

Osnova

Lesson Overview

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [LESSON OVERVIEW](#)

Welcome in lesson 2, good to see you again.

You have finished lesson 1, so **you should know**, how to:

- install Python and run script,
- work and operate with basic data types.

Let's jump into lesson 2. The **goal of this lesson** is to learn, how to:

- empower your program with deciding abilities by **using conditions**. Just imagine you are in pub with friends. Barman is about to decide whether you are more than 18 years old, because you want to buy a pint of beer. We will teach Python, how to decide these situations. Ready? Go :)

Osnova

00:54

REVIEW EXERCISES

Concatenation

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [REVIEW EXERCISES](#) / [CONCATENATION](#)

Create a python script named `official_name.py` that will perform the following actions:

- ask the user for the first name

100% z Lekce 2

so we can concatenate the whole name in **format**:

Osnova

'LastName, FirstName'

Example of running the script in terminal:

```
~/PythonBeginner/Lesson1$ python official_name.py
What is your first name? Bob
What is your last name? Sponge
Your name is: Sponge, Bob
```

Online Python Editor

You can create the script in you computer or here in our editor.

```
1 # Input
2
3 # Concatenation
4
5 # Printing
```

spustit kód

Osnova

Code solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution



Length

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [REVIEW EXERCISES](#) / [LENGTH](#)

Create a python script named `word_length.py` that will print out the length of the word 'quetzalcoatl' in a sentence similar to:

```
quetzalcoatl is X characters long
```

The placeholder X in the above string should be replaced by the actual length of the word.

Example of running the script in terminal:

```
~/PythonBeginner/Lesson1$ python word_length.py  
quetzalcoatl is 12 characters long
```

Online Python Editor

You can create the script in you computer or here in our editor.

```
1 # Getting lenght  
2  
3 # Printing
```

Osnova

[spustit kód](#)

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

[Click to see our solution](#)

Indexing

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [REVIEW EXERCISES](#) / [INDEXING](#)

Create a file **cities.py** . Inside the file do following:

1. Create a list containing following cities: New York, Los Angeles, Berlin, Prague, London
2. Print the list.

100% z Lekce 2

4. Print the city at the last index introduced by the string: **Last city, located at**
Osnova
...e. _um>, we have:

Example of running the script in terminal:

```
~/PythonBeginner/Lesson2$ python cities.py
Cities in my list: ['New York', 'Los Angeles', 'Berlin', 'Prague',
'London']
At index 2 we have: Berlin
Last city, located at index 4, we have: London
```

Online Python Editor

You can create the script in you computer or here in our editor.

```
1 # List
2
3 # Printing the list
4
5 # Printing the city at index 2
6
7 # Getting last index
8
9 # Printing the city with last index|
```

spustit kód

Osnova

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution



Slicing

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [REVIEW EXERCISES](#) / [SLICING](#)

We have 3 email addresses and we would like to **extract only the part preceding** the @ symbol. Print each extracted string to the terminal.

Here are the emails:

- mr.reilly@gmail.com
- john55@yahoo.com
- elgringo@atlas.sk

```
~/PythonBeginner/Lesson2 $ python extract.py
mr.reilly
john55
elgringo
```

Online Python Editor

You can create the script in you computer or here in our editor.

1 |

Osnova

[spustit kód](#)

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

[Click to see our solution](#)

ONSITE PROJECT

Solution from Lesson 1

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [ONSITE PROJECT](#) / [SOLUTION FROM LESSON 1](#)

We were asked to create a program that will allow user to reserve a trip to a given destination. We offer 6 destinations for the following prices:

Destination	Price
1 - Prague	1000
2 - Wien	1100
3 - Brno	2000
4 - Svitavy	1500
5 - Zlin	2300
6 - Ostrava	3400

Also, for destinations **Svitavy** and **Ostrava** we offer a special **discount of 25%**.

Once the user selects the destination, we want our program to calculate the price and



However, there are few constraints on the format, we want the collected data to be in:

- We cannot provide our services to clients under 15 years of age
- Emails have to contain @ symbol
- Password has to be at least 8 chars long, cannot begin and end with a number and has to contain both letters and numbers

If at least one of the above requirements is not met, the whole reservation process should be cancelled with appropriate message.

On the other hand, if everything went well the program should summarize the reservation details mentioning the user's name, destination and price.

