Osnova

Lesson Overview

PYTHON ACADEMY / 4. WHILE LOOPS / LESSON OVERVIEW

Welcome to lesson 4, we missed you!

In last lesson you have learned how to use:

- dictionaries
- & sets.

And now, we will focus on **while loops**.

This concept is very useful, because automatization saves your time. You can automatizate some routine action, e.g. repetitive tasks. A tiny part of code can ensure that you don´t have to ask Python to print some information for 10 times, but just once.

At the end of this lesson, a voluntary project for self-study is being introduced. We highly recommend you to work on it. You can test, what you have learned so far. Go for it :)

100% z Lekce 4

Osnova

00:58

REVIEW EXERCISES

Dictionary

PYTHON ACADEMY / 4. WHILE LOOPS / REVIEW EXERCISES / DICTIONARY

Goal of this task is to put different dictionaries into our main dictionary.

100% z Lekce 4

[illegible] dictionary: `database = {'id123': {},'id124': {},'id125':` [illegible] `}`

Osnova

And here are the others:
`FirstDict = {'name': 'Thomas', 'age': 45, 'Country': 'Czechia', 'City': 'Brno'}`
`SecondDict = {'name': 'Daniel', 'age': 34, 'Country': 'Czechia', 'City': 'Prague'}`
`ThirdDict = {'name': 'Eva', 'age': 43, 'Country': 'Czechia', 'City': 'Olomouc'}`

Example of running script:

```
{'id123': {'name': 'Thomas', 'age': 45, 'Country': 'Czechia', 'City': 'Brno'}, 'id124': {'name': 'Daniel', 'age': 34, 'Country': 'Czechia', 'City': 'Prague'}, 'id125': {'name': 'Eva', 'age': 43, 'Country': 'Czechia', 'City': 'Olomouc'}}
```

Online Python Editor

100% z Lekce 4

Osnova

spustit kód

Code Solution

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

Click to see our solution

```
database = {'id123': {},'id124': {},'id125': {},'id126': {}}

FirstDict = {'name': 'Thomas', 'age': 45, 'Country': 'Czechia', 'City': 'Brno'}
SecondDict = {'name': 'Daniel', 'age': 34, 'Country': 'Czechia', 'City': 'Prague'}
ThirdDict = {'name': 'Eva', 'age': 43, 'Country': 'Czechia', 'City': 'Olomouc'}

database.update(id123 = FirstDict)
database.update(id124 = SecondDict)
database.update(id125 = ThirdDict)

print(database)
```

Sets

100% z Lekce 4

A the ... we ... ve dictionary with tram stations. Our goal is to identify which stations are
.du ... Osnova .st

```
TramStations = {
'No.1' : ['Reckovice', 'Semilasso', 'Husitska', 'Jungmannova',
'Kartouzska', 'Sumavska', 'Hrnicrska', 'Pionyrska', 'Antoninska',
'Moravske nam.', 'Malinovske nam', 'Hlavni nadr.', 'Nove sady',
'Hybesova', 'Vaclavska', 'Mendlovo nam.', 'Vystaviste main',
'Vystaviste G2', 'Lipova', 'Pisarky'],
'No.2' : ['Zidenice', 'Kuldova', 'Vojenska nemocnice', 'Tkalcovska',
'Kornerova', 'Malinovske nam.', 'Hlavni nadr.', 'Nove Sady',
'Hybesova', 'Vaclavska', 'Mendlovo nam.', 'Porici', 'Nemocnice UM',
'Celni', 'Hluboka', 'Ustredni hrbitov'],
'No.4' : ['Husovice','Nam. republiky','Vozovna
Husovice','Mostecka','Travnickova', 'Tkalcovska', 'Kornerova',
'Malinovske nam.', 'Hlavni nadr.', 'Nove sady', 'Silingrovo nam.',
'Ceska', 'Komenskeho nam.', 'Obilni trh', 'Uvoz']
                 }
```

