle Dictionary Пример

```
triangle = {
    'side_a': _________,
    'side_b': _________,
    'side_c': _________,
    'height_a': _________
}

print(triangle['side_a'])
print(triangle['side_b'])
print(triangle['side_c'])
print(triangle['height_a'])

perimeter = get_perimeter(triangle)
print(perimeter)

area = get_area(triangle)
print(area)
```

Задача 6. Computer Dictionary

Напишете програма, която създава речника **computer** (**компютър**).

Речникът трябва да притежава следните **ключ-стойности**:

- **brand**: **String**
- **model**: **String**
- **ram**: **Number**
- **hard_disk**: **Number**
- **operating_system**: **String**
- **is_turn_on**: **Boolean**

Създайте функциите **turn_on()** и **turn_off()**, които **приемат речника като аргумент** и променят неговия **ключ is_turn_on**.

Пример:

Computer Dictionary Пример

```
computer = {
    'brand': _________,
    'model': _________,
    'ram': _________,
    'hard_disk': _________,
    'operating_system': _________,
    'is_turn_on': False
}

print(computer['brand'])
print(computer['model'])
print(computer['ram'])
print(computer['hard_disk'])
print(computer['operating_system'])
print(computer['is_turn_on']) # False

turn_on(computer)
print(computer['is_turn_on']) # True

turn_off(computer)
print(computer['is_turn_on']) # False
```

Задача 7. Upvotes & Downvotes Function

Напишете функция, която **намира как даден вот трябва да бъде представен**. **Положителните и отрицателните гласове са представени чрез речник**, който се подава като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Upvotes & Downvotes Function

`votes = {`
`    'upvotes': 12,`
`    'downvotes': 8`
`}`
 	
`result = calculate_votes(votes)`
`print(result) # 4`

Ето как програмата трябва да работи:

Вход	Изход
1 2	-1
Вход	**Изход**
15 2	13
Вход	**Изход**
1 1	0
Вход	**Изход**
5 17	-12

Задача 8. Box Volume Function

Напишете **функция**, която **намира обема на кутия**. **Кутията е представена чрез речник**, който се подава като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

Box Volume Function

`box = {`
`    'width': 5,`
`    'height': 6,`
`    'length': 7`
`}`
 	
`box_volume = calculate_volume(box)`
`print(box_volume) # 210`

Ето как програмата трябва да работи:

Вход	Изход
1 2 3	6
Вход	**Изход**
5 5 5	125
Вход	**Изход**
10 10 10	1000

Задача 9. City Information

Напишете **функция**, която представя информацията от **даден речник**. **Речникът**, който се подава като **параметър на функцията**, съдържа информация относно **име на град** и неговата **локация**.

Функцията би трябвало да се използва по следния начин:

City Information Function

`data = {`
`    'name': 'Paris',`
`    'continent': 'Europe'`
`}`
 	
`text = extract_information(data)`
`print(text) # Paris is situated in Europe.`

Ето как програмата трябва да работи:

Вход	Изход

`data = {`
`    'name': 'Paris',`
`    'continent': 'Europe'`
`}`
	`Paris is situated in Europe.`

Вход	**Изход**

`data = {`
`    'name': 'Tokyo',`
`    'continent': 'Asia'`
`}`
	`Tokyo is situated in Asia.`

Вход	**Изход**

`data = {`
`    'name': 'Berlin',`
`    'continent': 'Europe'`
`}`
	`Berlin is situated in Europe.`

Задача 10. International Greetings

Напишете програма, която принтира **поздрав**, отправен от **даден студент**, спрямо неговите **име** и **националност**.

Използвайте следния код:

Списък от речници

```
students_list = [
    { 'name': 'Randy', 'country': 'Germany' },
    { 'name': 'Wendy', 'country': 'United Kingdom' },
    { 'name': 'Norman', 'country': 'United States' },
    { 'name': 'Samantha', 'country': 'United Kingdom' },
    { 'name': 'Gregory', 'country': 'United Kingdom' },
    { 'name': 'Mark', 'country': 'United States' },
    { 'name': 'Zoe', 'country': 'France' },
    { 'name': 'John', 'country': 'United States' },
    { 'name': 'Zack', 'country': 'United Kingdom' },
    { 'name': 'Harold', 'country': 'Germany' }
]
```

Вход:

- Въвеждаме **името на даден студент**.

Изход:

- Програмата принтира **поздрав**, отправен от **студента**, спрямо неговите **име** и **националност**.

Ето как програмата трябва да работи:

Вход	Изход
Randy	Hello, I am Randy. I am from Germany.

Вход	Изход
John	Hello, I am John. I am from United States.

Задача 11. In Interval Function

Напишете функция, която **намира дали дадено число попада в даден интервал**. **Минималната и максималната стойности на интервала са представени чрез речник**, който се подава като **параметър на функцията**.

Функцията би трябвало да се използва по следния начин:

In Interval Function

`number = 15`
`range = {`
`    'min': 12,`
`    'max': 20`
`}`
 	
`in_interval = is_in_interval(range, number)`
`print(in_interval) # True`

Ето как програмата трябва да работи:

Вход	Изход
10 8 12	True
Вход	**Изход**
15 10 20	True
Вход	**Изход**
5 10 20	False

Задача 12. Email

Напишете програма, която принтира **електронната поща** на **даден студент**. В случай, че **студентът няма електронна поща**, програмата принтира **No email address**.

Използвайте следния код:

Списък от речници

```
students_list = [
    { 'name': 'Randy', 'email': 'randy@gmail.com' },
    { 'name': 'Wendy', 'email': None },
    { 'name': 'Norman', 'email': 'norman@gmail.com' },
    { 'name': 'Samantha', 'email': 'samantha@gmail.com' },
    { 'name': 'Gregory', 'email': 'gregory@gmail.com' },
    { 'name': 'Mark', 'email': 'mark@gmail.com' },
    { 'name': 'Zoe', 'email': 'zoe@gmail.com' },
    { 'name': 'John', 'email': None },
    { 'name': 'Zack', 'email': None },
    { 'name': 'Harold', 'email': 'harold@gmail.com' }
]
```

Вход:

- Въвеждаме **името на даден студент**.

Изход:

- Програмата принтира **електронната поща на студента**. Ако студентът **не разполага с електронна поща**, програмата принтира **No email address**.

Ето как програмата трябва да работи:

Вход	Изход
Randy	randy@gmail.com
Вход	**Изход**
John	No email address

Задача 13. Smoothie Ingredients

Напишете програма, която **пресмята крайната цена** на направено **смути**, спрямо **цените на неговите съставки**. **Съставките** и **цените** на **смутито** представляват съответно **ключове** и **стойности** на **даден речник**.

Вход:

- Въвеждаме **даден речник**, съдържащ продуктите и техните цени.

Изход:

- Програмата принтира **крайната цена на смутито**.

