Основи на програмирането на Python

В темата ще бъдат разгледани следните основни въпроси:

- Особеностите на Python като език за програмиране от високо ниво
- Типове данни в Python
- Управление на изпълнението на програмата
- Цикли
- Оператори break, continue и else в циклите

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Основи на програмирането на Python

Python е език за програмиране от високо ниво с отворен код, издаден под лиценз, съвместим с GNU General Public License (GPL). Авторските права върху Python се притежават от организацията с нестопанска цел Python Software Foundation (PSF). Програмният език Python е създаден в края на 80-те години от Гуидо Ван Росъм. Пуснат е през 1991 г. в Centrum Wiskunde & Informatica (CWI) в Холандия като наследник на езика ABC. Кръстен е на популярно комедийно шоу, наречено „Летящият цирк на Монти Пайтън“.

През последните няколко години популярността му нарасна неимоверно. Python е сред първите три най-обичани езика за програмиране с общо предназначение през последните три години и засега е на първо място сред езиците за програмиране на изкуствен интелект AI през 2022 г.

Други характеристики на Python са, че той е динамичен език за програмиране от високо ниво. Той е доста лесен за научаване и осигурява мощни възможности за въвеждане. Кодът на Python има много „естествен“ стил, тъй като е лесен за четене и разбиране (благодарение на липсата на точка и запетая, и скоби, които определят блоковете). Езикът за програмиране Python работи на всяка платформа, вариращa от Windows до Linux до Macintosh, Solaris и т.н.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Простотата на Python е това, което го прави толкова популярен. „Естественият“ стил на кода на Python се определя от следните негови характеристики:

- лесно четим код;
- елегантно и динамично писане на кода;
- по-малко синтактични изключения;
- лесна и функционална обработка на стингове;
- изчистено визуално оформление;
- използване на интерпретатор;
- подходящ за скриптове и бързо приложение;
- подходящ за много платформи.

За овладяване на основите на програмиране на Python е необходимо последователно да се изучат:

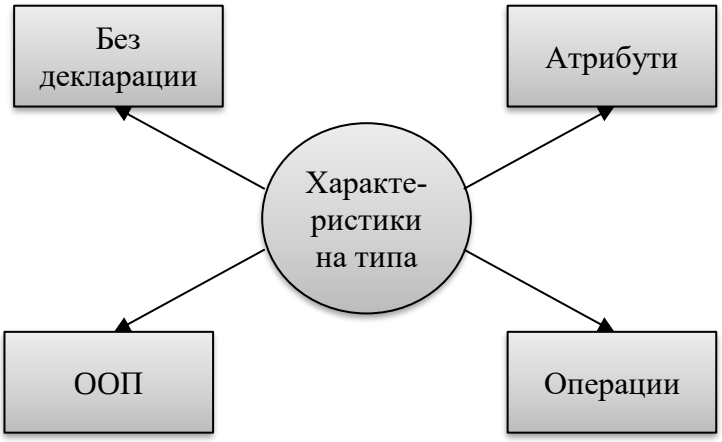
1. Типове данни
2. Управление на изпълнението на програмата
3. Функции
4. Работа с файлове
5. Обекти и класове

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

1. Типове данни в Python

Всички стойности на данни в Python са представени от обекти и всеки обект или стойност има типове данни.

Типовете данни в Python се определят с четири основни характеристики (фиг. 3.1).



Фиг. 3.1. Основни характеристики на типовете данни в Python

1. **ООП**: Първата основна характеристика на типовете данни в Python се определя от това, че Python е напълно обектно ориентиран език, а не статичен език за програмиране.
2. **Без декларации**: Python позволява променливите да не се декларират преди да се използват или да се декларира техния тип

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

3. **Атрибути**: Типът данни определя от една страна атрибутите на обекта и техните елементи, и от друга - дали обектът може да бъде променен.

4. **Операции**: Типът на обекта определя какви операции поддържа обекта, или, с други думи, какви операции може да се извършват със стойностите на данните.

Python поддържа осем вградени типа данни: булев; числов; стрингове; байтове и масиви от байтове; списъци; комплект; множества и речници.