Example running script:

```
{'Hlavni nadr.'}
```

Online Python Editor

```
1 |
```

100% z Lekce 4

Osnova

spustit kód

Code Solution

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

Click to see our solution

```
trams = {
'No.1' : ['Reckovice', 'Semilasso', 'Husitska', 'Jungmannova', 'Kartouzska', 'Sumavska', 'Hrnicrska', 'Pionyrska', 'Antoninska', 'Moravske nam.', 'Malinovske nam', 'Hlavni nadr.', 'Nove sady', 'Hybesova', 'Vaclavska', 'Mendlovo nam.', 'Vystaviste main', 'Vystaviste G2', 'Lipova', 'Pisarky'],
'No.2' : ['Zidenice', 'Kuldova', 'Vojenska nemocnice', 'Tkalcovska', 'Kornerova', 'Malinovske nam.', 'Hlavni nadr.', 'Nove Sady', 'Hybesova', 'Vaclavska', 'Mendlovo nam.', 'Porici', 'Nemocnice UM', 'Celni', 'Hluboka', 'Ustredni hrbitov'],
'No.4' : ['Husovice','Nam. republiky','Vozovna Husovice','Mostecka','Travnickova', 'Tkalcovska', 'Kornerova', 'Malinovske nam.', 'Hlavni nadr.', 'Nove sady', 'Silingrovo nam.', 'Ceska', 'Komenskeho nam.', 'Obilni trh', 'Uvoz']
        }

```

100% z Lekce 4

Osnova

```
ɪc    ıStations)
```

ONSITE PROJECT

Our Goal

PYTHON ACADEMY / 4. WHILE LOOPS / ONSITE PROJECT / OUR GOAL

Today we will work on a virtual shopping cart application. We want our shopping cart to be able to perform the following **actions**:

100% z Lekce 4

3. C l l t the t t l price
Osnova
4. Run the program until the user decides to terminate it

Optionally:

5. get basic statistics about counts of individual items in the basket
6. retrieve prices by name
7. compare the contents of our cart to a list items in offer
8. depict the cart contents in a neat way
9. remove the cart items

Before while loop

PYTHON ACADEMY / 4. WHILE LOOPS / ONSITE PROJECT / BEFORE WHILE LOOP

Create a program that will:

100% z Lekce 4

- add the items to list representing the shopping cart,

Osnova

- calculate the total price,
- print the cart contents and the total price.

Code Solution

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

Click to see our solution

```python
cart = []

item1 = float(input('Enter the price: '))
item2 = float(input('Enter the price: '))
item3 = float(input('Enter the price: '))

cart.append(item1)
cart.append(item2)
cart.append(item3)

total_price = cart[0] + cart[1] + cart[2]

print('CART: ' + str(cart))
print('Total Price: ' + str(total_price))
```

Repetition

PYTHON ACADEMY / 4. WHILE LOOPS / ONSITE PROJECT / REPETITION

What if we wanted to change the number of items to be entered? We would have to go back and

100% z Lekce 4

What if ... program to collect 100 price tags? We would have to add 100 lines.

Osnova

The solution is to use **loops**. Loop allows us to tell Python, which set of instructions to we want it to execute **repeatedly**.

Good candidates for change are these lines of code:

```python
item1 = float(input('Enter the price: '))
item2 = float(input('Enter the price: '))
item3 = float(input('Enter the price: '))

cart.append(item1)
cart.append(item2)
cart.append(item3)
```

What we are actually doing is repeating the following two commands:

```python
item = float(input('Enter the price: '))
cart.append(item)
```

Action 1 - Adding items

PYTHON ACADEMY / 4. WHILE LOOPS / ONSITE PROJECT / ACTION 1 - ADDING ITEMS

So in general, if want to say **while we have not collected 3 items, keep collecting them** the structure would look like this:

100% z Lekce 4

```
    c ``ecting them
```

Osnova

OR

```
while there are less than 3 items in a cart:
    keep collecting them
```

Action 1

Let's dive into our action list, starting with: 1. Add new items to the cart

Click to see our solution

```
while there are less than 3 items in a cart:
    item = float(input('Enter the price: '))
    cart.append(item)
```

Of course, we need the cart that we're filling:

Click to see our solution

```
cart = []
while there are less than 3 items in a cart:
    item = float(input('Enter the price: '))
    cart.append(item)
```

Code Task

100% z Lekce 4

• ... of i...ms in a cart

Osnova

• and express less than 3 items in a cart?

```

```

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 4

```
    ler'art) < 3:
        float(input('Enter the price: '))
    cart.append(item)
```

Osnova

Action 2 & 3 - List content & Total price

PYTHON ACADEMY / 4. WHILE LOOPS / ONSITE PROJECT / ACTION 2 & 3 - LIST CONTENT & TOTAL PRICE

Ok, so far we have the following code:

```
cart = []
while len(cart) < 3:
    item = float(input('Enter the price: '))
    cart.append(item)
```