Ето как програмата трябва да работи:

Вход	Изход

`smoothie_ingredients = {`
`    'banana': 1.2,`
`    'mango': 1.5,`
`    'pineapple': 2`
`}`
	`4.70`

Вход	**Изход**

`smoothie_ingredients = {`
`    'strawberry': 1.1,`
`    'apple': 0.6,`
`    'grapes': 1.9`
`}`
	`3.60`

Вход	**Изход**

`smoothie_ingredients = {`
`    'raspberries': 2.2,`
`    'blueberries': 2.5,`
`    'lemon': 2.2`
`}`
	`6.90`

Задача 14. Population Function

Напишете програма, която **пресмята възрастта на всеки човек от дадено множество от хора**, след **определен период от време**.

Вход:

- Въвеждаме **даден речник**, съставен от **имена на хора** и **техните години**.
- Въвеждаме **период от време**, измерван в **години**.

Изход:

- Програмата принтира **възрастта на всеки човек** след дадения **период от време** (**години**).

Ето как програмата трябва да работи:

Вход	Изход

`population = {`
`'Ivan': 42,`
`'Petar': 23,`
`'Georgi': 37,`
`'Anna': 29`
`}`
 	
`years = 10`
	52
	33
	47
	39

Вход	Изход

`population = {`
`'Anna': 22,`
`'Stefan': 29,`
`'Milen': 25,`
`'Dimitar': 34`
`}`
 	
`years = 15`
	37
	44
	40
	49

Задача 15. Products Price

Напишете програма, която принтира **общата цена на дадено множество от продукти**.

Използвайте следния код:

Списък от речници

```
products = [
    { 'id': 1, 'name': 'Watch', 'price': 122.99 },
    { 'id': 2, 'name': 'Diamond bracelet', 'price': 647.99 },
    { 'id': 3, 'name': 'Ring', 'price': 89.99 },
    { 'id': 4, 'name': 'Gaming mouse', 'price': 89.99 },
    { 'id': 5, 'name': 'T-Shirt', 'price': 20.99 },
    { 'id': 6, 'name': 'Headphones', 'price': 99.99 },
    { 'id': 7, 'name': 'Phone case', 'price': 12.99 },
    { 'id': 8, 'name': 'Socks', 'price': 5.99 },
    { 'id': 9, 'name': 'USB cable', 'price': 1.99 },
    { 'id': 10, 'name': 'Leather jacket', 'price': 94.99 },
    { 'id': 11, 'name': 'Black dress', 'price': 421.99 },
    { 'id': 12, 'name': 'Gloves', 'price': 12.99 }
]
```

Задача 16. Likes & Dislikes

Напишете програма, която **пресмята рейтинга** в **проценти** на **даден информационен ресурс**, спрямо неговите **харесвания** и **нехаресвания**.

Вход:

- Въвеждаме **даден речник**, съдържащ **броя харесвания** и **броя нехаресвания**.

Изход:

- Програмата принтира **рейтинга** на ресурса в **проценти**.

Ето как програмата трябва да работи:

Вход	Изход

`data = {`
`    'likes': 1000,`
`    'dislikes': 20`
`}`
	`98%`

Вход	**Изход**

`data = {`
`    'likes': 800,`
`    'dislikes': 400`
`}`
	`66%`

Вход	**Изход**

`data = {`
`    'likes': 700,`
`    'dislikes': 700`
`}`
	`50%`

Задача 17. Points Calculation

Напишете програма, която **пресмята общия брой точки**, спрямо **определен брой решени задачи**.

Таблица, **която показва точките за различните задачи**, спрямо тяхното **ниво на трудност**:

Ниво на трудност	Точки
Easy	5
Normal	20
Hard	50

Вход:

- Въвеждаме **даден речник**, съдържащ **броя решени задачи** за всяко **ниво на трудност**.

Изход:

- Програмата принтира **общия брой точки**.

Ето как програмата трябва да работи:

Вход	Изход

`solved_problems = {`
`    'easy': 12,`
`    'normal': 15,`
`    'hard': 5`
`}`
	`610`

Вход	**Изход**

`solved_problems = {`
`    'easy': 278,`
`    'normal': 45,`
`    'hard': 32`
`}`
	`3890`

Задача 18. The Most Expensive Car

Напишете програма, която принтира **марката на колата с най-висока цена от дадено множество от коли**.

Използвайте следния код:

Списък от речници

```
cars = [
    { 'brand': 'Honda', 'price': 5420 },
    { 'brand': 'Ford', 'price': 12100 },
    { 'brand': 'BMW', 'price': 15300 },
    { 'brand': 'Mercedes', 'price': 32000 },
    { 'brand': 'Citroen', 'price': 17000 },
    { 'brand': 'Ferrari', 'price': 79100 },
    { 'brand': 'Volvo', 'price': 1200 },
    { 'brand': 'Bentley', 'price': 89000 },
    { 'brand': 'Renault', 'price': 10000 },
    { 'brand': 'Audi', 'price': 12700 },
    { 'brand': 'Opel', 'price': 14200 },
    { 'brand': 'Nissan', 'price': 6500 }
]
```

Задача 19. The Cheapest Book

Напишете програма, която принтира името на книгата с **най-ниска цена** от дадено множество от книги.

Използвайте следния код:

Списък от речници

```
books = [
    { 'name': 'Introduction to Algorithms', 'price': 59.99 },
    { 'name': 'Design Patterns', 'price': 44.49 },
    { 'name': 'Programming with Python', 'price': 18.29 },
    { 'name': 'Programming with C#', 'price': 55.99 },
    { 'name': 'JavaScript Introduction', 'price': 36.67 },
    { 'name': 'Programming with C++', 'price': 18.19 },
    { 'name': 'Go Language', 'price': 32.99 },
    { 'name': 'Linux Introduction', 'price': 79.99 },
    { 'name': 'Programming with Ruby', 'price': 37.99 },
    { 'name': 'Functional Programming', 'price': 29.99 },
    { 'name': 'HTML5 and CSS3', 'price': 79.99 },
    { 'name': 'jQuery Introduction', 'price': 21.99 },
    { 'name': 'XML', 'price': 26.99 },
    { 'name': '.NET Framework', 'price': 29.99 },
    { 'name': 'Programming with Scratch', 'price': 26.99 },
    { 'name': 'DOM Manipulation', 'price': 25.99 },
    { 'name': 'Web Fundamentals', 'price': 37.99 },
    { 'name': 'PHP & WordPress', 'price': 48.59 },
    { 'name': 'Unreal Engine 5', 'price': 87.45 },
    { 'name': 'Java Programming', 'price': 13.99 }
]
```

Задача 20. The Highest Rated Game

Напишете програма, която принтира **името на играта с най-висок рейтинг от дадено множество от компютърни игри**.

