Python Basics

Автори на книгата:

Борис Костов

Мартин Хърсовски

Python Basics Course @ Atlas IT Academy

Съдържание

Предговор

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

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

Книгата запознава читателя с **основни понятия** от сферата на програмирането – **езици за програмиране**, **променливи**, **работа с данни** (числа и текст), **прости пресмятания** и **проверки**.

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 **80 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 желаят да направят своите **първи стъпки** в света на програмирането и компютрите.

Първи стъпки в програмирането

В тази глава ще обясним накратко **какво представляват компютрите**, ще се запознаем с **програмирането** и ще обясним какво означава понятието **език за програмиране**. Ще се запознаем с езика **Python**, както и със **средата за разработка Microsoft Visual Studio Code**.

Какво представляват компютрите?

Компютрите са машини, които обработват **информация**.

Съществуват различни видове компютри. Компютрите, които използваме в ежедневието, се наричат **персонални** (лични) **компютри** (PCs – Personal Computers).

Всеки персонален компютър е изграден от няколко **основни съставни елемента**:

- **Процесор**. Процесорът представлява "мозъка" на компютъра.
- **Оперативна памет** (**RAM** памет).
- **Памет за съхранение** (**HDD** или **SSD**).
- **Входни устройства**. Взаимодействие с компютъра може да се осъществи посредством входни устройства – мишка, клавиатура, микрофон. Чрез тези устройства ние подаваме команди / инструкции / данни / информация на компютъра.
- **Изходни устройства**. Обратно, посредством изходните устройства компютърът ни поднася информация. Изходни устройства са екран, колонки, слушалки и други. Екранът може да бъде и входно устройство (touch screen).

Според преносимостта си, персоналните компютри биват два вида – **настолни компютри** (desktop personal computers) и **лаптопи** (laptops).

Според приложението си, персоналните компютри могат да се разпределят в множество категории – компютри с общо предназначение, компютри подходящи за учене, бизнес, подходящи за игри и други.

Какво означава "да програмираме"?

"Да програмираме" означава да **задаваме команди** на компютъра да свърши определена работа. Тези команди се задават в **писмен вид**, използвайки специален **език** (набор от команди), наречен **език за програмиране**.

Езици за програмиране

Езикът за програмиране представлява множество от **команди**. Всяка команда има своето **предназначение**.

При човешките езици се използват понятията **синтаксис** и **семантика**.

Синтаксис означава как изписваме думите и кога една дума е изписана правилно или грешно.

Семантика означава какво значение има дадената дума.

Подобно на човешките езици, при програмните езици всяка команда има **синтаксис** (начинът, по който се изписва) и **семантика** (какво означава).

Съществуват стотици езици за програмиране. Например **C++**, **C#**, **Java**, **JavaScript**, **Python**, **PHP** и други. Те се различават по **синтаксис** (изписване на командите) и **употреба** (уеб приложения, настолни приложения, игри и др.).

Езикът Python

Налични са няколко причини, поради които решихме да изберем езика **Python**.

Езикът **Python** е **език за програмиране от високо ниво**. Това означава, че командите, които пишем на **Python**, са по-близки до разбирането на човека, отколкото на компютъра / машината. Езиците за програмиране от високо ниво се характеризират с **опростен** и **приятен синтаксис**.

Езикът **Python** е **един от най-широкоизползваните езици за програмиране**. Той намира приложение при разработката на сайтове, игри, настолни приложения, обработка и анализ на данни и други.

Среда за разработка

Среда за разработка наричаме програма, която улеснява писането на **софтуер**. Най-често средите за разработка ни помагат в **организацията на файловете** за даден проект, а също така **оцветяват командите**, помагайки ни да се ориентираме по-лесно, докато пишем код.

В този курс ще използваме безплатната **среда за разработка Microsoft Visual Studio Code**.

Пресмятания

В тази глава ще се запознаем с понятието **променлива**, както и с това какво представляват **типовете от данни**. Ще се запознаем с това как можем да принтираме резултат на екрана, както и кои са основните операции, които можем да използваме, а именно – **събиране**, **изваждане**, **умножение** и **деление**.

Пресмятания в програмирането

Както вече споменахме, компютрите са машини, които обработват **информация**. Основната дейност на компютрите е пресмятането.

Всички данни, които програмите използват, за да свършат своята работа, се записват в **променливи**. Променливите имат **имена** и **стойности**.

Променливи в езика Python

```
name = 'George'

age = 11

students_count = 6

letter = 'a'

money = 2.50
```

Променливите имат **имена**, които посочват какви **данни** съхраняваме – например име, години, брой ученици и други. Променливите имат и **стойности** – данните, които съхраняваме.

Имената на променливите не трябва да съдържат интервали! Ако името на променливата е съставено от повече от една дума, използвайте **_** (**долна черта**) за **разделител между думите**.

Основни типове от данни

Както споменахме, променливите съхраняват **данни** / **стойности**. Всяка стойност може да бъде от различен **тип**. При решаването на повечето проблеми се налага да работим с **цели числа**, **дробни** (**реални**) **числа**, **символи** и **текст**.

Цели числа

Естествените числа са множеството от числата, които **започват от 0** и продължават да **отброяват до безкрайност**:

{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ... }

Често казваме, че **естествените числа са числата**, **с които броим**. Те **не** включват отрицателни числа и дроби.

Целите числа включват **естествените числа** (числата, с които броим) и **отрицателните числа** (числата по-малки от нула). Отново, без дробите.

{ ... -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5 ... }

Променливи, които съхраняват цели числа

```
age = 15

days_count = 7

subscribers = 1600

likes = 400

pictures_count = 2
```

Реални числа

Реалните числа включват **всички цели числа** и **всички дробни числа**.

Променливи, които съхраняват реални числа

```
product_price = 2.50

excellent_grade = 5.50

person_weight_kg = 54.2

shop_bill = 20.80
```

Символи и текст

Компютрите съхраняват данни под формата на нули и единици. Съхранението на символи става възможно, когато се зададе **специално число** на **всеки символ** / **буква**.

Модерните програмни езици използват таблицата **Unicode**, която вече е присвоила число на всеки символ, познат на човечеството. Таблицата **Unicode** съдържа над **100 000 символа**.

Таблицата **Unicode** прилича на речник – всеки символ има съответстващо число. Записвайки символ, в компютърната памет се записва неговото съответстващо число от **Unicode** таблицата.

Шрифтът (font) представлява набор от **изображения**, чрез които символите могат да бъдат **визуализирани на екрана**.

За да запишем **символ** в езика **Python**, трябва да оградим символа, използвайки **'** (**единична кавичка**). Текстът всъщност представлява **множество от подредени символи**.

Променливи, които съхраняват символи и текст

```
symbol = 'a'

letter = 'z'

name = 'George'

city = 'Sofia'
```

Python използва **Unicode**. Това означава, че можем да използваме и записваме всякакви видове символи – букви от всякакви **азбуки** и **писмености**, включително китайски йероглифи.

Типът данни на дадена стойност има значение, тъй като има разлика между стойност от тип текст и стойност от тип число.

Разликата между текст и число

```
text = '2' # Текстът '2', съставен от 1 символ
number = 2 # Числото 2
```

Принтиране на резултат

Принтирането на резултат на екрана се осъществява с вградената функция **print()**. Между двете скоби на командата трябва да поставим стойността, която желаем да принтираме – число, текст или стойност на променлива.

Принтиране на числа

```
print(2)

print(3)

print(5.5)
```

Принтиране на символи и текст

```
print('a')

print('text')

print('George')

print('Sofia')
```

Принтиране на стойности на променливи

```
age = 12
print(age) # 12

product_price = 2.50
print(product_price) # 2.50

name = 'George'
print(name) # George

days_count = 7
print(days_count) # 7
```

Взаимодействие с потребителя

Програмите, които видяхме, не са **интерактивни**. Единственото, което можем да направим, е да запишем стойност в променлива, след което да принтираме стойността на променливата.

За да направим програмите **интерактивни**, трябва да взаимодействаме с потребителя на нашата програма.

Всяка стойност, която потребителят въвежда, когато програмата се изпълнява, е под формата на **текст**. За да накараме потребителя да въведе стойност, трябва да използваме вградената **функционалност input()**.

Въвеждане на потребителски вход под формата на текст

```
text = input()
print(text)
```

Вградената **функционалност input()** ще спре изпълнението на програмата, докато потребителя не въведе стойност и натисне бутона **[Enter]**. Стойността, която потребителят ще въведе, ще се запише в променливата **text**.

Ако желаем да въведем число, трябва да използваме **допълнителна функционалност**, която **да превърне текста в число**. Ето как можем да превърнем текста в число:

Въвеждане на потребителски вход под формата на цяло число

```
number = int(input())
print(number)
```

Вградената функционалност **int()** превръща даден текст в цяло число. В случая, текстът, който желаем да превърнем в число, е текстът, който потребителят въвежда.

Въвеждане на потребителски вход под формата на реално число

```
number = float(input())
print(number)
```

Вградената функционалност **float()** превръща даден текст в дробно число, т.е. позволява въвеждането на дробно число.

Аритметични операции

В математиката са налични няколко **основни операции** при работа с числа – **събиране**, **изваждане**, **умножение** и **деление** – т. нар. **аритметични операции**.

Тези операции са налични и в **Python**. В контекста на програмите, обаче, компютърът ще извършва всички пресмятания вместо нас.

Събиране

За да извършим **събиране** в **Python**, използваме знака **+** (**плюс**). Когато събираме числа, резултатът от събирането също е число, което можем да запишем в **променлива**.

Събиране на числата 5 и 7

```
first_number = 5
second_number = 7

result = first_number + second_number
print(result) # 12
```

Събиране на цели числа, въведени от потребителя

```
first_number = int(input())
second_number = int(input())

result = first_number + second_number
print(result)
```

Събиране на реални числа, въведени от потребителя

```
first_number = float(input())
second_number = float(input())

result = first_number + second_number
print(result)
```

Изваждане

За да извършим **изваждане** в **Python**, използваме знака **-** (**минус**). Когато изваждаме числа, резултатът от изваждането също е число, което можем да запишем в **променлива**.