We can now get to the 2nd and 3rd action point:

2. List the cart's content
3. Calculate the total price

The first one shouldn't be such a problem ;). Also, we'd be able to calculate the price f.e. by using indexing:

```
total_price = cart[0] + cart[1] + cart[2]
```

However, this is not a lesson on indexing! **Use a while loop** to sum all the 3 prices :)

```
1
```

100% z Lekce 4

Osnova

spustit kód

Code Solution

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

Click to see our solution

```
cart = []
while len(cart) < 3:
    item = float(input('Enter the price: '))
    cart.append(item)

# 3. Calculate the total price
total_price = 0
i = 0
while i < len(cart):
    total_price = total_price + cart[i]
    i = i + 1
```

100% z Lekce 4

```
    ' 'CART: ' + str(cart))
```

Osnova

```
# Printing total price
print('Total Price: ' + str(total_price))
```

Action 4 - Infinite asking

PYTHON ACADEMY / 4. WHILE LOOPS / ONSITE PROJECT / ACTION 4 - INFINITE ASKING

So we have already completed 3/4 tasks for this lesson, good job!:

1. **Add new items to it**
2. **List its content**
3. **Calculate the total price**
4. Run the program until the user decides to terminate it

So, our last task is to run the program until the user decides to terminate it. For this purpose we can use infinite loop. However, there is **bad and good** infinite loop.

Bad infinite loop

When using while loops, a infinite loop can occur. This can be bad, when **we do not have it under control**:

```
total_price = 0
i = 0
while i < len(cart):
    total_price = total_price + cart[i]
```

We need to be able to **change the variable**, that is being assessed in the loop's header.

```
total_price = 0
```

100% z Lekce 4

```
        t_price = total_price + cart[i]
   Osnova

    i = i + 1
```

Good infinite loop

Good infinite loop grant's the user possibility to **terminate the program by allowing for input**. What does that mean? A use case in our program is when we want to allow our users to add as many prices as they want and enter **'q'** to stop the process of price collection.

To create an **infinite loop** we need a a condition that will always evaluate `True`, unless we change the tested variable value inside the loop body.

Code Solution Summary

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

Click to see our solution

```
cart = []
repeat = True

# Infinite loop
while repeat:

    item = float(input('Enter the price: '))
    cart.append(item)

    answer = input('Press enter to continue or "q" to quit: ')

    if answer == 'q':
        repeat = False


```

100% z Lekce 4

Osnova

```
      :otal sum
while i < len(cart):
    total_price = total_price + cart[i]
    i = i + 1

print('CART: ' + str(cart))
print('Total Price: ' + str(total_price))
```

Action 4 - Infinite listing

PYTHON ACADEMY / 4. WHILE LOOPS / ONSITE PROJECT / ACTION 4 - INFINITE LISTING

Additionaly, we could keep listing the contents of the cart (you can use the cart below), until the user tells the program to stop, by entering letter **'q'**. Once we are at the end of the cart, we want to return to its beginning and show the first item and so forth.

Before we incorporate it into our program, let's try this first with the following cart:

```
cart = [1.02, 3.45, 6.82]
```

```
1
```

100% z Lekce 4

Osnova

spustit kód

Code Solution

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

Click to see our solution

```
cart = [1.02, 3.45, 6.82]
i = 0
repeat = True

while repeat:

    index= i % 3
    print(cart[index])

    answer = input('Press enter to continue or "q" to quit: ')

    if answer == 'q':
        repeat = False
    else:
        i = i+1
```

100% z Lekce 4

Osnova ıry

PYTHON ACADEMY / 4. WHILE LOOPS / ONSITE PROJECT / CODE SUMMARY

Let's now try to incorporate the infinite listing into our program:

```python
i = 0
repeat = True
# Infinite listing
while repeat:

    index = i % 3
    print(cart[index])

    answer = input('Press enter to continue or "q" to quit: ')

    if answer == 'q':
        repeat = False
    else:
        i = i + 1
```

So the program should:

1. keep **asking user for price** input until the key 'q' is pressed
2. keep **listing items** until the key 'q' is pressed
3. using while loop to **calculate the total price**