Използвайте следния код:

Списък от речници

```
games = [
    { 'name': 'GTA V', 'rating': 9.76 },
    { 'name': 'Fortnite', 'rating': 9.78 },
    { 'name': 'The Last of Us', 'rating': 8.95 },
    { 'name': 'Spellbreak', 'rating': 8.75 },
    { 'name': 'World of Warcraft', 'rating': 10 },
    { 'name': 'Diablo II', 'rating': 9.73 },
    { 'name': 'Mortal Kombat X', 'rating': 9.42 },
    { 'name': 'God of War 4', 'rating': 9.79 },
    { 'name': 'Fall Guys', 'rating': 9.71 },
    { 'name': 'Little Nightmares', 'rating': 9.84 },
    { 'name': 'Minecraft', 'rating': 9.78 },
    { 'name': 'Roblox', 'rating': 9.79 },
    { 'name': 'Counter Strike', 'rating': 9.75 },
    { 'name': 'FIFA 2k21', 'rating': 9.77 },
    { 'name': 'Super Mario', 'rating': 8.45 },
    { 'name': 'Paladins', 'rating': 9.65 },
    { 'name': 'PUBG', 'rating': 9.45 },
    { 'name': 'Hearthstone', 'rating': 9.89 }
]
```

Задача 21. Party Invitations

Напишете програма, която проверява **дали дадено лице е поканено на парти**.

Използвайте следния код:

Списък от речници

```
invited_guests = [
    { 'first_name': 'Ivan', 'last_name': 'Georgiev' },
    { 'first_name': 'Stoyan', 'last_name': 'Vladimirov' },
    { 'first_name': 'Petar', 'last_name': 'Marinov' },
    { 'first_name': 'Kostadin', 'last_name': 'Kostadinov' },
    { 'first_name': 'Krasimir', 'last_name': 'Anastasov' },
    { 'first_name': 'Alexander', 'last_name': 'Popov' },
    { 'first_name': 'Stanimir', 'last_name': 'Georgiev' },
    { 'first_name': 'Daniel', 'last_name': 'Ivanov' },
    { 'first_name': 'Elizabeth', 'last_name': 'Borisova' },
    { 'first_name': 'Georgi', 'last_name': 'Petrov' },
    { 'first_name': 'Milena', 'last_name': 'Petkova' },
    { 'first_name': 'Mladen', 'last_name': 'Stoychev' }
]
```

Вход:

- Въвеждаме **имената на даден човек**.

Изход:

- Програмата принтира **булева стойност**, която указва **дали лице с тези имена е поканено на парти**.

Ето как програмата трябва да работи:

Вход	Изход
`Stanimir Georgiev`	`True`
Вход	**Изход**
`Ivan Bilev`	`False`

Задача 22. Dictionary Keys

Напишете **функция**, която **преобразува стойностите на даден ключ на даден речник** в **списък**.

Вход:

- Въвеждаме **даден речник**.

Изход:

- Програмата принтира **списък**, **чийто елементи представляват ключовете на дадения речник**.

Ето как програмата трябва да работи:

Вход	Изход
`people = [ { 'name': 'George' }, { 'name': 'Tim' }, { 'name': 'Simon' }, { 'name': 'Harry' } ]`	`[George, Tim, Simon, Harry]`
Вход	**Изход**
`people = [ { 'name': 'Tom' }, { 'name': 'Jeff' }, { 'name': 'Mark' }, { 'name': 'John' } ]`	`[Tom, Jeff, Mark, John]`
Вход	**Изход**
`people = [ { 'name': 'Kelly' }, { 'name': 'Fred' }, { 'name': 'Bob' }, { 'name': 'Greg' } ]`	`[Kelly, Fred, Bob, Greg]`

Задача 23. Secret Society Function

Напишете функция, която намира **кодовото име** на **секретна мисия**.

Кодовото име на дадена **секретна мисия** представлява **текст**, съставен от **първите букви** на **собствените имена** на всеки **таен агент**, участващ в **мисията**.

Вход:

- Въвеждаме **списък**, съставен от **имената** и **годините** на **тайните агенти**, участващи в мисията.

Изход:

- Програмата принтира **кода на секретната мисия**.

Ето как програмата трябва да работи:

Вход	Изход

`agents = [`
`{ 'name': 'Tim' },`
`{ 'name': 'Ben' },`
`{ 'name': 'Jeff' },`
`{ 'name': 'Anna' }`
`]`
	`TBJA`

Вход	**Изход**

`agents = [`
`{ 'name': 'Anna' },`
`{ 'name': 'Robert' },`
`{ 'name': 'Petter' },`
`{ 'name': 'William' }`
`]`
	`ARPW`

Задача 24. Compose URL

Напишете програма, която **конструира URL адрес**, използвайки **стойностите на даден речник**.

Вход:

- Въвеждаме **даден речник**, съдържащ съставните части на **URL адрес**.

Изход:

- Програмата принтира **конструирания URL адрес**.

Ето как програмата трябва да работи:

Вход	Изход

`parts = {`
`    'protocol': 'https',`
`    'sub_domain': 'www',`
`    'domain': 'google.com',`
`}`
	`https://www.google.com`

Вход	**Изход**

`parts = {`
`    'protocol': 'https',`
`    'sub_domain': None,`
`    'domain': 'website.bg',`
`}`
	`https://website.bg`

Вход	**Изход**

`parts = {`
`    'protocol': 'https',`
`    'sub_domain': 'www',`
`    'domain': 'youtube.com',`
`}`
	`https://www.youtube.com`

Задача 25. Employees Data

Напишете програма, която **намира**:

- **Двете имена на най-младия служител** във фирмата.
- **Двете имена на най-възрастния служител** във фирмата.

Използвайте следния код:

Списък от речници

```
employees = [
    { 'firstName': 'Ivan', 'age': 23 },
    { 'firstName': 'Stoyan', 'age': 27 },
    { 'firstName': 'Vladimir', 'age': 28 },
    { 'firstName': 'Petar', 'age': 33 },
    { 'firstName': 'Kostadin', 'age': 29 },
    { 'firstName': 'Stefan', 'age': 22 },
    { 'firstName': 'Krasimir', 'age': 37 },
    { 'firstName': 'Maria', 'age': 43 },
    { 'firstName': 'Alexander', 'age': 41 },
    { 'firstName': 'Stanimir', 'age': 24 },
    { 'firstName': 'Hristo', 'age': 41 },
    { 'firstName': 'Daniel', 'age': 49 },
    { 'firstName': 'Anna', 'age': 29 },
    { 'firstName': 'Elizabeth', 'age': 31 },
    { 'firstName': 'Georgi', 'age': 27 },
    { 'firstName': 'Nikola', 'age': 23 },
    { 'firstName': 'Milena', 'age': 32 },
    { 'firstName': 'Yordan', 'age': 38 },
    { 'firstName': 'Mladen', 'age': 25 }
]
```

Задача 26. Free Shipping

Напишете програма, която принтира **имената на всички продукти**, които **попадат в категория продукти с безплатна доставка**.

