

Basic Syntax, Conditional Statements and Loops



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

sli.do

#fund-python

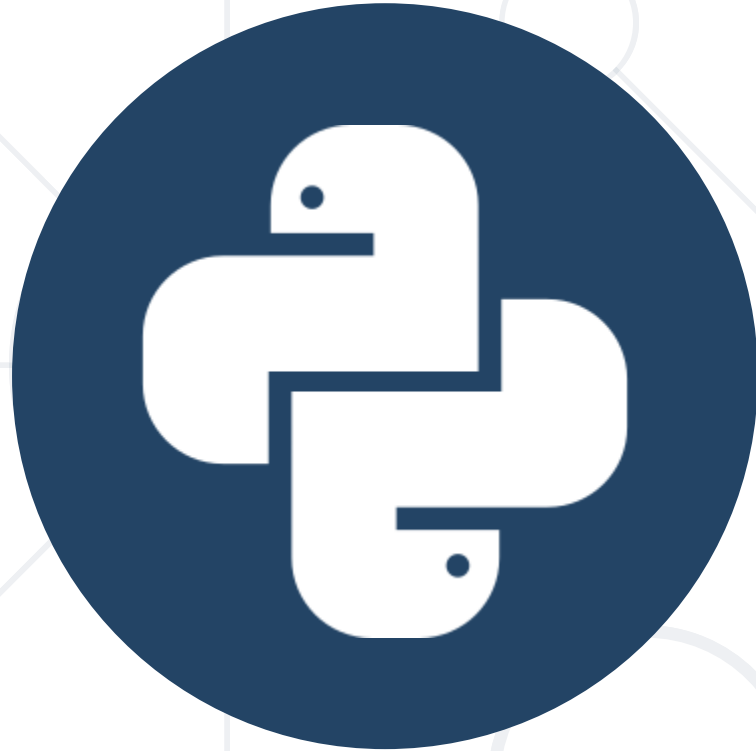
1. Basic Syntax and First Steps

2. Conditional Statements

- if, elif, else
- indentation
- and, or

3. Loops

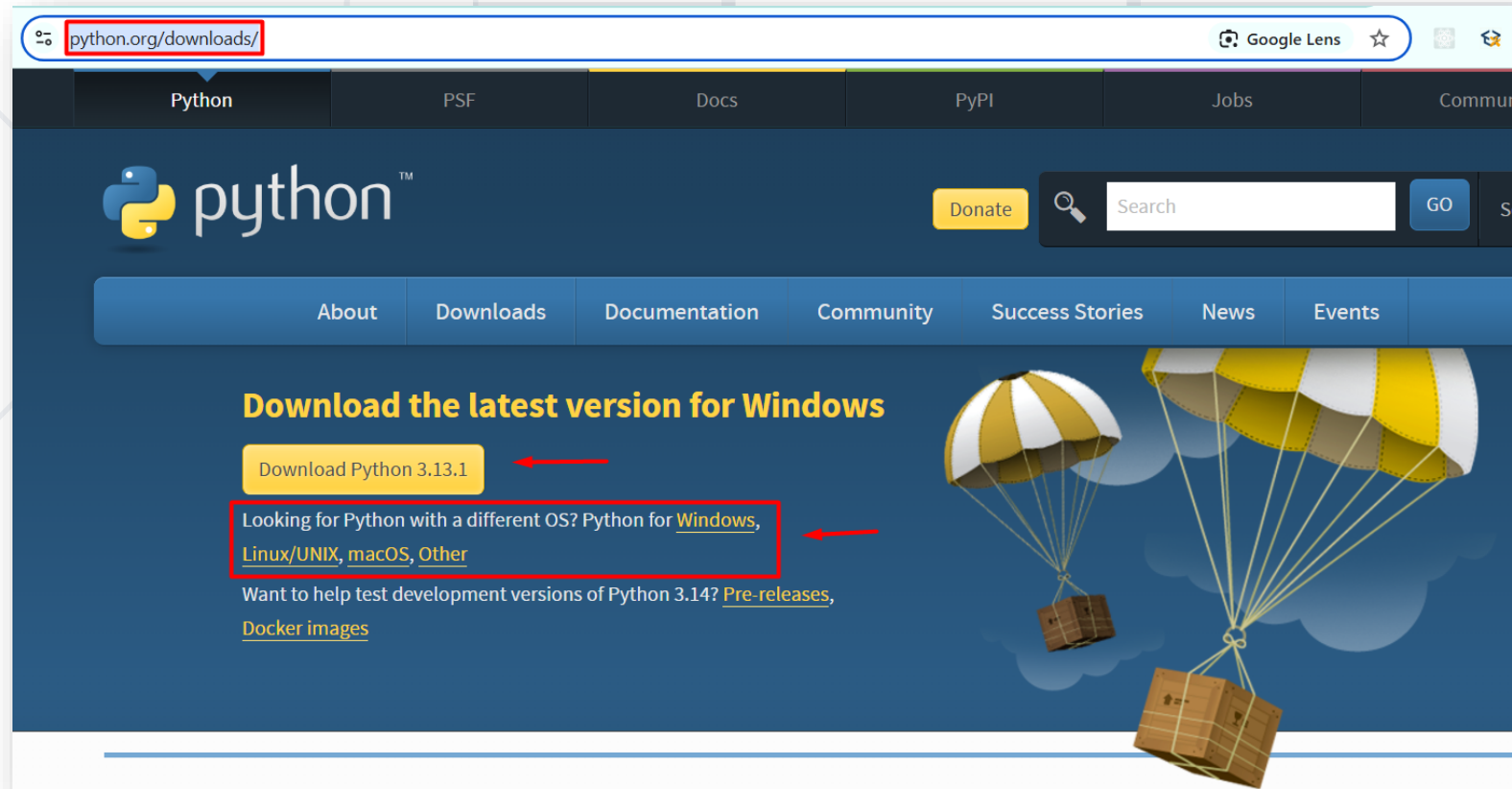




