Úsnova

Lesson Overview

Hi there! I hope you're doing well. What about the project from the last lesson? Have you done it yet? If not, let us know, we will help you. In last lesson you have learned while loops we can dive in lesson 5.

In this lesson we will discuss:

- **ranges** - addition to our collection data types,
- **for loop** - this is a new type of loop and will greatly expand the possibilities of working with loops :)

REVIEW EXERCISES

Prepend

Write a Python program, called `prepend.py`, that will accept a string as input. The script should prepend string "However," to the front of the input string.

The word that was originally the first in the sentence, has to be converted to lower case. If the input string already begins with "However, " then return the string unchanged.

Example of running the program:

```
Osnova
any sentence: My name is Bob
is Bob.
```

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

```
ce   input('Please write here any sentence: ')
```

Osnova

```
if sentence[0:9] == 'However, ':
    print(sentence)
else:
    print('However, ' + sentence[0].lower() + sentence[1:])
```

Find a word

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / REVIEW EXERCISES / FIND A WORD

Create a script that will ask for two inputs:

- String to be searched in
- Word we want to search for in the previous input

The program should return the index at which the input has been encountered. Otherwise -1

```
~/PythonBeginner/Lesson3 $ python find_word.py
Please enter the sentence to be searched: I do not want to work today.
Please enter the word you would like to search for: want
I have found this string at the index: 9
```

Online Python Editor

```
1
```

100% z Lekce 6

Osnova

spustit kód

Code Solution

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

Click to see our solution

```
Phrase = input('Please enter the sentence to be searched: ')

OurWord = input('Please enter the word you would like to search for: ')
print('I have found this string at the index: ' + str(Phrase.find(OurWord)))
```

Find the bug #1

100% z Lekce 6

There are [illegible] errors in the code below, try to find them and repair them. You will know, that [illegible] the program prints the sum of the numbers listed in the list nums.

Osnova

```
1  nums = ['421',867,65, '93', 45, 82]
2  i = 0
3  result = 0
4  while i < count(nums):
5      result = result + nums[i]
6
7  print(result)
```

spustit kód

Code Solution

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

Click to see our solution

```
        ['21',867,65, '93', 45, 82]
```

Osnova

```
result = 0

while i < len(nums):
    result = result + int(nums[i])
    i = i + 1

print(str(result))
```

Find the bug #2

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / REVIEW EXERCISES / FIND THE BUG #2

There are some bugs in the code below, try to find them and repair them. You will know, that the bugs are fixed, when the program values `True` or `False` based on the user input.

```
letter = input('Please enter a letter')
vowels = list('aeiouy')
if type(letter)==str and len(something)==1:
print(letter in vowels)
```

100% z Lekce 6

Osnova

Code Solution

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

Click to see our solution

Here's our solution of your `find_the_bug2.py` program:

```
1 letter = input('Please enter a letter: ')
2 vowels = list('aeiouy')
3
4 if letter.isalpha() and len(letter)==1:
5     print(letter in vowels)
```

Osnova

ONSITE EXERCISES

Today

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / ONSITE EXERCISES / TODAY

Today our main goal will be to introduce new type of loop - **for loop**. We will understand:

1. How does it work
2. When to use it instead of while loop
3. What are some tips and tricks using for loops

Along the way we will meet so called **dictionary views** and new data type - **range**.

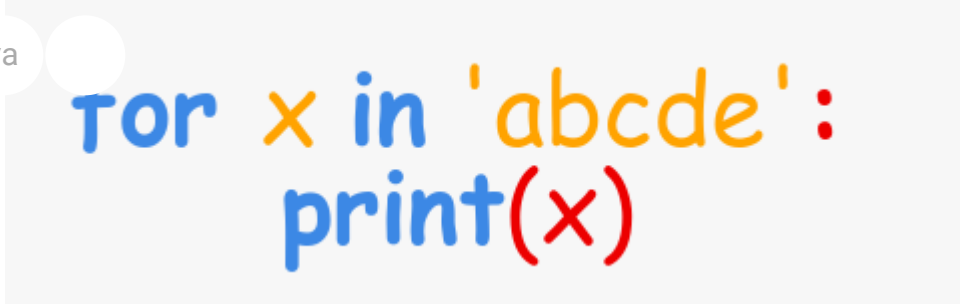
For Loop Syntax

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / ONSITE EXERCISES / FOR LOOP SYNTAX

Bellow we can see the **general syntax** of for loop, followed by an **example**. To visualize for loop we can use Python Tutor.

100% z Lekce 6

Osnova



Using For Loop

We already know while loop. But **when to use while and when for loop**?

- For loops are best used on finite collections of data.
- While loops are best used in scenarios, when we do not know in advance, when the loop will be terminated.

So let's say we have the following list: `names = ['Jakub', 'Jana', 'Petr', 'Klara']`. What we would like to do with it is to:

1. print all the names in the list,
2. print only names beginning with the letter "J"
3. and calculate the total number of letters that these names contain.

```
1  names = ['Jakub', 'Jana', 'Petr', 'Klara']
```

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. print all the names in the list:

```
1  names = ['Jakub', 'Jana', 'Petr', 'Klara']
2
3  for name in names:
4      print(name)
```

2. print only names beginning with the letter "J":

```
1  names = ['Jakub', 'Jana', 'Petr', 'Klara']
2
3  for name in names:
4      if name.startswith('J'):
5          print(name)
```

100% z Lekce 6

```
= ['Jakub', 'Jana', 'Petr', 'Klara']
```

Osnova

```
total = 0
for name in names:
    total = total + len(name)

print(total)
```

What can be iterated using for loop?


```
# Numbers?
for item in 1234567:
    print(item)
```

Output

```
# Strings?
for item in 'abcdefghi':
    print(item)
```

Output

```
a
b
c
d
```

Osnova

```
i
```

```
# Tuples?
for item in (1,2,3,4,5):
    print(item)
```

Output

```
1
2
3
4
5
```

```
# Sets?
for item in {1,2,3,4,5}:
    print(item)
```

Output

```
1
2
3
4
5
```

100% z Lekce 6

```
2 for item in {'name': 'Bob', 'age': 23, 'job':'plumber'}:
3     print(item)
```

Osnova

Output

```
name
age
job
```

```
1 # Ranges??
2 for item in range(10):
3     print(item)
```

Output

```
???
```

Range data type

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / ONSITE EXERCISES / RANGE DATA TYPE

We could meet ranges in elementary school's math classes. For example, we were checking whether a number belonged to an interval of numbers. **In Python, range:**

- range is a standalone **data type**,

100% z Lekce 6

- the difference between these values is by default 1 (but we can change that),
- the direction of ordering can be changed (we can generate ascending or descending sequences of numeric values).

Osnova

We can create all kinds of ranges:

```
# Numbers 0,1,2,3,4
r = range(5)

# Numbers 2,3,4,5
r = range(2,6)

# Numbers 2,4,6,8
r = range(2,9,2)

# Numbers 4,3,2,1
r = range(4,0,-1)

# Even numbers 8,6,4,2??
r = range(8,1,-2)
```