Даден продукт попада в категория продукти с **безплатна доставка** ако **цената му надвишава $200**.

Използвайте следния код:

Списък от речници

```
products = [
    { 'name': 'Flatscreen TV', 'price': '$199.50' },
    { 'name': 'MacBook', 'price': '$2883.99' },
    { 'name': 'Airfryer', 'price': '$152.89' },
    { 'name': 'Backpack', 'price': '$69.50' },
    { 'name': 'Camera', 'price': '$99.90' },
    { 'name': 'Razer headphones', 'price': '$358.50' },
    { 'name': 'Samsung watch', 'price': '$299.50' },
    { 'name': 'Keyboard', 'price': '$15.85' },
    { 'name': 'Playstation 5', 'price': '$1199' },
    { 'name': 'Gaming mouse', 'price': '$71.50' },
    { 'name': 'Adidas shoes', 'price': '$131.50' },
    { 'name': 'Gaming mousepad', 'price': '$23.00' },
    { 'name': 'Vacuum cleaner', 'price': '$29.99' },
    { 'name': 'Gaming desk', 'price': '$259.90' },
    { 'name': 'Washing machine', 'price': '$399.19' },
    { 'name': 'Smart TV', 'price': '$1999.29' },
    { 'name': 'Notebook', 'price': '$19.99' }
]
```

Задача 27. Equal Dictionaries

Напишете програма, която **проверява дали два речника съхраняват еднакви стойности**.

Вход:

- Въвеждаме **два речника**.

Изход:

- Програмата принтира **булева стойност**, която указва **дали двата речника съхраняват еднакви стойности еднакви**.

Ето как програмата трябва да работи:

Вход	Изход

`first_person = {`
`    'first_name': 'John',`
`    'last_name': 'Doe',`
`    'age': 23`
`}`
 	
`second_person = {`
`    'first_name': 'John',`
`    'last_name': 'Doe',`
`    'age': 23`
`}`
	`True`

Вход	Изход

`point = {`
`    'x': 2,`
`    'y': 3`
`}`
 	
`point = {`
`    'x': 5,`
`    'y': 6`
`}`
	`False`

Задача 28. Football Champions

Напишете програма, която принтира **името на отбора**, **отбелязал най-много точки**.

Точките на всеки отбор се сформират чрез следната формула:

Points = wins * 3 + 0 * loss + 1 * draws

Използвайте следния код:

Списък от речници

```
champions = [
    { 'name': 'Arsenal', 'win': 12, 'loss': 4, 'draw': 2 },
    { 'name': 'Chelsea', 'win': 10, 'loss': 4, 'draw': 4 },
    { 'name': 'Real Madrid', 'win': 15, 'loss': 4, 'draw': 2 },
    { 'name': 'Liverpool', 'win': 5, 'loss': 1, 'draw': 1 },
    { 'name': 'Fulham', 'win': 3, 'loss': 2, 'draw': 0 },
    { 'name': 'Aston Villa', 'win': 5, 'loss': 1, 'draw': 0 },
    { 'name': 'Southampton', 'win': 7, 'loss': 0, 'draw': 1 },
    { 'name': 'Juventus', 'win': 1, 'loss': 2, 'draw': 0 },
    { 'name': 'Atlanta', 'win': 3, 'loss': 2, 'draw': 2 },
    { 'name': 'Roma', 'win': 2, 'loss': 2, 'draw': 0 },
    { 'name': 'Sevilla', 'win': 5, 'loss': 2, 'draw': 1 },
    { 'name': 'Villarreal', 'win': 7, 'loss': 1, 'draw': 2 }
]
```

Задача 29. Students Average Grades

Напишете програма, която намира **средноаритметичната оценка на всеки ученик**.

Използвайте следния код:

Списък от речници

```
students = [
    { 'name': 'Hristo', 'grades': [4, 5, 6, 5.5, 4, 3, 5, 6] },
    { 'name': 'Maria', 'grades': [5, 4, 3, 5, 6, 5, 5, 5, 5] },
    { 'name': 'Dimitar', 'grades': [4, 4, 5, 4, 4, 5, 5, 4] },
    { 'name': 'Martin', 'grades': [2.5, 5, 3, 3, 3, 4, 2, 3] },
    { 'name': 'Stoyan', 'grades': [5, 5, 6, 5.5, 5, 4.5, 5] },
    { 'name': 'Tom', 'grades': [4, 5, 4, 4, 5.5, 5, 5, 5, 4] },
    { 'name': 'Georgi', 'grades': [4, 4, 3, 4, 4, 3, 4, 5] },
    { 'name': 'Kristiyan', 'grades': [2, 2, 3, 2, 2, 2, 3] },
    { 'name': 'Valentin', 'grades': [5, 5, 4, 5, 6, 6, 5, 5] },
    { 'name': 'Monica', 'grades': [4, 5, 6, 5, 4, 4, 3, 5] },
    { 'name': 'Ivan', 'grades': [4, 5, 5, 4, 5, 5, 5, 5, 5] },
    { 'name': 'Alexandra', 'grades': [4, 5, 6, 6, 3, 6, 6] },
    { 'name': 'Asen', 'grades': [4, 5, 5, 5, 5, 5, 4.5, 3.5] },
    { 'name': 'Nikolay', 'grades': [5, 4, 5, 6, 5, 4, 6, 5] },
    { 'name': 'Anton', 'grades': [4, 4.5, 6, 5, 4, 5, 6, 4] },
    { 'name': 'Eleonora', 'grades': [5, 3, 4, 5, 6, 6, 5, 4] },
    { 'name': 'Atanas', 'grades': [6, 5, 6, 4, 6, 5, 4, 5] },
    { 'name': 'Blagoy', 'grades': [4, 5, 6, 4, 6, 5, 6, 6] },
    { 'name': 'Boris', 'grades': [5, 5, 5, 5, 4, 6, 4, 5, 6] }
]
```

Задача 30. Students Max Grades

Напишете програма, която намира **максималната оценка на всеки ученик**.