First Steps

Installing Python

- Go to python.org and click the download link depending on your operating system



Run Python in Command Prompt

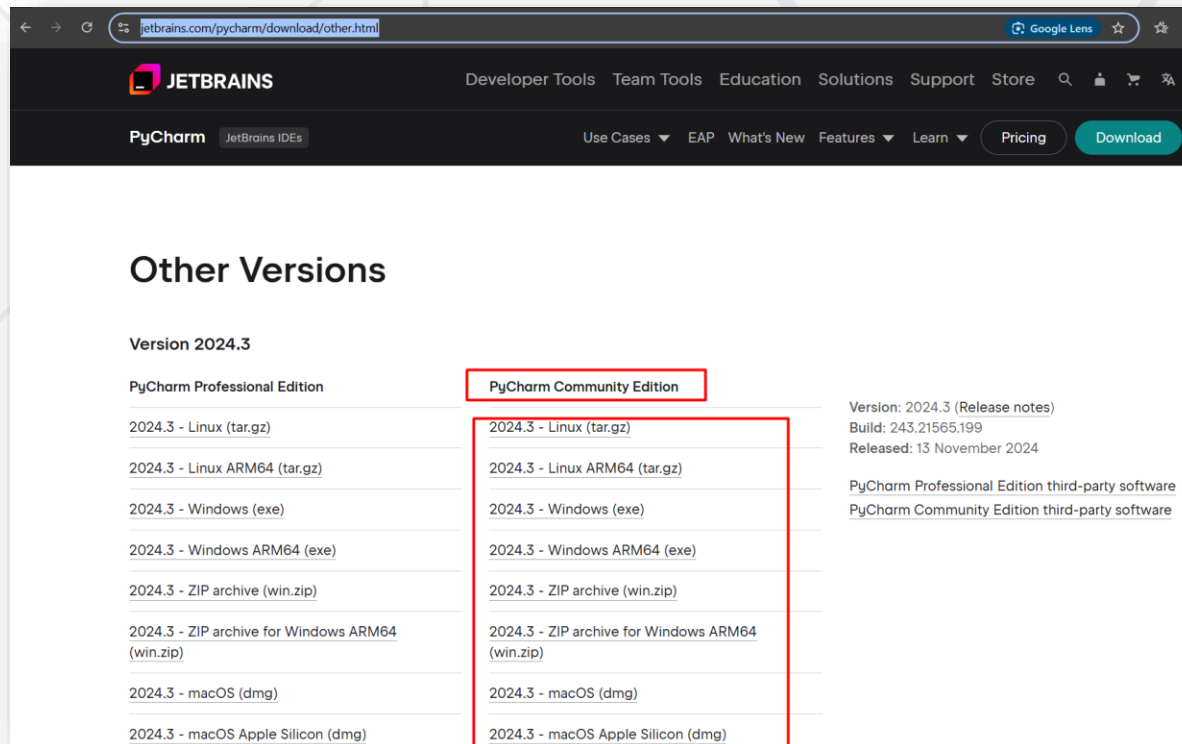
- You can code and execute python directly in the command prompt by typing "**python**" or "**py**"