The code so far

```
1  # Greet the client
2  print('=' * 80)
3  print('''Welcome to the DESTINATION,
4  place where people plan their trips''')
5  print('=' * 80)
6
7  # Offer destinations
8  print('We can offer you the following destinations:')
9  print('-' * 80)
10 print('')
11 1 - Prague   | 1000
12 2 - Wien    | 1100
13 3 - Brno    | 2000
14 4 - Svitavy | 1500
15 5 - Zlin    | 2300
16 6 - Ostrava | 3400
17 ''')
18 print('-' * 80)
19
```

```
1  Osnova = ['Prague', 'Wien', 'Brno', 'Svitavy', 'Zlin', 'Ostrava']
22 PRICES = [1000, 1100, 2000, 1500, 2300, 3400]
23 DISCOUNT_25 = ['Svitavy', 'Ostrava']
24
25
26 destination = DESTINATIONS[selection-1]
27 price = PRICES[selection-1]
28
29 print('=' * 80)
30 print('REGISTRATION:')
31 print('-' * 80)
32 print('In order to complete your reservations, please share few
    details about yourself with us.')
33 print('-' * 80)
34
35 name = input('NAME: ')
36 print('=' * 40)
37 surname = input('SURNAME: ')
38 print('=' * 40)
39 birth_year = input('YEAR of BIRTH: ')
40 print('=' * 40)
41 email = input('EMAIL: ')
42 print('=' * 40)
43 password = input('PASSWORD: ')
44 print('=' * 80)
45
46 print('Thank you ' + name)
47 print('We have made your reservation to ' + destination)
48 print('The total price is ' + str(price))
```

Checking the Conditions

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [ONSITE PROJECT](#) / [CHECKING THE CONDITIONS](#)

100% z Lekce 2

After we finish the final stage of creating our small app. At this phase, we need to begin to check if the conditions provided meet certain conditions.

For example, we would like to check, whether the person's age is below 15 years. If so, we would like to print the following string:

```
'Sorry, we cannot offer our services to babies'
```

To check, whether some condition or test is valid, we use **conditional statements**. These statements are composed of two parts:

1. **HEAD**

2. **BODY**

The general syntax is as follows:

```
1 HEAD:  
2     BODY  
3     BODY  
4     ...
```

Body

Even though the body, comes after the head, we will explain it first. Body is just a set of commands (statements) we would like to execute, if the condition in the head part is **True**.

The statements inside the body have to be **indented** to the right. The recommended indentation is **4 spaces**.

So actually, body of our condition that checks the user's age, would look like this:

```
1 print('Sorry, we cannot offer our services to babies')
```

Head

Head contains the actual test checking the condition. Each conditional statement has to begin with **keyword** **if** followed by the testing expression finished by the colon **:** at the end of the

```
1  # Osnova
2  if ( < 18 ):
    print('Sorry, we cannot offer our services to babies')
```

Code Task

However, we do not have the variable `age`, but we have the variable `birth_year`. Create a condition that will check, whether a person is under the age of 15 only by using the variable `birth_year`.

```
1 |
```

spustit kód

Osnova

The first step towards learning the conditions is behind us. Now we need to find out, how we can check multiple excluding conditions. That means if something is not **True** , then do something **else** .

Let's work with the case, when we want to check, whether the destination name is **'Svitavy'** . If it is so, we want to discount from the original price of 1500 CZK 25% and store the result in the variable **price** . Otherwise, we want to apply a discount of 50% to the original price and store it back in the **price** variable.

Actually, we have here **2 mutually excluding possibilities**:

1. **if** the destination is **Svitavy**
2. **else** it is something different

In this case we will use the following construction:

```
1 if destination == 'Svitavy':  
2     price = price * 0.75  
3 else:  
4     price = price * 0.50
```

The second part of the conditional statement (the **else** part) **does not check for any condition** as this is not needed - we just wanted to know, whether the destination is Svitavy.

Multiple Conditions

In case we would have to perform more than one excluding check, we would need to combine **else** and **if** .

Let's say, additionally to the above condition of Svitavy receiving 50% discount, that if the **destination** is Brno, we want to offer discount only 10%.

We could do it as follows:

```
1 if destination == 'Svitavy':
```

```
4     if destination == 'Brno':  
5         price = price * 0.90  
6     else:  
7         price = price * 0.50
```

What we have just seen is called **nesting** of conditional statements. It is ok to nest one conditional statement into another one as we have seen here. But what would happen, we had to check more than 2 conditions? Let's say 10 conditions. That would be 9 nested conditions. Nobody would be able to **read** such a code.

Therefore, Python offers us to use keyword **elif** (shorthand of else and if). We use it **on the same level as if and else** and so we avoid ugly nesting.

