# Машинно обучение

# Лабораторно упражнение №1

# Въведение в Python

Python е език от високо ниво, който позволява програмата да се раздели на модули, които могат да бъдат използвани, отново в други програми на Python. Езикът съдържа набор от вградени модули, които обезпечават различни функции като: файлов вход/изход (I/O), системни функции, сокети (sockets), програмни интерфейси към GUI библиотеки.

Python е интерпретируем език, като не са необходими компилиране и свързване. Интерпретаторът може да се използва интерактивно, което го прави лесен за експериментиране с възможностите на езика, или за тестване на функциите по време на разработка отдолу-нагоре.

Езикът е кръстен на шоуто на BBC "Monty Python's Flying Circus".

**1. Интерактивен режим.**

В интерактивен режим промпта представлява три знака по-голямо (>>>), а за продължение на ред се появява вторичен промпт, обикновено три точки (...). В този режим Python може да се използва като калкулатор.

Помощна информация за елементите на езика, може да получите като въведете help(). Появява се пояснителен текст и от появилия се списък, въвеждате търсения елемент.

2. **Идентификатори в Python. –** идентификаторите започват с буква или долно тире, съдържат латински главни и/или малки букви, долно тире и цифри, и са неограничени по дължина. Запазени идентификатори са тези идентификатори, които започват със следните символи: _*, _*_, __*.
3. **Ключови думи в Python. –** ключовите думи са запазени за езика и не могат да се използват като обикновени идентификатори.

```
False      class      finally    is         return
None       continue   for        lambda     try
True       def        from       nonlocal   while
and        del        global     not        with
as         elif       if         or         yield
assert     else       import     pass
break      except     in         raise
```

**4. Основни типове данни.**

```
int, float, long, complex, bool
```

В Python типа на операндите не се заявява предварително. Ключовите думи: int, float, long, comlex и bool се използват за преобразуване от един тип друг.

**5. Коментар. –** използва се знака #

**6. Свързване на два или повече физически реда. –** използва се наклонена черта \

***Пример :***

```
>>> x=[1,\
   2]
```

***Резултат:***

```
>>> x
[1, 2]
```

При създаване на списък, отделните елементи могат да се въвеждат на няколко реда. Коментари могат да се добавят на всеки ред и до всеки символ.

***Пример :***

```
>>> y=[1,2,# коментар
   3,4,5,
   6,7]
```

***Резултат*:**

```
>>> y
[1, 2, 3, 4, 5, 6, 7]
```

**7. Оператори.**

```
+       -       *       **      /       //      %
<<      >>      &       |       ^       ~
<       >       <=      >=      ==      !=
```

**8. Разделители.**

```
(       )       [       ]       {       }
,       :       .       ;       @       =       ->
+=      -=      *=      /=      //=     %=      &=      |=
^=      >>=     <<=     **=
```

**9. Аритметични операции. - +, -, * , **, /, //, %**

***Примери*:**

```
>>> 2**3 # степенуване
8
>>> 9/2   # деление
4.5
>>> 9//2 # целочислено деление
4
>>> 9%2 # остатък от деление
1
```

### 10. Оператори за сравнение.

```
<       >       <=      >=      ==      !=
```

### 11. Побитови операции.

```
<<      >>      &       |       ^       ~
```

### 12. Съставни оператори.

```
+=      -=      *=      /=      //=     %=      @=
&=      |=      ^=      >>=     <<=     **=
```

### 13. Логически оператори.

```
   and      or       not
```

### 14. Оператори за вход и изход.

#### 14.1. Оператор за вход.

**input([prompt]) -> string**

Чете низ от стандартния вход

***Пример 1:***

```
>>> name=input("Your name:")
Your name:Ana
>>> name
'Ana'
```

***Пример 2:***

```
>>> number=input("Enter integer:")
Enter integer:3
>>> number
```

```
'3'
```

В примера number  е string.

