

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare Informatică și Microelectronică**

**Departamentul Informatică și Ingineria Sistemelor**

**Disciplina: Programarea intereactivă**

**Lucrare de laborator №1**

**Tema: Introducere în limbajul Python. Expresii, variabile, funcţii. Instrucţiunile logice şi decizionale.**

A efectuat:

Pleșca Virgiliu,

*student*

IA-201

A controlat:

Strună Vadim,

*lector univ.*

**Chișinău 2021**

# Task

## Obiective:

* Instalare interpretor / IDE python
* Familiarizare cu limbajul python
  + Execuţie instrucţiuni simple în interpretor
  + Implementare/executare programe simple folosind Python IDLE

## Cerinţe:

* Rezolvaţi în timpul laboratorului problemele de la punctul I și II.

## I Probleme

1. Scrieţi o funcţie mile\_kilometri care ia un parametru *mile* şi returnează numărul de *kilometri*.
2. Scrieţi o funcţie seconde\_totale care ia trei parametri *ore, minute şi secunde* şi returnează numărul total de secunde pentru variabilele ore, minute şi secunde.
3. Scrieţi o funcţie suprafata\_dreptunghi care are doi parametri *lăţimea* şi *înălţimea* corespunzătoare lungimilor laturilor unui dreptunghi şi returnează perimetrul și aria dreptunghiului în cm.
4. Scrieţi o funcţie circumferinta\_cerc care are un parametru *raza* ce corespunde razei unui cerc în cm şi returnează circumferinţa și aria circumferinței unui cerc în cm. Pentru a furniza o aproximare mai mare la precizie pentru variabila π utilizaţi funcţia pi din modulul math. (ex. import math math.pi)
5. Scrieţi o funcţie depozit\_valoare care are trei parametri *valoare\_curenta*, *rata\_anuala* şi *ani\_depozit* şi returnează valoarea\_sumei\_viitoare\_de\_bani a valoare\_curenta investită la dobânda rata\_anuala în procente, formată anual.
6. Scrieţi o funcţie virsta\_persoana care are parametrul ziua de naştere (zi/luna/an), şi va returna vîrsta persoanei în zile.

## II. Probleme

1. Scrieţi o funcţie numar\_par care are ca intrare un număr întreg şi returnează True dacă numărul este par şi False dacă numărul este impar.

2. Scrieţi o funcţie gaseste\_nume care are ca input un nume şi returnează True dacă numele este fie "Ion", "Elena" sau "Maria" şi se întoarce False în celelalte cazuri.

3. Scrieţi o funcţie an\_bisect care ia ca date de intrare anul şi returnează True dacă anul (un număr întreg) este un an bisect, conform calendarului gregorian şi False în caz contrar. (Sfat Intră pe Wikipedia: este un algoritm simplu pentru a stabili dacă un an este un an bisect.)

4. Scrieţi o funcţie intersectare\_interval care ia parametrii a, b, c şi d, şi returnează True dacă intervalele [a, b] şi [c, d] se intersectează şi False în caz contrar. În timp ce acest test poate părea complicat, soluţia este de fapt foarte simplă.

5. Scrieţi o funcţie numele\_si\_virsta care ia ca date de intrare numele (un şir de caractere) şi vîrsta (un număr) şi returnează un şir de forma " % este % de ani." unde procentele sunt numele şi vîrsta sub forma de şir de caractere. Funcţia trebuie să includă o verificare de eroare pentru cazul în care vîrsta este mai mică decât zero. În acest caz, funcţia ar trebui să returneze şirul "Eroare: vârstă invalid".

6. Scrieţi o funcţie afiseaza\_numerele care ia un număr întreg din intervalul [0,100) şi afişează mesajul "Numarul consta din % zeci si % unitati." unde procentele ar trebui să fie înlocuite cu valorile corespunzătoare. Funcţia trebuie să includă o verificare de eroare pentru cazul în care numărul este negativ sau mai mare sau egal cu 100. În aceste cazuri, funcţia ar trebui să afişeze un mesaj de erare "Eroare: Numarul introdus nu este un număr de două cifre.".

7. Scrieţi o funcţie cauta\_nume care are ca parametru prenume, care corespunde unuia dintre ("Vasile", "Mihail", "Ion" sau "Stefan") şi apoi returnează numele lor corespunzătoare ("Moraru", "Perebinos", "Sirghi" sau "Buzurniuc"). Dacă prenume nu se potriveşte cu niciuna dintre aceste şiruri de caractere, returnează eroare "Eroare: Nu este profesor".