Използвайте следния код:

Списък от речници

```
students = [
    { 'name': 'Hristo', 'grades': [4, 5, 6, 5.5, 4, 3, 5, 6] },
    { 'name': 'Maria', 'grades': [5, 4, 3, 5, 6, 5, 5, 5, 5] },
    { 'name': 'Dimitar', 'grades': [4, 4, 5, 4, 4, 5, 5, 4] },
    { 'name': 'Martin', 'grades': [2.5, 5, 3, 3, 3, 4, 2, 3] },
    { 'name': 'Stoyan', 'grades': [5, 5, 6, 5.5, 5, 4.5, 5] },
    { 'name': 'Tom', 'grades': [4, 5, 4, 4, 5.5, 5, 5, 5, 4] },
    { 'name': 'Georgi', 'grades': [4, 4, 3, 4, 4, 3, 4, 5] },
    { 'name': 'Kristiyan', 'grades': [2, 2, 3, 2, 2, 2, 3] },
    { 'name': 'Valentin', 'grades': [5, 5, 4, 5, 6, 6, 5, 5] },
    { 'name': 'Monica', 'grades': [4, 5, 6, 5, 4, 4, 3, 5] },
    { 'name': 'Ivan', 'grades': [4, 5, 5, 4, 5, 5, 5, 5, 5] },
    { 'name': 'Alexandra', 'grades': [4, 5, 6, 6, 3, 6, 6] },
    { 'name': 'Asen', 'grades': [4, 5, 5, 5, 5, 5, 4.5, 3.5] },
    { 'name': 'Nikolay', 'grades': [5, 4, 5, 6, 5, 4, 6, 5] },
    { 'name': 'Anton', 'grades': [4, 4.5, 6, 5, 4, 5, 6, 4] },
    { 'name': 'Eleonora', 'grades': [5, 3, 4, 5, 6, 6, 5, 4] },
    { 'name': 'Atanas', 'grades': [6, 5, 6, 4, 6, 5, 4, 5] },
    { 'name': 'Blagoy', 'grades': [4, 5, 6, 4, 6, 5, 6, 6] },
    { 'name': 'Boris', 'grades': [5, 5, 5, 5, 4, 6, 4, 5, 6] }
]
```

Текстообработка

Защо обработваме текст?

Голяма част от съдържанието в **Интернет** е под формата на **текст** - блог статии, описания на продукти, документи, електронни книги и много, много други. Работата с текст включва разнообразна обработка - **търсене и заместване на думи и символи в текст**, **броене на думи в текст**, **изрязване на части от текст** и много други.

Символен низ

Символен низ наричаме **последователност от символи**. Всяка последователност от символи в езика **Python** трябва да бъде оградена в (**единични кавички**) `''` или (**двойни кавички**) `""`.

Пример за употреба на символни низове

```
user_greeting = 'Hello, user!'

print(userGreeting) # Hello, user!

user_message = "Hello, user!"

print(userMessage) # Hello, user!
```

Всеки един **символен низ** представлява **списък от символи** – **последователност от N на брой символи**.

Празен символен низ

Езиците за програмиране позволяват създаване на списък с **0 на брой елемента**. Тъй като символният низ представлява списък, то можем да създадем символен низ с **0 на брой елемента** – **празен символен низ**.

Празен символен низ

```
empty_string = ''

print(empty_string)
```

Сравняване на символни низове

Символните низове са **сравними типове**. Те могат да бъдат **сравнявани за еднаквост** (чрез операторите **==** и **!=**) или за **азбучна подредба** (чрез операторите **>**, **>=**, **<** и **<=**).

Сравняване на символни низове

```
ben = 'Ben'
tim = 'Tim'

print(ben == tim) # False
print(ben != tim) # True
print(ben > tim) # False
print(ben < tim) # True
```

Конкатениране на символни низове

Операторът **+** (**плюс**), поставен **между два символни низа**, има смисъл на **долепване** (**конкатениране**) на символните низове.

Конкатениране на символни низове

```
username = 'username'
email_provider = '@gmail.com'
email_address = username + email_provider

print(email_address) # username@gmail.com
```

В случай, че конкатенираме имена, трябва ръчно да добавим **разделител** (**интервал**) между двете имена.

Конкатениране на имена

```
first_name = 'John'
last_name = 'Silver'
full_name = first_name + ' ' + last_name

print(full_name) # John Silver
```

Полезни функции

Функцията lower()

Функцията lower() конвертира символна последователност, изписвайки я изцяло с малки букви. Функцията връща нова символна последователност.

Функцията lower()

```
text = 'Hello, World!'
result = text.lower()

print(result) # hello, world!
```

Функцията upper()

Функцията upper() конвертира символна последователност, изписвайки я изцяло с главни букви. Функцията връща нова символна последователност.

Функцията upper()

```
text = 'Hello, World!'
result = text.upper()

print(result) # HELLO, WORLD!
```

Функцията startswith()

Функцията startswith() проверява дали дадена символна последователност започва с даден символен низ. Функцията връща булева стойност.

Функцията startswith()

```
text = 'Hello, World!'
result = text.startswith('Hello')

print(result) # True
```

Функцията endswith()

Функцията endswith() проверява дали дадена символна последователност завършва с даден символен низ. Функцията връща булева стойност.

Функцията endswith()

```
text = 'Hello, World!'
result = text.endswith('World!')

print(result) # True
```

Функцията replace()

Функцията replace() заменя символна последователност от даден символен низ с друга. Функцията връща символна последователност.

Функцията replace()

```
text = 'Hello, World!'

hello_python = text.replace('World', 'Python')
print(hello_python) # Hello, Python!

hello_python = text.replace('!', '')
print(hello_python) # Hello, Python
```

Функцията split()

Функцията split() разделя дадена символна последователност по подаден разделител. Функцията връща списък, съставен от символни низове.

Функцията split()

```
numbers = '1, 2, 3, 4, 5, 6'
list = numbers.split(', ')

for x in list:
    print(x)
```

Функцията strip()

Функцията strip() премахва ненужни символи като интервали в началото и края на дадена символна последователност. Функцията връща символна последователност.

Функцията strip()

```
text = '        Hello, World!         '
result = text.strip()

print(result) # Hello, World!
```

Обработка на символни низове

Ключовата дума in

Ключовата дума in проверява дали дадена символна последователност се среща в даден символен низ. Изразът връща булева стойност.

Ключовата дума in

```
text = 'Hello, World!'

includes_comma = ',' in text
print(includes_comma) # True

includes_question_mark = '?' in text
print(includes_question_mark) # False

symbol = 'e'

if symbol in text:
    print('contains ' + symbol)
else:
    print('does not contain ' + symbol)
```

Изрязване на символен низ

Python позволява употребата на специален синтаксис, който се използва за изрязване на част от символен низ.

Изрязване на символни низове

```
text = 'Hello, World!'

world = text[7:]
print(world) # World!
print(world.lower()) # world!

hello = text[0:5]
print(hello) # Hello
print(hello.lower()) # hello

hello_world = text[0:12]
print(hello_world) # Hello, World
print(hello_world.lower()) # hello, world
```

Текстообработка: Упражнения

Задача 1. Count Symbols

Напишете програма, която принтира **броя символи** на произволен **символен низ**.

Вход:

- Въвеждаме **даден символен низ**.

Изход:

- Програмата принтира **броя символи**.

Ето как програмата трябва да работи:

Вход	Изход
`Hello`	`5`
Вход	**Изход**
`Hello, world!`	`13`
Вход	**Изход**
`Hello, my name is John!`	`23`

Задача 2. Three Letters

Напишете програма, която принтира **всички думи**, **с дължина 3 символа**, в даден **списък от думи**.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **всички думи**, **с дължина 3 символа**.