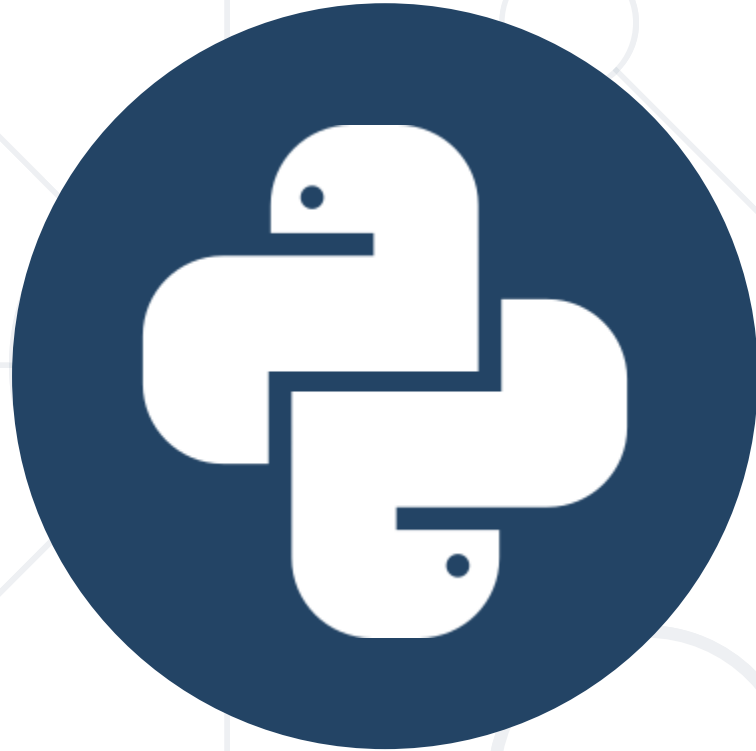
```
C:\Users\De11>python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>>
```

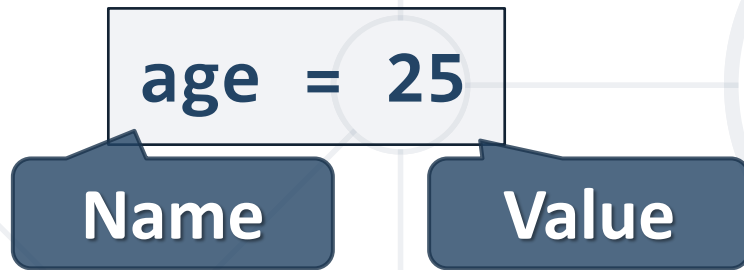
- You can also code in Python using **IDE** (for example: **PyCharm**)
- You can download **PyCharm** from here:
<https://www.jetbrains.com/pycharm/download/other.html>





Basic Syntax

- Variables - they are the way to **store information** and are characterized by **name**, **type**, and **value**



- Data types – variables are used to hold different data types
 - int** - integer number: 1, 2, 3, 4, ...
 - float** - real number: 0.5, 3.14, -0.5, ...
 - str** - string and chars: "a", "Hello", ...
 - bool** - boolean: True, False



Conditional Statements

Conditional Code Execution

The if-Statement

- An "if statement" is written by using the **if** keyword



```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```


The if-Statement

- Python supports the usual logical conditions from mathematics
 - Equals: $a == b$
 - Not Equals: $a != b$
 - Less than: $a < b$
 - Less than or equal to: $a <= b$
 - Greater than: $a > b$
 - Greater than or equal to: $a >= b$



Indentation


- Python relies on **indentation**, using whitespace, to define scope in the code
- Other programming languages often use curly-brackets for this purpose
- If statement, without indentation will raise an error



```
a = 33
b = 200
if b > a:
print("b is greater than a") # error
```

The else-Statement


- The **else** keyword catches anything which isn't caught by the preceding conditions