8. Pig Latin este un joc de limbaj care implică modificarea cuvintelor printr-un simplu set de reguli. Scrieţi o funcţie pig\_latin care are ca parametru un cuvînt şi aplică următoarele reguli pentru a genera un nou cuvînt în limbajul pig latin.

- Dacă prima literă din cuvînt este o consoană, adăugaţi consoana la sfîrşit şi adăugaţi "ay" la restul cuvîntului. De exemplu, pig\_latin ("pig") va rezulta "igpay".

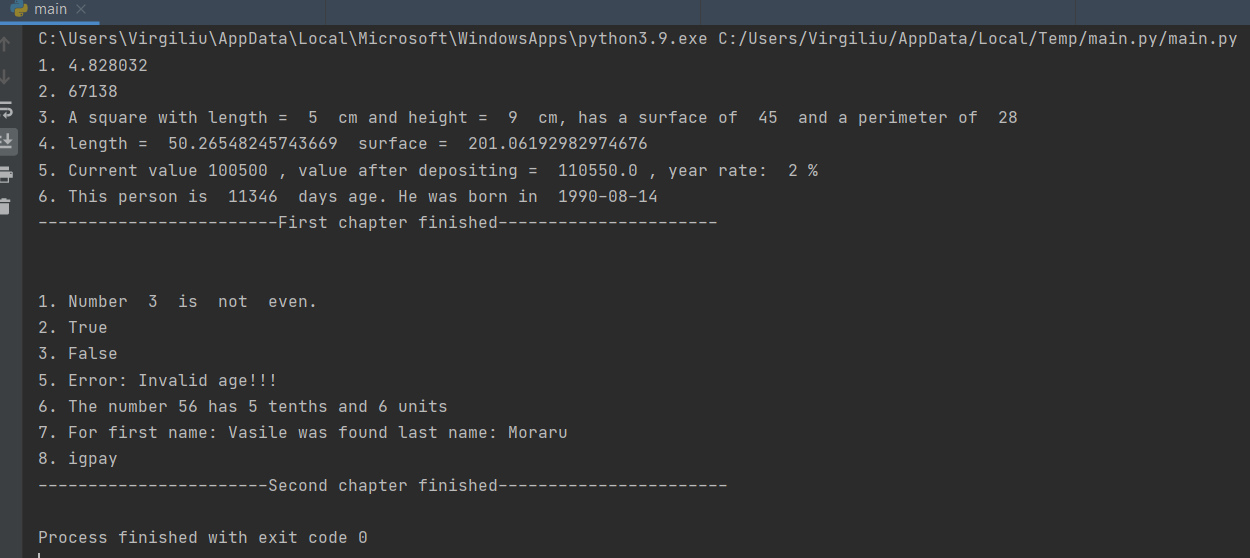
- Dacă prima literă din cuvînt este o vocală, adăugaţi "way" la sfîrşitul cuvîntului. De exemplu, pig\_latin ("awl") returnează "owlway".

# Solving the problems

## Source code

import math  
  
  
# 1.----------------------------------------------------------------  
# Suma numerelor impare pana la n  
def suma\_impare(n):  
 sum = 0  
 for i in range(1, n, 2): # Range 1 : \_n with step 2  
 sum = sum + i  
 return sum  
  
  
print(" 1. ", suma\_impare(10))  
  
  
# 2.---------------------------------------------------------------  
# Verifies if the prod if digits equals to p  
def prod\_verify(n, p):  
 a = n % 10  
 n = n // 10  
  
 b = n % 10  
 n = n // 10  
  
 c = n % 10  
  
 if a \* b \* c == p:  
 return True  
 else:  
 return False  
  
  
def prime\_nums\_prod(\_p):  
 prime = []  
 res = []  
  
  
 # Find the all prime numbers which are of 3 digits  
 for num in range(100, 999):  
 # all prime numbers are greater than 1  
 for i in range(2, num):  
 if (num % i) == 0:  
 break  
 else:  
 prime.append(num)  
  
 # Find all the prime numbers whose digit product equals to \_p  
 for n in prime:  
 if prod\_verify(n, \_p):  
 res.append(n)  
  
 return res  
  
  
print(" 2. ", prime\_nums\_prod(9))  
  