Ето как програмата трябва да работи:

Вход	Изход
`['fox', 'bear', 'cat', 'lion']`	`fox` `cat`
Вход	**Изход**
`['Ben', 'Anna', 'John', 'Tim']`	`Ben` `Tim`
Вход	**Изход**
`['earth', 'water', 'air']`	`air`

Задача 3. Credit Card

Напишете програма, която **прикрива номера на кредитна карта**.

Нека приемем, че **всеки номер е съставен от 16 цифри**.

Програмата **прикрива** номера на **кредитна карта** като заменя **първите 12 цифри** от всеки номер със **символа *** (**звезда**).

Вход:

- Въвеждаме **номера на кредитната карта**.

Изход:

- Програмата принтира номера на кредитната карта, **след прикриването на първите 12 цифри**.

Ето как програмата трябва да работи:

Вход	Изход
1236987854639512	************9512
Вход	**Изход**
6958954789658962	************8962
Вход	**Изход**
1239764584546594	************6594

Задача 4. Player Stats

Напишете програма, която **визуализира оставащите животи** на **даден герой**.

Вход:

- Въвеждаме **стойността на текущите животи**.
- Въвеждаме **стойността на максималните животи**.

Изход:

- Програмата принтира **текст**, **представящ животите на героя**.

Ето как програмата трябва да работи:

Вход	Изход
5 10	|||||-----
Вход	**Изход**
2 10	||--------
Вход	**Изход**
9 10	|||||||||-
Вход	**Изход**
1 10	|---------

Задача 5. Vowels Sum

Напишете програма, която **преброява колко гласни букви** се съдържат в дадена **дума**, написана на **английски език**.

Вход:

- Въвеждаме **дадена дума**.

Изход:

- Програмата принтира **броя на гласните букви в думата**.

Ето как програмата трябва да работи:

Вход	Изход
`Hello`	`2`
Вход	**Изход**
`Wolrd`	`1`
Вход	**Изход**
`JavaScript`	`3`
Вход	**Изход**
`JavaScript is awesome!`	`8`

Задача 6. Search in Text

Напишете програма, която проверява **дали дадена дума се среща в даден текст**.

Използвайте следния текст:

Текст
JavaScript is a scripting or programming language that allows you to implement features on web pages – every time a web page does more than just sit there and display static information for you to look at – displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. – you can bet that JavaScript is probably involved.

Вход:

- Въвеждаме **дадена дума**.

Изход:

- Програмата принтира **булева стойност**, която указва **дали думата се среща в текста**.

Ето как програмата трябва да работи:

Вход	Изход
`JavaScript`	`True`
Вход	**Изход**
`programmer`	`False`
Вход	**Изход**
`language`	`True`
Вход	**Изход**
`awesome`	`False`

Задача 7. Substring Function

Напишете програма, която **извлича подниз от даден символен низ**, **стартирайки от дадена позиция**.

Вход:

- Въвеждаме **даден символен низ**.
- Въвеждаме **стартова позиция** (**индекс**).

Изход:

- Програмата принтира **текста**.

Ето как програмата трябва да работи:

Вход	Изход
abc 1	bc
Вход	**Изход**
JavaScript 4	Script
Вход	**Изход**
Hello, world! 0	Hello, world!

Задача 8. Reverse Text

Напишете програма, която **обръща даден символен низ наобратно**.

Вход:

- Въвеждаме **даден символен низ**.

Изход:

- Програмата принтира **символния низ**, **изписан наобратно**.

Ето как програмата трябва да работи:

Вход	Изход
`Hello`	`olleH`
Вход	**Изход**
`JavaScript`	`tpircSavaJ`
Вход	**Изход**
`C++`	`++C`
Вход	**Изход**
`hacker`	`rekcah`

Задача 9. English Alphabet

Напишете програма, която **превръща даден списък в текст**. Всеки елемент на списъка представлява **пореден номер на буква** в **английската азбука**.

Елементите на списъка са числа в интервала [1 ... 26].

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **думата**, която се образува.

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 3]`	`abc`
Вход	**Изход**
`[8, 5, 12, 12, 15]`	`hello`
Вход	**Изход**
`[1, 16, 16, 12, 5]`	`apple`
Вход	**Изход**
`[25, 5, 19]`	`yes`

Задача 10. Letters & Digits

Напишете програма, която **преброява колко букви** и **колко цифри** се **съдържат в даден символен низ**.

Вход:

- Въвеждаме **даден символен низ**.

Изход:

- Програмата принтира **броя на буквите и броя на цифрите**.

Ето как програмата трябва да работи:

Вход	Изход
`Hello World`	`Letters: 10` `Digits: 0`
Вход	**Изход**
`H3110 W0r1d`	`Letters: 4` `Digits: 6`
Вход	**Изход**
`151516`	`Letters: 0` `Digits: 6`

Задача 11. Expand a Number

Напишете програма, която изписва **дадено число като сума**, използвайки **неговите цифри** и **степени на числото 10**.

Вход:

- Въвеждаме **произволно число**.

Изход:

- Програмата принтира **символен низ**, показващ **всички събираеми**, **участващи в сумата**.

Ето как програмата трябва да работи:

Вход	Изход
`123`	`100 + 20 + 3`
Вход	**Изход**
`47834`	`40000 + 7000 + 800 + 30 + 4`
Вход	**Изход**
`9834`	`9000 + 800 + 30 + 4`

Задача 12. Email Address Validation

Напишете програма, която **проверява дали даден символен низ представлява валиден email адрес**.

За да бъде валиден **email** адрес, **символният низ** трябва да **завършва** с един от изброените символни низове: **@gmail.com**, **@protonmail.com**, **@yahoo.com**.

Вход:

- Въвеждаме **даден символен низ**.

Изход:

- Програмата принтира **булева стойност**, която указва **дали символният низ представлява валиден email адрес**.

Ето как програмата трябва да работи:

Вход	Изход
valentin.petrov@gmail.com	True
Вход	**Изход**
business_mail@abv.bg	False
Вход	**Изход**
official.mail@yahoo.com	True
Вход	**Изход**
encrypted@protonmail.com	True

Задача 13. Name Validation

Напишете програма, която **проверява дали въведен текст представлява валидно име**.

Приемаме за **валидно име всеки символен низ**, който отговаря на следните **изисквания**:

- **Дължината на текста е между 3 и 20 символа**.
- **Текстът започва с главна буква**, **всички останали символи са малки букви**.
- **Текстът съдържа само букви от латинската азбука**.

Вход:

- Въвеждаме **даден символен низ**.