Code Solution

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

Click to see our solution

100% z Lekce 4

Osnova

```
# [illegible] asking
[illegible]:

    item = float(input('Enter the price: '))
    cart.append(item)

    answer = input('Press enter to continue or "q" to quit: ')

    if answer == 'q':
        repeat = False

i = 0
repeat = True
# Infinite listing
while repeat:

    index = i % len(cart)
    print(cart[index])

    answer = input('Press enter to continue or "q" to quit: ')

    if answer == 'q':
        repeat = False
    else:
        i = i + 1

i = 0
total_price = 0
# Calculating total sum
while i < len(cart):
    total_price = total_price + cart[i]
    i = i + 1

print('CART: ' + str(cart))
print('Total Price: ' + str(total_price))
```

100% z Lekce 4

WHILE LOOPS

Loops introduction

Computers are good friends when speaking about doing repetitive and boring tasks on our behalf. Repetitions in programming are called loops. For example, if we want to find and print out all the numbers between 3 and 1358979677 divisible by 3, it is better to leave this task to our PC. Actually a computer is designed to work like that, to run in cycles - its processor runs in cycles. The number of cycles per second is called Hertz. So these guys are nowadays doing billions of cycles per second. Why shouldn't we let them do, what they know the best?

In programming, performance of repeating tasks is called looping and Python recognizes 2 kinds of loops:

- **while loop**

Osnova

statements that means they consist of header and suite similarly to

Template for the while loop:

```
while test:
    your code
```

Template for the for loop:

```
for item in iterable:
    your code
```

Header contains the reserved keyword - `for` or `while` . Suite contains at least one statement. Statements that compose the suite (body) of the loop are repeatedly executed until a defined end state is reached.

The principle of While

PYTHON ACADEMY / 4. WHILE LOOPS / WHILE LOOPS / THE PRINCIPLE OF WHILE

While loop is a more general loop than for loop. Meanwhile Python has to perform some magic behind the scenes of for loop, while loop's principle is very straightforward.

With `while` loop, the code is repeatedly executed inside the while body as long as the test in the loop's header evaluates `True` :

```
while test:
    statements
    statement manipulating variable used in the header expression
```

While steps

1. Once the while statement header is encountered by program execution, the test in the header is evaluated as `bool(test)` by Python

100% z Lekce 4

2. I[illegible] [illegible] of the boolean test is `True`, then the statements in the loop's suite are [illegible] Osnova [illegible]se the suite is skipped.

3. Program execution returns back to the `while` header and again evaluates the test it contains.

4. In the moment, the test evaluates to `False`, the loop terminates and the program execution continues with lines after the while block.

As an **example**, try to run the following code on your computer:

```python
num = 5

while num > 0:
    print('My number is ' + str(num))
    num = num - 1
print('The loop has terminated')
```

spustit znovu

```
My number is 4
My number is 3
My number is 2
My number is 1
The loop has terminated
```

100% z Lekce 4

Osnova

PYTHON ACADEMY / 4. WHILE LOOPS / WHILE LOOPS / INFINITE LOOP

Using while loop brings with itself the danger of getting stuck in an infinite loop. Infinite loop never terminates and is caused by poor code design - when the test in the header can never evaluate to `False` .

Example of such situation is, when we count from 1 up and want to stop loops execution once the counter is equal to 0. The value in the variable `num` will never reach the limit 0 as it moves away from it.

```python
num = 1
while num > 0:
    print(num)
    num += 1
```

If you will run the code above, the program execution will enter infinite loop! In order to stop such a program we need to use keyboard shortcut **Ctrl+C**.

Important

One of the statements executed inside the loop's suite has to **manipulate variable** that forms part of the header test expression, otherwise:

- If this variable was not changed, the loop would run infinitely long.
- If value of this variable does not approach the limit, the loop runs infinitely long

Use case 1

PYTHON ACADEMY / 4. WHILE LOOPS / WHILE LOOPS / USE CASE 1

The `.get()` method is very useful, **when we want to count number of occurrences** of items. In order we can understand the below code, we need to know something about the `while` loop

100% z Lekce 4

Osnova

```
[' een', 'blue', 'black', 'red', 'red', 'yellow', 'blue',
    b  k' , 'red', 'green']
color_counts = {}

while colors:
    color = colors.pop()

    if color not in color_counts:
        color_counts[color]=0

    color_counts[color] = color_counts[color] + 1
```

or this way

```
colors = ['green', 'blue', 'black', 'red', 'red', 'yellow', 'blue',
'grey', 'black' , 'red', 'green']
color_counts = {}

while colors:
    color = colors.pop()
    color_counts[color] = color_counts.get(color,0) + 1
```

The `color_counts` variable should be at the end refer to a dictionary:

```
>>> color_counts
{'grey': 1, 'blue': 2, 'black': 2, 'red': 3, 'green': 2, 'yellow': 1}
```

Use case 2

PYTHON ACADEMY / 4. WHILE LOOPS / WHILE LOOPS / USE CASE 2

While loops are often used in cases **when we do not know in advance, how many repetitions** will be necessary to be executed. For example, when the program requires user's input which is unpredictable.