Изваждане на числата 9 и 7

```
first_number = 9
second_number = 7

result = first_number - second_number
print(result) # 2
```

Изваждане на числата 7 и 9

```
first_number = 7
second_number = 9

result = first_number - second_number
print(result) # -2
```

Изваждане на цели числа, въведени от потребителя

```
first_number = int(input())
second_number = int(input())

result = first_number - second_number
print(result)
```

Изваждане на реални числа, въведени от потребителя

```
first_number = float(input())
second_number = float(input())

result = first_number - second_number
print(result)
```

Умножение

За да извършим **умножение** в **Python**, използваме знака ***** (**звезда**). Когато умножаваме числа, резултатът от умножението също е число, което можем да запишем в **променлива**.

Умножение на числата 5 и 7

```
first_number = 5
second_number = 7

result = first_number * second_number
print(result) # 35
```

Умножение на числата 3 и 0

```
first_number = 3
second_number = 0

result = first_number * second_number
print(result) # 0
```

Умножение на цели числа, въведени от потребителя

```
first_number = int(input())
second_number = int(input())

result = first_number * second_number
print(result)
```

Умножение на реални числа, въведени от потребителя

```
first_number = float(input())
second_number = float(input())

result = first_number * second_number
print(result)
```

Деление

За да извършим **деление** в **Python**, използваме знака **/** (**наклонена черта**). Когато делим числа, резултатът от делението също е число, което можем да запишем в **променлива**.

Деление на числата 35 и 7

```
first_number = 35
second_number = 7

result = first_number / second_number
print(result) # 5.0
```

Деление на числата 5 и 1

```
first_number = 5
second_number = 1

result = first_number / second_number
print(result) # 5.0
```

Деление на цели числа, въведени от потребителя

```
first_number = int(input())
second_number = int(input())

result = first_number / second_number
print(result)
```

Деление на реални числа, въведени от потребителя

```
first_number = float(input())
second_number = float(input())

result = first_number / second_number
print(result)
```

Други операции

Намиране на остатък при целочислено деление

Остатъкът при целочислено деление показва колко единици от числото, което разделяме, не могат да бъдат разделени.

13 / **5** = **2** (остатък **3**)

За да извършим операцията намиране на **остатък** при целочислено деление, използваме знака **%** (**процент**). Когато намираме остатък при целочислено деление, резултатът от намирането на остатъка също е число, което можем да запишем в **променлива**.

Намиране на остатък при целочислено деление

```
first_number = 13
second_number = 5

remainder = first_number % second_number
print(remainder) # 3
```

Степенуване

Степенуването е **съкратен запис** на умножението на едно и също число **n на брой пъти**. Броят на повторенията при умножението се записва като **степен**.

$$\mathbf{2^5} = \mathbf{2} * \mathbf{2} * \mathbf{2} * \mathbf{2} * \mathbf{2}\ (\mathbf{5}\ \text{пъти})$$

За да извършим **степенуване** в **Python**, използваме знака ****** (**две звездички**). Когато степенуваме числа, резултатът от степенуването също е число, което можем да запишем в променлива.

Степенуване на числа

```
number = 2
power = 5

result = number ** power
print(result) # 32
```

Съединяване на текст

Освен за събиране, знакът **+** се употребява за **долепване** / **съединяване** на текст. Съединяването на текст се нарича **конкатениране**.

Долепяне на текст

```
first_name = 'George'
last_name = 'Ferris'
full_name = first_name + ' ' + last_name
print(full_name) # George Ferris

greeting = 'Hello, '
print(greeting + first_name) # Hello, George
print(greeting + full_name) # Hello, George Ferris
```

Изрази

Израз наричаме последователност от знаци и стойности, която има смисъл. В процеса на изпълнение на програмата, изразите се оценяват до конкретни стойности.

Изрази в Python

```
number = 1 + 2 - 5 + 10
print(number) # 8

another_number = number + 5
print(another_number) # 13
```

Както в математиката, в **Python** също можем да използваме **скоби**. Скобите се използват за задаване на **приоритет** на операциите, т.е. те показват кои операции трябва да бъдат пресметнати най-напред.

Използване на скоби

```
first_number = 1 + 2 * 5 + 10
print(first_number) # 21

second_number = (1 + 2) * 5 + 10
print(second_number) # 25
```

Пресмятания: Упражнения

Задача 1. Sum Numbers

Напишете програма, която **сумира две числа**, въведени от потребителя.

Вход:

- Въвеждаме първото число.
- Въвеждаме второто число.

Изход:

- Програмата принтира сумата на двете числа.

Следвайте стъпките:

1. Създайте променлива, която се казва **first_number** и съхранява стойността на първото число.
2. Създайте променлива, която се казва **second_number** и съхранява стойността на второто число.
3. Създайте променлива, която се казва **sum** и съхранява **сбора** (**сумата**) **на двете числа**.
4. Принтирайте стойността на променливата **sum**.

Ето как програмата трябва да работи:

Вход	Изход
2 5	7
Вход	**Изход**
5 9	14
Вход	**Изход**
25 36	61

Задача 2. Multiply Numbers

Напишете програма, която **умножава две числа**, въведени от потребителя.

Вход:

- Въвеждаме първото число.
- Въвеждаме второто число.

Изход:

- Програмата принтира произведението на двете числа.

Следвайте стъпките:

1. Създайте променлива, която се казва **first_number** и съхранява стойността на първото число.
2. Създайте променлива, която се казва **second_number** и съхранява стойността на второто число.
3. Създайте променлива, която се казва **product** и съхранява **произведението на двете числа**.
4. Принтирайте стойността на променливата **product**.

Ето как програмата трябва да работи:

Вход	Изход
2 3	6
Вход	**Изход**
20 3	60
Вход	**Изход**
12 12	144

Задача 3. BGN to EURO

Напишете програма, която обръща валути – **лева** към **евро**.

Левът е **паричната единица**, която използваме в **България**, а евро е паричната единица, която се използва в **Европа**.

Един лев е приблизително равен на половин евро (50 евро цента).

Вход:

- Въвеждаме сумата в лева.

Изход:

- Програмата принтира приблизителната сума в евро.

Следвайте стъпките:

1. Създайте променлива, която се казва **amount_bgn** и съхранява стойността на левовете.
2. Създайте променлива, която се казва **amount_euro** и съхранява приблизителната стойност на еврото след обръщане.
3. Принтирайте стойността на променливата **amount_euro**.

Ето как програмата трябва да работи:

Вход	Изход
2	1
Вход	**Изход**
14	7
Вход	**Изход**
16	8

Задача 4. Greet User

Напишете програма, която поздравява **потребител**.

Вход:

- Въвеждаме името на потребителя.

Изход:

- Програмата принтира съобщение, което поздравява потребителя.

Следвайте стъпките:

1. Създайте променлива, която се казва **username** и съхранява името на потребителя.
2. Създайте променлива, която се казва **user_greeting** и съхранява цялото съобщение, което поздравява потребителя.
3. Принтирайте стойността на променливата **user_greeting**.

Ето как програмата трябва да работи:

Вход	Изход
`George`	`Hello, George!`
Вход	**Изход**
`Ben`	`Hello, Ben!`
Вход	**Изход**
`Thomas`	`Hello, Thomas!`

Задача 5. Square Perimeter

Напишете програма, която пресмята **периметъра** (обиколката) **на квадрат** по въведена **дължина на страна**.

Квадратът е **фигура**, която има **4 страни**. Всичките **4 страни** на квадрата имат **еднаква дължина**.

Има два начина да намерим **периметъра** (обиколката) **на квадрата**. Първият начин е като **съберем дължините на всичките му страни**, а вторият – като **умножим дължината на едната му страна по 4**.

Вход:

- Въвеждаме страната на квадрата.

Изход:

- Програмата принтира периметъра на квадрата.

Следвайте стъпките:

1. Създайте променлива, която се казва **square_side** и съхранява дължината на страната на квадрата.
2. Създайте променлива, която се казва **square_perimeter** и съхранява стойността на периметъра на квадрата.
3. Принтирайте стойността на променливата **square_perimeter**.

Ето как програмата трябва да работи:

Вход	Изход
2	8
Вход	**Изход**
3	12
Вход	**Изход**
5	20

Задача 6. Square Area

Напишете програма, която пресмята **лицето** (площта) **на квадрат**.

Лицето на квадрат се пресмята като **умножим дължината на една от страните** на квадрата с дължината на **нейна съседна страна**.

Има два начина да намерим **лицето** (площта) **на квадрат**. Първият начин е като **умножим дължината на една негова страна по дължината на друга негова страна**, а вторият – като **повдигнем дължината на едната му страна на втора степен**.

Вход:

- Въвеждаме страната на квадрата.

Изход:

- Програмата принтира лицето на квадрата.

Следвайте стъпките:

1. Създайте променлива, която се казва **square_side** и съхранява дължината на страната на квадрата.
2. Създайте променлива, която се казва **square_area** и съхранява стойността на лицето на квадрата.
3. Принтирайте стойността на променливата **square_area**.

Ето как програмата трябва да работи:

Вход	Изход
2	4
Вход	**Изход**
5	25
Вход	**Изход**
8	64

Задача 7. Apples Count

Тим има **X ябълки**, Кейт има **2 пъти повече** ябълки от Тим, а Бен има **3 пъти повече** ябълки от Кейт.

Напишете програма, при която потребителят въвежда колко ябълки има Тим и пресмята колко ябълки имат Тим, Кейт и Бен.

Вход:

- Въвеждаме количеството ябълки на Тим.

Изход:

- Програмата принтира колко ябълки имат Тим, Кейт и Бен.