Изход:

- Програмата принтира **булева стойност**, която указва **дали символният низ представлява валидно име.**

Ето как програмата трябва да работи:

Вход	Изход
`Hello`	`True`
Вход	**Изход**
`E4t-34k`	`False`
Вход	**Изход**
`Alexander`	`True`

Задача 14. PIN Validation

Напишете програма, която **проверява дали въведен текст представлява валиден уникален идентификационен номер**.

Приемаме за **валиден уникален идентификационен номер всеки символен низ**, който отговаря на следните **изисквания**:

- **Дължината на текста е между 4 и 6 символа**.
- **Текстът съдържа само символите**, **представляващи арабските цифри** – (**0 – 9**).
- **Текстът не съдържа интервали**.

Вход:

- Въвеждаме **даден символен низ**.

Изход:

- Програмата принтира **булева стойност**, която указва **дали символният низ представлява валидно име**.

Ето как програмата трябва да работи:

Вход	Изход
`1234`	`True`
Вход	**Изход**
`35 65`	`False`
Вход	**Изход**
`45b54a`	`False`

Задача 15. Phone Number Formatting

Напишете програма, която **преобразува даден списък от цифри** в **телефонен нормер** с **определен формат**.

Вход:

- Въвеждаме **елементите на списъка**.

Изход:

- Програмата принтира **телефонен номер**, съставен спрямо показания **формат**.

Ето как програмата трябва да работи:

Вход	Изход
`[1, 2, 3, 4, 5, 6, 7, 8, 9, 0]`	`(123) 456-7890`
Вход	**Изход**
`[5, 1, 9, 5, 5, 5, 4, 4, 6, 8]`	`(519) 555-4468`
Вход	**Изход**
`[3, 4, 5, 5, 0, 1, 2, 5, 2, 7]`	`(345) 501-2527`
Вход	**Изход**
`[1, 3, 4, 5, 6, 2, 4, 9, 7, 4]`	`(134) 562-4974`

Задача 16. Word in Plural

Напишете програма, която **преобразува дадена дума** на **английски език от единствено число в множествено число**.

Вход:

- Въвеждаме **дадена дума на английски език**.

Изход:

- Програмата принтира **думата в множествено число**.

Ето как програмата трябва да работи:

Вход	Изход
potato	potatoes
Вход	**Изход**
butterfly	butterflies
Вход	**Изход**
car	cars
Вход	**Изход**
wolf	wolves
Вход	**Изход**
peach	peaches

Задача 17. Extract File Name

Напишете програма, която **извлича името на даден файл**, **спрямо пълния път на файла**.

Вход:

- Въвеждаме **валиден път на файл**.

Изход:

- Програмата принтира **името на файла**.

Ето как програмата трябва да работи:

Вход	Изход
`C:\Users\Users\avatar.png`	`avatar.png`
Вход	**Изход**
`E:\Downloads\game.exe`	`game.exe`
Вход	**Изход**
`F:\Music\Adele\Hello.mp3`	`Hello.mp3`

Задача 18. Palindrome Phrase

Напишете програма, която **проверява дали даден символен низ е палиндром**.

Палиндром наричаме **последователност** от **символи** или **числа**, която независимо в коя посока бива прочетена, носи един и същ смисъл.

Вход:

- Въвеждаме **даден символен низ**.

Изход:

- Програмата принтира **булева стойност**, която указва **дали символният низ е палиндром**.

Ето как програмата трябва да работи:

Вход	Изход
`racecar`	`True`
Вход	**Изход**
`value`	`False`
Вход	**Изход**
`madam`	`True`
Вход	**Изход**
`hello`	`False`

Задача 19. Pig Latin

Напишете програма, която **преобразува даден символен низ**, **променяйки всяка дума**.

За всяка дума се прилагат следните **действия**:

- **Първата буква става последна**.
- **Добавя се сричката ау** (на английски език) **в края на думата**.

Вход:

- Въвеждаме **даден символен низ**.

Изход:

- Програмата принтира **символния низ**, **след преобразуването**.

Ето как програмата трябва да работи:

Вход	Изход
`javascript`	`avascriptjay`
Вход	**Изход**
`programmer`	`rogrammerpay`
Вход	**Изход**
`brown`	`rownbay`
Вход	**Изход**
`fox`	`oxfay`

Задача 20. Password Generator

Напишете програма, която **при въведена дума генерира тайна парола**.

Думата трябва да бъде съставена точно от 9 букви.

Пример:

- **pineapple**

Следвайте стъпките:

- Първите **3 букви** се преобразуват, превръщайки се в следващите букви от английската азбука. **pin → qjo**. Буквата **z** се преобразува в буквата **a**.
- Вторите **3 букви** се изписват **наобратно**. **eap → pae**
- Последните 3 букви се преобразуват, заменяйки се с поредния им номер в английската азбука. **ple → 16125**

Вход:

- Въвеждаме **дадена дума**.

Изход:

- Програмата принтира **генерираната тайна парола**.

Ето как програмата трябва да работи:

Вход	Изход
pineapple	qjopae16125
Вход	**Изход**
signature	tjhtan21185
Вход	**Изход**
chocolate	diploc1205

Задача 21. H4ck3r Sp33ch

Напишете програма, която изписва **даден символен низ** на **английски език**, **заменяйки символите** в оригиналния текст.

Таблица със символите:

Символ	Преобразуване
`O или o`	`0`
`I или i`	`1`
`E или e`	`3`
`A или a`	`4`
`S или s`	`5`
`T или t`	`7`
`G или g`	`9`

Вход:

- Въвеждаме даден **символен низ**, съдържащ **латински букви**.

Изход:

- Програмата принтира **символния низ**, **след преобразуването**.

Ето как програмата трябва да работи:

Вход	Изход
`python is cool`	`py7h0n 15 c00l`
Вход	**Изход**
`programming is fun`	`pr09r4mm1n9 15 fun`

Задача 22. Roman Numerals

Напишете програма, която преобразува **римско число в арабско число**.

Използвайте следната таблица:

Римско число	Арабско число
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Вход:

- Въвеждаме **римското число**.

Изход:

- Програмата принтира **стойността**, **изписана с арабски цифри**.

Ето как програмата трябва да работи:

Вход	Изход
MMCCCLXV	2365
Вход	**Изход**
CCLIV	254
Вход	**Изход**
VIII	8

Задача 23. HTML Element

Напишете програма, която **извлича съдържанието на даден HTML елемент**.

HTML елементите са съставени от отварящ таг, **съдържание и затварящ таг**.

Вход:

- Въвеждаме **валиден HTML елемент**.

Изход:

- Програмата принтира **съдържанието на HTML елемента**.

Ето как програмата трябва да работи:

Вход	Изход
`<p>I am a paragraph</p>`	`I am a paragraph`
Вход	**Изход**
`<h1>Heading 1</h1>`	`Heading 1`
Вход	**Изход**
`<div>content</div>`	`content`

Задача 24. Isogram Function

Напишете **функция**, която **проверява дали дадена дума е изограма**. **Думата** се подава като **параметър на функцията**.