```
a = 200
b = 33
if b > a:
    print("b is greater than a")
else:
    print("b is not greater than a")
```

The elif-Statement

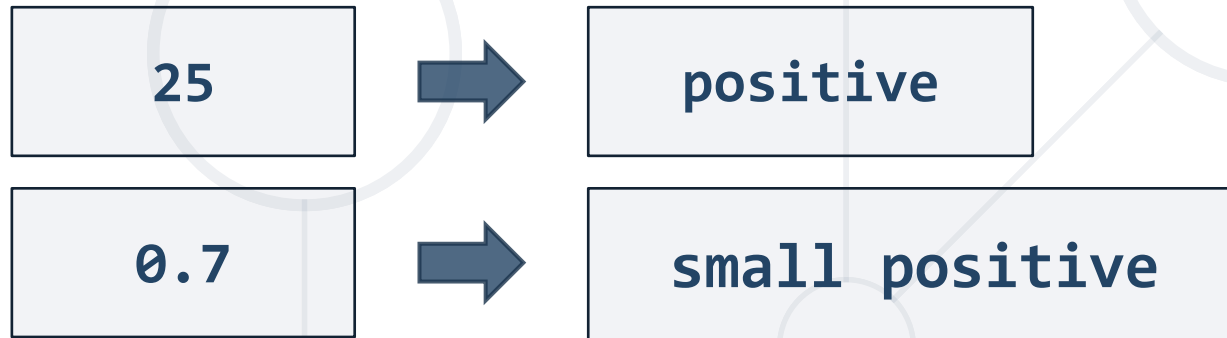
- The **elif** keyword is pythonic way of saying "if the previous conditions were not true, then try this condition"



```
a = 33
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
```

Problem: Number Definer

- Write a program that
 - Reads a floating-point number
 - Prints **zero** if the number is zero or otherwise prints **positive** or **negative**
 - Adds **small** if the absolute value of the number < 1 , or **large** if the number $> 1\,000\,000$



Solution: Number Definer

```
number = float(input())
if number == 0:
    print("zero")
elif number > 0:
    if number < 1:
        print("small positive")
    elif number > 1000000:
        print("large positive")
    else:
        print("positive")
# TODO
```



- They are used to **combine** conditional statements

```
if a > b or a > c:  
    print("At least one of the conditions is True")
```

```
if a > b and c > a:  
    print("Both conditions are True")
```

```
if not a > c:  
    print("The condition is False")
```

- **or** has a lower priority than **and**
- **and** has a lower priority than **not**

```
if 2 > 1 or 3 < 4 and not 4 > 0:  
    print('This will be printed')
```

```
if (2 > 1 or 3 < 4) and not 4 > 0:  
    print('This will NOT be printed')
```

Check Number Range

- If you want to check whether a number is in a given range, you can use the following syntax

```
a = int(input())  
if 1 <= a <= 10:  
    print("a is in the range 1 and 10")
```

1 ... 10

Problem: Largest of Three Numbers

- Write a program which
 - Reads three whole numbers from the console
 - Prints the largest number

3
-1
5



5

0
-1
-2



0



Solution: Largest of Three Numbers

```
first_num = int(input())
second_num = int(input())
third_num = int(input())
if first_num > second_num and first_num > third_num:
    print(first_num)
elif second_num > first_num and second_num > third_num:
    print(second_num)
else:
    print(third_num)
```



Loops

Repeating Blocks of Code

For-Loops

- A **for** loop is used to iterate over a sequence of iterable types like
 - string
 - list
 - other iterable types
- The for loop does not require an indexing variable to set beforehand



The range() Function

- To loop through a set of code a specified number of times, we can use the **range()** function



```
for x in range(3):  
    print(x)
```

```
# 0
```

```
# 1
```

```
# 2
```

Problem: Word Reverse

- Write a program that
 - Receives a single word from a user
 - Reverses it and prints it




Solution: Word Reverse

```
word = input()
reversed_word = ""
for i in range(len(word) - 1, -1, -1):
    reversed_word += word[i]
print(reversed_word)
```

The Break Statement

- The **break** statement **stops** the loop before it has looped through all the items




```
for x in range(3):  
    if x == 1:  
        break  
    print(x)
```



0

The Continue Statement

- The **continue** statement **skips** the current iteration of the loop and continues with the next