Следвайте стъпките:

1. Създайте променлива, която се казва **tim_apples_count** и съхранява броя ябълки, които Тим има.
2. Създайте променлива, която се казва **kate_apples_count** и съхранява броя ябълки, които Кейт има.
3. Създайте променлива, която се казва **ben_apples_count** и съхранява броя ябълки, които Бен има.
4. Принтирайте стойностите на трите променливи.

Ето как програмата трябва да работи:

Вход	Изход
2	2 4 12
Вход	**Изход**
3	3 6 18
Вход	**Изход**
5	5 10 30

Задача 8. FPS Calculator

Напишете програма, която пресмята колко **фрейма** се **визуализират** от екран за определено време, при въведени **минути** и **FPS**.

FPS е съкращение от **Frames per Second** (**фрейма за секунда**).

Вход:

- Въвеждаме минутите.
- Въвеждаме броя фреймове за секунда.

Изход:

- Програмата принтира колко фрейма са се визуализирали.

Следвайте стъпките:

1. Създайте променлива, която се казва **minutes** и съхранява минутите.
2. Създайте променлива, която се казва **fps** и съхранява броя фреймове за секунда.
3. Създайте променлива, която се казва **total_frames** и съхранява броя на всички фрейма за въведените минути.
4. Принтирайте стойността на променливата **total_frames**.

Ето как програмата трябва да работи:

Вход	Изход
30 20	36000
Вход	**Изход**
10 60	36000
Вход	**Изход**
15 120	108000

Задача 9. Shop Bill

Напишете програма, която пресмята **крайната цена** на **продукти** в магазин, спрямо закупеното **количество** и техните **цени**.

Осъществени покупки:

Продукт	Цена	Количество
Минерална вода	2 лв	2 бултилки
Пица	4 лв	3 броя
Сладолед	3 лв	2 броя

Изход:

- Програмата принтира крайната цена на продуктите.

Следвайте стъпките:

1. Създайте 3 променливи, които се казват съответно **water_bottle_price**, **pizza_price** и **ice_cream_pack_price** и съхраняват цените на съответните продукти.
2. Създайте 3 променливи, които се казват съответно **water_bottles_count**, **pizzas_count**, **ice_cream_packs_count** и съхраняват количествата на съответните продукти.
3. Създайте променлива, която се казва **total_price** и съхранява крайната цена на продуктите, спрямо техните цени и количества.
4. Принтирайте стойността на променливата **total_price**.

Ето как програмата трябва да работи:

Вход	Изход
	22

Задача 10. Speed Calculation

Напишете програма, която пресмята **скоростта**, с която се движи дадено превозно средство.

Превозното средство изминава **път** с **дължина 250 километра** за **2 часа**.

Формула за скорост: **v = S / t**

При горепосочената формула, с **v** отбелязваме **скоростта** на обекта, със **S** отбелязваме **дължината на пътя**, който обектът е изминал, а с **t** – **времето**, за което е изминат този път.

Изход:

- Програмата принтира скоростта на превозното средство.

Следвайте стъпките:

1. Създайте променлива, която се казва **S** и съхранява дължината на изминатия път в километри.
2. Създайте променлива, която се казва **t** и съхранява времето, за което даденият път е изминат.
3. Създайте променлива, която се казва **v** и съхранява скоростта, с която обектът се е движил.
4. Принтирайте стойността на променливата **v**.

Ето как програмата трябва да работи:

Вход	Изход
	125

Задача 11. Rectangle Properties

Напишете програма, която пресмята **периметъра** (обиколката) и **лицето** (площта) на **правоъгълник**.

Правоъгълникът е **фигура**, която има **4 страни**. Срещуположните страни на правоъгълника имат **еднакви дължини**.

Формула за периметър: **P = a + a + b + b = 2 * (a + b)**

Формула за лице: **S = a * b**

Вход:

- Въвеждаме двете страни на правоъгълника.

Изход:

- Програмата принтира периметъра и лицето на правоъгълника.

Следвайте стъпките:

1. Създайте променлива, която се казва **side_a** и съхранява дължината на едната страна на правоъгълника.
2. Създайте променлива, която се казва **side_b** и съхранява дължината на другата страна на правоъгълника.
3. Създайте променлива, която се казва **rectangle_perimeter** и съхранява стойността на периметъра на правоъгълника.
4. Създайте променлива, която се казва **rectangle_area** и съхранява стойността на лицето на правоъгълника.
5. Принтирайте стойностите на двете променливи.

Ето как програмата трябва да работи:

Вход	Изход
2 3	10 6
Вход	**Изход**
3 6	18 18

Задача 12. Triangle Perimeter

Напишете програма, която пресмята **периметъра** (обиколката) **на триъгълник**.

Триъгълникът е **фигура**, която има **3 страни**. Страните на триъгълника могат да бъдат с **различна дължина**.

Формула за периметър: **P = a + b + c**

Вход:

- Въвеждаме трите страни на триъгълника.

Изход:

- Програмата принтира периметъра на триъгълника.

Следвайте стъпките:

1. Създайте три променливи, които имат описателни имена и съхраняват трите страни на триъгълника.
2. Създайте променлива, която се казва **triangle_perimeter** и съхранява стойността на периметъра на триъгълника.
3. Принтирайте стойността на променливата **triangle_perimeter**.

Ето как програмата трябва да работи:

Вход	Изход
4 3 4	11
Вход	**Изход**
3 7 7	17
Вход	**Изход**
5 5 5	15

Задача 13. Triangle Area

Напишете програма, която пресмята **лицето** (площта) **на триъгълник**.

Площта на триъгълник се пресмята като **умножим дадена страна на триъгълника с височината**, **спусната към страната**, **и разделим произведението на 2**.

Формула за лице: **$S = a * h_a / 2$**

Вход:

- Въвеждаме едната страна на триъгълника.
- Въвеждаме височината, спусната към тази страна.

Изход:

- Програмата принтира лицето на триъгълника.

Следвайте стъпките:

1. Създайте променлива, която се казва **side_a** и съхранява дължината на **страната a**.
2. Създайте променлива, която се казва **height_a** и съхранява дължината на **височината**, **спусната към страната a**.
3. Създайте променлива, която се казва **triangle_area** и съхранява стойността на **лицето на триъгълника**.
4. Принтирайте стойността на променливата **triangle_area**.

Ето как програмата трябва да работи:

Вход	Изход
4 3	6
Вход	**Изход**
3 6	9
Вход	**Изход**
5 6	15

Задача 14. Trapezoid Perimeter

Напишете програма, която пресмята **периметъра** (обиколката) **на трапец**.

Трапецът е **фигура**, която има **4 страни**. **Страните a** и **b** на **трапеца** се наричат **основи**. Основите на трапеца са **успоредни**. Страните на трапеца могат да бъдат с различни дължини.

Формула за периметър: **P = a + b + c + d**

Вход:

- Въвеждаме четирите страни на трапеца.

Изход:

- Програмата принтира периметъра на трапеца.

Следвайте стъпките:

1. Създайте четири променливи, които имат описателни имена и съхраняват четирите страни на трапеца.
2. Създайте променлива, която се казва **trapezoid_perimeter** и съхранява стойността на периметъра на трапеца.
3. Принтирайте стойността на променливата **trapezoid_perimeter**.

Ето как програмата трябва да работи:

Вход	Изход
4 3 6 6	19
Вход	**Изход**
9 10 4 4	27

Задача 15. Trapezoid Area

Напишете програма, която пресмята **лицето** (площта) **на трапец**.

Лицето на трапец се пресмята като **умножим сбора на двете основи на трапеца по височината към едната основа** и резултатът **разделим на 2**.

Формула за лице: **S = (a + b) * h / 2**

Вход:

- Въвеждаме двете основи на трапеца.
- Въвеждаме височината на трапеца.

Изход:

- Програмата принтира лицето на трапеца.

Следвайте стъпките:

1. Създайте две променливи, които имат описателни имена и съхраняват дължините на двете основи и дължината на височината.
2. Създайте променлива, която се казва **trapezoid_area** и съхранява стойността на лицето на трапеца.
3. Принтирайте стойността на променливата **trapezoid_area**.

Ето как програмата трябва да работи:

Вход	Изход
4 3 4	14
Вход	**Изход**
9 10 4	38
Вход	**Изход**
5 6 2	11

Задача 16. Computer Memory

Напишете програма, която конвертира от **TB** (терабайт) към **MB** (мегабайт).

Един **TB** (терабайт) е равен на **1024 GB** (гигабайта), а един **GB** (гигабайт) е равен на **1024 MB** (мегабайта).

Вход:

- Въвеждаме стойността на терабайтите.

Изход:

- Програмата принтира стойността на мегабайтите.

Следвайте стъпките:

1. Създайте променлива, която се казва **terabytes** и съхранява стойността на терабайтите.
2. Създайте променлива, която се казва **megabytes** и съхранява стойността на мегабайтите.
3. Принтирайте стойността на променливата **megabytes**.

Ето как програмата трябва да работи:

Вход	Изход
2	2097152
Вход	**Изход**
3	3145728
Вход	**Изход**
5	5242880

Задача 17. GBP to BGN

Напишете програма, която обръща **британски лири** към **лева**.

Британска лира (паунд) е **паричната единица**, която се използва в **Обединеното Кралство** (Англия, Шотландия, Ирландия, Уелс).

Един паунд е приблизително равен на 2.29 лева.

Вход:

- Въвеждаме стойността на паундите.

Изход:

- Програмата принтира сумата в лева.

Ето как програмата трябва да работи:

Вход	Изход
2	4.58
Вход	**Изход**
14	32.06
Вход	**Изход**
16	36.64

Задача 18. Centimeters to Inches

Напишете програма, която **конвертира дължина от сантиментри към инчове**.

Инчовете са официална **мерна единица за дължина**, използвана в **Обединеното Кралство** и **Съединените Американски Щати**.

Един сантиметър е приблизително равен на 0.39 инча.