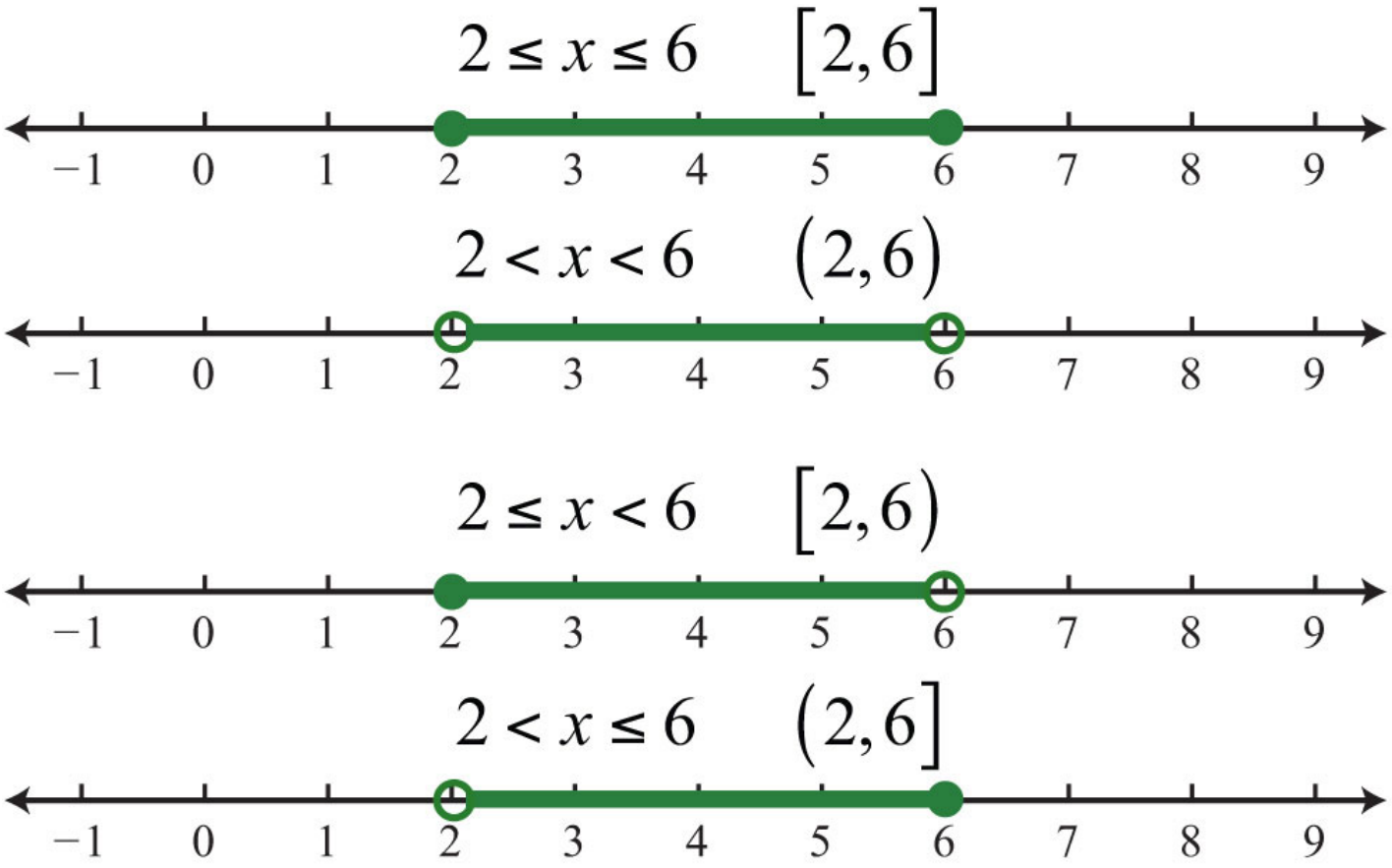
spustit kód

Therefore, we can defined 3 parameters when creating a range object:

100% z Lekce 6

Osnova

- What if Step > 0 and Start > Stop?
- What if Step < 0 and Start < Stop?



For loop vs Range

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / ONSITE EXERCISES / FOR LOOP VS RANGE

Most basic example:

```
1 for num in range(5):
2     print(num)
```

100% z Lekce 6

Output

Osnova

```
0
1
2
3
4
```

We usually use `range()` in for loop, to generate numbers later used as **indexes**:

```
1 names = ['Helmut', 'Helga', 'Harold', 'Hammet', 'Hetfield']
2
3 for index in range(len(names)):
4     print(names[index])
```

Output

```
Helmut
Helga
Harold
Hammet
Hetfield
```

However, and you know what's about to come, the above example is **not very Pythonic**. This is plausible for other programming languages, but in Python we do not need to generate index numbers - for that we have while loop. Therefore we can simply write the for loop as follows:

```
1 names = ['Helmut', 'Helga', 'Harold', 'Hammet', 'Hetfield']
2
3 for name in names:
```

Osnova

```
Helmut
Helga
Harold
Hammet
Hetfield
```

Ranges can become handy, when we need to work with **every nth item** in the sequence:

```
1 names = ['Helmut', 'Helga', 'Harold', 'Hammet', 'Hetfield']
2
3 for i in range(0,len(names),2):
4     print(i,names[i])
```

Output

```
0 Helmut
2 Harold
4 Hetfield
```

If we need to work with both the **sequence item and its index**, we use a built-in function `enumerate()` :

```
1 numbers = [321,45,32,12,54]
2 for index, number in enumerate(numbers):
3     if index % 2 == 0:
4         print(number, ':', number**2)
```

100% z Lekce 6

Output

Osnova

```
321 : 103041
32 : 1024
54 : 2916
```

Searching for words

Create a program that will ask the user for any word and will then begin to search for the word in the text we have attached bellow. If the word is found, the program should print the number representing **word's position** (order) in the given string:

```
text = '''
Situated about 10 miles west of Kemmerer,
Fossil Butte is a ruggedly impressive
topographic feature that rises sharply
some 1000 feet above Twin Creek Valley
to an elevation of more than 7500 feet
above sea level. The butte is located just
north of US 30N and the Union Pacific Railroad,
which traverse the valley.'''
```

Example:

```
SEARCH WORD: miles
POSITION: 4
```

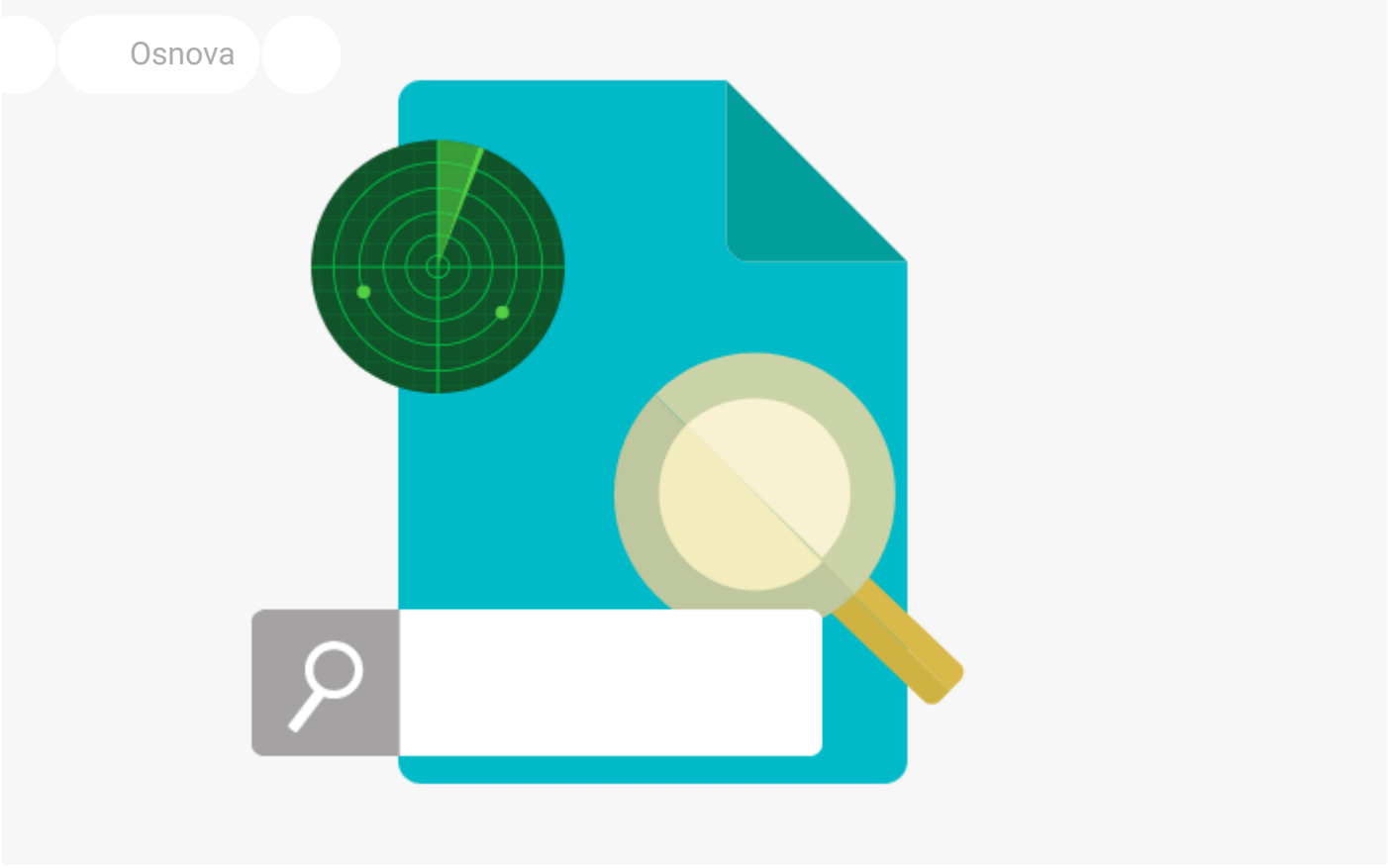
```
SEARCH WORD: milesss
NO SUCH WORD
```

1. [illegible] program
Osnova
2. Adjust the program for cases when there are **multiples occurrences of a given string.**

```
text = '''
Situated about 10 miles west of Kemmerer,
Fossil Butte is a ruggedly impressive
topographic feature that rises sharply
some 1000 feet above Twin Creek Valley
to an elevation of more than 7500 feet
above sea level. The butte is located just
north of US 30N and the Union Pacific Railroad,
which traverse the valley.'''
```

spustit kód

100% z Lekce 6



Code Solution

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

Click to see our solution

1. the program

```
1  target = input('SEARCH WORD: ')
2
3  text = '''
4  Situated about 10 miles west of Kemmerer,
5  Fossil Butte is a ruggedly impressive
6  topographic feature that rises sharply
7  some 1000 feet above Twin Creek Valley
```

```
    of 30N and the Union Pacific Railroad,
Osnova  se the valley.'''

words = text.split()
found = False
position = 0

for i,word in enumerate(words):
    word = word.strip(';,._/:')
    if word == target:
        found = True
        position = i + 1

if found:
    print('POSITION: ' + str(position))
else:
    print('NO SUCH WORD')
```

2. multiple occurrences of a string

If we wanted to stop our search with the word's first occurrence we would need some kind of break out mechanism. With for loops as well as with while loops we can use the `break` keyword to immediately terminate the loop.