```
for x in range(3):  
    if x == 1:  
        continue  
    print(x)
```




0

2

The else Clause

- The **else** clause is executed when the loop finishes iterating without hitting the **break** statement



```
for x in range(3):  
    if x == 3:  
        break  
  
else:  
    print("Finish")
```



Finish

Problem: Even Numbers

- Write a program that
 - Receives a number n and then receives n **different numbers**
 - If it receives an **odd number**, print "{num} is odd!" and end the program
 - If all numbers are **even**, print "All numbers are even"

2
12
286



All numbers are even.




Solution: Even Numbers

```
n = int(input())  
  
for i in range(n):  
    number = int(input())  
    if not number % 2 == 0:  
        print(f"{number} is odd!")  
        break  
else:  
    print("All numbers are even.")
```


While-Loops

- With a `while` loop we can execute a set of statements as long as the condition is true



```
i = 1
while i < 6:
    print(i)
    i += 1
```

- **Note:** remember to increment **i**, or else the loop will continue forever

Problem: Number Between 1 and 100

- Write a program that
 - Reads floating-point numbers from the console until it receives a number between 1 and 100 inclusive
 - When the correct number is received, stop reading and print **"The number {number} is between 1 and 100"**

-3
0.9
44



The number 44 is between 1 and 100

Solution: Number Between 1 and 100

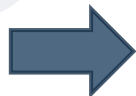
```
number = float(input())  
while not (1 <= number <= 100):  
    number = float(input())  
print(f'The number {number} is between 1 and 100')
```



Problem: Shopping

- Write a program that
 - Reads a **budget** and then **prices of each product** you need to buy until it receives the command "End"
 - If there is **not enough budget** left for the next product, print "You went in overdraft!" and end the program
 - If you **bought everything** needed and the program receives "End", prints "You bought everything needed."

```
100
5
End
```



```
You bought everything needed.
```

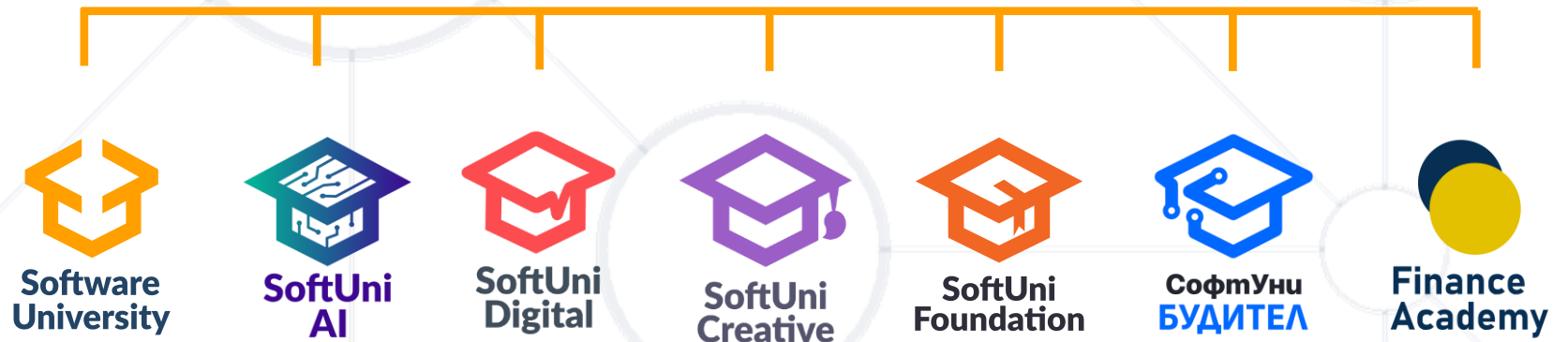
```
budget = int(input())
command = input()
while command != "End":
    product_price = int(command)
    budget -= product_price
    if budget < 0:
        print("You went in overdraft!")
        break
    command = input()
else:
    print("You bought everything needed.")
```



- We learned how to:
 - **Execute code** based on different conditions
 - Use loops to execute a **block of code** multiple times on different elements
 - **Stop / skip iterations** in loops



Questions?



SoftUni Diamond Partners



**SUPER
HOSTING
.BG**



INDEAVR
Serving the high achievers



THE CROWN IS YOURS

VIVACOM

- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, softuni.org

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity



- This course (slides, examples, demos, exercises, homework, documents, videos, and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