Вход:

- Въвеждаме стойността на сантиметрите.

Изход:

- Програмата принтира стойността в инчове.

Ето как програмата трябва да работи:

Вход	Изход
2	0.78
Вход	**Изход**
14	5.46
Вход	**Изход**
16	6.24

Задача 19. Celsius to Fahrenheit

Напишете програма, която **конвертира градуси по Целзий към градуси по Фаренхайт**.

Формула за конвертиране: **°F = °C * 9 / 5 + 32**

Вход:

- Въвеждаме градусите по Целзий.

Изход:

- Програмата принтира градусите по Фаренхайт.

Ето как програмата трябва да работи:

Вход	Изход
25	77
Вход	**Изход**
30	86
Вход	**Изход**
35	95

Задача 20. Average Number

Напишете програма, която пресмята **средноаритметичната стойност** на **3 числа**, въведени от потребителя.

Средноаритметичната стойност на **n числа** се намира като **разделим сбора на числата на числото n**.

Вход:

- Въвеждаме три числа.

Изход:

- Програмата принтира средноаритметичната им стойност.

Ето как програмата трябва да работи:

Вход	Изход
3 9 6	6
Вход	**Изход**
3 3 3	3
Вход	**Изход**
5 7 12	8

Условни конструкции

В тази глава ще се запознаем с **условните конструкции** в езика **Python**. Преди това, обаче, ще спомена как можем да сравняваме стойности в езика **Python**. Използвайки **условните конструкции**, нашите програми ще могат да изпълняват **различни команди**, в зависимост от дадено **условие**.

Оператори за сравнение

Оператор наричаме **знак** или **символ**, който има определено предназначение. Знаците, които използваме, за да укажем операциите **събиране** (**+**), **изваждане** (**-**), **умножение** (*****), **деление** (**/**), **степенуване** (******) и **намиране на остатък при деление** (**%**) се наричат **оператори**.

Python, както и всеки друг език за програмиране, разполага с т. нар. **оператори за сравнение**, познати от математиката:

- Оператор **<** (по-малко)
- Оператор **>** (по-голямо)
- Оператор **<=** (по-малко или равно)
- Оператор **>=** (по-голямо или равно)
- Оператор **==** (равно)
- Оператор **!=** (различно)

Резултатът от операцията **сравнение** е **булева стойност** – **вярно** (**True**) или **невярно** (**False**).

Оператори за сравнение

```
a = 5
b = 6

print(a > b) # False

print(a < b) # True

print(a == b) # False

print(a != b) # True

print(a == 5) # True
```

Булевите стойности също могат да бъдат записани в променливи.

Булеви променливи

```
a = 5
b = 6

areEqual = (a == b)
print(areEqual) # False

areDifferent = (a != b)
print(areDifferent) # True
```

Условни конструкции

Програмите, които досега писахме, са **линейни** – изпълняват се **команда след команда**. Условните конструкции ни позволяват да изпълним **различни команди**, спрямо **истинността** на дадено **условие**. По този начин **разклоняваме изпълнението на програмата**.

if конструкция в Python

```
grade = float(input())

if grade == 6:
    print('Excellent!')
```

Проверките в почти всички програмни езици се извършват с т. нар. **if конструкция**. Ако условието бъде оценено до **истина** (**True**), кодът, поместен в тялото на условната конструкция, ще се изпълни.

Обърнете внимание, че кодът след условието в условната конструкция е **изместен надясно** с една **табулация** (**[Tab]** - бутонът на клавиатурата, който се намира над бутона **[CapsLock]**). Освен това, **Python** изисква да поставим символа **:** (**двуеточие**) след условието.

Важно! Кодът, **написан на Python**, **трябва да бъде форматиран!**

Ако правилата за изписване (**форматиране**) на командите не се следват, ще получим **съобщение за грешка**. **Python** представлява език, при който форматирането и правилното подреждане / подравняване на кода е от съществено значение.

Примери за грешно форматиране в Python

```
grade = float(input())

if grade == 6:
print('Excellent!')

if grade == 6: print('Excellent!')
```

В общия случай, условната конструкция **if** има вида:

if конструкция в Python

```
if <условие>:
    <код, който ще се изпълни, ако условието се оцени до True>
```

По този начин, обаче, не сме обявили какво ще се случи, ако условието се оцени до **False**. В случай, че желаем даден код да се изпълни, ако условието се оцени до **False**, можем да използваме **else** клауза.

If-else конструкция в Python

```
grade = float(input())

if grade == 6:
    print('Excellent!')
else:
    print('Not excellent.')
```

Отново, **кодът трябва да бъде форматиран правилно**.

Пример за грешно форматиране в Python

```
grade =
float(input())

if grade == 6:
print('Excellent!')

else:
print('Not excellent.')
```

В общия случай, условната конструкция **if-else** има вида:

if-else конструкция в Python

```
if <условие>:
    <код, който ще се изпълни, ако условието се оцени до True>
else:
    <код, който ще се изпълни, ако условието се оцени до False>
```

Едно условие може да се оцени само до **True** или **False** и **никоя друга стойност**. Това означава, че **if-else** конструкцията винаги ще изпълни даден код – ако **условието е вярно** ще се изпълни кодът в **if блока**, ако **условието е невярно** ще се изпълни кодът в **else блока**.

Ако **if конструкцията** не разполага с **else** клауза и условието се оцени до **False**, тогава **няма** да се изпълни **нито една команда**, която е част от условната конструкция.

Множество проверки

Понякога се налага да извършим **множество проверки**.

В такъв случай, можем да добавим **допълнителни условия** с помощта на **elif** клауза. Ключовата дума **elif** представлява **съкратен запис** на **else if**, което бихме превели като **иначе ако**.

if-elif-else конструкция в Python

```
age = int(input())

if age < 6:
    print('Kindergarden')
elif age <= 18:
    print('School')
elif age <= 24:
    print('University')
else:
    print('Job')
```

Можем да поставим **неограничен брой условия**, използвайки ключовата дума **elif**. Не е задължително да поставим ключовата дума **else**.

Условни конструкции: Упражнения

Задача 1. Employee Payment

Напишете програма, която пресмята **дневната заплата** на служител за **8 часов работен ден**, спрямо **сумата**, която служителя получава за всеки **работен час**.

Вход:

- Въвеждаме сумата, която служителя получава за работен час.

Изход:

- Програмата принтира дневната заплата на служителя.

Ето как програмата трябва да работи:

Вход	Изход
2	16
Вход	**Изход**
14	112
Вход	**Изход**
16	128

Задача 2. Winery

Напишете програма, която пресмята колко **литра вино** ще произведе винарна, обработвайки дадена **площ** с размери **X м²** на **Y м²**.

Винарната произвежда **500 милилитра вино** за всеки **квадратен метър**.

Вход:

- Въвеждаме дължината **X** на площта.
- Въвеждаме широчината **Y** на площта.

Изход:

- Програмата принтира колко литра вино ще бъдат произведени.

Ето как програмата трябва да работи:

Вход	Изход
500 600	150000
Вход	**Изход**
200 200	20000
Вход	**Изход**
100 100	5000

Задача 3. Excellent Grade

Напишете програма, която проверява дали дадена **оценка** е **отлична**.

Вход:

- Въвеждаме оценка.

Изход:

- Програмата принтира **Excellent!**, ако оценката е равна на **6**.

Следвайте стъпките:

1. Създайте променлива, която се казва **grade** и съхранява оценката, въведена от потребителя.
2. Ако оценката е равна на **6** принтирайте текста **Excellent!**.

Ето как програмата трябва да работи:

Вход	Изход
6	`Excellent!`
Вход	**Изход**
5	
Вход	**Изход**
4	

Задача 4. Age Checker

Напишете програма, която проверява дали **стойността на годините** на даден човек е число, **по-малко от 12**.

Вход:

- Въвеждаме стойността на годините.

Изход:

- Програмата принтира **You are not old enough!**, ако стойността на годините е **по-малка от 12**.

Следвайте стъпките:

1. Създайте променлива, която се казва **person_age** и съхранява годините на даден човек.
2. Ако стойността на годините е **по-малка от 12** принтирайте текста **You are not old enough!**.

Ето как програмата трябва да работи:

Вход	Изход
6	You are not old enough!
Вход	**Изход**
5	You are not old enough!
Вход	**Изход**
14	

Задача 5. Positive Number

Напишете програма, която проверява дали дадено **число** е **положително**.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира **Positive**, ако числото е **по-голямо от 0**.

Следвайте стъпките:

1. Създайте променлива, която се казва **number** и съхранява стойността на числото.
2. Ако числото е **по-голямо от 0** принтирайте текста **Positive**.

Ето как програмата трябва да работи:

Вход	Изход
`6`	`Positive`
Вход	**Изход**
`0`	
Вход	**Изход**
`14`	`Positive`

Задача 6. Negative Number

Напишете програма, която проверява дали дадено **число** е **отрицателно**.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира **Negative**, ако числото е **по-малко от 0**.

Следвайте стъпките:

1. Създайте променлива, която се казва **number** и съхранява стойността на числото.
2. Ако числото е **по-малко от 0** принтирайте текста **Negative**.

Ето как програмата трябва да работи:

Вход	Изход
6	
Вход	**Изход**
-2	Negative
Вход	**Изход**
-6	Negative

Задача 7. Shop Bill

Напишете програма, която проверява дали дадена **сума пари** е **достатъчна**, за да **плати** дадена **сметка** в магазин.

Вход:

- Въвеждаме наличната сума пари.
- Въвеждаме стойността на крайната сметка.

Изход:

- Програмата принтира **Yes**, ако наличната сума е по-голяма или равна на стойността на крайната сметка, или **No** в противен случай.