So it could look like this:

```
words = text.split()
position = -1
for i,word in enumerate(words):
    word = word.strip(';,._/:')
    if word == target:
        position = i + 1
        print('POSITION: ' + str(position))
        break # <---------------------------- BREAK

if position == -1:
    print('NO SUCH WORD')
```

Osnova

```
    ... = ...t.split()
... in enumerate(words):
    word = word.strip(';,._/:')
    if word == target:
        position = i + 1
        print('POSITION: ' + str(position))
        break # <----------------------------- BREAK
else: # <----------------------------------- ELSE
    print('NO SUCH WORD')
```

For loop vs Dictionaries

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / ONSITE EXERCISES / FOR LOOP VS DICTIONARIES

We have the following dictionary: `employee = {'f_name':'John', '_name':'Smith','age': 23, 'job':'plumber'}`

When looping, we will want to access:

Keys

If we wanted to loop over dictionary keys, we could just use the name of dictionary:

```
for k  in employee:
    print(k)
```

...or we could use the method `.keys()`

```
for k  in employee.keys():
    print(k)
```

Output

100% z Lekce 6

```
f
    Osnova
age
job
```

Values

In the case of values, and as we will see also in the case of pairs key:value, we must use a method:

```
1 for v in employee.values():
2     print(v)
```

Output

```
John
Smith
23
plumber
```

Both keys and values

```
1 for k,v in employee.items():
2     print(k,v)
```

Output

```
f_name John
```

Osnova

Nested Loops

We have data parsed out of the csv file. There are rows and columns and we need to iterate over values in the first two rows. How would we print each value on a separate line?

```
data = [['ID','Name', 'Price', 'Amount', 'Supplier'],
        ['321','Ibalgin', 40.50, 2841, 'Zentiva'],
        ['534','Paralen', 19.90, 3229, 'Zentiva'],
        ['327','Smecta', 68.00, 2709, 'Sipsen'],
        ['242','Uniclophen', 76.00, 476, 'UNIMED']]
```

Code Tasks

1. What is the total price of our inventory?
2. What is the total amount of items in our warehouse?
3. How many items do we have from the company called Zentiva?
4. Let's transfer the above table into a dictionary of dictionaries. How would we do that?

```
data = [['ID','Name', 'Price', 'Amount', 'Supplier'],
        ['321','Ibalgin', 40.50, 2841, 'Zentiva'],
        ['534','Paralen', 19.90, 3229, 'Zentiva'],
        ['327','Smecta', 68.00, 2709, 'Sipsen'],
        ['242','Uniclophen', 76.00, 476, 'UNIMED']]
```

Osnova

spustit kód

Code Solution

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

Click to see our solution

```
data = [['ID','Name', 'Price', 'Amount', 'Supplier'],
        ['321','Ibalgin', 40.50, 2841, 'Zentiva'],
        ['534','Paralen', 19.90, 3229, 'Zentiva'],
        ['327','Smecta', 68.00, 2709, 'Sipsen'],
        ['242','Uniclophen', 76.00, 476, 'UNIMED']]
```

1. Total price of our inventory

```
total_price = 0

for row in data[1:]:
    total_price = total_price + row[2] * row[3]

```

? Osnova r unt of items in our warehouse

```
num_items = 0

for row in data[1:]:
    num_items = num_items + row[3]

print(num_items)
```

3. Items from Zentiva

```
num_items = 0

for row in data[1:]:
    if row[4] == 'Zentiva':
        num_items = num_items + row[3]

print(num_items)
```

4. Table into a dictionary of dictionaries.

```
d = {}

header = data[0][1:]

for row in data[1:]:
    id = row[0]
    d[id] = {}

    for i,item in enumerate(row[1:]):
        key = header[i]
        d[id].update({key: item})

print(d)
```

Osnova

```
    ['ID','Name', 'Price', 'Amount', 'Supplier'],
    ['21','Ibalgin', 40.50, 2841, 'Zentiva'],
    ['534','Paralen', 19.90, 3229, 'Zentiva'],
    ['327','Smecta', 68.00, 2709, 'Sipsen'],
    ['242','Uniclophen', 76.00, 476, 'UNIMED']]

# 1. Total price of our inventory
total_price = 0
for row in data[1:]:
    total_price = total_price + row[2] * row[3]
print("The total price of our inventory: ", total_price)

# 2. Total amount of items in our warehouse
num_items = 0
for row in data[1:]:
    num_items = num_items + row[3]
print("The total amount of items in our warehouse: ", num_items)

# 3. Items from Zentiva
num_items = 0
for row in data[1:]:
    if row[4] == 'Zentiva':
        num_items = num_items + row[3]
print("The amount of Zentiva produts: ", num_items)

# 4. Table into a dictionary of dictionaries.
d = {}

header = data[0][1:]

for row in data[1:]:
    id = row[0]
    d[id] = {}

    for i,item in enumerate(row[1:]):
        key = header[i]
        d[id].update({key: item})
print("Database dictionary: ", d)
```

100% z Lekce 6

Osnova

We would like to have a program that will prompt for one number NUM and then generates a chess board of size NUM x NUM

Example of board with size 10 x 10:

```
# # # # #
 # # # # #
# # # # #
 # # # # #
# # # # #
 # # # # #
# # # # #
 # # # # #
# # # # #
 # # # # #
```

1

spustit kód

Osnova

Code Solution

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

Click to see our solution

```
size = 10
sym = ['#',' ']
desk = []

for r,row in enumerate(range(size)):
    line = []
    for c,cell in enumerate(range(size)):
        line.append(sym[(r+c)%len(sym)])
    desk.append(''.join(line))

print('\n'.join(desk))
```

Playing with tables

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / ONSITE EXERCISES / PLAYING WITH TABLES

Create a program that will allow the user to perform the following actions:

100% z Lekce 6

3. Insert a column into the given table
4. Update a cell in the table
5. Calculate total for a selected column
6. Calculate total for a selected row
7. Print the whole table or selected rows
8. Export the table into a dictionary

Osnova

The program should run in a loop until the user tells it to terminate. We recommend you do this exercise in your computer as our Online Python Editor might be unsuitable for it ;).

An example of program running:

```
----------------------------------------
What do we want to do? Enter the number
    1-Create table | 2-Insert new row | 3-Insert new column |
    4-Update a cell | 5-Column Total | 6-Row Total |
    7-Print Table | 8-Export do List of Dicts |
OPTION: 1
----------------------------------------
Enter header names separated by comma (e.g. Name,Age,Email)
Name,Age,Email
----------------------------------------
Press enter to repeat or q to quit:
----------------------------------------
What do we want to do? Enter the number
    1-Create table | 2-Insert new row | 3-Insert new column |
    4-Update a cell | 5-Column Total | 6-Row Total |
    7-Print Table | 8-Export do List of Dicts |
OPTION: 2
----------------------------------------
ROW NUMBER: 2
Enter the comma separated values (e.g. Bob,23,bob@abc.com):
Ann,43,ann@abc.com
----------------------------------------
```

100% z Lekce 6

```
What do we want to do? Enter the number
    1-Create table | 2-Insert new row | 3-Insert new column |
    4-Update a cell | 5-Column Total | 6-Row Total |
    7-Print Table | 8-Export do List of Dicts |
OPTION: 2
---------------------------------------
ROW NUMBER: 3
Enter the comma separated values (e.g. Bob,23,bob@abc.com):
Bob,23,bob@abc.com
---------------------------------------
Press enter to repeat or q to quit:
---------------------------------------
What do we want to do? Enter the number
    1-Create table | 2-Insert new row | 3-Insert new column |
    4-Update a cell | 5-Column Total | 6-Row Total |
    7-Print Table | 8-Export do List of Dicts |
OPTION: 3
---------------------------------------
COLUMN NAME: Job
ENTER VALUE: Ann|43|ann@abc.com|Manager
ENTER VALUE: Bob|23|bob@abc.com|Plumber
---------------------------------------
Press enter to repeat or q to quit:
---------------------------------------
What do we want to do? Enter the number
    1-Create table | 2-Insert new row | 3-Insert new column |
    4-Update a cell | 5-Column Total | 6-Row Total |
    7-Print Table | 8-Export do List of Dicts |
OPTION: 7
---------------------------------------
FROM ROW: 0
TILL ROW: 200
Skip rows? [y/n]: n
---------------------------------------
Name | Age | Email | Job
```

```
Osnova
                    peat or q to quit: q
```