***Пример 3:***

```
>>> number=int(input("Enter integer:"))
Enter integer:3
>>> number
3
```

В примера number  е цяло число.

### 14.2. Оператор за изход.

**print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)**

За повече информация help(print).

***Пример:***

```
>>> print("number=",number)
number= 3
```

**15. Списък.** (List)- съставен тип данни за групиране на стойности, от различен и/или един и същи тип, изписани като поредица и разделени със запетая между квадратни скоби. Достъпът до елемент се осъществява чрез индекс, цяло число. Дължината на всеки списък може да се променя и да не се определя предварително. Списъците могат да се влагат един в друг.

```
>>> ime='Ivan'
>>> familia='Ivanov'
>>> student=['ime:',ime,'familia:',familia,4,'kurs']
```

*Резултат*:

```
>>> student
['ime:', 'Ivan', 'familia:', 'Ivanov', 4, 'kurs']
```

### 15.1. Индексиране.

```
>>> student[0]
'ime:'
>>> student[-1] # извежда последния елемент
'kurs'
```

```
>>> student[0:2] # извежда елементи от индекс 0 до
индекс 1
['ime:', 'Ivan']
>>> student[:3] # извежда първите три елемента
['ime:', 'Ivan', 'familia:']
>>> student[:3] # извежда всичко, освен първите три
елемента
['ime:', 'Ivan', 'familia:']
>>> student[3:]
['Ivanov', 4, 'kurs']
```

**15.2. Замяна на елементи.**

```
>>> student[1]='Dian'
```

*Резултат*:

```
>>> student
['ime:', 'Dian', 'familia:', 'Ivanov', 4, 'kurs']
```

**15.3. Конкатенация на списъци.**

```
>>> fnomer=['fN',123456]
>>> student=student + fnomer
```

*Резултат*:

```
>>> student
['ime:', 'Dian', 'familia:', 'Ivanov', 4, 'kurs', 'fN',
123456]
```

**15.4. Изтриване на елементи.**

```
>>> student[6:]=[ ]
```

*Резултат*:

```
>>> student
['ime:', 'Dian', 'familia:', 'Ivanov', 4, 'kurs']
```

***15.5. Създаване на празен списък.***

```
>>> student1=list()
>>> student1
[ ]
```

**15.6. Визуализация на списък.**

```
>>> list(student)
    ['ime:', 'Ivan', 'familia:', 'Ivanov', 4, 'kurs']
```

**15.7. Списъкът може да се третира и като двумерен масив.**

```
>>> student[1][0]
    'I'
```

**15.8. Списъци. Методи.**

**15.8.1. Разширяване на списък чрез extend.**

```
    >>> student.extend(fnomer)
```

*Резултат*:

```
    >>> student
['ime:', 'Dian', 'familia:', 'Ivanov', 4, 'kurs', 'fN',
123456]
```

***Но:***

```
     >>> student.extend('TU-Varna')
```

*Резултат*:

```
     >>> student
['ime:', 'Dian', 'familia:', 'Ivanov', 4, 'kurs', 'fN',
123456, 'T', 'U', '-', 'V', 'a', 'r', 'n', 'a']
```

**15.8.2. Разширяване на списък чрез append.**

```
    >>> student.append('TU-Varna')
```

*Резултат*:

```
    >>> student
    ['ime:', 'Dian', 'familia:', 'Ivanov', 4, 'kurs', 'fN',
123456, 'T', 'U', '-', 'V', 'a', 'r', 'n', 'a', 'TU-Varna']
```

***Но:***

```
>>> year=['year',2018]
>>> student.append(year)
```

*Резултат*:

```
>>> student
    ['ime:', 'Dian', 'familia:', 'Ivanov', 4, 'kurs', 'fN',
123456, 'T', 'U', '-', 'V', 'a', 'r', 'n', 'a', 'TU-Varna',
['year', 2018]]
```

**15.8.3. Преброяване на повтарящите се елементи.**

```
>>> student1=['Koev', 2,2,3,4,5,5,5]
>>> student1.count(5)
3
```

**15.8.4. Индекси на списък.**

***L.index(value, [start, [stop]]) -> integer -- return first index of value.***

```
>>> student1.index('Koev')
0
>>> student1.index(2,2,7)
2
```

**Самостоятелна задача 1:**

Разгледайте самостоятелно методите: **clear**, **insert**, **pop**, **remove**, **reverse**, **sort**, **reverse**. За повече подробности: help(list). Тествайте със собствени примери.