Следвайте стъпките:

1. Създайте променлива, която се казва **money_amount** и съхранява стойността на наличната сума.
2. Създайте променлива, която се казва **bill_amount** и съхранява стойността на сметката в магазина.
3. Ако наличната сума е по-голяма или равна на стойността на сметката в магазина, принтирайте текста **Yes**. В противен случай, принтирайте текста **No**.

Ето как програмата трябва да работи:

Вход	Изход
50 40	Yes
Вход	**Изход**
100 30	Yes
Вход	**Изход**
50 120	No

Задача 8. Password Checker

Напишете програма, която проверява дали въведената от потребителя **парола** е правилната парола. **Правилната парола е qwerty123**.

Вход:

- Въвеждаме паролата.

Изход:

- Програмата принтира **Correct password!**, ако е въведена правилната парола, или **Incorrect password!** в противен случай.

Следвайте стъпките:

1. Създайте променлива, която се казва **input_password** и съхранява стойността на въведената от потребителя парола.
2. Ако въведената от потребителя парола съвпада с правилната парола принтирайте текста **Correct password!**. В противен случай, принтирайте текста **Incorrect Password!**.

Ето как програмата трябва да работи:

Вход	Изход
`123456`	`Incorrect password!`
Вход	**Изход**
`parola123`	`Incorrect password!`
Вход	**Изход**
`qwerty123`	`Correct password!`

Задача 9. Pool Capacity

Напишете програма, която **проверява** дали дадени **литри вода** са **достатъчни**, за да **запълнят басейн** с определена **вместимост**.

Вход:

- Въвеждаме литрите вода.
- Въвеждаме вместимостта на басейна в литри.

Изход:

- Програмата принтира **Yes**, ако водата е достатъчна да запълни басейна, или **No** в противен случай.

Следвайте стъпките:

1. Създайте променлива, която се казва **water_liters** и съхранява стойността на литрите вода.
2. Създайте променлива, която се казва **pool_capacity_liters** и съхранява вместимостта на басейна в литри.
3. Ако водата е достатъчна да запълни басейна принтирайте текста **Yes**. В противен случай принтирайте текста **No**.

Ето как програмата трябва да работи:

Вход	Изход
600 450	Yes
Вход	**Изход**
630 620	Yes
Вход	**Изход**
870 900	No

Задача 10. Max Number

Напишете програма, която намира **по-голямото число измежду две числа**, въведени от потребителя.

Вход:

- Въвеждаме първото число.
- Въвеждаме второто число.

Изход:

- Програмата принтира по-голямото число измежду двете числа.

Следвайте стъпките:

1. Създайте променлива, която се казва **first_number** и съхранява стойността на първото число.
2. Създайте променлива, която се казва **second_number** и съхранява стойността на второто число.
3. Създайте променлива, която се казва **max_number** и съхранява **числото 0**.
4. Ако първото число е по-голямо от второто, запишете стойността му в променливата **max_number**.
5. Ако второто число е по-голямо от първото, запишете стойността му в променливата **max_number**.

Ето как програмата трябва да работи:

Вход	Изход
6 8	8
Вход	**Изход**
3 6	6
Вход	**Изход**
9 23	23

Задача 11. Min Number

Напишете програма, която намира **по-малкото число измежду две числа**, въведени от потребителя.

Вход:

- Въвеждаме първото число.
- Въвеждаме второто число.

Изход:

- Програмата принтира по-малкото число измежду двете числа.

Следвайте стъпките:

1. Създайте променлива, която се казва **first_number** и съхранява стойността на първото число.
2. Създайте променлива, която се казва **second_number** и съхранява стойността на второто число.
3. Създайте променлива, която се казва **min_number** и съхранява **числото 0**.
4. Ако първото число е по-малко от второто, запишете стойността му в променливата **min_number**.
5. Ако второто число е по-малко от първото, запишете стойността му в променливата **min_number**.

Ето как програмата трябва да работи:

Вход	Изход
6 8	6
Вход	**Изход**
3 6	3
Вход	**Изход**
9 23	9

Задача 12. Number Checker

Напишете програма, която приема **число**, въведено от потребителя, и проверява дали **числото** е **малко** или **голямо**.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира **Small**, ако числото е по-малко или равно на **10**, или **Large**, ако числото е по-голямо или равно на **100000**.

Следвайте стъпките:

1. Създайте променлива, която се казва **number** и съхранява стойността на числото.
2. Ако стойността на числото е **по-малка или равна на 10** принтирайте текста **Small**.
3. Ако стойността на числото е **по-голяма от 100000** принтирайте текста **Large**.

Ето как програмата трябва да работи:

Вход	Изход
`1`	`Small`
Вход	**Изход**
`10000000`	`Large`
Вход	**Изход**
`5`	`Small`

Задача 13. Car Fuel

Напишете програма, която пресмята дали кола разполага с **необходимото гориво**, за да измине **определено разстояние**.

Вход:

- Въвеждаме литрите гориво.
- Въвеждаме разхода на гориво за 100 километра.
- Въвеждаме разстоянието в километри.

Изход:

- Програмата принтира **We have enough fuel!**, ако литрите гориво са достатъчни, или **Unfortunately, we do not have enough fuel!** в противен случай.

Следвайте стъпките:

1. Създайте променлива, която се казва **fuel_liters** и съхранява литрите бензин в резервоара на колата.
2. Създайте променлива, която се казва **fuel_liters_per_100_km** и съхранява литрите бензин, които са необходими, за да се изминат 100 километра.
3. Създайте променлива, която се казва **distance_km** и съхранява разстоянието в километри.
4. Програмата трябва да принтира **We have enough fuel!**, ако имаме достатъчно гориво, или **Unfortunately, we do not have enough fuel!**, ако горивото не ни достига.

Ето как програмата трябва да работи:

Вход	Изход
22 8 250	We have enough fuel!
Вход	**Изход**
34 6 600	Unfortunately, we do not have enough fuel!

Задача 14. Number Sign

Напишете програма, която принтира **знака на дадено число**.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира **+** (плюс), **-** (минус) или **0** (нула) в зависимост от стойността на числото.

Следвайте стъпките:

1. Създайте променлива, която се казва **number** и съхранява числото.
2. Ако числото е **по-голямо от 0**, принтирайте **плюс**.
3. Ако числото е **равно на 0**, принтирайте **нула**.
4. Ако числото е **по-малко от 0**, принтирайте **минус**.

Ето как програмата трябва да работи:

Вход	Изход
`5`	`+`
Вход	**Изход**
`0`	`0`
Вход	**Изход**
`-3`	`-`

Задача 15. Luggage Price

Напишете програма, която пресмята **цената** за **транспорт на багаж**, спрямо неговото **тегло**.

Разписание на цените:

Тегло	Цена
≤ 20 килограма	12
≤ 50 килограма	24
≤ 100 килограма	36

Вход:

- Въвеждаме теглото на багажа.

Изход:

- Програмата принтира цената за неговия транспорт.

Следвайте стъпките:

1. Създайте променлива, която се казва **luggage_kg** и съхранява теглото на багажа в килограми.
2. Пресметнете цената за транспорт, спрямо горепосочения ценоразпис.

Ето как програмата трябва да работи:

Вход	Изход
5	12
Вход	**Изход**
56	36
Вход	**Изход**
23	24

Задача 16. Grade Calculation

Напишете програма, която пресмята **крайната оценка** от **изпит**, в зависимост от **получените точки** на изпита.

Таблица за пресмятане на крайната оценка:

Точки	Оценка
≥ 80 точки	6
≥ 70 точки	5
≥ 60 точки	4
≥ 50 точки	3
< 50 точки	2

Вход:

- Въвеждаме получените точки от изпита.

Изход:

- Програмата принтира крайната оценка от изпита.

Следвайте стъпките:

1. Създайте променлива, която се казва **points** и съхранява броя на получените точки.
2. Пресметнете оценката, спрямо горепосочената таблица.

Ето как програмата трябва да работи:

Вход	Изход
`50`	`3`
Вход	**Изход**
`95`	`6`

Задача 17. Water Degrees

Напишете програма, която спрямо **градусите** на **водата** принтира информация за нейното **състояние**.

Таблица за състоянията на водата:

Градуси	Състояние
100 градуса	Boiling water
≥ 70 градуса	Very hot water
≥ 40 градуса	Hot water
≥ 20 градуса	Warm water
≥ 1 градуса	Cool water
0 градуса	Ice

Вход:

- Въвеждаме градусите на водата.

Изход:

- Програмата принтира информация за нейното състояние.

Ето как програмата трябва да работи:

Вход	Изход
`50`	`Hot water`
Вход	**Изход**
`95`	`Very hot water`
Вход	**Изход**
`0`	`Ice`

Задача 18. Days of Week

Напишете програма, която при въведено **число в интервала [1 ... 7]** принтира съответния **ден от седмицата**, **написан на български език**.

Ако въведеното число е извън горепосочения интервал принтирайте текста **Невалиден ден от седмицата!**.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира деня от седмицата, написан на български език.

Ето как програмата трябва да работи:

Вход	Изход
1	Понеделник
Вход	**Изход**
5	Петък
Вход	**Изход**
3	Сряда
Вход	**Изход**
55	Невалиден ден от седмицата!

Задача 19. Month Printer

Напишете програма, която при въведено **число в интервала [1 ... 12]** принтира съответния **месец от годината**, **написан на български език**.

Ако въведеното число е извън горепосочения интервал принтирайте текста **Невалиден месец!**.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира месеца на български език.

Ето как програмата трябва да работи:

Вход	Изход
1	Януари
Вход	**Изход**
6	Юни
Вход	**Изход**
19	Невалиден месец!
Вход	**Изход**
3	Март
Вход	**Изход**
4	Април

Задача 20. Digit to Text

Напишете програма, която при въведена **цифра в интервала [0 ... 9]** я изписва на **български език**.

Ако въведената цифра е извън горепосочения интервал принтирайте текста **Невалидна цифра!**.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира числото, написано на български език.