Code Solution

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

Click to see our solution

```
# our training table - in case we do not want to insert new
# rows and columns at the beginning
table = [['Name','Age','Email'],
         ['bob', '23', 'bob@abc.com'],
         ['ann', '25', 'ann@abc.com'],
         ['fred', '43', 'fred@abc.com']]

while True:

    print('-'*40)
    print('What do we want to do? Enter the number')
    print('''
    1-Create table | 2-Insert new row | 3-Insert new column |
    4-Update a cell | 5-Column Total | 6-Row Total |
    7-Print Table | 8-Export do List of Dicts |
    ''')
    selection = int(input('OPTION: '))
    print('-'*40)

    # user wants to create a table
    if selection == 1:
        table = []
        print('Enter header names separated by comma (e.g.
Name,Age,Email)')
        header = input()
        table.append(header.split(','))
```

Osnova

```
    elif selection == 2:
        where = int(input('ROW NUMBER: '))
        print('Enter the comma separated values (e.g. Bob,23,bob@abc.com): ')
        row = input().split(',')
        if 0 < where < len(table):
            table.insert(where, row)
        elif where >= len(table):
            table.append(row)

    # user wants to insert a new column
    elif selection == 3:
        col_name = input('COLUMN NAME: ')
        table[0].append(col_name)

        for i,row in enumerate(table[1:]):
            val = input('ENTER VALUE: ' + '|'.join(row) + '|')
            table[i+1].append(val)

    # user wants to update a cell
    elif selection == 4:
        row_num = int(input('ROW NUMBER: '))
        col_num = int(input('COLUMN NUMBER: '))
        if row_num in range(len(table)) and \
           col_num in range(len(table[0])):
            val = input('VALUE (current value: ' + table[row_num][col_num] + '):')
            table[row_num][col_num] = val
        else:
            print('No such row or column')

    # user wants to calculate the total sum of numbers in a column
    elif selection == 5:
        total = 0
        print('Select the column name')
        print('|'.join(table[0]))
        col_name = input()
        if col_name in table[0]:
```

100% z Lekce 6

Osnova

```
            for row in table:
                if row[col_num].isnumeric():
                    total += int(row[col_num])
            print('COLUMN TOTAL FOR ' + col_name + ': ' +str(total))
        else:
            print('No column named ' + col_name)

    # user wants to calculate the total sum of numbers in a row
    elif selection == 6:
        total = 0
        print('Select the row number in range 1 to ' +
str(len(table)-1) + '(excluding header)')
        row_num = int(input())

        if 0 < row_num < len(table):
            for cell in table[row_num]:
                if cell.isnumeric():
                    total += int(cell)
            print('ROW TOTAL FOR ROW #' + str(row_num) + ': ' +
str(total))

    # user wants to print table contents - can select which rows
    elif selection == 7:
        start = int(input('FROM ROW: '))
        end = int(input('TILL ROW: '))
        skip=1
        if input('Skip rows? [y/n]: ')=='y':
            skip = input('How many? (2 for every second etc.): ')

        start = 0 if start not in range(len(table)) else start
        end = len(table) if end>len(table) else end

        print('-'*40)
        for num in range(start,end,skip):
            row = table[num]
            print(' | '.join(row))

    # user wants to export the table (list of lists) into a
```

100% z Lekce 6

Osnova

```
            data = []
            for row in table[1:]:
                d = {}
                for i,header in enumerate(table[0]):
                    d[header] = row[i]
                data.append(d)

            print(data)


    print('-'*40)
    repeat = input('Press enter to repeat or q to quit: ')

    if repeat == 'q':
        break
```

100% z Lekce 6

Osnova

principle

Range is **immutable** data type similarly to string, bytes, tuple. The only way, we can create a range object is by using the `range()` constructor. Range belong among sequence data types (str, tuple, list).

Range Arguments

There are 3 ways, how a range can be created:

1. range(stop)
2. range(start, stop)
3. range(start,stop,step)

The `range()` constructor arguments have to be integers:

- **Stop** - is the number at which the generation of integer sequence should stop. The stop number will not be generated. Range begins to generate the sequence at int 0 if no start argument specified. That way range(10) is a sequence of numbers from 0 to 9 including (but excluding 10).
- **Start** - if specified, also the stop argument has to be included. Range sequence does not have to begin at 0, but can begin at any integer specified by the Start argument.
- **Step** - allows to generate every nth integer between Start and Stop. For example we can generate every 2nd item in the sequence btw. 0 and 10 by writing range(0,10,2). If step is a negative integer, the sequence is generated backwards: range(10,0,-1) generates numbers from 10 to 1.

```
>>> r1 = range(10)
>> r1
range(0, 10)
```

If we want to get printed the numbers generated by `range()` constructor, we have to input the

Creating range(stop)

Osnova

```
>>> tuple(range(10))
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

Creating range(start, stop)

```
>>> tuple(range(5,10))
(5, 6, 7, 8, 9)
```

Creating range(start, stop, step)

```
>>> tuple(range(5,10,2))
(5, 7, 9)
```

Principle

Range generates the numbers "on demand" and not at once. If we wanted to generate them all at once we can use `list()` or `tuple()` constructors. That is why we will not see the whole sequence if we pass range object to the `print()` function. Why this is so is discussed in the module covering the concept of Generators. Meanwhile we should be aware of the fact that printing bare `range()` object will not print the sequence of items it generates.

Experimenting with range() arguments

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / RANGE / EXPERIMENTING WITH RANGE() ARGUMENTS

Before you run the code bellow, try to guess the outputs. These examples will help you understand the nature of this data type further:

```
1  # 1. start > stop
2  print("1.", list(range(5,1)))
3
4  # 2. stop == 0
5  print("2.", list(range(0)))
```

100% z Lekce 6

```
9
:    Osnova :    :op and step < 0
11 .   ,  .',  .st(range(10,0,-1)))
12
13 # 5. step > 2
14 print("5.", list(range(0,10,3)))
15
16 # 6. step < -1
17 print("6.", list(range(10,0,-2)))
18
19 # 7. step == 0
20 print("7.", list(range(1,10,0)))
```

spustit znovu

```
Traceback (most recent call last):
  module __main__ line 19, in runBrythonCode
    _ = exec(script, {'__name__':'__main__'})
  module <module> line 20
    print("7.", list(range(1,10,0)))
```

Number of range() arguments matters

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / RANGE / NUMBER OF RANGE() ARGUMENTS MATTERS

Range determines, what argument it receives based on the number of arguments provided:

Case	Example	Input Significance	Result
1 argument	`range(5)`	always STOP	0,1,2,3,4
2 arguments	`range(3,5)`	always START & STOP	3,4
3 arguments	`range(3,12,2)`	always START & STOP & STEP	3,5,7,9,11

100% z Lekce 6

1. Print every element in the range 0 - 19

Osnova

2. Print every element in the range 15 - 19
3. Print every third element in the range 10 - 19
4. Create a list `nums` of every number divisible by 5 in range btw. 1 and 100

```
1  # 1.
2
3  # 2.
4
5  # 3.
6
7  # 4.
```

spustit kód

Code Solution

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

100% z Lekce 6

Osnova

```
print(list(range(20)))

# 2.
print(list(range(15,20)))

# 3.
print(list(range(10,20,3)))

# 4.
nums = list(range(5,101,5))
print(nums)
```

Range operations - Indexing

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / RANGE / RANGE OPERATIONS - INDEXING

Just like in the case of string, list, or tuple, range also supports the operation of indexing. To revise indexing you can go back to Lesson 1 - Indexing.

Examples for range:

```
>>>range(10)[0]
0
>>>range(10)[5]
5
```

Range operations - Slicing & Striding

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / RANGE / RANGE OPERATIONS - SLICING & STRIDING

100% z Lekce 6

```
>>> [-]
Osnova
```

Or for striding, we can cut out every nth item from a range:

```
>>> list(range(10)[::3])
[0, 3, 6, 9]
```

Range operations - Membership

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / RANGE / RANGE OPERATIONS - MEMBERSHIP

We can test whether an integer is present in a range we use the operator in:

```
>>> 6 in range(10)
True
```

Range operations - Conversion

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / RANGE / RANGE OPERATIONS - CONVERSION

Range is specific in that it accepts only integers as arguments and so does not really allow for converting from another data type into range:

```
range([1,2,3])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'list' object cannot be interpreted as an integer
```

Operations NOT supported

T to ut of the following code:

Osnova

1. Concatenation: `range(15,20) + range(20,25)`

Output

Ranges do not support concatenation operations:

```
>>> range(15,20) + range(20,25))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'range' and 'range'
```

2. Repetition: `range(5) * 3`

Output

Nor do they support repetition:

```
>>> range(5) * 3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for *: 'range' and 'int'
```

Range Usage

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / RANGE / RANGE USAGE

100% z Lekce 6

Osnova

...re ...st often used to generate sequences of integers. Such sequences of ...ge... ...s... ...n `for loop` for example.

