

Base Types	
int 783 0 -192 0b010 0o642 0xF3	null binary octal hexa
float 9.23 0.0 -1.7e-6	
bool True False	x10 ⁶
str "One\nTwo"	Multiline string: """\tX\tY\tZ 1\t2\t3"""
bytes b'toto\xfe\x775'	escaped new line 'I\tm' escaped tab hexadecimal octal
	immutable

Container Types	
list [1, 5, 9] ["x", 11, 8.9]	["mot"]
tuple (1, 5, 9) 11, "y", 7.4	("mot",)
Non modifiable values (immutable)	expression with just commas → tuple
str bytes (ordered sequences of chars / bytes)	
key containers, no a priori order, fast key access, each key is unique	
dictionary dict {"key": "value"} (key/value associations) {1: "one", 3: "three", 2: "two", 3.14: "pi"}	dict(a=3, b=4, k="v")
collection set {"key1", "key2"} # keys=hashable values (base types, immutables...)	{1, 9, 3, 0} set()
	frozenset immutable set

Identifiers	
for variables, functions, modules, classes... names	
a-zA_Z_ followed by a-zA_Z_0-9	
diacritics allowed but should be avoided	
language keywords forbidden	
lower/UPPER case discrimination	
o a toto x7 y_max BigOne	
o by and for	
= Variables assignment	
1) evaluation of right side expression value	
2) assignment in order with left side names	
# assignment ↔ binding of a name with a value	
x=1.2+8+sin(y)	
aa=bb=c=0 assignment to same value	
y, z, r=9.2, -7.6, 0 multiple assignments	
a, b=b, a values swap	
a, *b=seq unpacking of sequence in *a, b=seq item and list	
x+=3 increment ↔ x=x+3	
x-=2 decrement ↔ x=x-2	
x=None < undefined > constant value	
del x remove name x	

Conversions	
type(expression)	
int("15") → 15	can specify integer number base in 2 nd parameter
int("3f", 16) → 63	truncate decimal part
int(15.56) → 15	rounding to 1 decimal (0 decimal → integer number)
float("-11.24e8") → -112400000.0	representation string of x for display (cf. formating on the back)
round(15.56, 1) → 15.6	bool(x) False for null x, empty container x, None or False x; True for other x
str(x) → ... representation string of x for display (cf. formating on the back)	chr(64) → '@' ord('@') → 64 code ↔ char
repr(x) → ... literal representation string of x	bytes([72, 9, 64]) → b'H\t@'
list("abc") → ['a', 'b', 'c']	list([1, 2, 3]) → [1: 'one', 3: 'three']
dict([(3, "three"), (1, "one")]) → {1: 'one', 3: 'three'}	set(["one", "two"]) → {'one', 'two'}
separator str and sequence of str → assembled str	':'.join(['toto', '12', 'pswd']) → 'toto:12:pswd'
str splitted on whitespaces → list of str	"words with spaces".split() → ['words', 'with', 'spaces']
"words with spaces".split(str) → list of str	"1, 4, 8, 2".split(",") → ['1', '4', '8', '2']
sequence of one type → list of another type (via comprehension list)	[int(x) for x in ('1', '29', '-3')] → [1, 29, -3]

for lists, tuples, strings, bytes...						
negative index	-5	-4	-3	-2	-1	
positive index	0	1	2	3	4	
lst=[10, 20, 30, 40, 50]						
positive slice	0	1	2	3	4	5
negative slice	-5	-4	-3	-2	-1	

Access to sub-sequences via lst[start slice:end slice:step]

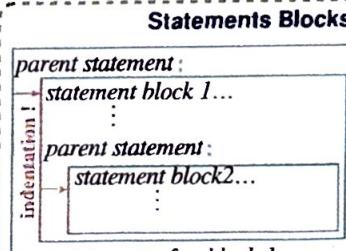
lst[:-1]→[10, 20, 30, 40] lst[::-1]→[50, 40, 30, 20, 10] lst[1:3]→[20, 30] lst[:3]→[10, 20, 30]
 lst[1:-1]→[20, 30, 40] lst[::2]→[50, 30, 10] lst[-3:-1]→[30, 40] lst[3:]→[40, 50]
 lst[::2]→[10, 30, 50] lst[:]→[10, 20, 30, 40, 50] shallow copy of sequence

Missing slice indication → from start / up to end.