## 16. Комплекти. – "tuple"

**Комплектите** са съставен тип данни за групиране на стойности, от различен и/или един и същи тип, изписани като поредица и разделени със запетая и без/със заграждащи скоби

```
>>> x='hi'
>>> y=x,'world',2018
>>> y
('hi', 'world', 2018)
```

В резултата комплектите винаги са затворени в скоби

### 16.1. Празен комплект.

```
>>> c=()
```

### 16.2. Списък с един елемент.

```
>>> d=1,
>>> d
(1,)
```

### 16.3. Дължина на комплект.

```
>>> len(d)
 1
```

### 16.4. Присвояване на стойности на повече от една променлива.

```
>>> i,j=1,2
>>> i
1
>>> j
2
```

### 16.5. Извличане на стойности от комплекта – разкплектоване.

```
>>> a,b,c=y
>>> a,b,c
('hi', 'world', 2018)
>>> a
'hi'
>>> b
'world'
>>> c
2018
```

### 16.6. Едновременно създаване на комплект и сравняване.

```
>>> 1,2<1,3
(1, False, 3)
```

### 16.7. Сравняване на комплекти.

Сравнява първия елемент от първия комплект с първия елемент от втория комплект, ако е лъжа сравнява следващата двойка елементи.

```
>>> (4, 5) < (3, 5)
False
>>> (2, 6) < (3, 5)
True
```

### 16.8. Комплекти. Методи.

#### 16.8.1. Преброяване на повтарящите се елементи.

```
>>> y.count(2018)
1
```

#### 16.8.2. Индекси на елемент от комплект.

```
>>> y.index(2018)
2
```

**17. Оператори за членство. -** използват се за проверка дали дадена стойност или променлива се съдържа в дадена последователност - string, list, tuple, set и dict.

| Оператор | Значение | Пример |
|---|---|---|
| in | True, ако стойността/променливата е намерена в дадената последователност - string, list, tuple, set и dictionary | 2 in x |

| | | |
|---|---|---|
| not in | True, ако стойността/променливата не е намерена в дадената последователност - string, list, tuple, set и dictionary | 2 not in x |

***Пример*:** Проверка дали даден елемент се съдържа в множеството – със служебната дума *in*

```
>>> 2 in x
True
```

**18. Оператори за идентичност. –** използват се за проверка дали две променливи сочат към една и съща клетка от паметта. Две променливи, които са равни, може да са идентични, може и да не са.

| Оператор | Значение | Пример |
|---|---|---|
| is | True, ако операндите са идентични | x is y |
| is not | True, ако операндите не са идентични | x is not y |

**Пример 1.** Променливите са от цял тип.

```
>>> u=9
>>> v=9
>>> u==v
True
>>> u is v
True
>>> id(u),id(v)
(1878320112, 1878320112)
```

**Извод:** Стойностите им са както равни, така и идентични.

И двете променливи сочат към една и съща клетка от паметта. Идентификационните им номера са равни. За повече информация help(id).

**Самостоятелна задача 2:**

Проверете идентични ли са 2 променливи, които съдържат еднакви низове.