```
1 for number in range(10):
2     print(number)
```

spustit znovu

```
0
1
2
3
4
```

100% z Lekce 6

Osnova

FOR LOOP

Syntax

For loop allows to step through sequence of objects. For loops work only on so called iterable Python objects - for example collections like tuples, lists, dictionaries, strings, ranges etc.

```
1 for item in iterable:
2     statements
```

Loop header consists of:

- `for` keyword
- variable name (in our example called num)
- `in` keyword
- iterable object - object which we walk through (range(10))
- colon `:`

Code Task

100% z Lekce 6

```
2 print(num)
```

Osnova

spustit kód

Loop principle

- For loop steps through items in item order (in case of sequences) or in arbitrary order (sets and dictionaries).
- To step through means that the reference to one item from iterable per cycle is stored in the header variable (in our example below called `char`). This variable can be called whatever we want.

- The iterator variable inside the header variable (`char`) can then be used inside the loop's body (e.g. to convert the letter into upper or lower case).

Osnova

- There are only as many repetitions as many items are there inside the iterable object in the loop header - we will only step through each letter in the string `'Python'` and then the loop will automatically terminate.

```
converted_word = ''
for char in 'Python':
    if char.isupper():
        converted_word = converted_word + char.lower()
    else:
        converted_word = converted_word + char.upper()

print(converted_word)
```

Output:

```
pYTHON
```

You can use Python Tutor to visualize the above code execution. Just paste it inside the edit pane and click "Visualize Execution".

The example above can be translated as follows: `for` each character (temporarily stored in the variable `char`) inside the string `'Python'` convert the character to lower or upper case.

Iterating over range() object

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / FOR LOOP / ITERATING OVER RANGE() OBJECT

Range object represents a sequence of integers. That allows us to iterate over it and use the number for example to retrieve by index items from another sequence:

```
>>> my_str = 'Hello World'
```

100% z Lekce 6

Osnova

```
e
l
l
o
W
o
r
l
d
```

In the code above, we generate range from 0 up to the last index number of `my_str` using the `len()` function. However, `range(len(object))` is **not very Pythonic!** It would be better if we wrote the above `for loop` this way, without using range:

```
>>> for i in my_str:
...     print(i)
...
H
e
l
l
o
W
o
r
l
d
```

Python knows, how to step through the collection of data in `my_str`, therefore we do not have to provide it with index numbers.

Iterating over other sequences & sets

100% z Lekce 6

Osnova

...iterable... ...list... ...uples, sets, ranges or strings, the for loop header should contain only 1 ...ab... ...le... ...ns in:

```
1 for number in range(10):
2     print(number,number**2)
```

If we provide more variables (`number` , `i`) than necessary, error will be raised:

```
1 for number,i in range(10):
2     print(number,number**2)
```

Output:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not iterable
```

Trick

However, tuple of 2-item tuples can be iterated this way - by providing 2 header variables:

```
>>> data = (('Age',43),('Name','John'),('Surname','Smith'))
>>> for category, value in data:
...     print(category,':',value)
...
Age : 43
Name : John
Surname : Smith
```

You can use Python Tutor to visualize the above code execution.

The use case of the above 2-item tuples would be a collection of (x,y) coordinates referring to a position of a pixel in an image. This way, we would be able to collect both x and y coordinates in one cycle.

Code task

```
1 data = (('Age',43,True),('Name','John',True),('Surname','Smith',False))
```

Osnova

spustit kód

Click to see our solution

```
1 data = (('Age',43,True),('Name','John',True),
  ('Surname','Smith',False))
2
3 for a,b,c in data:
4     print(a,b,c)
```

100% z Lekce 6

Osnova

dictionaries

We have a dictionary: `employees = {'First_Name': 'John', 'Last_Name': 'Smith', 'Age':43}`

We will now use our knowledge of so called dictionary views to iterate only over keys, values or both.

Only Keys

We want to use the for loop to **go over dictionary keys** and print them to the terminal. We do this as follows:

```
>>> for category in employees:
...     print(category)
...
Age
First_Name
Last_Name
```

That means, the looping is done on dictionary keys by default.

Only Values

After we know how to loop over dictionary keys, we should learn how to go over the **dictionary values**. For that purpose we have to use the `dict.values()` method in the for loop header:

```
>>> for value in employees.values():
...     print(value)
...
43
John
Smith
```

100% z Lekce 6

Finally, we want to extract both key and value. In that case we have to:

- provide **2 variables in the for loop header** to store the key and value inside
- use `dict.items()` method on the iterated dictionary

```
>>> for key,value in employees.items():
...     print(key,value)
...
Age 43
First_Name John
Last_Name Smith
```

Iterating backwards

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / FOR LOOP / ITERATING BACKWARDS

It is pretty clear, how to iterate over an iterable forwards. But how to do that backwards?

This may be the first thought how to do it:

```
>>> my_string = 'Python'
>>> for index in range(len(my_string)-1,-1,-1):
>>>     char = my_string[index]
>>>     print(char, end='')
nohtyP
```

But that is **not Pythonic.** Instead, we should use the built-in `reversed` function:

```
>>> my_string = 'Python'
>>> for char in reversed(my_string):
>>>     print(char, end='')
nohtyP
```

It is more readable, saves space and typing and it is faster because the `reversed()` function is optimized for this kind of job. The `reversed()` function orders the original collection

Osnova

note, that the `print()` function now contains a new input - `end=''` . h letter will be **printed on the same line** and not on a new one.

Code Task

Write a loop that will print the string `'New York'` reversed on the same line.

```
1
```

spustit kód

Code Solution

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

100% z Lekce 6

Click to see our solution

Osnova

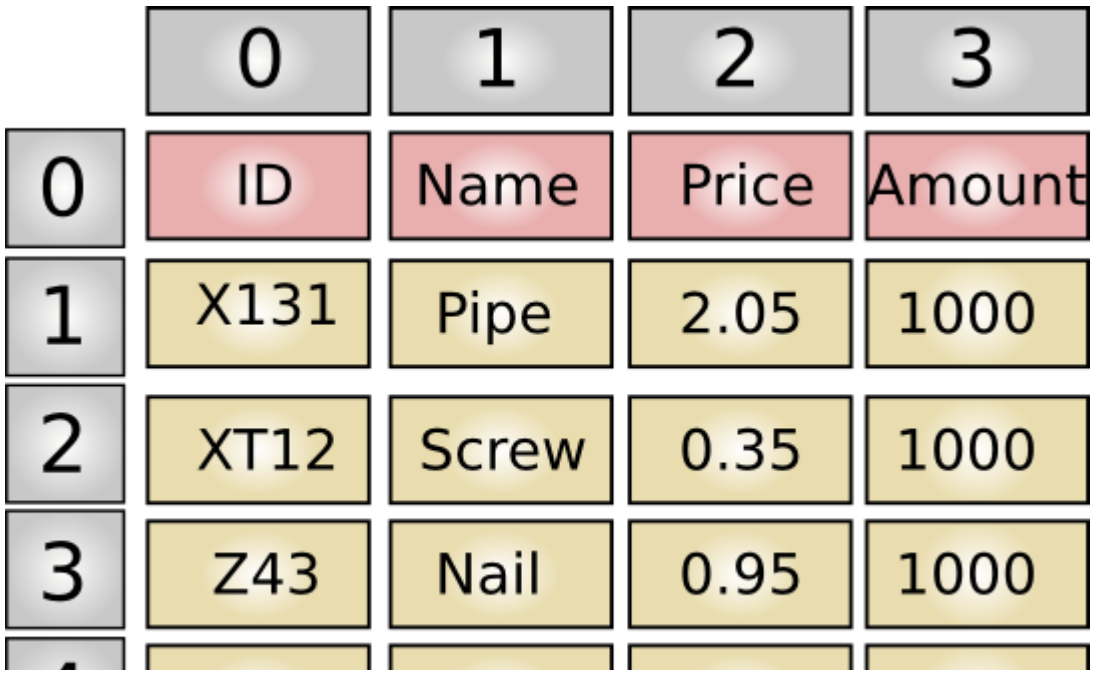
```
my_str = 'New York'

for char in reversed(my_str):
    print(char, end='')
```

Nested Loops

Let's imagine we have a table in an Excel file. Each table has rows and columns. To refer to any of the table cells, we need to use the number of the row and the number of the column it is located in. These two numbers can be called coordinates.