We can rewrite the above condition as follows:

```
1  if destination == 'Svitavy':  
2      price = price * 0.75  
3  elif destination == 'Brno':  
4      price = price * 0.90  
5  else:  
6      price = price * 0.50
```

```
1  destination = 'Svitavy'  
2  price = 1500
```

Osnova

[spustit kód](#)

Number of Items in a Sequence

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [ONSITE PROJECT](#) / [NUMBER OF ITEMS IN A SEQUENCE](#)

Let's sort out some details about our program.

What if the user enters a number higher than the number of destinations we offer? We will have to set up a condition that will check how many items are there in a list. So how can we find out, how many items are there?

We can use the command `len()`. Let's try it out.

1. What are the lengths of the below lists and strings?
2. Maybe we can create a code that will asks us for a and input and will print string `'Too long'` if the input is longer than 6 characters or will print string `'This is ok'`, if the string is between 4 to 6 characters. Otherwise it should print `'Too Short'`

```
1 text1 = 'Hello'
2 text2 = text1 * 5
3 list1 = list(text1)
4 list2 = [text1] + list(text2)
```


Osnova

[spustit kód](#)

Is This in a Sequence?

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [ONSITE PROJECT](#) / [IS THIS IN A SEQUENCE?](#)

Another thing, we will need to know, is whether a selected destination is among those with discount. The word `in` is crucial here. Python recognizes it as a keyword.

For example, here we are checking, whether a letter `'e'` is present in the string `'Hello'` :

```
>>> 'e' in 'Hello'
```

And here we are checking, whether 'Hello' is present in the list `words` :

```
>>> words = ['hello', 'Hello', 'World']  
>>> 'Hello' in words
```

As we should already know, this operation is called **membership testing**. But what actually is the value that is returned by membership testing?

```
1 city = 'New York'
2     Osnova      .n city
3
4
5 names = ['Helga', 'Herta', 'Helmut']
6
7 result2 = 'Helmi' in names
```

spustit kód

Understanding the Truth

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [ONSITE PROJECT](#) / [UNDERSTANDING THE TRUTH](#)

Membership testing as well as comparison operations return as a result one of the two words:

- **True**
- **False**

100% z Lekce 2

They represent for us a new data type called **boolean**. There are only these two values, i.e. data type. They tell us, whether something is true (e.g. $2 < 3$) or false (e.g. $2 > 3$). Actually based on these values Python or executes a body of a conditional branch or it skips it. They serve as **signals**.

Even though these two values look like a string, they are actually numbers behind the scenes - number 1 is **True** and number 0 is **False**.

Actually each value in Python can be judged from the perspective of truthiness or falsiness. These are the values considered to be **False** (all other are **True**):

- 0 or 0.0
- empty string, list ...
- False
- None

To check, whether a value is **True** or **False**, we use command **bool**. Actually this command converts any value into a boolean value.

1 |

Logical Operations

PYTHON ACADEMY / 2. CONDITIONS / ONSITE PROJECT / LOGICAL OPERATIONS

The password requirements in our program are quite tricky. Let's review them again:

Password has to:

1. be at least 8 chars long,
2. cannot begin with a number and
3. cannot end with a number and
4. has to contain both letters and numbers

How can we check all those things at once? We will need to use some kind of glue, that will check, whether at least one of the requirements fails. That means that:

1. the password is too short
2. **or** it begins with number
3. **or** it ends with a number
4. **or** it does not contain both letters and numbers.

The word **or** is exactly what we need. And it is fortunately part of Python language.

Besides the word **or** we can use words **and** and **not**. These words are called **LOGICAL OPERATORS**.

So what are the results if using **or**, **and** or **not**? The best way to demonstrate it is to use



Operation	Result
True and True	True
True and False	False
False and False	False

OR

Operation	Result
True or True	True
True or False	True
False or False	False

NOT

Operator **not** is used on single value:

Operation	Result
not True	False
not False	True

Let's see how they work:

1 |

Osnova

[spustit kód](#)

Final Program

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [ONSITE PROJECT](#) / [FINAL PROGRAM](#)

It is time to add all those conditions, that are missing in our program. Here they are:

- Check, whether the user entered a destination number in a correct range
- Check, whether the selected destination is among those with discount
- We cannot provide our services to clients under 15 years of age
- Emails have to contain @ symbol
- Password has to be at least 8 chars long, cannot begin and end with a number and has to contain both letters and numbers

We would like the interaction with our program to look like this:

```
=====
Welcome to the DESTINATION,
place where people plan their trips
```