**Какъв е извода?**

***Пример 2.*** Променливите са от реален тип. Стойностите им са равни, но не са идентични.

```
>>> a=1.234
>>> b=1.234
>>> a==b
True
>>> a is b
False
>>> id(a),id(b)
(48802528, 48801840)
```

**Но, ако свържем двете променливи със знак за равенство**

```
>>> a=b
>>> a is b
True
>>> id(a),id(b)
(48801840, 48801840)
```

**Извод:** Стойностите на двете променливи от реален тип след свързване със знака за равенство са равни и идентични.

**Самостоятелна задача 3:**

Проверете идентични ли са 2 променливи, които съдържат еднакви списъци от елементи.

**Какъв е извода?**

## 19. Условни оператори.

```
if (условен израз):      # двете точки са задължителни
    оператор             # задължително трябва да има отстъп
elif:                    # двете точки са задължителни
    оператор             # задължително трябва да има отстъп
else:                    # двете точки са задължителни
    оператор             # задължително трябва да има отстъп
```

## 20. Циклични оператори.

### 20.1. Оператор за цикъл while.

```
while (условен израз):     # двете точки са задължителни
  оператор 1               # задължително трябва да има отстъп
```

```
    оператор 2          # задължително трябва да има отстъп
    . . .
    Оператор n          # задължително трябва да има отстъп
```

### 20.2. Оператор за цикъл for.

```
for x1 in x2:     # двете точки са задължителни
    оператор 1          # задължително трябва да има отстъп
    оператор 2          # задължително трябва да има отстъп
    . . .
    Оператор n          # задължително трябва да има отстъп
```

където:

**x1** – управляваща_променлива

**x2** – списък, редица или низ

- С оператора **break** се осъществява излизане от цикъл
- Оператора **continue** прекъсва текущата итерация и продължава изпълнението на цикъла от следващия елемент

**21. Функция range(). -** генерира списъци, съдържащи аритметични прогресии

**Синтаксис: range(stop)** или **range(start, stop[, step])**

За повече подробности: **help(range)**

***Примери*:**

```
>>> range(5)
range(0, 5)
```

Това са числата от 0 до 4 включително, т.е. 0,1,2,3,4

```
>>> for i in range(5):
        print(i,end=" ")
```

***Резултат*:**

```
0 1 2 3 4
>>> for i in range(0,10,2):
        print(i,end=" ")
```

***Резултат*:**

```
0 2 4 6 8
```

Често операторът range се комбинира с оператора len.

```
>>> x="ab"
>>> for i in range(len(x)):
     print(x[i])
```

***Резултат*:**

```
a
b
```

**22. Математически функции. – чрез добавяне на библиотеката math**

**Примери:**

```
>>> import math
>>> math.sqrt(9)
3.0
>>> math.exp(1)
2.718281828459045
>>> math.log(1)
0.0
>>> math.log(math.exp(1))
1.0
>>> math.log2(8)
3.0
>>> math.log10(100)
2.0
>>> pow(2,4)
16
>>> math.factorial(4)
24
```

***За повече информация: help(math)***

**23. Работа с комплексни числа.**

**23.1.** реална част – z.real
**23.2.** имагинерна част - z.imag

*Примери*:

```
>>> z=8+3j
>>> z.real
```

```
8.0
>>> z.imag
3.0
```

## 24. Самостоятелни задачи.

### 24.1. Самостоятелна задача 4:

С помощта на оператора while или for, изчислете и изведете числата на Фибоначи.

Отворете нов файл от File→New File. Запишете файла като име_на_файл.py. Стартирайте го от Run→Run Module или F5.

### 24.2. Самостоятелна задача 5:

Създайте програма на Python, която да проверява дали въведена дума от потребител е палиндром.

### 24.3. Самостоятелна задача 6:

Създайте програма на Python, която да преброява гласните във въведена дума от потребител.

Използвайте функциите lower() и count().

## 25. Полезни връзки.

**25.1.** https://docs.python.org/3/
**25.2.** https://www.programiz.com/python-programming
**25.3.** https://www.tutorialspoint.com/python/index.htm
**25.4.** https://anh.cs.luc.edu/python/hands-on/3.1/handsonHtml/ch1.html