100% z Lekce 4

Osnova

```
[illegible]ort sleep

num_seconds = int(input('How many seconds to you need?: '))

while num_seconds:
    sleep(1)
    print(num_seconds)
    num_seconds = num_seconds - 1
```

Output:

```
5
4
3
2
1
```

Iteration techniques

Let's go over 3 iteration techniques to further demonstrate the use of this loop.

Looping using a number

Loop will go on until the variable value is not equal to 0:

```
number = 10
while number:
    print(number)
    number = number- 1
print('Happy New Year')
```

If number variable acquires value 0, the loop body is not executed and the loop code block is

[illegible] iterables

Osnova

Cutting of pieces of a string (or other iterable):

```python
my_str = 'while loops are more genEral'
while my_str:
    if my_str[0].isupper():
        print('I have found capital:',my_str[0])
    my_str = my_str[1:]
```

The program above checks whether there are any capital letters in the string my_str. Similarly to number being equal to 0, if iterable is empty (all the chars cut off), then the loop exits.

Check it out in Python Tutor.

Using index to retrieve an item

Less elegant and less Pythonic than the previous two examples:

```python
my_str = 'while loops are more genEral'
index = 0
while index < len(my_str):
    if my_str[index].isupper():
        print('I have found capital:',my_str[index])
    index += 1
```

100% z Lekce 4

Osnova

QUIZ

While Loop

PYTHON ACADEMY / 4. WHILE LOOPS / QUIZ / WHILE LOOP

1/9

What is the purpose of the while loop?

A. To repeat designated set of commands until the tested condition is not met

B. To pause the code execution

C. To repeat designated set of commands until the tested condition is met

D. To perform condition testing

100% z Lekce 4

HOME EXERCISES

Student Names

We have a class of students. All the student names are stored in the list `students` .

```
students = ['Adam, Baker','Chelsea, Archer',
            'Marcus, Archer','Oliver, Cook',
            'Alex, Dyer', 'Sandra, Turner',
            'Ann, Fisher', 'Glenn, Hawkins',
            'Samuel, Baker','Clara, Slater',
            'Tyler, Hunt', 'Alex, Smith',
            'Clara, Woodman','Abraham, Mason',
            'Marcus, Sawyer','Alex, Glover',
            'Glenn, Cook','Clara, Fisher',
            'Alfred, Dyer', 'Curt, Head',
              'Samuel, Hawkins', 'Ann, Woodman',
     Osnova    Sandra, Slater', 'Curt, Dyer']
```

Our task is to extract an overview of what unique names and surnames do we have in the class.

```
~/PythonBeginner/Lesson2 $ python student_names.py
Extracting ...
Unique names:
{'Ann', 'Curt', 'Clara', 'Abraham', 'Chelsea', 'Oliver', 'Glenn',
'Samuel', 'Alfred', 'Marcus', 'Alex', 'Adam', 'Tyler', 'Sandra'}
Unique surnames:
{'Woodman', 'Head', 'Dyer', 'Smith', 'Cook', 'Hunt', 'Slater', 'Baker',
'Parker', 'Turner', 'Fisher', 'Sawyer', 'Mason', 'Archer', 'Glover',
'Hawkins'}
```

Online Python Editor

```
1
```

spustit kód

100% z Lekce 4

Code Solution

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

Click to see our solution

In the `students` list, names and surnames are repeated (e.g. surname 'Dyer' in 'Alex, Dyer' and 'Curt,Dyer'). However, we do not want names neither surnames to repeat inside the container. Therefore we need to do one of the following:

- check, whether the name resp. surname has already been collected using if condition
- or collect names and surnames into two sets. That way, we will not have to check whether an item is already present in the container or not. Set does not add any item, that is already present in it (it keeps items unique inside itself). In the example below, we chose to go use sets.

