

# Instrucțiuni

# Instrucțiuni

- Instrucțiunea de atribuire

```
x = 1
```

```
x = y = 1 #atriburie multipla
```

# Instrucțiuni

- Instrucțiunea de atribuire

```
x = 1
```

```
x = y = 1
```

```
x, y = 1, 2 #atriburie compusa/tuple assignment
```

# Instrucțiuni

- Instrucțiunea de atribuire

```
x = 1
```

```
x = y = 1
```

```
x, y = 1, 2
```

```
x, y = y, x #!!! Interschimbare
```

# Instrucțiuni

- Instrucțiunea de atribuire

```
x = 1
```

```
x = y = 1
```

```
x, y = 1, 2
```

```
x, y = y, x #!!! Interschimbare
```

```
x, y = min(x,y), max(x,y)
```

```
print("intervalul [" + str(x) + ", " + str(y) + "])"
```

# Instrucțiuni

- Instrucțiunea de decizie (condițională) `if`

Varianta 1. Instrucțiunea de decizie

```
if expresie_logică:  
    instructiuni
```

---

```
x = int(input())  
if x<0:  
    print('valoare incorecta')
```

# Instrucțiuni

- Instrucțiunea de decizie (condițională) `if`

## Varianta 2. Instrucțiunea alternativă

```
if expresie_logică:  
    instructiuni_1  
else:  
    instructiuni_2
```

---

```
if x%2 == 0:  
    print('numar par')  
else:  
    print('numar impar')
```

# Instrucțiuni

- Instrucțiunea de decizie (condițională) `if`

## Varianta 3. Instrucțiuni alternative imbricate

```
if expresie_logică:  
    instructiuni_1  
elif expresie_logică_2:  
    instructiuni_2  
elif expresie_logică_3:  
    ...  
else:  
    instructiuni
```

← poate lipsi



# Instrucțiuni

- Instrucțiunea de decizie (condițională) `if`

```
k = int(input())  
print('ultima cifra a lui 3**',k, 'este',end=" ")
```

# Instrucțiuni

- Instrucțiunea de decizie (condițională) `if`

```
k = int(input())
print('ultima cifra a lui 3**',k, 'este',end=" ")
r = k%4
if r == 0:
    print(1)
elif r == 1:
    print(3)
elif r == 2:
    print(9)
else:
    print(7)
```

- 
- `else` poate lipsi

# Instrucțiuni

- Instrucțiunea de decizie/potrivire multiplă `match`

`match expresie:`

`case tipar1: instructiune`

`case tipar2: instructiune`

`case _: instructiune` *#poate lipsi, similar default*

# Instrucțiuni

- Instrucțiunea de decizie/potrivire multiplă `match`

```
x = int(input())
match x:
    case 1: print("luni")
    case 2: print("marti")
    case 3: print("miercuri")
    case 4|5: print("ultimele doua zile cu ore")
    case _: print("nu avem ore") #poate lipsi
```

# Instrucțiuni

- Instrucțiunea de decizie/potrivire multiplă `match`

```
x = int(input())
match x:
    case n if n<10:
        print("o cifra")
    case n if n<100:
        print("doua cifre")
    case _:
        print("multe cifre")
```

# Instrucțiuni

- Instrucțiunea repetitiva cu test inițial while

`#suma cifrelor unui numar`

# Instrucțiuni

- Instrucțiunea repetitiva cu test inițial while

```
while expresie_logică:  
    instrucțiuni
```

# Instrucțiuni

- Instrucțiunea repetitiva cu test inițial while

```
#suma cifrelor unui numar
```

```
m = n = int(input())
```

```
s = 0
```

```
while n>0:
```



# Instrucțiuni

- Instrucțiunea repetitiva cu test inițial while

```
#suma cifrelor unui numar
```

```
m = n = int(input())
```

```
s = 0
```

```
while n>0:
```

```
    s += n%10
```

```
    n //= 10 #!!nu /
```

```
print("suma cifrelor lui", m, "este",s)
```

# Instrucțiuni

- **Instrucțiunea repetitiva cu test inițial while**
  - while poate avea else
  - Nu există do... while

# Instrucțiuni

- Instrucțiunea repetitiva cu număr fix de iterații (for)

Doar “for each”, de forma

*for variabila in colectie\_iterabila*

de exemplu:

```
for litera in sir:
```

```
for element in lista:
```

# Instrucțiuni

```
s = "abcde"
```

```
for litera in s:
```

```
    litera="a"
```

```
print(s)
```

```
for i in [0,1,2,3,4]:
```

```
    s[i]='a'
```

# Instrucțiuni

```
s = "abcde"
```

```
for litera in s:
```

```
    litera="a"
```

```
print(s)      #nu se modifica, nu da eroare
```

```
for i in [0,1,2,3,4]:
```

```
    s[i]='a'  #eroare
```

```
#TypeError: 'str' object does not support item assignment
```