If we wanted (and this is actually done so), we could represent any table as a list of lists. Let's take the following item table and convert it into a list of lists:



	0	1	2	3
0	ID	Name	Price	Amount
1	X131	Pipe	2.05	1000
2	XT12	Screw	0.35	1000
3	Z43	Nail	0.95	1000

Osnova

represented in a list of lists (each row represented as a separate list). In in each list, we will need to use two for loops, nested within each other. This code will print each item of the above code:

```
table = [['ID','NAME','PRICE','AMOUNT'],
         ['X131', 'Pipe', 2.05, 1000],
         ['XT12', 'Screw', 0.35, 1000],
         ['Z43', 'Nail', 0.95, 1000],
         ['P843', 'Tape', 1.39, 1000]
]

for row in table:
    for item in row:
        print(item)
```

spustit znovu

```
ID
NAME
PRICE
AMOUNT
X131
```

Continue & Break

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / FOR LOOP / CONTINUE & BREAK

Loops do not have to run till the end - this can be controlled by the use of `continue` and

100% z Lekce 6

- [illegible] program execution back to the top of the loop (**continue**) or
- just jump out of the loop immediately and execute the code that follows the loop block (**break**)

Continue and break statements can be use **in both** `while` loop as well as `for` loop. Let's take a more detailed look at them.

Continue Statement

Once encountered in a loop, continue statement causes the loop execution to jump to the header of the loop:

- in for loop Python goes to the next item in the iterated collection
- in while loop Python returns to the loop header and evaluates its test

Below we have 2 equivalent examples using for and while loops:

Continue in for loop

```
for number in range(20):
    if number % 2 == 0:
        continue
    print(number)
```

Continue in while loop

```
number = -1

while number < 20:
    number += 1
    if number % 2 == 0:
```

While loop requires explicit adjustment of the loop variable, meanwhile in the for loop Python does it. For loop code looks also much cleaner.

Code Task

Guess what would happen if we did not wrap the continue statement inside a conditional statement?

So the **for loop** would look like this:

```
for number in range(20):
    continue
    print(number)
```

Click to see our solution

If used in for loop, no code after the continue statement will be executed - in our case, no number is printed to the terminal. **The loop will eventually terminate**

As for **while loop**:

```
number = -1
while number < 20:
    continue
    number += 1
    print(number)
```

Click to see our solution

If the loop variable `number` is changed after the continue statement, **then the loop will never**

Or [illegible] of [illegible]e below, the loop below will terminate, but nothing will be printed to the [illegible]

Osnova

```
number = -1
while number < 20:
    number += 1
    continue
    print(number)
```

Break Statement

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / FOR LOOP / BREAK STATEMENT

Break statement **terminates the loop** and moves the program execution to the code following the loop body. Similarly to continue statement, once encountered inside a loop, the lines following `break` will not get executed inside the loop.

In contrast to `continue`, the code execution does not return back to the loop header, but **exits the loop altogether** - the program execution "breaks out" of the loop. Again, `break` should be wrapped inside a conditional statement.

Break in for loop

```
for number in range(100):

    if 40 < number < 60:
        break
    print(number)

print('END')
```

Break in while loop

100% z Lekce 6

```
    while number < 100:
        if 40 < number < 60:
            break
        print(number)

        number += 1

print('END')
```

Only numbers 0 to 40 will be printed by the above loops. Once the variable `number` will be equal to 41, the condition `if 40 < number < 60:` will be true and `break` will be executed followed by `print('END')`

Else Block

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / FOR LOOP / ELSE BLOCK

Both `for` and `while` loop statements can have `else` block appended at its end. `else` keyword's indentation level has to be the same as the one of the loop header. Code inside the else block gets evaluated if the loop it is part of, have terminated normally - that means, without encountering **break statement.**

Else and for loop

If the letters have not been found, the `break` will never be reached, and `print('There are no letters "xz" inside the string')` statement will be executed:

```
checked_string = 'Python Loop Statements'
for letter in checked_string:
    if letter in 'xz':
        break
else:
    print('There are no letters "xz" inside the string')
```

```
Osnova ers "xz" inside the string
```

If letters 'x' or 'z' were encountered in the searched string, the statement `print('We have found it')` will be executed:

```
checked_string = 'Python Loop x Statements'
for letter in checked_string:
    if letter in 'xz':
        print('We have found it')
        break
else:
    print('There are no letters "xz" inside the string')
```

Output:

```
We have found it
```

Else and while loop

The `else:` block works also with `while` loops:

```
checked_string = 'Python Loop Statements'
while checked_string:
    if checked_string[0] in 'xz':
        break
    checked_string = checked_string[1:]
else:
    print('There are no letters "xz" inside the string')
```

Output:

```
There are no letters "xz" inside the string
```

Osnova

...or loop using `range()` object we showed a way, how to loop over a sequence generating indexes using `range()`. Subsequently we used the index number to access the item in a given sequence:

```
some_string = 'For loops support iteration protocol'
    for index in range(len(some_string)):
        char = some_string[index]
        if index % 2 == 0:
            char = char.upper()
        print(char, end = '')
```

Output:

```
FoR LoOpS SuPpOrT ItErAtIoN PrOtOcOl
```

Using this technique is however less Pythonic. Prefered way how to use the for loop is to iterate over items, not over indexes. That way we save few lines of code. And if we need to make decision based on item's index number, we should use built-in function `enumerate()` which iterates with indices and items at the same time:

```
some_string = 'For loops support iteration protocol'
for index, char in enumerate(some_string):
    if index % 2 == 0:
        char = char.upper()
    print(char, end = '')
```

Instead of calculating the length of the sequence and then generating the another sequence in form of range object, we have used the enumerate function which returns 2-item tuples of (item index, item) values:

```
>>> list(enumerate('For loops support iteration protocol'))
[(0, 'F'), (1, 'o'), (2, 'r'), (3, ' '), (4, 'l'), (5, 'o'), (6, 'o'),
(7, 'p'), (8, 's'), (9, ' '), (10, 's'), (11, 'u'), (12, 'p'), (13, 'p'),
(14, 'o'), (15, 'r'), (16, 't'), (17, ' '), (18, 'i'), (19, 't'), (20,
'e'), (21, 'r'), (22, 'a'), (23, 't'), (24, 'i'), (25, 'o'), (26, 'n'),
(27, ' '), (28, 'p'), (29, 'r'), (30, 'o'), (31, 't'), (32, 'o'), (33,
```

Osnova

```
1 some_string = 'For loops support iteration protocol'
2 for index, char in enumerate(some_string):
3     print(index, "->", char)
```

Output:

```
0 -> F
1 -> o
2 -> r
3 ->
4 -> l
...
34 -> o
35 -> l
```

When iterating over the enumerate object, a tuple of two items is returned on each cycle. The first item of the tuple, is the order number (index) of the item and the second is the item itself.

Function `enumerate()` returns a special enumerate object.

```
>>> enumerate('abcde')
<enumerate object at 0x7f6800e50ab0>
```

100% z Lekce 6

Osnova

QUIZ

Range

1/11

What is the main use case for range objects?

A. Generation of a range of float numbers or integers

B. Generation of a range of integers

C. Generation of random integers between the start and the stop point

D. Generation of random floats between the start and the stop point

For Loop

100% z Lekce 6

Osnova

Which of the following for loop headers is correct if we wanted to loop over the string `'Python'` stored in variable `my_str`?

A. for in letter my_str:

B. letter in for my_str:

C. for my_str in letter:

D. for letter in my_str:

Osnova

HOME EXERCISES

String to list

Write a Python program which prompts and accepts a string of comma-separated numbers from a user and generates a list of those individual numeric strings converted into numbers.

The program should print the resulting list to the terminal.

Also take care of possible spaces that could be located at the beginning or the end of the string. In case you do not know, how to get rid of blank spaces at the beginning or end of a string.

Example of running the program:

```
~/PythonBeginner/Lesson3 $ python str_to_list.py
Hello, please write your numbers and press enter to confirm: '23,54,
645, 76'
List: [23, 54, 645, 76]
```

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

- First of all, our string will have to be split into individual numeric strings
- For that purpose we will use the `str.split()` method
- We will want to split the input string on `','` delimiter, therefore the command will look like

```
1 inpt = input('Hello, please write your numbers and press enter to
  confirm: ')
2 nums = inpt.split(',')
3 result = []
4
5 for num in nums:
6     num = int(num.strip())
7     result.append(num)
8
9 print('List:', result')
```

100% z Lekce 6

Write a Python script that will calculate the sum of digits for every number in the list: [1, 22, 321, 64221, 5657, 8321]. The result of this operation should be stored in a new list and printed out to the terminal.

Example running the program (using list [123 , 345, 67]):

```
~/PythonBeginner/Lesson3 $ python sum_digits.py
[6, 12, 13]
```

Online Python Editor

```
1
```

spustit kód

Osnova

Code Solution

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

Click to see our solution

We will want to use nested looping to solve this problem.