Ето как програмата трябва да работи:

Вход	Изход
3	Три
Вход	**Изход**
9	Девет
Вход	**Изход**
51	Невалидна цифра!
Вход	**Изход**
1	Едно
Вход	**Изход**
6	Шест

Сложни условни конструкции

В тази глава ще се сблъскаме с **по-сложни примери** за употреба на **условните конструкции**. Ще илюстрираме как работят **логическите оператори И** (**and**) и **ИЛИ** (**or**). Накрая, ще разгледаме какво представляват **вложените условни конструкции.**

Логически Оператори

В програмирането съществуват т. нар. **логически оператори**, които се прилагат в случай, че разполагаме с **множество условия**.

Най-използваните логически оператори в **Python** са:

- **Логическо И** – **and**
- **Логическо ИЛИ** – **or**

Логическият оператор И

Логическият оператор and може да бъде поставен **между две условия**. Резултатът от израза също е условие, което се оценява до **True** или **False**.

Проверка дали въведена възраст е валидна

```
age = int(input())

if age >= 1 and age <= 100:
    print('Valid age')
else:
    print('Invalid age')
```

Логическият оператор and използваме, когато желаем **и двете условия да се оценят до True**, за да се изпълни кода, поместен в конструкцията.

В горепосочения пример, желаем **едновременно** да са изпълнени и двете условия. Тогава и само тогава, стойността, която потребителят е въвел, може да се приеме за **валидна възраст**.

По-долу е представена таблица, която представя как би се оценило **по-сложното условие** (**X and Y**), спрямо **истинността** на **X** и **Y**. **X** и **Y** могат да бъдат заменени с **булеви изрази**, **булеви стойности** или **булеви променливи**. Резултатът от израза **X and Y** ще бъде булева стойност – **True** или **False**.

X	Y	X and Y
True	True	True
True	False	False
False	True	False
False	False	False

Логическият оператор and

```
print(True and False) # False

x = 5 > 6 # False
y = 4 < 6 # True

print(x and True) # False

print(x and False) # False

print(y and True) # True

print(y and False) # False

print(x and y) # False
```

Логическият оператор ИЛИ

Логическият оператор or може да бъде поставен **между две условия**. Резултатът от израза също е условие, което се оценява до **True** или **False**.

Логическият оператор or

```
age = int(input())

if age < 1 or age > 100:
    print('Invalid age')
else:
    print('Valid age')
```

Логическият оператор or използваме, когато желаем **поне едно от двете условия да се оцени до True**, за да се изпълни кода, поместен в конструкцията.

В горепосочения пример, желаем **поне едно** от двете условия да бъде изпълнено. Тогава и само тогава, стойността, която потребителят е въвел, може да се приеме за **невалидна възраст**.

По-долу е представена таблица, която представя как би се оценило **по-сложното условие** (**X or Y**), спрямо **истинността** на **X** и **Y**. **X** и **Y** могат да бъдат заменени с **булеви изрази**, **булеви стойности** или **булеви променливи**. Резултатът от израза **X or Y** ще бъде булева стойност – **True** или **False**.

X	Y	X or Y
True	True	True
True	False	True
False	True	True
False	False	False

Логическият оператор or

```
print(True or False) # True

x = 5 > 6 # False
y = 4 < 6 # True

print(x or True) # True

print(x or False) # False

print(y or True) # True

print(y or False) # True

print(x or y) # True
```

Вложени условни конструкции

Понякога се налага да използваме **вложени условни конструкции**.

Вложена условна конструкция, представлява условна конструкция поместена в тялото на друга условна конструкция. Независимо от това къде са поместени, условните конструкции работят по един и същ начин.

Вложени условни конструкции

```
city = input()
product = input()
price = 0

if city == 'Sofia':
    if product == 'coffee':
        price = 1.20
    elif product == 'milk':
        price = 2.10
    elif product == 'tea':
        price = 1.50
elif city == 'Varna':
    if product == 'coffee':
        price = 1.10
    elif product == 'milk':
        price = 2.20
    elif product == 'tea':
        price = 0.90
elif city == 'Plovdiv':
    if product == 'coffee':
        price = 0.90
    elif product == 'milk':
        price = 1.90
    elif product == 'tea':
        price = 1.80

print(price)
```

Подготовка за изпит: Упражнения

Задача 1. Circle Area

Напишете програма, която пресмята **лицето** (площта) **на кръг**, спрямо въведен **радиус**.

Формула за лице на кръг: **S = π * r * r**

Вход:

- Въвеждаме радиуса на кръга.

Изход:

- Програмата принтира лицето на кръга.

Ето как програмата трябва да работи:

Вход	Изход
2	12.56
Вход	**Изход**
3	28.26
Вход	**Изход**
5	78.5

Задача 2. Circle Circumference

Напишете програма, която пресмята **обиколката на кръг**, спрямо въведен **радиус**.

Формула за обиколка на кръг: **C = 2 * π * r**

Вход:

- Въвеждаме радиуса на кръга.

Изход:

- Програмата принтира обиколката на кръга.

Ето как програмата трябва да работи:

Вход	Изход
2	12.56
Вход	**Изход**
3	18.84
Вход	**Изход**
5	31.4

Задача 3. Triangle Degrees

Напишете програма, която намира стойността на **градусите на третия ъгъл** на **триъгълник**, при въведени другите **два ъгъла**.

За всеки триъгълник е валидно, че **сумата от градусите на трите ъгъла е равна на 180 градуса**.

Вход:

- Въвеждаме стойностите на градусите на двата ъгъла.

Изход:

- Програмата принтира стойността на третия ъгъл.

Ето как програмата трябва да работи:

Вход	Изход
60 60	60
Вход	**Изход**
50 60	70
Вход	**Изход**
30 30	120

Задача 4. Salary Increase

Напишете програма, която пресмята **новата заплата** на служител **след 10% увеличение** на старата заплата.

Един процент е равен на **1** / **100** (**една стотна**) от едно **число**. Процентите се използват за означаване на **част от цялото**.

Вход:

- Въвеждаме заплатата на служителя.

Изход:

- Програмата принтира стойността на заплатата след увеличението.

Ето как програмата трябва да работи:

Вход	Изход
1200	1320
Вход	**Изход**
1500	1650
Вход	**Изход**
2000	2200
Вход	**Изход**
2500	2750

Задача 5. Budgeting

Напишете програма, която разпределя даден **бюджет**, спрямо **процентно съотношение**.

Методът 50 / 30 / 20 е широкоизползван в бюджетирането. Методът гласи, че личният бюджет трябва да се разпределя както следва:

- **50%** от бюджета трябва да бъде заделен за нужди – храна, плащане на сметки, облекло и др.
- **30%** от бюджета трябва да бъде инвестиран.
- **20%** от бюджета трябва да бъде спестен.

Вход:

- Въвеждаме бюджета.

Изход:

- Програмата принтира разпределението на бюджета.

Ето как програмата трябва да работи:

Вход	Изход
1200	600 360 240
Вход	**Изход**
1000	500 300 200
Вход	**Изход**
5000	2500 1500 1000

Задача 6. Last Digit

Напишете програма, която намира **последната цифра** на дадено число.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира последната цифра на числото.

Ето как програмата трябва да работи:

Вход	Изход
12	2
Вход	**Изход**
25	5
Вход	**Изход**
87	7
Вход	**Изход**
19	9

Задача 7. Elevator Capacity

Напишете програма, която пресмята колко пъти трябва да бъде използван **асансьор** с даден **капацитет** за транспотирането на определен **брой хора**.

Капацитетът на асансьора позволява транспортирането на **до 6 лица**.

Вход:

- Въвеждаме броя хора, които желаят да използват асансьора.

Изход:

- Програмата принтира колко пъти трябва да бъде използван асансьора, спрямо неговия капацитет.

Ето как програмата трябва да работи:

Вход	Изход
62	11
Вход	**Изход**
8	2
Вход	**Изход**
13	3

Задача 8. Even or Odd

Напишете програма, която проверява дали въведено от потребителя число е **четно** (even) или **нечетно** (odd).

За всички **четни числа** е валидно, че **остатъкът при деление на 2** е **0**.

За всички **нечетни числа** е валидно, че **остатъкът при деление на 2** е **1**.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира дали числото е четно или нечетно.

Ето как програмата трябва да работи:

Вход	Изход
18	even
Вход	**Изход**
3	odd
Вход	**Изход**
12	even

Задача 9. Taxi Service

Напишете програма, която пресмята цената на транспорт, спрямо **тарифата** на такси, **частта от денонощието** и **престоят** в таксито.

Таблица с тарифите спрямо денонощието:

Част от денонощието	Цена за минута престой
day	0.80
night	0.90

Вход:

- Въвеждаме частта от денонощието.
- Въвеждаме минутите престой.

Изход:

- Програмата принтира цената на транспорта.

Ето как програмата трябва да работи:

Вход	Изход
`day` `20`	`16`
Вход	**Изход**
`night` `16`	`14.40`
Вход	**Изход**
`night` `30`	`27`

Задача 10. Three Numbers

Напишете програма, която принтира **максималното число** измежду **три числа**, въведени от потребителя.

Вход:

- Въвеждаме три числа.

Изход:

- Програмата принтира максималното число.

Ето как програмата трябва да работи:

Вход	Изход
1 3 5	5
Вход	**Изход**
3 9 65	65
Вход	**Изход**
123 654 352	654

Задача 11. Figures Types

Напишете програма, която спрямо въведен **брой стени** принтира **типа фигура**.

Типове фигури, **спрямо броя стени:**

Брой стени	Фигура
3	Triangle
4	Rectangle
5	Pentagon
6	Hexagon
7	Heptagon
8	Octagon

Вход:

- Въвеждаме броя стени на фигурата.

Изход:

- Програмата принтира типа на фигурата.

Ето как програма трябва да работи:

Вход	Изход
5	Pentagon
Вход	**Изход**
8	Octagon
Вход	**Изход**
3	Triangle

Задача 12. Abbreviations

Напишете програма, която спрямо въведена **абревиатура** принтира нейното **значение**.