100% z Lekce 2

Osnova

1 - Prague		1000
2 - Wien		1100
3 - Brno		2000
4 - Svitavy		1500
5 - Zlin		2300
6 - Ostrava		3400

Please enter the destination number to select: 4

Lucky you! You have just earned 25% discount for your destination -
Svitavy

Press enter to continue

=====

REGISTRATION:

In order to complete your reservations, please share few details about
yourself with us.-----
NAME: Bob

=====

SURNAME: Rob

=====

YEAR of BIRTH: 2000

=====

EMAIL: bob@rob.com

=====

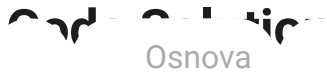
PASSWORD: passwor1d

=====

Thank you Bob

We have made your reservation to Svitavy

The total price is 1125.0



Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution



CONDITIONS

Introduction

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [CONDITIONS](#) / [INTRODUCTION](#)

100% z Lekce 2


```
2   ... to be executed if test expression boolean value is  
Osnova
```

Test expression

Test expression is tested for the truth value as if it would be inserted into bool constructor - `bool(test_expression)` . Only if this operation evaluates to `True` , the code inside the `if` statement is executed.

If the test evaluates to `False` , then the suite part of the `if` statement is skipped. That code will not be executed.

```
1  a = 5  
2  if a:  
3      print(a**2)
```

In the example above `bool(5)` evaluates to `True` . It is convenient that we do not have to write the test like this:

```
1  a = 5  
2  if a !=0 and a != None and a!= [] and a!= () etc.:  
3      print(a**2)
```

... because `bool(test)` evaluates all those cases for us.

Colon ":"

It is mandatory to use colon at the end of the header. This tells Python that:

- here ends the header part of the statement
- on the next line expect indented code block

Why is it important to distinguish between **code blocks**?

Because it tells Python, which lines of code should not be executed if the test expression evaluates to `False` .

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [CONDITIONS](#) / [IF - ELSE STATEMENT](#)

Statement **if-else** contains two branches:

```
1 if test_expression:
2     statements executed if test expression boolean value is True
3 else:
4     statements executed if test expression is False
```

Using only **if** statements causes, that the code inside the conditional statement is executed or not executed at all. However there can be situations, when we want to execute code under a specific condition or otherwise execute something else - but not both at the same time. For that purpose **else** part of the conditional statement is introduced.

```
1 a = 26
2 b = a / 3.14
3 if b > 9:
4     print('Greater than 9')
5 else:
6     print('Less than 9')
```

Scenario of executing one or the other set of instructions is actually analogy of **logical operation**. However, logical operations allow us to execute **only one expression** instead of set of statements. For more see the module dedicated to boolean operations.

The **if-else** statement can be reflected in the **or** boolean operation used as analogy to conditional statement:

```
>>> name = ''
>>> name or print('No name specified')
No name specified
```

Osнова shorthand for **else if**. Statement **if-elif-else** can contain three and more branches. So in this scenario we could need to check multiple conditions. In such case **elif** statements are added after the **if** statement:

```
1 if test_expression1:
2     statements executed if test expression boolean value is True
3 elif test_expression2:
4     statements executed if previous test expressions boolean value is
    False
5 elif test_expression3:
6     statements executed if previous test expressions boolean value is
    False
7 else:
8     statements executed if all test expressions boolean value is
    False
```

And when taking example with actual values:

```
1 city = 'Berlin'
2
3 income = 50000
4
5 if city == 'Monaco':
6     net_income = income * 0.9
7 else:
8     if city == 'Luxembourg':
9         net_income = income * 0.6
10    else:
11        if city == 'Dublin':
12            net_income = income * 0.85
13        else:
14            net_income = income * 0.75
```

Code above demonstrates how **difficult is to read** conditional statements if only **if** and **else** statements were used. Readability is improved by introducing **elif**. Also, we do not have to

```
1  # 'Re - in'
2  Osnova
3  income = 50000
4
5  if city == 'Monaco':
6      net_income = income * 0.9
7  elif city == 'Luxembourg':
8      net_income = income * 0.6
9  elif city == 'Dublin':
10     net_income = income * 0.85
11 else:
12     net_income = income * 0.75
```

CONDITIONS +

Osnova : Conditional operator

PYTHON ACADEMY / 2. CONDITIONS / CONDITIONS + / TERNARY CONDITIONAL OPERATOR

The **if-else** version of conditional statement can be written in Python with **one line** of code. This line of code is called ternary operator as it imitates the ternary operator in C language. The result of this expression is then stored in variable.

