Технически Университет – София,
Факултет „Компютърни системи и технологии“

Упражнение 2 – Съставяне, въвеждане и настройка на програми, илюстриращи операциите в езика и основните типове данни

Пример 1 – първа програма

```
# first pyton code
print('Hello world !!!')
```

Програмата е последователност от команди, които указват на компютъра какво да прави, за да се постигне даден резултат.

В нашия пример програмата се състои от една команда, която принтира на конзолата текста ***Hello world !!!***

Първият ред от програмата представлява коментар.

Програмите на Python са файлове с разширение .py

В езика Python не се слага точка и запетая (;) в края на командите. Командите се разделят една от друга чрез нов ред или индентация (отстъп).

Понятие за променлива

Променливата в програмните езици има за цел да съхранява данни. Променливите са именувани области от паметта, които пазят данни от определен тип, например число или текст. Всяка една променлива в Python има име и стойност.

Пример2: дефиниране на променлива и присвояване на стойност:

```
a = 10
b = 2
c = a * b
print(a + b)
print(c)
```

Типове данни

В Python съществуват типове данни. При присвояване на стойност на променливата типът данни се избира автоматично, съобразно присвоената стойност .

Вградени типове : булев, низ, цяло число , число с плаваща запетая

Колекции : списък, кортеж(неизменяем списък), речник

bool- логически тип данни. Може да съдържа само 2 стойности: true или false, които съответстват на числата 1 и 0.

complex- комплексни числа

- ✓ Това са числа, които имат **реална** и **имагинерна** част.
- ✓ В Python се записват с буквата **j** за имагинерната част.

dict- речник. Това е структура от данни в Python, която съхранява информация под формата **ключ: стойност**. Ключът е думата, която търсиш. Стойността е обяснението или значението на тази дума.

```
student = {
   "име": "Иван",
   "възраст": 21,
   "специалност": "Информатика"
}
print(student["име"])  # Иван
print(student["възраст"])   # 21
```

float- е тип данни за **реални (дробни) числа** — числа с десетична част 3.14, -0.5, 2.0

int- цели числа. Размерът на числото може да бъде толкова голям , колкото позволява размерът на достъпната оперативна памет

list- списък. Аналогичен е на масива в другите езици за програмиране Може да съдържа различни типове данни и се **променя** (можем да добавяме/изтриваме елементи).

Подрежда елементите в **ред**, който се запазва.

fruits = ["ябълка", "банан", "череша"]

set- множество

- ✓ Съдържа набор от **уникални елементи** (няма повтаряне).
- ✓ Няма запазен ред на елементите.
- ✓ Подходящ е, ако искаме да премахнем дубликати или да правим математически операции (обединение, сечение).

numbers = {1, 2, 3, 3, 2}

print(numbers) # {1, 2, 3} → дубликатите изчезват

str- низ

- ✓ Това е **текст** в Python.
- ✓ Представлява поредица от символи (букви, цифри, знаци).
- ✓ Записва се в **кавици**: '...' или "...".
- ✓ Можем да работим с него като с масив от букви – да взимаме по индекс, да режем, да съединяваме.

tuple- кортеж (наредени n-торки)

- ✓ Подобен е на **списък (list)**, но с една ключова разлика → **не може да се променя** (immutable).
- ✓ Използва се, когато искаме да съхраним няколко стойности заедно и да сме сигурни, че няма да се променят.
- ✓ Подрежда елементите в **определен ред**.

point = (3, 4)

colors = ("red", "green", "blue")

print(point[0]) # 3

print(colors[2]) # blue

- ✓ **Броенето започва от 0**:

  point[0] → първият елемент

  point[1] → вторият елемент

✓ Кортеж може да съдържа различни типове

data = ("Иван", 21, True)

✓ Кортеж с един елемент трябва да има запетая

single = (5,)

Всички типове данни в Python могат да се разделят на : изменяеми и неизменяеми. Към неизменяемите типове данни спадат числата, низовете, кортежите. Към изменяемите се отнасят списъците и речниците.