  
# 3.--------------------------------------------------------------------  
# Suma divizorilor lui n  
def sum\_divisors(\_n):  
 s = 0  
 for i in range(1, \_n + 1):  
 if \_n % i == 0:  
 s = s + i  
 return s  
  
  
def friend\_nums(a, b):  
 b = b + 1  
 d = {}  
 res = []  
  
 for i in range(a, b):  
 d[i] = sum\_divisors(i) # Dictionarul cu suma divizorilor pentru fiecare numar din intervalul [a, b]  
  
 for i in range(a, b):  
 for j in range(a, b):  
 if i == j: # Excludem elementele cu valorile tuplului identice  
 continue  
 elif res.\_\_contains\_\_((i, j)) or res.\_\_contains\_\_((j, i)): # Excludem tuplurile cu elementele asemanatoare  
 continue  
 elif d[i] == d[j]:  
 res.append((i, j))  
 return res  
  
  
print(" 3. ", friend\_nums(10, 25))  
  
  
# 4.-------------------------------------------------------------------------------------------------------  
  
def div\_com(nr, div):  
 n = 0  
 print(" 4. ")  
 for i in range(1, nr + 1):  
 for j in range(i + 1, nr + 1):  
 if math.gcd(i, j) == div:  
 n = n + 1  
 print((i, j))  
 print(n, " perechi")  
  
div\_com(20, 6)  
  
# 5.-------------------------------------------------------------------------------------------------------  
  
  
def is\_consec(n):  
 a = n % 10  
 n = n // 10  
  
 b = n % 10  
 n = n // 10  
  
 c = n  
  
 if a > b > c:  
 return True  
 else:  
 return False  
  
  
def sum\_digits(n):  
 # Calculeaza suma cifrelor unui numar  
 a = n % 10  
 n = n // 10  
  
 b = n % 10  
 n = n // 10  
  
 c = n  
 return a + b + c  
  
  
def sum(s):  
 arr = []  
  
 for i in range(100, 1000):  
 if is\_consec(i) and sum\_digits(i) == s:  
 arr.append(i)  
  
 print(" 5. ", arr)  
  
  
sum(18)  
  
# 6.-------------------------------------------------------------------------------  
  
  
def desc(x):  
 arr = []  
 prim = []  
 p = 0  
 i = 0  
  
 # Numerele prime pana la x  
 for num in range(2, x + 1):  
 # all prime numbers are greater than 1  
 if num > 1:  
 for i in range(2, num):  
 if (num % i) == 0:  
 break  
 else:  
 prim.append(num)  
 i = 0  
 p = 0  
 while x > 1:  
 while x % prim[i] == 0:  
 p = p + 1  
 x = x / prim[i]  
  
 if p == 0:  
 i = i + 1  
 continue  
  
 arr.append((prim[i], p))  
 p = 0  
 i = i + 1  
  
 print(" 6. ", arr)  
  
  
desc(512)  
  
# 7.---------------------------------------------------------------------  
  
  
def reverse(x):  
 return int(str(x)[::-1])  
  
  
i = 0  
a = []  
  
  
def \_7():  
 for i in range(10, 100):  
 if i\*\*2 == reverse(reverse(i)\*\*2):  
 a.append(i)  
 print(" 7. ", a)  
  
  
\_7()  
  
# 8.------------------------------------------------------------------  
  
  
def \_8(n):  
 d = []  
 r = []  
  
 # Gasim divizorii proprii  
 for i in range(1, n):  
 k = 0  
 for j in range(1, i):  
 if(i % j == 0 and j != 1 and j != i):  
 k = k + 1  
 d.append((i, k))  
 # Gasim numerele cu cei mai multi divizori proprii  
 max = -1  
 ind = 0  
 for i in range(len(d)):  
 if(d[i][1] >= max):  
 max = d[i][1]  
 ind = i  
 # Numaram cate numere cu nr maxim de div proprii  
 a = []  
 for i in range(len(d)):  
 if(d[i][1] == max):  
 a.append(i)  
  
 for i in a:  
 r.append(d[i])  
 print(" 8. ", r)  
  
\_8(20)  
  
# 9.------------------------------------------------------  
  
  
def \_9(k):  
 i = 1  
 while(True):  
 count = 0  
 for j in range(1, i + 1):  
 if(i % j == 0):  
 count = count + 1  
 if(count == k):  
 print(" 9. ", i)  
 break  
 i = i + 1  
  
\_9(4)  
  
  
# 10.---------------------------------------------------  
  
k = 0  
  
  
def is\_prime(x):  
 global k  
 if x == 2:  
 return True  
 for i in range(2, x):  
 if(x % i == 0):  
 return False  
 elif(i == x - 1):  
 return True  
  
  
r = []  
  
  
def \_10():  
 prime = []  
 r = []  
  
 # generam numere prime  
 for num in range(100, 999):  
  
 for i in range(2, num):  
 if(num % i == 0):  
 break  
 elif(i == num - 1):  
 prime.append(num)  
  