# Instrucțiuni

- Instrucțiunea repetitiva cu număr fix de iterații (for)

```
for i in [0,1,2,3,4]
```



```
for i in [0,..., n] ????!
```

# Instrucțiuni

- Instrucțiunea repetitiva cu număr fix de iterații (for)

```
for i in [0,1,2,3,4]
```

```
for i in [0,..., n] ????
```



Funcția `range ()`

# Instrucțiuni

- Instrucțiunea repetitiva cu număr fix de iterații (for)

```
for i in [0,1,2,3,4]
```

```
for i in [0,..., n] ????
```



Funcția `range()`

```
for i in range(0,n+1):
```



# Instrucțiuni

- Funcția `range()` – clasa `range`, o secvență (iterabilă)

`range(b)`    =>    de la 0 la  $b-1$

`range(a,b)`   =>   de la  $a$  la  $b-1$

`range(a,b,pas)`   =>    $a, a+p, a+2p...$

↑  
`pas` poate fi negativ

# Instrucțiuni

- Funcția `range()` – clasa `range`, o secvență (iterabilă)

`range(b)`  $\Rightarrow$  de la 0 la  $b-1$

`range(a,b)`  $\Rightarrow$  de la  $a$  la  $b-1$

`range(a,b,pas)`  $\Rightarrow a, a+p, a+2p...$

↑  
`pas` poate fi negativ

- memorie puțină, **un element este generat doar cand este nevoie de el**, nu se memorează toate de la început (secvența este generată element cu element)

# Instrucțiuni

`range(10) =>`

`range(1,10) =>`

`range(1,10,2) =>`

`range(10,1,-2) =>`

`range(1,10,-2) =>`

# Instrucțiuni

`range(10) => 0 1 2 3 4 5 6 7 8 9`

`range(1,10) => 1 2 3 4 5 6 7 8 9`

`range(1,10,2) =>`

`range(10,1,-2) =>`

`range(1,10,-2) =>`

# Instrucțiuni

`range(10) => 0 1 2 3 4 5 6 7 8 9`

`range(1,10) => 1 2 3 4 5 6 7 8 9`

`range(1,10,2) => 1 3 5 7 9`

`range(10,1,-2) => 10 8 6 4 2`

`range(1,10,-2) => vid`

# Instrucțiuni

```
print(*range(1,10,2))
```

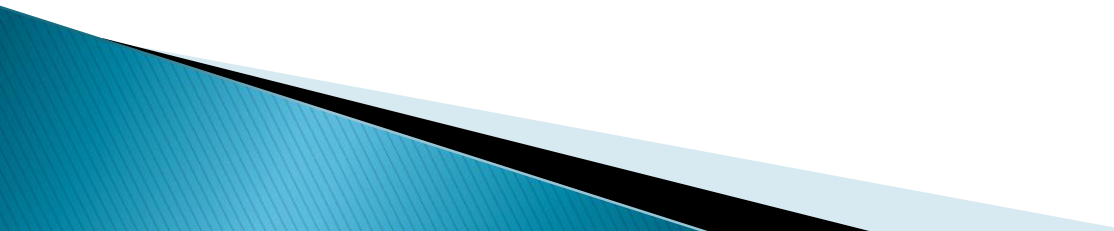
```
print(*range(10,1,-2))
```

```
print(*range(1,10,-2))
```

```
s = "abcde"
```

```
for i in range(len(s)):
```

```
    print(s[i])
```



# Instrucțiuni

- **break, continue + clauza else pentru instrucțiuni repetitive**
  - **break, continue** – aceeași semnificație ca în C

# Instrucțiuni

- **break, continue + clauza else pentru instrucțiuni repetitive**
  - **break, continue** – aceeași semnificație ca în C

```
while True:
```

```
    comanda = input('>> ')
```

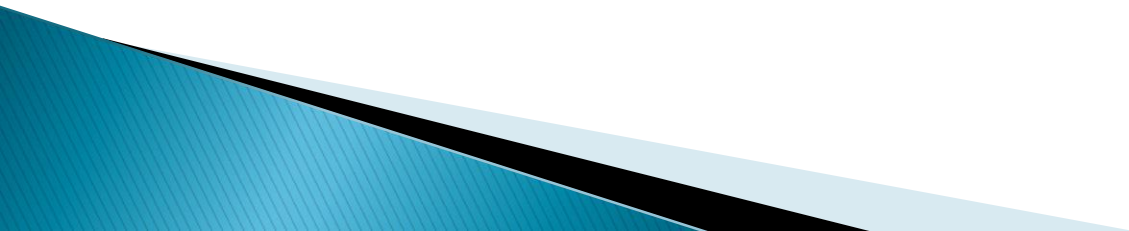
```
    if comanda == 'exit()':
```

```
        break
```