Оператори

Операторите извършват определени действия с данни. Например математическите оператори изпълняват аритметични изчисления.

Оператор за присвояване =

За присвояване на стойност се използва оператора =

На променливите може да бъде присвоена :

- Обикновена стойност (константата)

a = 1 ***# на променливата a се присвоява стойност 1***
name = ' Mladen' ***# на променливата се присвоява константата Mladen***

-стойност на друга променлива

b = a

- Резултат от изчисление на израз

c = a + b * 5

- Резултат от действието на функция

sum=func(a)

Математически оператори

-събиране +

-изваждане -

-умножение *

-обикновено деление /

-деление с остатък //

-остатък от деление %

Оператори за работа с последователности

+ конкатенация- обединява две последователности

Пример: 'Hello '+ 'World ' -> Hello World ***# низове***

[1,2,3] + [4,5,6] -> [1,2,3,4,5,6] ***# списъци***

*** повторение -създава нова последователност**

Пример: 'b '*4 -> ' bbbb '

[5] * 3 -> [5,5,5]

In проверка за наличие

Пример: ' e ' in 'hello' -> True

' s ' in 'hello' -> False

Побитови оператори- използват се за манипулиране на отделните битове

~ побитова инверсия(стойността на бита се променя-> 1 става 0, 0 става 1)

& побитово И **(&)** в Python е оператор, който извършва логическа операция И върху всеки бит от двете числа. Резултатът е 1 само когато и двата съответстващи бита са 1.

a = 5 *# 101 в двоичен вид*

5 // 2 = 2, 5 % 2 = 1 → запазваш 1

2 // 2 = 1, 2 % 2 = 0 → запазваш 0

1 // 2 = 0, 1 % 2 = 1 → запазваш 1

1 - 2*0 = 1

Четем остатъците отдолу нагоре: **101**.

a = 5 *# 101 в двоичен вид*

b = 3 *# 011 в двоичен вид*

result = a & b *# 001 в двоичен вид = 1*

Таблица на истинност за &

Бит 1	Бит 2	Резултат
0	0	0
0	*1*	*0*
1	0	0
1	1	1

| побитово ИЛИ

Побитово ИЛИ (|) в Python е оператор, който извършва логическа операция ИЛИ върху всеки бит от двете числа. Резултатът е 1 когато поне един от съответстващите битове е 1.

Побитовото ИЛИ | (побитово OR) сравнява двата операнда **бит по бит**. За всеки бит:

- ✓ 0 | 0 = 0
- ✓ 0 | 1 = 1
- ✓ 1 | 0 = 1
- ✓ 1 | 1 = 1

```
a = 5   # 101 в двоичен вид
b = 3   # 011 в двоичен вид
result = a | b  # 111 в двоичен вид = 7
```

^ **побитово изключващо ИЛИ** или **XOR** в Python е оператор, който извършва логическа операция изключващо ИЛИ върху всеки бит от двете числа. Резултатът е 1 само когато битовете са различни (един е 0, другият е 1).

```
a = 5   # 101 в двоичен вид
b = 3   # 011 в двоичен вид
result = a ^ b  # 110 в двоичен вид = 6
```

Таблица на истинност за ^

Бит 1	Бит 2	Резултат
0	0	0

0 1 1

1 0 1

1 1 0

<< изместване вляво

- ✓ Всички битове се местят наляво.
- ✓ Отдясно се добавят нули.
- ✓ Числото става **по-голямо**.
- ✓ Това е същото като умножение по 2^n.

Пример:
5 = 101 (двойчно)
5 << 1 → 1010 = 10
5 << 2 → 10100 = 20

>> изместване вдясно

- ✓ Всички битове се местят надясно.
- ✓ Най-десните битове се изхвърлят.
- ✓ Числото става **по-малко**.
- ✓ Това е същото като целочислено деление на 2^n.