1. We need to loop over each item in the list `lst` ,
2. once we have the number from the `lst` available, we need to loop over this number's digits in order to sum them together
3. also, we have pay attention to populate the new list `sums` each time we extract a number from `lst` list - that is why we use `sums.append(0)` . We prepare a new place - number - in the currently built list to which we will add individual digits inside the nested loop
4. to extract individual digits from each integer, we first need to convert it into a string, then extract the digit by looping and convert the digit back to the integer

We could write our code using while or for loop. Both solutions are available below:

For loop

```
lst = [1, 22, 321, 64221, 5657, 8321]
sums = []

# 1.
for num in lst:
    # 4.
    num_str = str(num)
    # 3.
    sums.append(0)

```

```
    Osnova
    s[-1] = sums[-1] + int(digit)

print(sums)
```

While loop

While loop requires a bit more code in order to

5. shorthen the length of list `lst`
6. and string `num`

```
lst = [1, 22, 321, 64221, 5657, 8321]
sums = []

# 1.
while lst:
    # 4.
    num = str(lst[0])
    # 5.
    lst = lst[1:]
    # 3.
    sums.append(0)

    # 2.
    while num:
        # 3.
        sums[-1] += int(num[0])
        # 6.
        num = num[1:]

print(sums)
```

Osnova

04:53

Vowels & Consonants

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / HOME EXERCISES / VOWELS & CONSONANTS

Write a Python program that will count how many vowels and consonants are in a given string. The program should count the number of consonants and vowels inside the sentence: **'a speech sound that is produced by comparatively open configuration of the vocal tract'**

The program should print the resulting counts in format `'No. consonants: <cons_count> | No. vowels: <vowel_count>'`

- `cons_count` should be the number specifying the number of consonants in the sentence
- `vowel_count` should be the number specifying the number of vowels in the sentence

Example of running the program:

```
~/PythonBeginner/Lesson3 $ python count_vow_cons.py
No. consonants: 43 | No. vowels: 30
```

To solve this task you will need the following knowledge:

- Membership testing
- String Methods

100% z Lekce 6

Osnova

If you decide to generate the whole alphabet programmatically, you will probably want to

- use your knowledge of sets - **Mutable Set Operations**
- look up information about built-in function `chr()` resp. `ord()`

Online Python Editor

```
1
```

spustit kód

Code Solution

100% z Lekce 6

Osnova ution

1. We need to have sequences of consonants and vowels
2. then we can iterate over the whole string
3. and check each character, whether it is a letter
4. and whether it is a consonant
5. or a vowel
6. The dictionary that keep the counts, should be created before we begin to loop over the sentence. It will have two categories mapped to actual counts, which are at the beginning set to 0.

```
# 1.
vowels = 'aeiouy'
consonants = 'bcdfghjklmnpqrstvwxz'

# 6.
counts = {'cons':0,'vows':0}

sentence = 'a speech sound that is produced by comparatively open configuration of the vocal tract'

# 2.
for char in sentence:
    # 3.
    if char.isalpha():
        # 4.
        if char.lower() in vowels:
            counts['vows'] = counts['vows'] + 1
        # 5.
        else:
            counts['cons'] = counts['cons'] + 1

print('No. consonants: ' + str(counts['cons']) + ' | No. vowels: ' + str(counts['vows']))
```

100% z Lekce 6

If you want to make an impression, we could also generate the whole alphabet

First we need to know that:

1. computers represent lowercase letters a to z by numbers between 97 and 122
2. function `chr()`, that converts a integers to a corresponding character (chars a to z are mapped to values 97 to 122)

Then we use the knowledge of set operations:

3. if we make a difference among the whole alphabet
4. and the set of vowels, we get consonants (line 9)

```
1  vowels = set('aeiouy')
2
3  letters = set()
4  # 1.
5  letter_a = 97
6  letter_z = 123
7
8  # 3.
9  for num in range(letter_a, letter_z):
10     # 2.
11     letters.add(chr(num))
12
13 # 4.
14 consonants = letters - vowels
15
16 counts = {'cons':0,'vows':0}
17
18 sentence = 'a speech sound that is produced by comparatively open
   configuration of the vocal tract'
19
20 for char in sentence:
21     if char.isalpha():
22         if char.lower() in vowels:
23             counts['vows'] = counts['vows'] + 1
24         else:
```

```
Osnova  'No. consonants: ' + str(counts['cons']) + ' | No. vowels: ' +
          vows']))
```

Divisor

Write a program that will work with three inputs:

- integer `start`
- integer `stop`
- integer `divisor`

All of them should be provided by the user.

The program should:

- generate a collection of integers in range between `start` (including) and `stop` (included in the collection)
- print a list of only those numbers in range `start` - `stop`, that are divisible by `divisor`

If divisor is 0, the program should print to the terminal a string `'Cannot divide by zero'` instead of the list of divisible numbers

Example of running the program:

```
~/PythonBeginner/Lesson5 $ python divisible.py
START: 3
STOP: 9
DIVISOR: 3
Numbers in range(3,10) divisible by 3:
[3,6,9]
```

```
Osnova
Cannot divide by zero
```

Online Python Editor

We should basically perform two levels of checks:

1. Whether the user input is not 0
2. Whether a number is divisible by divisor

```
1
```

spustit kód

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

Osnova

Click to see our solution

1. The first check to be performed has to be that for number 0. If a user provided incorrect input, the program can be terminated,
2. if the divisor is correct, we can iterate the range of integers and check, whether the remainder of each number divided by divisor is zero,
3. if it is, then we want to collect this number into our result variable

```
start = int(input('START: '))
stop = int(input('STOP: '))
divisor = int(input('DIVISOR: '))

result = []

# 1.
if divisor:

    for num in range(start,stop+1):
        # 2.
        if num % divisor == 0:
            # 3.
            result.append(num)

    print('Numbers in range(' + str(start) +', ' + str(stop) + ') divisible by ' + str(divisor) + ':')
    print(result)

else:
    print('Cannot divide by zero')
```

Osnova

ters

At the beginning, we have a sample sentence: **'It was a bright, sunny day in May, and the church bell was just ringing'**

Your program will calculate how many times each letter occurs in a sentence

The output should looks like:

```
~/PythonBeginner/Lesson3 $ python letters.py
{'B': 1, 'y': 3, 'l': 4, ...
```

Online Python Editor

```
1
```

spustit kód

100% z Lekce 6

Osnova

Code Solution

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

Click to see our solution

1. We need to create an empty dictionary,
2. second step is going through the string, the letter after letter,
3. if the letter is not in our dictionary, we want to save it and set the value 1,
4. if the letter is already in our dictionary, we want to increase the value by 1,
5. last step is printing our results.

```
message = 'It was a bright, sunny day in May, and the church bell was just ringing'
# 1.
letters = {}

# 2.
for letter in message:
    # 3.
    letters.setdefault(letter, 0)
    # 4.
    letters[letter] = letters[letter] + 1

# 5.
print(letters)
```

100% z Lekce 6

Create a program that will generate a list of length and width specified by the user. The length and the width should be the same number so the user should be prompted only for one number.

Example of such a list of size 10 can be seen below:

```
[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]]
```

Example of running the program:

```
~/PythonBeginner/Lesson5 $ python numbers.py
Grid Size: 5
[[0, 1, 2, 3, 4],
[0, 1, 2, 3, 4],
[0, 1, 2, 3, 4],
[0, 1, 2, 3, 4],
[0, 1, 2, 3, 4]]
```

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

If we do not care about each inner list to be printed on its own line, the following code will suffice:

```
size = int(input('Grid Size: '))
table = []

for row in range(size):
    line = []

    for num in range(size):
        line.append(num)

```

100% z Lekce 6

```
print('table')
```

Osnova

Or we can do this small trick introducing the line `from pprint import pprint as pp` and changing `print` for `pp` :

```
from pprint import pprint as pp

size = int(input('Grid Size: '))
table = []

for row in range(size):
    line = []

    for num in range(size):
        line.append(num)

    table.append(line)

pp(table)
```

Our own sorting program

Let's create a program that will sort any list of strings according to alphabetical order. It is not permitted to use methods and functions as `sort()` or `sorted()` .

To test your program, use the following list of names:

```
names = ['Michal', 'Pepa', 'Honza',
         'Pavel', 'Lukas', 'Matej',
         'Iva', 'Klara', 'Jana',
         'Honza', 'Vasek','Milan', 'Michal'
         ]
```

```
~/P          ar   /Lesson5 $ python sorter.py
[ H     ,    on    , 'Iva', 'Jana', 'Klara', 'Lukas', 'Matej', 'Michal',
'Michal', 'Milan', 'Pavel', 'Pepa', 'Vasek']
```

Osnova

Online Python Editor

```
1
```

spustit kód

Code Solution

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

100% z Lekce 6

Osnova

Here we need to create a new sorted listed (`sorted_names`). In order to generate a new sorted list, we need to take each value from the original list and compare it to each value in the newly generated list. The value from the old list should be inserted in the place, where it is not greater than the value currently occupying that place.