1. We will need to split each name into a name and surname,
2. then we will have some containers, into which we will be adding each name resp. surname,
3. splitting each item of `students` list and adding it into a container is a repetitive job therefore it will be the best if we use the **while loop** to execute these two tasks repeatedly,
4. at the end we just print the results with introductory comments.

```
students = ['Adam, Baker','Chelsea, Archer',
            'Marcus, Archer','Oliver, Cook',
            'Alex, Dyer', 'Sandra, Turner',
            'Ann, Fisher', 'Glenn, Hawkins',
            'Samuel, Baker','Clara, Slater',
            'Tyler, Hunt', 'Alex, Smith',
            'Clara, Woodman','Abraham, Mason',
            'Marcus, Sawyer','Alex, Glover',
            'Glenn, Cook','Clara, Fisher',
```

Osnova

```
            'Samuel, Hawkins', 'Ann, Woodman',
            'Sandra, Slater', 'Curt, Dyer']

# 2.
names = set()
surnames = set()

print('Extracting... ')

# 3.
while students:
    # 1.
    name, surname = students.pop().split(', ')
    names.add(name)
    surnames.add(surname)

# 4.
print('Unique names:')
print(names)
print('Unique surnames:')
print(surnames)
```

Difference - Odd vs.Even

PYTHON ACADEMY / 4. WHILE LOOPS / HOME EXERCISES / DIFFERENCE - ODD VS.EVEN

Write a Python script that will sum all the even numbers and odd numbers separately. At the end the program should print to terminal **the absolute value of the difference** between the two sums of odd and even numbers.

Example of how the script should work:

1. We have a list of numbers: `[1,2,3,4,5,6,7,8]`

100% z Lekce 4

3. T[illegible] ''l s' all the odd numbers and the result store in the variable `odd = 1 + 3 +`

Osnova

4. Finally we want to get the difference among the two sums
5. We should make sure that the difference will not be negative number (you may want to look at built-in function for numeric data types in the lesson 1)

Of course your task is to find out, how to iterate over each item of the number sequence and not to write the summation manually.

For your script, please use the following list of numbers:

```
nums = [ 386, 462, 47, 418, 907, 344, 236, 375, 823,
         566, 597, 978, 328, 615, 953, 345, 399, 162,
         758, 219, 918, 237, 412, 566, 826, 248, 866,
         950, 626, 949, 687, 217, 815, 67, 104, 58, 512,
         24, 892, 894, 767, 553, 81, 379, 843, 831, 445,
         742, 717, 958,743, 527]
```

Example of running the script:

```
~/PythonBeginner/Lesson2 $ python diff_odd_even.py
The difference is: 96
```

100% z Lekce 4

Osnova

spustit kód

Code Solution

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

Click to see our solution

1. we have stored the list of our numbers in `nums` variable
2. we create two variables where we will keep the sums of odd resp. even numbers
3. values in `odd` and `even` variables will be increased repeatedly inside the while loop
4. inside the while loop, we will extract next number from the end of the nums list and store it in `num` variable
5. we will increase the value stored in `even` variable if the result of expression `num % 2` is equal to 0
6. we will increase the value stored in `odd` variable if the result of expression `num % 2` is equal to 1
7. at the end we calculate the difference among odd and even sums and turn the result to absolute value using `abs()` function

```
# 1.
nums = [386, 462, 47, 418, 907, 344, 236, 375, 823,
        566, 597, 978, 328, 615, 953, 345, 399, 162,
        758, 219, 918, 237, 412, 566, 826, 248, 866,
        950, 626, 949, 687, 217, 815, 67, 104, 58, 512,
```

100% z Lekce 4

Osnova

```python
odd = 0
even= 0

# 3.
while len(nums) > 0:

    # 4.
    num = nums.pop()

    # 5.
    if num%2==0:
        even = even + num
    # 6.
    else:
        odd = odd + num
# 7.
result = abs(odd - even)
print('The difference is:', result)
```

Nicer solution:

1. we do not need two separate variable for odd and even sums - we can use a list `sums` . Our sums list have only two indices (0,1). Now, what are the possible remainders when floor dividing any integer by 2? It is also 0 and 1
2. we use this fact to access sum of even numbers `sums[0]` (remainder 0) and to access sum of odd numbers `sums[1]` (remainder 1) (line 13)
3. lastly we just calculate the difference of `sums[0]` and `sums[1]`