Let's imagine we wanted to write the following few lines of code:

```
1 hour = 15
2 day_time = ''
3 if hour < 12:
4     day_time = 'Morning'
5 else:
6     day_time = 'Afternoon'
7
8 print('Good',day_time )
```

... On **one line**:

```
1 hour = 15
2 day_time = 'Morning' if hour < 12 else 'Afternoon'
3 print('Good ',day_time )
```

General Syntax

variable_name = expression1 if condition else expression2

Result of **expression1** is assigned to variable if condition evaluates to **True** , otherwise the result of **expression2** is assigned to the variable.

Code Task

Try to re-code the following lines of code using ternary operator:

100% z Lekce 2

```
3     Osnova
4     if ... title():
5         result = 'Titlecased'
6 else:
7     result = 'Not titlecased'
```

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution



QUIZ

1/10

What do the conditional statements consist of?

A. header and suite

B. header, suite and ending

C. header, conditional statement and ending

D. header and conditional statement

Osnova

HOME EXERCISES

The year of the birth

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [HOME EXERCISES](#) / [THE YEAR OF THE BIRTH](#)

Your task is to create a script called `birth_year.py` that will:

- ask the user for his/her age
- calculate the year the person was born in
- print the resulting year

Example of running the script:

```
/Users/PythonBeginner/Lesson1$ python birth_year.py
How old are you? 35
You were (probably) born in 1982
```

Online Python Editor

```
1 # Try to code the following exercises on your own - you always have to solution :
2
3 # Feel free to delete all this text now and write your code.
4
5 # Good luck! :)|
```

Osnova

[spustit kód](#)

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

[Click to see our solution](#)

Day converter

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [HOME EXERCISES](#) / [DAY CONVERTER](#)

Your task is to create a script called `convert_day.py` that will:

- ask for a number between 1 to 7
- return the name of corresponding weekday (1 - `'Monday'` , 2- `'Tuesday'` , 3 - `'Wednesday'` , 4- `'Thursday'` , 5- `'Friday'` , 6- `'Saturday'` , 7- `'Sunday'`)
- if no input has been provided (user hitting enter without typing anything), the program should print: 'No input provided'

100% z Lekce 2

Forma "running" script:
Osnova

```
~/PythonBeginner/Lesson1$ python convert_day.py  
Please enter the number of the day: 3  
Wednesday
```

```
~/PythonBeginner/Lesson1$ python convert_day.py  
Please enter the number of the day: abc  
Enter only numbers, please
```

```
~/PythonBeginner/Lesson1$ python convert_day.py  
Please enter the number of the day:  
No input provided
```

Online Python Editor

1 |

spustit kód

100% z Lekce 2

Osnova

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution



Determining the String Type

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [HOME EXERCISES](#) / [DETERMINING THE STRING TYPE](#)

Your task is to create a script called `string_type.py` that will:

- Ask user for any string
- Determine, whether the string entered
 - contains only numbers - digits - in that case the program should print to the terminal: `'Numeric'`
 - contains only letters - in that case the program should print to the terminal: `'Letters Only'`
 - otherwise print to terminal: `'Mixed'`

Example of running the script:

```
/Users/PythonBeginner/Lesson1$ python string_type.py
Give me some input: Abc
Letters Only
```

```
/Users/PythonBeginner/Lesson1$ python string_type.py
Give me some input: 4every1
Mixed
```

100% z Lekce 2

aktivní
Osnova
čas: 99

Online Python Editor

1 |

spustit kód

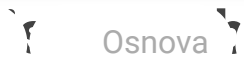
Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution



100% z Lekce 2



even 2 points [H]

[PYTHON ACADEMY](#) / [2. CONDITIONS](#) / [HOME EXERCISES](#) / [DISTANCE BETWEEN 2 POINTS \[H\]](#)

Your task is to create a script called **dist.py**. The program should ask for x and y coordinates for 2 points and calculate the distance between those 2 points. The output should be a float, therefore let's round the result to 2 decimal places.

- The distance should be straight line between the two points.
- The coordinates cannot be negative numbers.

Example of running the script:

```
/Users/PythonBeginner/Lesson1$ python dist.py
Point A, X Coordinate: 234
Point A, Y Coordinate: 34
Point B, X Coordinate: 27
Point B, Y Coordinate: 114
The distance between the points A and B is 221.92
```

You may want to look at the:

- [math libraries - method sqrt\(\)](#),
- the built-in function [round\(\)](#).
- and [abs\(\)](#).

Online Python Editor

1 |

Osnova

spustit kód

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution



DALŠÍ LEKCE