# Instrucțiuni

#primul divizor propriu



# Instrucțiuni

```
#primul divizor propriu
```

```
x = int(input())
```

```
dx = None
```

```
for d in
```



# Instrucțiuni

```
#primul divizor propriu
```

```
x = int(input())
```

```
dx = None
```

```
for d in range(2,x//2+1):
```

```
    if x%d == 0:
```

```
        dx = d
```

```
        break
```



# Instrucțiuni

```
#primul divizor propriu
```

```
x = int(input())
```

```
dx = None
```

```
for d in range(2,x//2+1):
```

```
    if x%d == 0:
```

```
        dx = d
```

```
        break
```

```
if dx: #if dx is not None:
```

```
    print("primul divizor propriu:",dx)
```

```
else:
```

```
    print("numar prim")
```

# Instrucțiuni

Clauza `else` a unei structuri repetitive: nu se executa daca s-a iesit din ciclu cu `break`

# Instrucțiuni

Clauza `else` a unei structure repetitive: nu se executa daca s-a iesit din ciclu cu `break`

```
x = int(input())  
  
for d in range(2,x//2+1):  
    if x%d == 0:  
        print("primul divizor propriu:",d)  
        break  
  
else: #al for-ului, nu al if-ului  
    print("numar prim")
```

# Instrucțiuni

```
#numarul de divizori proprii - cu continue
```

```
x = int(input())
```

```
k = 0
```

```
for d in range(2,x//2+1):
```

```
    if x%d != 0:
```

```
        continue
```

```
    k+=1
```

```
print("numarul de divizori proprii:",k)
```



# Instrucțiuni

**Exercițiu:** Date  $a$  și  $b$ , să se determine cel mai mic număr prim din intervalul  $[a, b]$



# Instrucțiuni

**#cel mai mic număr prim din intervalul [a,b]**

```
a = int(input("a="))
```

```
b = int(input("b="))
```

```
for x in range(a,b+1):
```



# Instrucțiuni

#cel mai mic număr prim din intervalul [a,b]

```
a = int(input("a="))
```

```
b = int(input("b="))
```

```
for x in range(a,b+1):
```

```
    for d in range(2,x//2+1):
```

```
        if x%d == 0:
```

```
            break
```



# Instrucțiuni

#cel mai mic număr prim din intervalul [a,b]

```
a = int(input("a="))
b = int(input("b="))
for x in range(a,b+1):
    for d in range(2,x//2+1):
        if x%d == 0:
            break
    else:
        print(x)
        break
```

# Instrucțiuni

#cel mai mic număr prim din intervalul [a,b]

```
a = int(input("a="))
b = int(input("b="))
for x in range(a,b+1):
    for d in range(2,x//2+1):
        if x%d == 0:
            break
    else:
        print(x)
        break
else:
    print("Nu exista numar prim in interval")
```

# Instrucțiuni

#cel mai mic număr prim din intervalul [a,b]

```
a = int(input("a="))
b = int(input("b="))
for x in range(a,b+1):
    for d in range(2, int(x**0.5)+1):
        if x%d == 0:
            break
    else:
        print(x)
        break
else:
    print("Nu exista numar prim in interval")
```

# Instrucțiuni

#cel mai mic număr prim din intervalul [a,b]

```
a = int(input("a="))
b = int(input("b="))
for x in range(a,b+1):
    for d in range(2, int(x**0.5)+1):#???sqrt?
        if x%d == 0:
            break
    else:
        print(x)
        break
else:
    print("Nu exista numar prim in interval")
```

# Instrucțiuni

- **pass**

```
x = int(input())
```

```
if x < 0:
```

```
    pass #urmeaza sa fie implementat
```

# Funcții predefinite

- **Modulul builtins**

<https://docs.python.org/3/library/functions.html#built-in-funcs>



# Funcții predefinite

- **Conversie**

– constructori `int()`, `float()`, `str()`

```
print(bin(23), hex(23)) #str
```

# Funcții predefinite

- **Matematică**

```
print(abs(-5))
```

```
print(min(5,2))
```

```
x = 3.0
```

```
print(x.is_integer())
```

# Funcții predefinite

- **Matematică**

```
import math
```

```
print(math.sqrt(4))
```

```
print(math.factorial(5))
```

# Funcții predefinite

- **Matematică**

```
from math import sqrt
```

```
print(sqrt(5))
```

```
print(factorial(5)) #eroare, se putea: from math  
import sqrt, factorial
```

```
print(math.sqrt(4)) #eroare
```

```
print(math.factorial(4)) #eroare
```