Кратко описание и примери за приложението им е показано в таблица 3.1.

Таблица 3.1. *Кратко описание и примери за приложението на основните типове данни в Python*

Тип данни	**Кратко описание и видове**	**Примери**

Булев
Boolean
	Тип, който заема само две стойности истина или лъжа.
	True/False
		if (number % 2) = 0:
		even = True
		else:
		even = False

Числов Numbers	Числовите типове могат да бъдат цели числа (Integers), реални числа (Floats), дроби (Fraction) и	a = 7 b = 7.3 Fraction('3/7')

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

	комплексни числа (Complex).	c = 7 + 3j
Стрингове Strings	Последователност от Unicode символи	s = “This is a string.”
Байт и масив от байтове Byte & ByteArray	Състои се от единични байтове	m = ‘a’ n = ‘a\nb\nc\n’
Списък List	Подредена последователност от стойности	L = [3, 1.2, “Python”]
Комплект Tuple	Подредена непроменлива последователност от стойности. Може да се разглежда и като наредена n-торка от стойности.	t = [3.2, “Tuple”, 2]
Множества Set	Неподредено множество от стойности	week = {“Mon”, “Tue”, ‘Wed”, “Thu”, “Fri”, “Sat”, “Sun”}
Речник Dictionary	Неподредено множество от двойки ключ-стойност	d = {‘value’:7, ‘key’:127}

Използването на основните типове данни в Python ще бъде пояснено с кода от фиг. 3.2.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

```
number = [1,2,3,4,5]
boolean = 3 in number #Boolean
print(boolean)

num1 = 5**3 #Numbers
num2 = 31//3
num3 = 31/3
print('num1 is',num1)
print('num2 is',num2)
print('num3 is',num3)

str1 = "Welcome" #Strings
str2 = " to AI Programming with Python"
str3 = str1 + str2
print('str3 is',str3)
print(str3[0:13])
print(str3[-6:])
print(str3[:-6])

names = ["Maria", "Ivan", "Petko Petrov", "Petar"] #Lists
print(len(names))
print(names)
names.append('Silvia')
print(names)
names.insert(2, 'Atanas')
print(names)

sports_tuple = ('Cricket', 'Basketball', 'Football') #Tuples
sports_list = list(sports_tuple)
sports_list.append('Baseball')
print(sports_list)
print(sports_tuple)
book = { #Dictionary
  "name": "Pod Igoto",
  "author": "Ivan Vazov",
  "year": 1894,
}
Print(book)
book["year"]=2017 #Modifying
print('Poslednoto izdanie na knigata e', book)
```

Фиг. 3.2. Примерен код за използването на основните типове данни в Python

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Изходът от изпълнението на кода от фигура 3.2 е показан на фиг. 3.3.

```
True

num1 is 125
num2 is 10
num3 is 10.333333333333334

str3 is Welcome to AI Programming with Python
Welcome to AI
Python
Welcome to AI Programming with

4
['Maria', 'Ivan', 'Petko Petrov', 'Petar']
['Maria', 'Ivan', 'Petko Petrov', 'Petar', 'Silvia']
['Maria', 'Ivan', 'Atanas', 'Petko Petrov', 'Petar', 'Silvia']

Списъкът е  ['Cricket', 'Basketball', 'Football', 'Baseball']
Комплектът е  ('Cricket', 'Basketball', 'Football')

{'name': 'Под игото', 'author': 'Иван Вазов', 'year': 1894}
Последното издание на книгата е {'name': 'Под игото', 'author':
'Иван Вазов', 'year': 2017}
```

Фиг. 3.3. Изход от примерния код за използването на основните типове данни в Python от фиг. 3.2

2. Управление на изпълнението на програмата

Управлението на изпълнението на програмата позволява да се дефинира определена последователност от оператори с цел да се имитират

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

ситуации от реалния свят. За това е необходимо всеки език да притежава оператори, които да управляват изпълнението на програмите, като оператори за разклонение и цикли, които в общия случай се наричат „управление на програмите“.