Also, it can happen, that the value we want to insert is greater than all the values in the currently built list. In that case the insert and neither break statements will not be executed and therefore it is convenient to use the **else** case to append the value.

```
1  names = ['Michal', 'Pepa', 'Honza',
2          'Pavel', 'Lukas', 'Matej',
3          'Iva', 'Klara', 'Jana',
4          'Honza', 'Vasek','Milan', 'Michal']
5
6  sorted_names = [names.pop(0)]
7
8  for name in names:
9
10     for i,s_name in enumerate(sorted_names):
11         if name < s_name:
12             sorted_names.insert(i,name)
13             break
14     else:
15         sorted_names.append(name)
16
17 print(sorted_names)
```

Odd - Even

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / HOME EXERCISES / ODD - EVEN

Write a Python program that will prompt user for a number and then will determine, whether the

100% z Lekce 6

```
~/PythonBeginner/Lesson3 $ python odd_even.py
Please, enter a number: 35
ODD
```

```
~/PythonBeginner/Lesson3 $ python odd_even.py
Please, enter a number: 36
EVEN
```

Online Python Editor

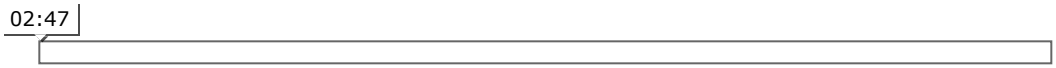
```
1
```

spustit kód

Code Solution

Osnova ution

```
num = int(input('Please, enter a number: '))
result = 'EVEN' if num % 2 == 0 else 'ODD'
print(result)
```



Get counts [H]

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / HOME EXERCISES / GET COUNTS [H]

Create a program that will print out how many times does every object occur inside any list, tuple or a string (for string we want to get counts of individual characters).

100% z Lekce 6

- **key** - string representation of the counted item
- **value** - integer reflecting the number of occurrence of the counted item

If the **input** sequence looks like this:

```
[1,21,5,3,5,8,321,1,2,2,32,6,9,1,4,6,3,1,2]
```

We want the **output** look like this:

```
{'1': 4, '8': 1, '321': 1, '4': 1, '3': 2, '6': 2, '21': 1, '32': 1, '5':
2, '2': 3, '9': 1}
```

Example of running the code:

```
~/PythonBeginner/Lesson3 $ python get_counts.py
{'1': 4, '8': 1, '321': 1, '4': 1, '3': 2, '6': 2, '21': 1, '32': 1, '5':
2, '2': 3, '9': 1}
```

You will probably need the knowledge of:

- conditions
- sequences
- dictionaries
- for loops

Bonus

Print the result as a **table** with nicely aligned numbers ordered from the most frequent occurrence to the least frequent occurrence.

Example of running the code depicting the above dictionary printed to the terminal:

```
~/PythonBeginner/Lesson3 $ python get_counts.py
 Item |Count
=============
```

```
3
        Osnova
5
6       |   2
32      |   1
4       |   1
321     |   1
8       |   1
9       |   1
21      |   1
```

Here you will probably use knowledge of:

- String Methods - String Positioning
- Dictionary methods: `.keys()`, `.values()`, `.items()`

Online Python Editor

```
1
```

100% z Lekce 6

Osnova

Code Solution

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

Click to see our solution

Our example will work with list: `[1,21,5,3,5,8,321,1,2,2,32,6,9,1,4,6,3,1,2]`

1. first we can create a dictionary that will collect counts of each item in the sequence
2. then we just need to loop over the input sequence item by item and:
    - Establish the counter for each item, if it does not exist yet and set the associated count value to 0
    - Increase the count of the item by 1
3. Lastly we just print the result

The two steps of the second point are performed in one line by the code
`counts[str(item)] = counts.setdefault(str(item),0) + 1`

```
seq = [1,21,5,3,5,8,321,1,2,2,32,6,9,1,4,6,3,1,2]

# 1.
counts = {}

# 2.
for item in seq:
    counts[str(item)] = counts.setdefault(str(item),0) + 1

# 3.
print(counts)
```

If we want to print the results in a tabular form we need to perform **2 extra actions**:

Osnova

1. DETERMINE EACH COLUMN'S LENGTH
    - the column length has to be min. as wide as the header
    - subseqenutly we just loop over each key-value pair
2. PRINT THE RESULTING TABLE
    - here we want to adjust the left column to the left (`str.ljust()` method) and the right column should be centered (`str.center()`)

To order the key in the dictionary, we use the trick with `sorted()` built-in function - `sorted(counts, key=counts.get, reverse=True)` :

- the parameter `key` tells python to compare only dictionary values, which we get using `counts.get` method
- the reverse parameter tells python to sort the dictionary keys from largest to smallest (opposite to default ordering)

```
1  seq = [1,21,5,3,5,8,321,1,2,2,32,6,9,1,4,6,3,1,2]
2  counts = {}
3
4  # GET COUNTS FOR EACH VALUE
5  for item in seq:
6      counts[str(item)] = counts.setdefault(str(item),0) + 1
7
8  # DETERMINE EACH COLUMN'S LENGTH
9  col1_width = len(' Item |') + 2
10 col2_width = len(' Count ') + 2
11
12 for k,v in counts.items():
13
14     k_len = len(str(k)) + 2
15     v_len = len(str(v)) + 2
16
17     if k_len > col1_width:
18         col1_width = k_len
19
```

100% z Lekce 6

Osnova

```
    RESULTING TABLE
print(' Item '.ljust(col1_width) + '|' + 'Count'.center(col2_width))
print('='* (col1_width +col2_width))

ordered_data = sorted(counts, key=counts.get, reverse=True)

for category in ordered_data:
    print(' ' +str(category).ljust(col1_width) + '|' +
str(counts[category]).center(col2_width))
```

04:44

Connect [H]

PYTHON ACADEMY / 5. FOR LOOPS & RANGES / HOME EXERCISES / CONNECT [H]

Write a Python program that will:

100% z Lekce 6

2. th ngths of each word from the original string
Osnova
3. and extract 1st character from each of these strings.
4. These characters should be then concatenated together based on lengths of the original words in the sentence and separate newly created words by space.

Example of our original string: **'Today is a nice day on Australian beach.'**

2. so collecting lengths of each word in the sentence: `5, 2, 1, 4, 3, 2, 10, 6`
3. then extracting first characters from each word: `T, i, a, n, d, o, A, b`
4. The resulting string will contain only 3 words, because there is not enough extracted letters to fill all the `lengths` : `'Tiand oA b'` - the first word should contain 5 characters, second 2 and third 1

Here is the **input** string you should be working with:

```
The Czech Republic also known as Czechia, is a nation state in Central
Europe bordered by Germany to the west, Austria to the south, Slovakia to
the east and Poland to the northeast.[12] The Czech Republic covers an
area of 78,866 square kilometres (30,450 sq mi) with mostly temperate
continental climate and oceanic climate. It is a unitary parliamentary
republic, has 10.5 million inhabitants and the capital and largest city
is Prague, with over 1.2 million residents. The Czech Republic includes
the historical territories of Bohemia, Moravia, and Czech Silesia.
```

And here is the expected **output:**

```
'TCR akaCi ansiCEbb Gttw AttsS tt eaPttnTC Rc a ao7sk( smwmt cc aocIiau
prh1mi atcalciP wo 1mrTCRi th toB MaCS'
```

Online Python Editor

1

100% z Lekce 6

Osnova

spustit kód

Code Solution

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

Click to see our solution

Here we can combine for and while loop.

With for we:

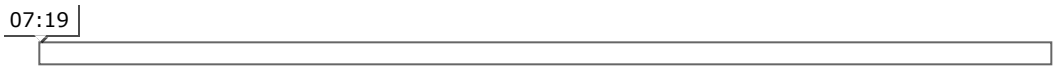
1. collect the lengths,

Th[illegible] lo[illegible]s used for the second part of the program, where we do not know in [illegible] Osnova [illegible] [illegible]l run out of initial letters extracted from each word. Each one iteration we generate 1 new word from initial letters during one iteration inside the while loop. That means we need to:

3. take the length of the next word to be built and check, whether:
4. the length of the next word will be longer than the number of available initial letters - in that case we want to concatenate the rest of the inits with the newly built string (sentence) and clear the inits list
5. the length of the next word does not use all the inits (line 16) - we concatenate the required number of inits with the currently built string and remove those initial letters from the `inits` list

```
1  words = my_str.split()
2  inits = []
3  lengths = []
4
5  for w in words:
6      # 1.
7      lengths.append(len(w))
8      # 2.
9      inits.append(w[0])
10
11 result_string = ''
12
13 while inits:
14     # 3.
15     l = lengths.pop(0)
16     # 4.
17     if l >= len(inits):
18         result_string = result_string + ''.join(inits)
19         inits.clear()
20     # 5.
21     else:
22         result_string = result_string + ''.join(inits[:l]) + ' '
23         del inits[:l]
```

Osnova



DALŠÍ LEKCE