 # verificam care numere sunt inverse si prime  
 for num in prime:  
 if(is\_prime(reverse(num))):  
 if(not r.\_\_contains\_\_((num, reverse(num))) and not r.\_\_contains\_\_((reverse(num), num)) and reverse(num) != num):  
 r.append((num, reverse(num)))  
  
 print(" 10. ", r)  
  
\_10()  
  
# 11.--------------------------------------------------------------------------------------------------------------------  
# Numere perfecte  
def \_11(x):  
 s = 0  
  
 for i in range(1, x):  
 if(x % i == 0):  
 s = s + i  
  
 if(s == x):  
 print(" 11. Numarul", x, "este pefect. Suma divizorilor este:", s)  
 else:  
 print(" 11. Numarul", x, "nu este pefect. Suma divizorilor este:", s)  
  
  
\_11(30)  
  
# 12.------------------------------------------------------------------------------  
  
  
  
def \_12(a, b):  
 r = []  
 for i in range(a, b + 1):  
 for j in range(i, b + 1):  
 if i == reverse(j):  
 r.append((i, j))  
 print(" 12. ", r)  
\_12(10, 40)  
  
# 13.----------------------------------------------------------------------------  
def sum\_digits(x):  
 s = 0  
 for digit in str(x):  
 s = s + int(digit)  
 return s  
  
  
def \_13(n):  
 r = []  
 for i in range(n):  
 if(is\_prime(sum\_digits(i))):  
 r.append(i)  
 print(" 13. ", r)  
  
\_13(25)  
  
# 14.-------------------------------------------------------  
  
def \_14(n):  
 if(n < 1 and n > 1000):  
 print("Erorr: Ati introdus o valoare care nu se incadreaza in interval")  
 return -1  
  
 d = []  
 s = 0  
 for i in range(1, n + 1):  
 if(n % i == 0):  
 d.append(i)  
 s = s + i  
 print(" 14. ", d)  
 print(" Suma divizorilor:", s)  
  
  
\_14(20)  
  
  
# 15.-------------------------------------------------------------------  
  
def \_15(n, m):  
 r = []  
 i = 1  
 while(len(r) < n):  
 if(sum\_digits(i) <= m):  
 r.append(i)  
 i = i + 1  
 print(" 15. ", r)  
  
\_15(10, 4)  
  
  
# 16.----------------------------------------------------------------  
  
def \_16(n):  
 r = []  
 for i in range(1, n):  
 if(i % sum\_digits(i) == 0):  
 r.append(i)  
 print(" 16. ", r)  
  
\_16(25)  
  
# 17.------------------------------------------------------------  
  
# Returneaza  
def closest\_num\_to\_divide(n):  
 i = 2  
 while(True):  
 if(n % i == 0):  
 return i  
 i = i + 1  
  
  
def \_17(n):  
 r = []  
 print(" 17. ")  
 while(True):  
 if(n == 1):  
 break  
  
 print(" ", n, " | ", closest\_num\_to\_divide(n))  
 n = n / closest\_num\_to\_divide(n)  
 print(" ", 1, " |")  
  
\_17(3268)  
  
# 18.--------------------------------------------------------------  
  
def is\_odd(n):  
 if(n % 2 != 0):  
 return True  
 else:  
 return False  
  
def \_18(n):  
 r = []  
 for i in range(n + 1):  
 if(is\_odd(sum\_digits(i))):  
 r.append(i)  
 print(" 18. ", r)  
  
\_18(15)  
  
  
# 19.-------------------------------------------------  
  
def \_19():  
 r = []  
 for i in range(10, 100):  
 cat = i // 15  
 rest = i % 15  
 if cat \*\* 2 == rest:  
 r.append(i)  
 print(" 19. ", r)  
  
\_19()  
  
# 20.---------------------------------------------------------  
  
def \_20(p, q):  
 r = []  
 k = 0  
 for i in range(p, q + 1):  
 if(is\_prime(i)):  
 r.append(i)  
 k = k + 1  
 print(" 20. ", r, "\n Au fost gasite", k, "numere prime")  
  
\_20(10, 25)

## Output



# Conclusion

After doing this laboratory work i have familiarised with Python programming language. Now i am more confident using Python. I think that i understood perfectly basic statements, variable declaration and that stuff which is nearly the same in each of the programming languages. The only thing i had to get used with was the sintax. Also I’ve made a conclusion about Python that it is a language for “lazy people”, because there are lots and lots of librarys thar can do the work for you, the only thing you have to do is to type *pip install <package>* in console and then use it in your projects. In this way people are developing projects in Python with a speed close to the speed of light :). There is even a quote about this: “Sloth is the mother of invention”.