Note the way we have written the while condition `while nums:` - this tells Python to loop meanwhile the nums list is not empty

```python
1  nums = [386, 462, 47, 418, 907, 344, 236, 375, 823,
2          566, 597, 978, 328, 615, 953, 345, 399, 162,
3          758, 219, 918, 237, 412, 566, 826, 248, 866,
4          950, 626, 949, 687, 217, 815, 67, 104, 58, 512,
```

100% z Lekce 4

Osnova

```python
sums = [0,0]

while nums:

    num = nums.pop()
    index = num % 2
    # 2.
    sums[index] = sums[index] + num
# 3.
result = abs(sums[0] - sums[1])

print('The difference is:', result)
```

Echo

PYTHON ACADEMY / 4. WHILE LOOPS / HOME EXERCISES / ECHO

Write a Python program that will create "echo sentences". Each word in the sentence we will feed in, should be repeated n number of times. The number of repetitions and the sentence to be manipulated are inputs provided by the user.

Example:
If the supplied number of repetitions is 3 and the sentence: `'I do not want to work today'` .

Output:
`'I I I do do do not not not want want want to to to work work work today today today'`

The resulting sentence cannot begin with space, unless the input sentence contained it.

Example of running the script:

100% z Lekce 4

```
Ent      of  petitions: 3
Osnova
         I do not want to work today
I I I do do do not not not want want want to to to work work work today
today today
```

Online Python Editor

spustit kód

Code Solution

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

100% z Lekce 4

Click to see our solution

Osnova

In order we can repeat each word, we have to isolate each word from the sentence. To isolate each word from a sentence, we can use `str.split()` method, that splits the string on delimiter (in this case the delimiter is blank space). The result is returned as a list of strings (we can save this list into a variable called f.e. `words`).

When we have all the words isolated in a list we can iterate over the list and repeat each word 3 times.

But wait, where will we store the repeated word? Back in the list `words` ? Better not - we may complicate our lives pretty much. Better we should create a new empty list, to which we will append all triplicated strings.

Our while loop should keep looping until the `word` list is empty - the `words` variable is each cycle evaluated as `bool(words)` . If it were empty, the expression would return `False` and the code inside the while loop would not execute.

Inside the loop we:

1. extract the next word from the beginning of the words list - `words.pop(0)`
2. enclose it inside a list brackets `[words.pop(0)]`
3. repeat that list `num_repeat` times (3 times)
4. add the result of the repetition to the current state of the `result` list

The above is repeated until there is nothing inside words list.

Before printing the result, we have to create a string. We do that by using `str.join()` method. We join words in the result list on space character (before we split the string on space character and now we are joining it back) - `" ".join(result)`

Finally, we can print the result string.

```python
num_repeat = int(input('Enter the # of repetitions: '))
sentence = input('Enter the string: ')

words = sentence.split()
```

100% z Lekce 4

Osnova

```
    word = [words.pop(0)]
    word = word * num_repeat
    result = result + word

result = " ".join(result)
print(result)
```

Alternative solution

We could solve the problem without collecting strings into a list, bud directly into a string

1. In this case we have to be careful and first concatenate space `" "` with popped word,
2. then we can replicate the word with space included
3. finally concatenate the result string with the repeated word
4. at the end we have to still strip blank space from the left side of the string (`result.lstrip()`)

```
sentence = 'I do not want to work today'
num_repeat = 3

words = sentence.split()

result = ''

while words:
    # 1.
    word = " " + words.pop(0)
    # 2.
    word = word * num_repeat
    # 3.
    result = result + word

# 4.
print(result.lstrip())
```

100% z Lekce 4

rer'' . `

Osnova

```
while words:
    result = result + (" " + words.pop(0)) * num_repeat

print(result.strip())
```

Longest word

PYTHON ACADEMY / 4. WHILE LOOPS / HOME EXERCISES / LONGEST WORD

Write a program that will take a list of words as input and will print to the terminal the longest word and its length in one tuple.

Please use the following list:

```
words = ['Python', 'is', 'a', 'widely', 'used',
         'high-level', 'programming', 'language',
         'for', 'general-purpose', 'programming,',
         'created', 'by', 'Guido', 'van', 'Rossum',
         'and', 'first', 'released', 'in', '1991.']
```

Example of running the script:

```
~/PythonBeginner/Lesson2 $ python longest_word.py
('general-purpose', 15)
```

Online Python Editor

1

100% z Lekce 4

Osnova

spustit kód

Code Solution

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

Click to see our solution

Our main goal is to iterate over the whole list keeping the length of the longest word yet seen.