Пример:
20 = 10100 (двойчно)
20 >> 1 → 1010 = 10
20 >> 2 → 101 = 5

Печатане на резултат на екрана

Функция **print (…)**- с нея можем да принтираме - стойността на променлива, текст или число

```
print(33)   # печатане на число
print('Hello World!')   # печатане на текст
word ='Hello World!'
print(word)      # печатане на стойност на променлива
```

Четене на потребителски вход

Използваме вградената **функция input(…)** за четене на текстов ред от конзолата и **функция int(…)** за преобразуване на текстова стойност към цяло число или **функция float(…)** за преобразуване на текстова стойност към дробно число.

Ако не преобразуваме входа към число, за програмата всяко едно число ще бъде просто текст, с който не можем да извършваме аритметични операции. При извикването на input (…) можем да подадем подканващо съобщение за потребителя, с което му казваме какво трябва да въведе,

Например: **s = int(input('Size : '))**

- ✓ input('Size : ') — показва подсказка Size : и чака потребителят да напише нещо и да натисне Enter. Всичко, което напише, се връща като текст (стринг).
  Пример: ако напише 5, input() връща "5" (тип str).
- ✓ int(...) — взима този стринг и се опитва да го превърне в цяло число. За "5" това става и резултатът е 5 (тип int).
- ✓ Резултатът се записва в променливата s. Така s става числото 5 и вече може да се използва в аритметика или цикли.

Отпечатване на форматиран текст в Python

В езика Python има няколко начина да отпечатаме форматиран текст, т.е. текст, смесен с числа, стойности на променливи и изрази.

1.Конкатенация на текст с оператора +

Вече знаем как да съединяваме текст и числа с оператора +.

```
a = 3
b = 9
print('a = ' + str(a) + ' b = ' + str(b) + ' S = ' + str(a * b))
```

Резултатът е :

a=3 b=9 S=27

Форматиращи низове %d, %s, %f

```
a= 3
b= 9
s= "area"
```

print('a = %d ; b = %d ; %s = %d' % (a, b, s, a * b))

Използваме оператор %, който замества в текста стойности, подадени като поредица от елементи в скоби. Използват се следните основни форматни спецификатори:

%d обозначава цяло число

%f обозначава дробно число

%s обозначава текст

Когато форматираме дробни числа можем да закръгляме до определен брой цифри след десетичната запетая, например с %.3f отпечатваме дробно число с 3 знака след десетичната запетая.

Форматиране с .format(…)

Можем да форматираме текст и числа чрез метод **.format(…)** като използваме номерирани шаблони {0}, {1}, {2} и т.н. Ето пример демонстриращ този метод:

```
a= 3
b= 9
print('a = {}  b = {}  S = {}'.format(a, b, a * b))
```

Форматиране с f-string

Поставяме префикс **f** пред стринга и в него поставяме на желани позиции стойности на променливи и изрази във фигурни скоби **{ }** .

Пример :

```
a= 3
b= 9
print(f'a = {a} ; b = {b} ; area = {a * b}')
:.2f – закръгляване два знака след запетаята
print(f'a = {a} ; b = {b:.2f} ; area = {a * b:.2f}')
```

Задачи :

1. Напишете програма, която чете от конзолата три числа a, b и h и пресмята лицето на трапец с основи a и b и височина h.

   (a + b) * h / 2

   Принтирайте резултата на екрана, като го закръглите до втория знак след десетичната запетая.

2. Напишете програма, която чете от конзолата число r и пресмята и отпечатва лицето и периметъра на окръжност с радиус r. Принтирайте резултата, като го закръглите до 3 знака след десетичната запетая.

3. Напишете програма, която да подкани потребителя да въведе броя на часовете и тарифа за час. Да се изчисли и принтира брутното заплащане.

   Примерен вход:

   Enter Hours: 35

   Enter Rate: 2.75

   Изход:

   Pay: 96.25