Функции

Функция –Блок от код, който има име и който може да бъде изпълнен посредством указване на името в правилен контекст (извикване на функция). Преди да можем да извикаме функцията, трябва, естествено да я опише. Python разполага с вградени функции, които може да използваме наготово по всяко време. Такива функции са: abs(), min(), max(), round(), filter() и др.

За да създаваме собствени функции използваме следния синтаксис:
def <име на функция>(параметър: тип):

Описание на функцията

```
def hi():
    print("Hello!")

hi()
```

```
C:\Users\SatellitePro\PycharmProje
Hello!

Process finished with exit code 0
```

На фиг. 1 е създадена функцията „hi“ , без параметър, която отпечатва поздрав. По надолу е извикана с *hi()*

Фиг.1

Функцията може да връща или да не връща стойност. Ако функцията връща стойност, използваме инструкцията *return*, след която се указва връщаната стойност. Изпълнението на инструкцията *return* в тялото на функцията води до прекратяване изпълнението на кода на функцията. Ако при това след return е указана някаква стойност, тя се превръща в резултат на функцията. Не е задължително функцията да връща резултат.

На фиг. 2 е създадена функцията „rect_area“ , с 2 параметъра, която пресмята лице на правоъгълник и връща резултат. По надолу е стойността и е запазена в променлива с *result = rect_area(a, b)*

```
1  # Фунцкия с аргументи, връща резултат
2  def rect_area(height, width):
3      area = height * width
4      return area
5
6  a = int(input())
7  b = int(input())
8
9  # Извикване на функцията
10 result = rect_area(a, b)
11
12 print(result)
```

```
3
4
12
```

Фиг.2

Променливите, които получават свои стойности в тялото на функция, се наричат локални. Те са достъпни само в тялото на функцията. Затова, ако в две различни функции се използват променливи с еднакви названия, на практика това са различни променливи. Всяка от тях е достъпна само в своята функция. Същото правило се отнася и до аргументите на функциите.

Задачи за проекта:

Добавете следния текст в програмата, над менюто. Тази функция служи за преобразуване на числото от двоична към десетична бройна система, като използва специалните функции на Python, за екстракция на цифрите от последната към първата. След това, използвайки for – цикъл се сумира произведението на последната цифра със степента на нейният индекс. (фиг. 3)

```
# Речник за превръщане на стойности 10-15 в букви A-F
HEX_CHISLA = "0123456789ABCDEF"


def bin_to_dec(b):
    desetichno = 0
    for i, digit in enumerate(reversed(b)):
        if digit == '1':
            desetichno += 2 ** i
    return desetichno
```

Фиг. 3

В началото променливата „HEX_CHISLA” служи за речник, в който се съдържат всички цифри и символи от шестнадесетична бройна система.

Следващата функция служи за преобразуване на числа от десетична към двоична бройна система. При нея се използва while – цикъл, който продължава докато остатъка от модулното деление на 2 не остане 0. Всеки остатък се записва в ред. Фиг. 4

Цифрите се записват в локална променлива „res” от тип низ.

```
def dec_to_bin(n):
    if n == 0: return "0"
    res = ""
    while n > 0:
        res = str(n % 2) + res
        n //= 2
    return res
```

Фиг. 4

Следната функция преобразува число от десетична към шестнадесетична бройна система, като използва същата функционалност от първата функция, с разликата, че използва вградена функция *.upper()* за превръщане на малките букви в главни с цел те да бъдат сравнени с речника в началото. Фиг. 5

```
def hex_to_dec(h):
    h = h.upper()
    desetichno = 0
    for i, char in enumerate(reversed(h)):
        val = HEX_CHISLA.index(char)
        dec += val * (16 ** i)
    return desetichno
```

Фиг. 5

При шестнадесетична бройна система, стойности по – големи от 9 се изписват с латинските букви A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.

Функцията dec_to_hex служи за преобразуване на число от десетична към шестнадесетична бройна система, като работи на подобен принцип като функция dec_to_bin, но вместо да дели числото на 2, го дели на 16 и сравнява с речника на шестнадесетичните числа (HEX_CHISLA). Фиг. 6

```
def dec_to_hex(n):
    if n == 0: return "0"
    res = ""
    while n > 0:
        res = HEX_CHISLA[n % 16] + res
        n //= 16
    return res
```

Фиг. 6

За да използваме функциите, трябва да ги „извикаме“. Това става като запишем името на всяка функция, заедно с необходимия параметър. Фиг. 7

```
if izbor == 1:
    v = input("Въведете двоично число: ")
    print(f"-> Резултат (Десетична): {bin_to_dec(v)}")
elif izbor == 2:
    v = int(input("Въведете десетично число: "))
    print(f"-> Резултат (Двоична): {dec_to_bin(v)}")
elif izbor == 3:
    v = input("Въведете шестнадесетично число: ")
    print(f"-> Резултат (Десетична): {hex_to_dec(v)}")
elif izbor == 4:
    v = int(input("Въведете десетично число: "))
    print(f"-> Резултат (Шестнадесетична): {dec_to_hex(v)}")
else:
    print("Невалиден избор!")
```