Once we encounter a longer word, we should record it and also change the information about the maximum length of the word we have yet seen.

1. We decided to keep the longest word and its length in a tuple.

Each time we will find something longer, we will change it for another tuple of the same structure `(longest_word, word_length)` . So inside the while loop we repeatedly do the following:

100% z Lekce 4

2. [illegible] ord [illegible] m the beginning of the words list,

Osnova

3. [illegible] extracted word is longer that the longest word we have see so far,

4. if the newly extracted word is longer, we store it with its length in a tuple. We do not have to enclose them inside parentheses - Python automatically considers objects separated by comma to be tuples).

5. At the end we just print the resulting tuple.

```python
words = ['Python', 'is', 'a', 'widely', 'used',
         'high-level', 'programming', 'language',
         'for', 'general-purpose', 'programming,',
         'created', 'by', 'Guido', 'van', 'Rossum',
         'and', 'first', 'released', 'in', '1991.']

# 1.
max_word = ('',0)

while words:
    # 2.
    word = words.pop(0)
    # 3.
    if len(word) > max_word[1]:
        # 4.
        max_word = word, len(word)
# 5.
print(max_word)
```

Sum powers

PYTHON ACADEMY / 4. WHILE LOOPS / HOME EXERCISES / SUM POWERS

Write a Python script that will ask the user to enter a number from which it will compute the result. The result should be the sum of numbers less than or equal to the input number, each raised to power of its value. Then the script should print the result to the terminal.

100% z Lekce 4

- if t... ente... number 5, the program should compute the sum as: `1**1 + 2**2 + ... 4 ... 5**5`.

Osnova

- if the user enters 6, then it should be: `1**1 + 2**2 + 3**3 + 4**4 + 5**5 + 6**6`
- ...and so on.

Example of running the script:

```
~/PythonBeginner/Lesson2 $ python sum_powers.py
Please enter your number: 5
The result is: 3413
```

Online Python Editor

```
1
```

spustit kód

100% z Lekce 4

Osnova

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

Click to see our solution

Simply put, we will want to repeatedly:

1. raise a number to the power of the same magnitude and add it to the result,
2. and then decrease the number by 1.

```
num = int(input('Please enter your number: '))
result = 0

while num:
    # 1.
    result = result + num**num
    # 2.
    num = num - 1

print(result)
```

Looping over a dict

PYTHON ACADEMY / 4. WHILE LOOPS / HOME EXERCISES / LOOPING OVER A DICT

We have a dictionary:

```
film = {'name':'Forrest Gump',
        'made':1994,
        'director':'Robert Zemeckis',
```

100% z Lekce 4

```
        'f   fact':'''The house used in Forrest Gump is
                     the same house used in The Patriot
                     (2000). Some paneling was changed
                     for interior shots  in the latter
                     film.'''}
```

Osnova

Create a script that will print each key - value pair to the terminal in format: `"Key: <value> | Value: <value>"`

Example of running the script:

```
~/PythonBeginner/Lesson2 $ python list_items.py
Key: starring | Value: Tom Hanks
Key: director | Value: Robert Zemeckis
Key: made | Value: 1994
Key: name | Value: Forrest Gump
Key: soundtrack | Value: Multiple
Key: fun_fact | Value: The house used in Forrest Gump is the same house
used in The Patriot (2000). Some paneling was changed for interior shots
in the latter film.
```

Online Python Editor

```

```

100% z Lekce 4

Osnova

spustit kód

Code Solution

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

Click to see our solution

- Each repetition inside the while loop, we will want to extract both key and a value from the dictionary `film`
- We can extract key:value pair from a dictionary using `dict.popitem()` method
- Doing this repeatedly, we will gradually destroy the film dictionary, therefore we can permit ourselves to use the `while film` condition in the while header.
- Once film dictionary will be empty, the code inside the while loop will not be executed

```
film = {'name':'Forrest Gump',
        'made':1994,
        'director':'Robert Zemeckis',
        'soundtrack':'Multiple',
        'starring':'Tom Hanks',

        'fun_fact':'''The house used in Forrest Gump
                    is the same house used in The Patriot (2000).
                    Some paneling was changed forinterior shots
                    in the latter film.'''}

```

100% z Lekce 4

Osnova

DALŠÍ LEKCE