Таблица с познати абревиатури:

Абревиатура	Значение
lol	Laughing out loud
rofl	Rolling on the floor laughing
lmk	Let me know
tbh	To be honest
brb	Be right back
afk	Away from keyboard
btw	By the way

Вход:

- Въвеждаме абревиатурата.

Изход:

- Програмата принтира значението на абревиатурата.

Ето как програмата трябва да работи:

Вход	Изход
`afk`	`Away from keyboard`
Вход	**Изход**
`lol`	`Laughing out loud`
Вход	**Изход**
`tbh`	`To be honest`

Задача 13. Chinese Zodiac

Напишете програма, която спрямо въведена **година** принтира **тип животно** спрямо **китайския зодиак**.

Таблица с китайските зодии:

Година	Животно
2000	Dragon
2001	Snake
2002	Horse
2003	Goat
2004	Monkey
2005	Rooster

Вход:

- Въвеждаме година.

Изход:

- Програмата принтира китайската зодия, спрямо въведената година.

Ето как програмата трябва да работи:

Вход	Изход
`2000`	`Dragon`
Вход	**Изход**
`2005`	`Rooster`
Вход	**Изход**
`2002`	`Horse`

Задача 14. Valid Age

Напишете програма, която проверява дали въведена от потребителя стойност на **години** са **валидна възраст**. Приемаме, че **валидните години** са в **интервала [1 ... 110]**.

Вход:

- Въвеждаме стойността на годините.

Изход:

- Програмата принтира **Valid**, ако стойността на годините е валидна, или **Invalid** в противен случай.

Следвайте стъпките:

1. Създайте променлива, която се казва **user_age** и съхранява стойността на годините.
2. Ако стойността на годините е по-голяма или **равна на 1 и по-малка или равна на 110**, принтирайте текста **Valid**. В противен случай, принтирайте текста **Invalid**.

Ето как програмата трябва да работи:

Вход	Изход
`5`	`Valid`
Вход	**Изход**
`0`	`Invalid`
Вход	**Изход**
`123`	`Invalid`

Задача 15. Number in Interval

Напишете програма, която проверява дали **дадено число** е **валидно**. Валидните числа са числата, които се намират в **интервала [100 ... 200]**.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира **Valid**, ако числото е в интервала, или **Invalid** в противен случай.

Следвайте стъпките:

1. Създайте променлива, която се казва **number** и съхранява числото.
2. Ако числото е в горепосочения **интервал**, принтирайте текста **Valid**. В противен случай, принтирайте текста **Invalid**.

Ето как програмата трябва да работи:

Вход	Изход
`168`	`Valid`
Вход	**Изход**
`0`	`Invalid`
Вход	**Изход**
`123`	`Valid`

Задача 16. House Building

Напишете програма, която пресмята за колко дни приблизително ще бъде построена една къща, спрямо броя работници.

Брой дни за построяване, **спрямо броя работници**:

Брой работници	Брой дни
10 **≤** брой работници **<** 15	300
15 **≤** брой работници **<** 20	240
20 **≤** брой работници **<** 25	180

Вход:

- Въвеждаме броя работници.

Изход:

- Програмата принтира за колко дни къщата ще бъде построена.

Ето как програмата трябва да работи:

Вход	Изход
`15`	`240`
Вход	**Изход**
`12`	`300`
Вход	**Изход**
`21`	`180`

Задача 17. Special Number

Напишете програма, която проверява дали дадено **число** е **специално число**. Едно число наричаме специално, ако то **се дели едновременно на 3**, **5 и 7 без остатък**.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира дали числото е специално.

Ето как програмата трябва да работи:

Вход	Изход
`105`	`Yes`
Вход	**Изход**
`3`	`No`
Вход	**Изход**
`315`	`Yes`

Задача 18. Triangle Sides

Напишете програма, която приема **три числа** и проверява дали числата изпълняват необходимите **условия**, за да се нарекат **страни на триъгълник**.

Ако имаме страните **a**, **b** и **c**, то необходимите условия са:

- **a + b > c**
- **a + c > b**
- **b + c > a**

Вход:

- Въвеждаме три числа.

Изход:

- Програмата принтира дали числата, разглеждани като дължини на страни, могат да сформират триъгълник.

Ето как програмата трябва да работи:

Вход	Изход
4 3 5	Yes
Вход	**Изход**
9 9 9	Yes
Вход	**Изход**
1 100 1000	No

Задача 19. Letter Checker

Напишете програма, която при въведена **буква** от английската азбука проверява дали буквата е **гласна** или **съгласна**.

Вход:

- Въвеждаме една буква от английската азбука.

Изход:

- Програмата принтира дали буквата е гласна или съгласна.

Ето как програмата трябва да работи:

Вход	Изход
a	vowel
Вход	**Изход**
b	consonant
Вход	**Изход**
e	vowel

Задача 20. Triangle Type

Напишете програма, която при въведени **три страни** на един **триъгълник** проверява дали триъгълникът е **равностранен**, **разностранен** или **равнобедрен**.

Триъгълникът е **равностранен**, когато е в сила равенството:
a = b = c

Триъгълникът е **разностранен**, когато е в сила неравенството:
a ≠ b ≠ c

Триъгълникът е **равнобедрен**, когато е в сила едно от условията:
a = b и a ≠ c и b ≠ c
b = c и a ≠ b и a ≠ c
a = c и a ≠ b и b ≠ c

Вход:

- Въвеждаме трите страни на един триъгълник.

Изход:

- Програмата принтира дали триъгълникът е равностранен, разностранен или равнобедрен.

Ето как програмата трябва да работи:

Вход	Изход
4 3 5	Разностранен
Вход	**Изход**
9 9 9	Равностранен
Вход	**Изход**
4 5 5	Равнобедрен

Задачи за шампиони

Задача 1. Pythagorean Triplet

Напишете програма, която при въведени **две Питагорови числа**, намира **третото Питагорово число**.

За всеки три Питагорови числа е валидно равенството: $a^2 + b^2 = c^2$

Вход:

- Въвеждаме стойността на **a**.
- Въвеждаме стойността на **b**.

Изход:

- Програмата принтира стойността на **c**.

Ето как програмата трябва да работи:

Вход	Изход
3 4	5
Вход	**Изход**
6 8	10
Вход	**Изход**
10 24	26
Вход	**Изход**
15 20	25

Задача 2. Century Calculation

Напишете програма, която при въведена **валидна година** принтира **века**.

Вход:

- Въвеждаме **валидна година**.

Изход:

- Програмата принтира **века**.

Ето как програмата трябва да работи:

Вход	Изход
1555	16
Вход	**Изход**
2001	21
Вход	**Изход**
1999	20
Вход	**Изход**
1899	19
Вход	**Изход**
2025	21

Задача 3. Cash Withdraw

Напишете програма, която проверява дали човек може да **изтегли дадена сума от банкомат**, спрямо **наличната сума** в неговата сметка, включвайки заплащането при банковата транзакция.

Всяка банкова транзакция струва $0.5.

При теглене от банкомат, **сумата трябва да бъде кратна на 5**.

Вход:

- Въвеждаме стойността на сумата, която човекът желае да изтегли.
- Въвеждаме стойността на наличната сума в банковата сметка.

Изход:

- Програмата принтира стойността на банковата сметка, след опита за теглене на пари.

Ето как програмата трябва да работи:

Вход	Изход
30 120	$89.50
Вход	**Изход**
42 120	$77.50
Вход	**Изход**
300 100	$100

Задача 4. Color Invert

Напишете програма, която **преобразува цвят**, подаден в **RGB формат**.

RGB е **абревиатура** на **Red**, **Green**, **Blue**. **Червен**, **зелен** и **син** са **основните цветове**, чрез които може да се образува всеки друг цвят.

Форматът RGB представлява три числа, всяко от които е със стойност в интервала **[0 ... 255]**, които показват **наситеността на всеки от трите цвята**.

Примери:

Цвят	RGB формат
Бяло	RGB(255, 255, 255)
Черно	RGB(0, 0, 0)
Червено	RGB(255, 0, 0)
Зелено	RGB(0, 255, 0)
Синьо	RGB(0, 0, 255)

Вход:

- Въвеждаме стойността на **червения цвят**.
- Въвеждаме стойността на **зеления цвят**.
- Въвеждаме стойността на **синия цвят**.

Изход:

- Програмата принтира стойността на преобразувания цвят в **RGB формат**.

Ето как програмата трябва да работи:

Вход	Изход

`165`
`170`
`221`
	`RGB(90, 85, 34)`

Задача 5. Ships Types

Напишете програма, която спрямо въведен **клас на кораб** принтира неговия **тип**.

Типове кораби, **спрямо техния клас:**

Клас	Тип
B или **b**	Battle Ship
C или **c**	Cruiser
D или **d**	Destroyer
F или **f**	Frigate

Вход:

- Въвеждаме класа на кораба.

Изход:

- Програмата принтира типа на кораба.

Ето как програма трябва да работи:

Вход	Изход
`b`	`Battle Ship`
Вход	**Изход**
`C`	`Cruiser`
Вход	**Изход**
`B`	`Battle Ship`
Вход	**Изход**
`d`	`Destroyer`

Задача 6. Animals Types

Напишете програма, която спрямо въведен **вид животно** принтира неговия **клас**.

Таблица с животни и техните класове:

Животно	Клас
lion	mammal
snake	reptile
crocodile	reptile
eagle	birds
dolphin	mammal
owl	birds

Вход:

- Въвеждаме вида животно.

Изход:

- Програмата принтира неговия клас.

Ето как програмата трябва да работи:

Вход	Изход
`lion`	`mammal`
Вход	**Изход**
`eagle`	`birds`
Вход	**Изход**
`snake`	`reptile`

Задача 7. Fruit or Vegetable

Напишете програма, която спрямо въведен **плод или зеленчук** принтира неговия **вид**.