On mutable sequences (list), remove with del lst[3:5] and modify with assignment lst[1:4]=[15, 25]

Boolean Logic	
Comparators: < > <= >= == !=	(boolean results) ≤ ≥ = ≠
a and b logical and both simultaneously	
a or b logical or one or other or both	
# pitfall : and and or return value of a or b (under shortcut evaluation).	
⇒ ensure that a and b are booleans.	
not a logical not	
True	
False	True and False constants

floating numbers... approximated values
 Operators: + - * / // % **
 Priority (...) x ÷ ↑ ↑ a^b
 integer ÷ ÷ remainder
 @ → matrix × `numpy`
 (1+5.3)*2→12.6
 abs(-3.2)→3.2
 round(3.57, 1)→3.6
 pow(4.3)→64.0



angles in radians Maths

```

from math import sin, pi
sin(pi/4)→0.707...
cos(2*pi/3)→-0.4999...
sqrt(81)→9.0 √
log(e**2)→2.0
ceil(12.5)→13
floor(12.5)→12
  
```

modules math statistics random

Sequence Containers Indexing

Individual access to items via lst[index]

len(lst)→5

index from 0 (here from 0 to 4)

1st[0]→10 ⇒ first one 1st[1]→20
 1st[-1]→50 ⇒ last one 1st[-2]→40

On mutable sequences (list), remove with del lst[3] and modify with assignment
 lst[4]=25

lst[3]=25

lst[1:3]→[20, 30] lst[:3]→[10, 20, 30]
 lst[-3:-1]→[30, 40] lst[3:]→[40, 50]

shallow copy of sequence

Modules/Names Imports

```

module true⇒file truc.py
from monmod import nom1, nom2 as fct
  →direct acces to names, renaming with as
import monmod →acces via monmod.nom1...
# modules and packages searched in python path (cf sys.path)
statement block executed only if a condition is true
  
```

Conditional Statement

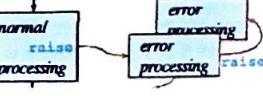
if logical condition:
 — statements block

Can go with several elif, elif... and only one final else. Only the block of first true condition is executed.

with a var x:
 if bool(x)==True: ⇔ if x:
 if bool(x)==False: ⇔ if not x:
 state="Retired"
 else:
 state="Active"

Signaling an error:

raise Exception(...)



Exceptions on Errors

Errors processing:
 try:
 —→ normal processing block
 except Exception as e:
 —→ error processing block

condition is true

while logical condition :
statements block



s = 0 initializations before the loop
i = 1 condition with at least one variable value (here 1)
while i <= 100:
 s = s + i2**
 i = i + 1 make condition variable change!
print("sum:", s)

print("v=", 3, "cm :", x, ", ", y+4)

items to display : literal values, variables, expressions

print options:

- **sep=" "** items separator, default space
- **end="\n"** end of print, default new line
- **file=sys.stdout** print to file, default standard output

S = input("Instructions:")

input always returns a string, convert it to required type
(cf. boxed Conversions on the other side).

len(c) → items count

min(c) max(c) sum(c) Note: For dictionaries and sets, these operations use keys.

sorted(c) → list sorted copy

val in c → boolean, membership operator in (absence not in)

enumerate(c) → iterator on (index, value)

zip(c1, c2...) → iterator on tuples containing c1 items at same index

all(c) → True if all c items evaluated to true, else False

any(c) → True if at least one item of c evaluated true, else False

Specific to ordered sequences containers (lists, tuples, strings, bytes...)