2.1. Оператор за разклонение `if`

Съставният оператор `if` на Python позволява условно да се изпълняват блокове от оператори (`statement`).

Синтаксисът на оператора *if* е показан на фиг. 3.4. а., а пример за неговото използване е показан на фиг. 3.4. б. Примерът въвежда две числа и проверява кое от тях е по-голямо или дали двете числа са равни.

Операторът *if* може да има нула или повече *elif* части. Частта *else* е незадължителна. Ключовата дума *elif* е съкратено от *else if* и е полезна за избягване на излишния отстъп. Поредицата if . . .elif . . . elif . . . замества конструкциите switch или case, които имат другите езици за програмиране.

Операторите `if` могат да се влагат един в друг. Две и повече условия могат да се комбинират с логическите оператори И (`and`) и ИЛИ (`or`). Операторите *`if`* не може да бъде без съдържание в тялото (`statement`), ако е необходимо да се използва какъв *`if`* оператор, то е необходимо да се използва ключовата дума `pass` в тялото.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

```
if expression:
     statement (s)
elif expression:
     statement (s)
elif expression:
     statement (s)
...
else:
    statement (s)
```

а)

```
a = input("Input number a: ")
b = input("Input number b: ")
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

б)

Фиг. 3.4. а. Синтаксис на оператора *if*

б., Пример за използване на оператора *if*

3. Цикли

В Python има два вида цикли while и for.

Цикълът while се изпълнява дотогава, докато условието (expression) е вярно. В Python, подобно на C++, всяка ненулева стойност е истина и само нула е лъжа. Също така, условието може да бъде символен низ или списък и каквато и да е редица (sequence). Всичко с ненулева дължина е истина, а празните редици са лъжа.

На фиг. 3.5. а. е показан синтаксиса на цикъла while , а пример за пресмятане и отпечатване на редицата на Фибоначи е представен на фиг. 3.5.б.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

```
while expression:
      statement (s)
```

а)

```
# Редица на Фибоначи:
a, b = 0, 1
while b < 50:
    print (b, end = ' ')
    a, b = b, a+b
```

б)

Фиг. 3.5. а. Синтаксис на цикъла *while*

б., Пример за използване на цикъла *while*

Операторът for в Python се различава от този, който се използва в C++. Вместо да се дава възможност на потребителя да определя стъпката на итерация и условието за спиране, като в C++, в Python операторът for итерира върху елементите на каквато и да е редица в реда, по който те се появяват в нея.

На фиг. 3.6. а. е показан синтаксиса на цикъла for , а а пример за използване на вложени оператори for е представен на фиг. 3.6.б.

```
for target in iterable:
    statement (s)
```

а)

```
feature = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]
for x in feature:
  for y in fruits:
    print(x, y)
```

б)

Фиг. 3.6. а. Синтаксис на цикъла *for*

б., Пример за използване на вложени цикли *for*

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

4. Оператори break, continue и else в циклите

Операторът break, както в C++, излиза от най-вътрешния for или while цикъл. Операторът continue, също както в C++, продължава със следващата итерация в цикъла. Циклите могат да имат клауза else. Тя се изпълнява когато цикълът завърши чрез изчерпване на списъка при for или когато условието стане лъжа при while, но не и когато цикълът завърши с оператор break. Пример за използване на оператора break е демонстриран в цикъла от фиг. 3.7, който търси и отпечатва прости числа.

```
for n in range(2, 12):
    for x in range(2, n):
        if n % x == 0:
            print (n, "е равно на", x, '*', n/x)
            break
    else:
        print (n, "е просто число"
```

а)

```
2 е просто число
3 е просто число
4 е равно на 2 * 2.0
5 е просто число
6 е равно на 2 * 3.0
7 е просто число
8 е равно на 2 * 4.0
9 е равно на 3 * 3.0
10 е равно на 2 * 5.0
11 е просто число
```

б)

Фиг. 3.7. Пример за използване на оператора break в цикъл, който търси и отпечатва прости числа **а.** Код на Python **б.** Изход от програмата

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.