Таблица с плодове и зеленчуци:

Храна	Вид
banana	fruit
tomato	fruit
pepper	vegetable
onion	vegetable
orange	fruit
potato	vegetable

Вход:

- Въвеждаме име на плод или име на зеленчук.

Изход:

- Програмата принтира неговия вид.

Ето как програмата трябва да работи:

Вход	Изход
`potato`	`vegetable`
Вход	**Изход**
`orange`	`fruit`
Вход	**Изход**
`banana`	`fruit`

Задача 8. Compare Numbers

Напишете програма, която проверява дали **две числа са равни**, спрямо **дадена стойност**.

Пример:

- Приемаме че, числата **1.000001** и **1.00001** са **равни**, ако **разликата между тях е по-малка** от **числото 0.0001**.

Вход:

- Въвеждаме стойността, спрямо която осъществяваме **сравнение**.
- Въвеждаме стойностите на двете числа.

Изход:

- Програмата принтира дали двете числа са равни.

Ето как програмата трябва да работи:

Вход	Изход

`0.0001`
`1.000001`
`1.00001`
	`true`

Вход	**Изход**

`0.000001`
`5.4354566666`
`5.435452`
	`false`

Вход	**Изход**

`0.001`
`6.442`
`6.441`
	`false`

Задача 9. Basic Calculator

Напишете програма, представлява **калкулатор**.

Вход:

- Въвеждаме **първия аргумент**.
- Въвеждаме **оператор**.
- Въвеждаме **втория аргумент**.

Изход:

- Програмата принтира **крайната стойност на израза**.

Ето как програмата трябва да работи:

Вход	Изход
12 + 12	24
Вход	**Изход**
1 * 1	1
Вход	**Изход**
5 - 2	3
Вход	**Изход**
10 / 5	2

Задача 10. Blackjack

Напишете програма, която проверява кой е **победителят измежду двама играчи** при игра на **блекджек**.

Целта на играта **блекджек** е, използвайки две карти, играчът да събере **сума**, **която максимално се приближава до числото 21**, **без да го надвишава**.

Картата асо се брой за **1 или 11**, а **картите вале** (**J**), **дама** (**Q**) и **поп** (**K**) носят **по 10 точки**. Всички останали карти носят толкова точки, колкото числото, което е нарисувано на тях.

Вход:

- Въвеждаме двете карти на първия играч.
- Въвеждаме двете карти на втория играч.

Изход:

- Програмата принтира побителят в играта.

Ето как програмата трябва да работи:

Вход	Изход
6 J K 2	First player
Вход	**Изход**
2 A 10 A	Second player
Вход	**Изход**
J K Q 2	First player

Задача 11. Rock, Paper, Scissors

Напишете програма, която проверява кой е **победителят измежду двама играчи** при игра на **Камък**, **ножица**, **хартия**.

Вход:

- Въвеждаме хода на първия играч.
- Въвеждаме хода на втория играч.

Изход:

- Програмата принтира побителят в играта.

Ето как програма трябва да работи:

Вход	Изход
rock paper	Second player
Вход	**Изход**
scissors paper	First player
Вход	**Изход**
rock scissors	First player
Вход	**Изход**
rock rock	Play again!

Задача 12. Days Count

Напишете програма, която преобразува **дни** в период, съдържащ **години**, **седмици** и **дни**.

Вход:

- Въвеждаме стойността на дните.

Изход:

- Програмата принтира стойността на годините, седмиците и дните.

Ето как програмата трябва да работи:

Вход	Изход
350	0 years, 50 weeks, 0 days
Вход	**Изход**
373	1 year, 1 week, 1 day
Вход	**Изход**
820	2 years, 12 weeks, 6 days

Задача 13. Time Conversion

Напишете програма, която пресмята **колко минути** и **колко секунди** се съдържат в **даден брой секунди**.

Вход:

- Въвеждаме стойността на секундите.

Изход:

- Програмата принтира стойността на минутите и стойността на секундите, **форматирани с водеща нула**.

Ето как програмата трябва да работи:

Вход	Изход
350	05:50
Вход	**Изход**
780	13:00
Вход	**Изход**
820	13:40

Задача 14. Time Calculation

Напишете програма, която при въведен **валиден час добавя 15 минути** към него.

Вход:

- Въвеждаме **валиден час под формата на текст**.

Изход:

- Програмата принтира **часа след 15 минути**.

Ето как програмата трябва да работи:

Вход	Изход
12:00	12:15
Вход	**Изход**
13:45	14:00
Вход	**Изход**
21:10	21:25
Вход	**Изход**
00:00	00:15
Вход	**Изход**
23:50	00:05

Задача 15. Playback Duration

Напишете програма, която пресмята **времетраенето на видеоклип**, в зависимост от неговата **продължителност в часове**, **минути и секунди** и **скоростта на възпроизвеждане**.

Вход:

- Въвеждаме стойността на часовете.
- Въвеждаме стойността на минутите.
- Въвеждаме стойността на секундите.
- Въвеждаме **скоростта на възпроизвеждане на видеоклипа**.

Изход:

- Програмата принтира **времетраенето на видеоклипа**.

Ето как програмата трябва да работи:

Вход	Изход
0 30 0 2	00:15:00

Вход	Изход
1 20 0 1.5	00:53:20

Вход	Изход
51 20 09 0.5	102:40:18

Задача 16. Armstrong Number

Напишете програма, която проверява **дали дадено трицифрено число е число на Армстронг**.

За всяко число на Армстронг е валидно, че числото е равно на сумата от цифрите на числото, всяка от тях повдигната на 3-та степен.

Пример:

- **371 е число на Армстронг**, тъй като: $3^3 + 7^3 + 1^3 = 371$.
- **153 е число на Армстронг**, тъй като: $1^3 + 5^3 + 3^3 = 153$.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира дали числото е **число на Армстронг**.

Ето как програмата трябва да работи:

Вход	Изход
371	true
Вход	**Изход**
123	false
Вход	**Изход**
255	false
Вход	**Изход**
153	true

Задача 17. Quadratic Equation

Напишете програма, която намира корените на квадратно уравнение **$ax^2 + bx + c = 0$**, при въведени коефициенти **a**, **b** и **c**.

Вход:

- Въвеждаме трите коефициенти – **a**, **b** и **c**.

Изход:

- Програмата извежда информация относно **корените на квадратното уравнение**.

Ето как програмата трябва да работи:

Вход	Изход
-1 -1 2	x1 = 1 x2 = -2
Вход	**Изход**
-2 1 3	x1 = 6 x2 = -4
Вход	**Изход**
1 1 1	No real roots

Задача 18. Logarithm

Напишете програма, която намира **логаритъм на дадено число** при подадена **основа**.

Логаритъмът представлява операция, която прилича на **обърнато степенуване**. **Резултатът от логаритъмът представя степен**.

Примери:

- **$\log_2 32 = 5 \rightarrow 2^5 = 32$**. **Числото 2** наричаме **основа**, а **числото 32** наричаме **число**, **подадено на логаритъма**.
- **$\log_4 64 = 4 \rightarrow 4^4 = 64$**. **Числото 4** наричаме **основа**, а **числото 64** наричаме **число**, **подадено на логаритъма**.

Вход:

- Въвеждаме основата на логаритъма.
- Въвеждаме числото, подадено на логаритъма.

Изход:

- Програмата принтира степента, на която трябва да повдигнем основата на логаритъма, за да получим числото, подадено на логаритъма.

Ето как програмата трябва да работи:

Вход	Изход
2 32	5

Вход	Изход
10 1000	3

Вход	Изход
5 25	2

Задача 19. Holiday Expenses

Напишете програма, която пресмята **крайната цена за почивка** на **дадено семейство**, съставено от **двама възрастни и три деца**. Ценообразуването е спрямо дестинацията и продължителността на почивката.

Таблица с възможните дестинации:

Дестинация	Цени
Море Хотелска стая	Възрастен – 10 лв. / ден Дете – 7.50 лв. / ден
Море Къща за гости	Възрастен – 12 лв. / ден Дете – 8 лв. / ден
Планина Хотелска стая	Възрастен – 8 лв. / ден Дете – 6.50 лв. ден
Планина Къща за гости	Възрастен – 9 лв. / ден Дете – 7 лв. / ден

Вход:

- Въвеждаме дестинацията.
- Въвеждаме мястото за почивка.
- Въвеждаме дните престой.

Изход:

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

Ето как програмата трябва да работи:

Вход	Изход

`Планина`
`Хотелска стая`
`7`
	`248.50 лв.`

Задача 20. Number to Text

Напишете програма, която при въведено **число в интервала [0 ... 100]** го изписва на **английски език**.

Ако въведеното число е извън горепосочения интервал принтирайте текста **Invalid number!**.

Вход:

- Въвеждаме произволно число.

Изход:

- Програмата принтира числото, написано на **английски език**.

Ето как програмата трябва да работи:

Вход	Изход
3	three
Вход	**Изход**
99	ninety nine
Вход	**Изход**
51	fifty one
Вход	**Изход**
21	twenty one
Вход	**Изход**
-6	Invalid number!

Заключение

Прочитайки цялата книга и решавайки правилно всички задачи, вече сте направили свойте **първи стъпки в програмирането**!

Поздравления!

В случай, че програмирането Ви е допаднало и намирате книгата и решаването на задачи по програмиране за интересни, **продължавайте да се обучавате**! Колкото по-отрано започнете, толкова по-добре.

Следващите стъпки в света на програмирането продължават с курса **Python Fundamentals**, където се изучават доста по-сложни концепции като **цикли**, **масиви**, **списъци**, **функции** и други.

Курсът **Python Fundamentals** е една предпоставка за изучаване и разбиране в дълбочина на други технологии, например **HTML** и **CSS**, чрез които се изграждат уеб сайтове, уеб приложения и игри.

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