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 :)

REVIEW EXERCISES

Concatenation

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

- ask the user for the first name
- ask the user for the last name

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 your computer or here in our editor.

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

spustit kód

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

Click to see our solution

1. ask user for inputs
2. concatenate the inputs putting `", "` between them
3. inside the print() function concatenate string `'Your name is: '` with the full name we have created on the previous line

```
# 1
name = input('What is your first name? ')
surname = input('What is your last name? ')

# 2
full_name = surname + ', ' + name

# 3
print('Your name is: ' + full_name)
```

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.

100% z Lekce 2

```
quetzalcoatl is 12 characters long
```

Online Python Editor

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

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

spustit kód

Code Solution

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

- to find out the length of a string, we can use `len()` function and store it inside a variable `word_length`
- once we have the length calculated, we can print our result sentence
- in the result sentence we have to perform concatenation
- concatenation can be however performed only among sequenes of the same data type
- `word_length` is however an integer and therefore we have to use the function `str()` to convert it into a string (line 2)

```
1 word_length = len('quetzalcoatl')
2 print('quetzalcoatl is ' + str(word_length) + ' characters long')
```

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.
3. Print out the city at the index 2 introduced by the string: `At index 2 we have:`
4. Print out the city at the last index introduced by the string: `Last city, located at index <index_num>, 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']
```

Online Python Editor

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

```
# List

# Printing the list

# Printing the city at index 2

# Getting last index

# Printing the city with last index
```

spustit kód

Code Solution

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

Click to see our solution

- first, we create a variable `cities`, where we store list[] containing names of cities
- if we want to print city at the index 2, we need to use `cities[2]` statement
- you know that index starts from 0, so last index can be determined as length of list - 1, so `len(cities) - 1`

```
1 cities = ['New York', 'Los Angeles', 'Berlin', 'Prague', 'London']
2 print('Cities in my list: ' + str(cities))
3 print('At index 2 we have: ' + cities[2])
4 last_index = len(cities) - 1
5 print('Last city, located at index ' + str(last_index) + " we have:
  " cities[last_index])
```

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
```

100% z Lekce 2

```
1
```

spustit kód

Code Solution

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

Click to see our solution

- Emails are represented as strings

- To do that, we need to find out, where the slice should begin (start index) and where it should end (stop index)
- We actually already know, that the start index is 0 - we want to extract substring from from the beginning up to "@"
- So our task here is to determine, at which index the `"@"` symbol encounters itself
- There are multiple ways, we can accomplish this:
    1. use `email.find(substr)` method - this operation looks for a substring `subtstr` ("@") in the `email` and returns the index at which it was found
    2. user `email.split(substr)` method - this operation splits the `email` string on delimiter `substr` ("@") and returns a list of parts, the original string was broken into

Method .find()

```
email1 = 'mr.reilly@gmail.com'
email2 = 'john55@yahoo.com'
email3 = 'elgringo@atlas.sk'

stop1 = email1.find('@')
stop2 = email2.find('@')
stop3 = email3.find('@')

print(email1[:stop1])
print(email2[:stop2])
print(email3[:stop3])
```

Method .split()

First we split the email in 2 parts and then we extract the `index [0]`. You can use `print()` function to inspect what is actually happening in the first and the second step.

100% z Lekce 2

```
4
5  email1_part = email1.split('@')[0]
6  email2_part = email2.split('@')[0]
7  email3_part = email3.split('@')[0]
8
9  print(email1_part)
10 print(email2_part)
11 print(email3_part)
```

ONSITE PROJECT

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 subsequently ask the user for registration collecting the following data: name, surname, year of birth, email & password

Task for Lesson 2

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 apropriate message.

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

100% z Lekce 2

```
print('=' * 80)
print('''Welcome to the DESTINATIO,
place where people plan their trips''')
print('=' * 80)

# Offer destinations
print('We can offer you the following destinations:')
print('-' * 80)
print('''
1 - Prague  | 1000
2 - Wien    | 1100
3 - Brno    | 2000
4 - Svitavy | 1500
5 - Zlin    | 2300
6 - Ostrava | 3400
''')
print('-' * 80)

selection = int(input('Please enter the destination number to select: '))
DESTINATIONS = ['Prague', 'Wien','Brno','Svitavy','Zlin','Ostrava']
PRICES = [1000, 1100, 2000, 1500, 2300, 3400]
DISCOUNT_25 = ['Svitavy', 'Ostrava']


destination = DESTINATIONS[selection-1]
price = PRICES[selection-1]

print('=' * 80)
print('REGISTRATION:')
print('-' * 80)
print('In order to complete your reservations, please share few details about yourself with us.')
print('-' * 80)

```

```
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

We are getting to the final stage of creating our small app. At this phase, we need to begin to check, whether the data 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
```

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:

```
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 line:

```
if age < 15:
    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`.

```
```

spustit kód

Multiple Conditions

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:

```
else:
    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:

```
if destination == 'Svitavy':
    price = price * 0.75
else:
    if destination == 'Brno':
        price = price * 0.90
    else:
        price = price * 0.50
```

What we have just seen is called **nesting** of conditional statements. It is ok two 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:

```
if destination == 'Svitavy':
    price = price * 0.75
elif destination == 'Brno':
```

100% z Lekce 2

```
6       price = price * 0.50
```

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

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.

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)
```

spustit kód

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?