Изограмата е дума, **при която нито една буква не се повтаря**.

Функцията би трябвало да се използва по следния начин:

Isogram Function

`word = 'Javascript'`
 	
`isogram = is_isogram(word)`
`print(isogram) # False`

Ето как програмата трябва да работи:

Вход	Изход
python	True
Вход	**Изход**
java	False
Вход	**Изход**
kotlin	True
Вход	**Изход**
HTML	True

Задача 25. Pangram Function

Напишете **функция**, която **проверява** дали **дадено изречение** е **панграма**. **Изречението** се подава като **параметър на функцията**.

Панграмата е изречение, **при което се използват всички букви в азбуката**.

Функцията би трябвало да се използва по следния начин:

Pangram Function

`sentence = 'Javascript is awesome!'`
 	
`is_pangram = is_pangram_sentence(sentence)`
`print(is_pangram) # False`

Ето как програмата трябва да работи:

Вход	Изход
`Python is the best!`	`False`
Вход	**Изход**
`The quick brown fox jumps over the lazy dog.`	`True`
Вход	**Изход**
`The five boxing wizards jump quickly.`	`True`

Задача 26. Dictionary to String

Напишете програма, която **преобразува даден речник**, **съхраняващ данни за даден град**, в **текст**.

Вход:

- Въвеждаме **даден речник**.

Изход:

- Програмата принтира **символен низ**, **представящ информация за града**.

Ето как програмата трябва да работи:

Вход	Изход

`city = {`
`  'name': 'Paris',`
`  'population': 2140526,`
`  'continent': 'Europe'`
`}`
	`Paris has a population of 2,140,526 and is situated in Europe.`

Вход	**Изход**

`city = {`
`  'name': 'Sofia',`
`  'population': 1242568,`
`  'continent': 'Europe'`
`}`
	`Sofia has a population of 1,242,568 and is situated in Europe.`

Задача 27. Playlist

Напишете програма, която намира **името на песента** с **най-дълга продължителност**.

Използвайте следния код:

Списък от речници

```
playlist = [
    { 'title': 'Around the World', 'length': '07:09' },
    { 'title': 'Purple Rain', 'length': '08:40' },
    { 'title': 'Here I Go Again', 'length': '05:08' },
    { 'title': 'Thunderstruck', 'length': '04:52' },
    { 'title': 'Perfect Stranges', 'length': '05:28' },
    { 'title': 'People Are Strange', 'length': '02:10' },
    { 'title': 'Billie Jean', 'length': '04:53' },
    { 'title': 'Send Me An Angel', 'length': '04:33' },
    { 'title': 'Back In Black', 'length': '04:15' },
    { 'title': 'Wish You Were Here', 'length': '05:04' },
    { 'title': 'Light My Fire', 'length': '07:09' },
    { 'title': 'Under the Bridge', 'length': '04:24' },
    { 'title': 'Robot Rock', 'length': '04:47' },
    { 'title': 'Californication', 'length': '05:29' },
    { 'title': 'Careless Whisper', 'length': '05:00' },
    { 'title': 'Wind Of Change', 'length': '05:12' },
    { 'title': 'Still of the Night', 'length': '06:37' },
    { 'title': 'Otherside', 'length': '04:15' },
    { 'title': 'Always Be My Baby', 'length': '04:18' },
    { 'title': 'Money', 'length': '06:33' }
]
```

Задача 28. IPv4 Validation

Напишете програма, която **проверява дали дадена символна последователност представлява валиден IPv4 адрес**.

IPv4 е формат, **съставен от 4 числа в интервала [0 ... 255]**, **разделени със символа . (точка)**.

Вход:

- Въвеждаме **даден символен низ**.

Изход:

- Програмата принтира **булева стойност**, която указва **дали текстът представлява валиден IPv4 адрес**.

Ето как програмата трябва да работи:

Вход	Изход
`123.43.23.12`	`True`
Вход	**Изход**
`445.534.423.12`	`False`
Вход	**Изход**
`45.12.143.43`	`True`

Задача 29. Balanced Brackets

Напишете програма, която **приема символна последователност**, **съставена от скоби** – **{ }**, **( ) или [ ]**. Програмата трябва да провери **дали изразът е балансиран**.

Изразът е балансиран, **когато**:

- **Съдържа точно толкова отварящи скоби**, **колкото затварящи**.
- **Всяка отваряща скоба има съответстваща затваряща от същия вид**.

Вход:

- Въвеждаме **дадена символна последователност**.

Изход:

- Програмата принтира **булева стойност**, която указва **дали изразът е балансиран**.

Ето как програмата трябва да работи:

Вход	Изход
([{}])	True
Вход	**Изход**
()[]{}	True
Вход	**Изход**
(([]{)(]	False
Вход	**Изход**
{{{{}}}}	True

Задача 30. Alphabet Soup

Напишете програма, която **преобразува дадена дума**. Програмата трябва да преобразува думата така, че **нейните букви да бъдат сортирани** (**подредени**) **по азбучен ред**.

Вход:

- Въвеждаме **дадена дума**.

Изход:

- Програмата принтира **думата**, **след преобразуването**.

Ето как програмата трябва да работи:

Вход	Изход
`hello`	`ehllo`
Вход	**Изход**
`javascript`	`aacijprstv`
Вход	**Изход**
`hacker`	`acehkr`

Заключение

Прочитайки цялата книга и решавайки правилно всички задачи, вече сте усетили многократно чувствата на удовлетворение и щастие, които изпитвате, когато решението на дадена задача е коректно и написания код работи правилно.

Достигайки до този момент, вече сте покорили **първите си върхове в света на програмирането**!

Поздравления!

Продължавайте да се обучавате!

Следващите стъпки продължават с курса **Въведение в HTML и CSS**, където се изучават доста по-прости концепции като **HTML елементи** и **CSS стилизиране**. Езиците **HTML** и **CSS** са **изключително лесни и приятни за употреба**.

HTML и **CSS** са **технологиите**, които се използват за **задаване на съдържанието** и **стилизирането** на **Уеб документи** (**Уеб страници** и **Уеб сайтове**).

Курсът **Въведение в HTML и CSS** е една предпоставка за изучаване и разбиране в дълбочина на **механизмите**, чрез които се изграждат **Уеб сайтове** и **Уеб приложения**.

Комбинирайки познанията си по **HTML**, **CSS** и **Python** ще можете да създавате **Уеб сайтове**, достъпни през Вашия **браузър** (например, **Google Chrome** или **Mozilla Firefox**).

Комбинацията от **HTML**, **CSS** и **Python** позволява както **писането на потребителския интерфейс** за **дадено Уеб приложение** (**front-end частта**), така и писането на **бизнес логиката** (**back-end частта**).

Авторският екип на книгата Ви пожелава **успех** и късмет, както в личен, така и в професионален план!