Let's try to perform few checks.

```
1  city = 'New York'
2
3  result1 = 'ew' in city
4
5  names = ['Helga','Herta','Helmut']
6
7  result2 = 'Helmi' in names
```

Understanding the Truth

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

- `True`
- `False`

So what are these words?

They are **not strings**, they represent for us a new data type called **boolean**. There are only these two values in boolean 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.

spustit kód

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

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 boolean values:

AND

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

Operation	Result
not True	False
not False	True

Let's see how they work:

```
1
```

spustit kód

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
================================================================================
We can offer you the following destinations:
--------------------------------------------------------------------------------
-------
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
```

Code Solution

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

Click to see our solution

```python
# Offer destinations
print('We can offer you the following destinations:')
print('-' * 80)
print('''
1 - Prague  | 1000
2 - Wien    | 1100
3 - Brno    | 2000
4 - Svitavy | 1500
```

```
print('-' * 80)


selection = int(input('Please enter the destination number to select: '))
DESTINATIONS = ['Prague', 'Wien','Brno','Svitavy','Zlin','Ostrava']
PRICES = [1000, 1100, 2000, 1500, 2300, 3400]
DISCOUNT_25 = ['Svitavy', 'Ostrava']


# Check, whether entered valid input
if not 0 < selection <= len(DESTINATIONS):
    print('We are sorry, but we can offer only those ' + str(len(DESTINATIONS)) + ' destinations')
else:
# Calculate the price & check whether discount applicable for the selected destination
    destination = DESTINATIONS[selection-1]
    price = PRICES[selection-1]
    if destination in DISCOUNT_25:
        print('Lucky you! You have just earned 25% discount for your destination - ' + destination)
        input('Press enter to continue')
        price =  price * 0.75
    print('=' * 80)


# Before we can confirm your reservation, you will need to register yourself
print('registration:'.upper())
print('-' * 80)
print('In order to complete your reservations, please share few details about yourself with us.')
print('-' * 80)

name = input('NAME: ')
print('=' * 40)
```

```
print('=' * 40)
email = input('EMAIL: ')
print('=' * 40)
password = input('PASSWORD: ')
print('=' * 80)

# Final check
if 2017 - int(birth_year) < 15:
    print('Sorry, we cannot offer our services to babies')

elif '@' not in email:
    print('Sorry, you have provided incorrect email')

elif (password.isnumeric() or password.isalpha()
      or password[0].isnumeric() or password[-1].isnumeric()
      or len(password) < 8):
    print('''
    Our passwords have to:
    * contain numbers and letters
    * be min 8 chars long
    * cannot begin or end with digit

    We cannot accept your password
    ''')

else:
# SUMMARY - name, price, date
    print('Thank you ' + name)
    print('We have made your reservation to ' + destination)
    print('The total price is ' + str(price))
```

CONDITIONS

Introduction

So far we know, that computers can perform calculations by evaluating expressions and these expressions can be chained in a single statement. Each data type has its own set of operations that can be represented in expressions. However, once we have a result of an expression, we need to do something with it. We need to decide, what to do next. For this purpose the concept of **conditional statements** has been introduced.

Conditional statements are needed in order to allow the program to perform decisions based on results computed previously. Use of conditions in our code imply, that the program will probaby not execute all lines of code - only those, where conditions are met.

Conditional statements can be identified in Python code by presence of **keywords**:

- `if` - required - test the first condition - it is the first statement of the conditional statement
- `elif` - optional - used to test further conditions

Syntax

As conditional statements are compound statements, they consist of header and suite.

- **Header** is the place, where conditional keywords are used and the boolean test is performed.
- **Suite** is a set of indented statements performed if tested condition is `True` or executed if all tests evaluated `False` (this is the case of statements under the else header).

```
Header:
    Suite
    ...
```

If statement

PYTHON ACADEMY / 2. CONDITIONS / CONDITIONS / IF STATEMENT

In general, the if statement looks like this:

```
if test_expression:
    statements to be executed if test expression boolean value is True
```

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.

```
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` .

If - else statement

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
```

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.

```
a = 26
b = a / 3.14
if b > 9:
    print('Greater than 9')
else:
    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
```

If - elif - else statement

PYTHON ACADEMY / 2. CONDITIONS / CONDITIONS / IF - ELIF - ELSE STATEMENT

The word `elif` is a 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:

```
if test_expression1:
    statements executed if test expression boolean value is True
elif test_expression2:
    statements executed if previous test expressions boolean value is
False
```

```
    False
else:
    statements executed if all test expressions boolean value is
False
```

And when taking example with actual values:

```
city = 'Berlin'

income = 50000

if city == 'Monaco':
    net_income = income * 0.9
else:
    if city == 'Luxembourg':
        net_income = income * 0.6
    else:
        if city == 'Dublin':
            net_income = income * 0.85
        else:
            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 nest when using elif:

```
city = 'Berlin'

income = 50000

if city == 'Monaco':
    net_income = income * 0.9
elif city == 'Luxembourg':
    net_income = income * 0.6
elif city == 'Dublin':
    net_income = income * 0.85
```

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
```

```
    day_time = 'Morning'
else:
    day_time = 'Afternoon'

print('Good',day_time )
```

... On **one line**:

```
hour = 15
day_time = 'Morning' if hour < 12 else 'Afternoon'
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:

```
my_str = 'Python'
result = ''

if my_str.istitle():
    result = 'Titlecased'
else:
    result = 'Not titlecased'
```

Code Solution

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

Click to see our solution

```
my_str = 'Python'
result = 'Titlecased' if my_str.istitle() else 'Not titlecased'
```

QUIZ

Conditions

1/10

A. header and suite

B. header, suite and ending

C. header, conditional statement and ending

D. header and conditional statement

HOME EXERCISES

100% z Lekce 2

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 i
2
3  # Feel free to delete all this text now and write your code.
4
5  # Good luck! :)
```

spustit kód

100% z Lekce 2

Code Solution

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

Click to see our solution

```
age = int(input('How old are you? '))
curr_year = 2018
birth_year = curr_year - age

print('You were (probably) born in', birth_year)
```

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'
- if the input is not a number the program should print: 'Enter only numbers, please'

Example of running the script:

```
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

Click to see our solution

```
week = ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday', 'Sunday')
day = input('Please enter the number of the day: ')

if not day:
    print('No input provided')
elif day not in ['1', '2', '3', '4', '5', '6', '7'] :
    print('Enter only numbers between 1 and 7, please')
else:
    index = int(day) - 1
    print(week[index])
```

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'`

```
/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
```

```
/Users/PythonBeginner/Lesson1$ python string_type.py
Give me some input: 99
Numeric
```

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

We ask for the user input using `input()` function. We store the input inside the variable `user_input`.

After that we check, whether the variable `user_input` refers to a string that:

1. Contains letters only - `if user_input.isalpha():`
2. Contains numbers only - `elif user_input.isnumeric():`

If none of the above is True, the code inside the `else:` clause is executed.

```
user_input = input('Give me some input: ')

if user_input.isalpha():
    print('Letters Only')
elif user_input.isnumeric():
    print('Numeric')
else:
    print('Mixed')
```

Distance between 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,

- 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 ak the:

- math libraries - method sqrt(),
- the built-in function round()
- and abs()

Online Python Editor

```
1
```

100% z Lekce 2

spustit kód

Code Solution

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

Click to see our solution

In this task we have to glue together knowledge from multiple areas:

1. We need to find out, how to calculate the distance between two points at given coordinates
2. We need to get user input, that has to be **converted** into integer (because everything that `input()` function returns is a string
3. We need to calculate square root of a numeric value - we will use Pythagoras's theorem ($a^2+b^2=c^2$) to determine the distance between two points.

First of all, we will import the `math` module, that contains the `sqrt()` function. This function calculates square root

```
1  import math
```

Then we will create two variables `A` and `B`, that will represent position of a given point. Position will be expressed as a list of two values. The default coordinates will be set to `[0,0]`. Having values in place will make it easier to store user inputs at given coord position.

```
1  A = [0,0]
2  B = [0,0]
```

We should also make sure the **inputs are positive numbers**. We could achieve that by informing the user within the input message, or we can use the built-in function **abs()** which changes the inputs to its absolute values. So f.e. -3 -> 3.

Remember, it is a good practice when a change the users might not expect occurs, we need to inform them.

```
A[0] = abs(int(input('Point A, X Coordinate: ')))
A[1] = abs(int(input('Point A, Y Coordinate: ')))
B[0] = abs(int(input('Point B, X Coordinate: ')))
B[1] = abs(int(input('Point B, Y Coordinate: ')))

print("Your inputs have been changed to absolute values.")
```

Then we will calculate the lengths of two triangle sides called legs calculated as a difference between coordinate X and Y of a given point. The built-in function `round()` rounds the result to two decimal places.

```
a = round(A[0] - B[0], 2)

b = round(A[1] - B[1] ,2)
```

Having the length of the two sides of right-angled triangle, we can use Pythagoras's theorem:

```
result = round(math.sqrt(a**2 + b**2),2)
```

And finally we print the result:

```
print(result)
```

Summarized Code

```
import math

A = [0,0]
B = [0,0]
```

```
8  B[0] = abs(int(input('Point B, X Coordinate: ')))
9  B[1] = abs(int(input('Point B, Y Coordinate: ')))
10 print("Your inputs have been changed to absolute values.")
11
12 a = round(A[0] - B[0], 2)
13 b = round(A[1] - B[1], 2)
14 result = round(math.sqrt(a**2 + b**2),2)
15
16 print(result)
```

DALŠÍ LEKCE