reversed(c) → inverted iterator

c*5 → duplicate c+c2 → concatenate

c.index(val) → position

Generic Operations on Containers

copy.copy(c) → shallow copy of container

copy.deepcopy(c) → deep copy of container

modify original list

lst.append(val)

add item at end

lst.extend(seq)

add sequence of items at end

lst.insert(idx, val)

insert item at index

lst.remove(val)

remove first item with value val

lst.pop([idx]) → value

remove & return item at index idx (default last)

lst.sort() lst.reverse() sort / reverse liste in place

Operations on Lists

Operations on Dictionaries

d[key]=value

d.clear()

d[key] → value

del d[key]

d.update(d2) update/add

Associations

d.keys() → iterable views on

Associations

d.values() → keys/values/associations

d.items() → key/value pairs

d.pop(key[,default]) → value

d.popitem() → (key,value)

d.get(key[,default]) → value

d.setdefault(key,default) → value

Operators:

| → union (vertical bar char)

& → intersection

- ^ → difference/symmetric diff.

< <= > >= → inclusion relations

Operators also exist as methods.

Operations on Sets

s.update(s2) s.copy()

s.add(key) s.remove(key)

s.discard(key) s.clear()

s.pop()

storing data on disk, and reading it back

f = open("file.txt", "w", encoding="utf8")

file variable name of file

for operations on disk

(+path...)

cf. modules os, os.path and pathlib

opening mode

▫ 'r' read

▫ 'w' write

▫ 'a' append

encoding of

chars for text

files:

utf8 ascii

writing

f.write("coucou")

f.writelines(list of lines)

read empty string if end of file reading

f.read([n]) → next chars

if n not specified, read up to end!

f.readlines([n]) → list of next lines

f.readline() → next line

text mode t by default (read/write str). possible binary mode b (read/write bytes). Convert from/to required type!

f.close() # dont forget to close the file after use!

f.flush() write cache

f.truncate([taille]) resize

reading/writing progress sequentially in the file, modifiable with:

f.tell() → position

f.seek(position[,origin])

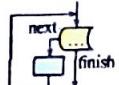
Very common: opening with a guarded block (automatic closing) and reading loop on lines

with **open(...)** as f:
for line in f :

item of a container or iterator

Loop Control
break immediate exit
continue next iteration
t → use block for normal loop exit

for var in sequence:
statements block



Go over sequence's values

t = "Some text" initializations before the loop
cnt = 0 loop variable, assignment managed by for statement
e = "e" character to count
cnt == "e": condition to check
cnt = cnt + 1 increment
print("found", cnt, "'e'") print result

Algo: count number of e in the string.

Display

loop on dict/set ⇒ loop on keys sequences
use slices to loop on a subset of a sequence

Go over sequence's index

modify item at index

access items around index (before / after)

lst = [11, 18, 9, 12, 23, 4, 17]

lost = []

for idx in range(len(lst)):

val = lst[idx]

if val > 15:

lost.append(val)

lst[idx] = 15

print("modif:", lst, "-lost:", lost)

Algo: limit values greater than 15, memorizing of lost values.

Go simultaneously on sequence's index and values:

for idx, val in enumerate(lst):

Integers Sequences

start default 0, fin not included in sequence, pas signed default 1

range(5) → 0 1 2 3 4 **range(2, 12, 3) → 2 5 8 11**

range(3, 8) → 3 4 5 6 7 **range(20, 5, -5) → 20 15 10**

range(len(seq)) → sequence of index of values in seq

range provides an immutable sequence of int constructed as needed

function name (identifier)

named parameters

def fact(x, y, z):

 """documentation"""

statements block, res computation, etc.

return res ← result value of the call, if no computed result to return: **return None**

parameters and all variables of this block exist only in the block and during the function call (think of a "black box")

Advanced: **def fact(x, y, z, *args, a=3, b=5, **kwargs):**

*args variable positional arguments (→tuple), default values,

**kwargs variable named arguments (→dict)

Function Call

fct() → **fct**

storage/use of one argument per returned value

this is the use of function Advanced:

name with parenthesis *sequence

which does the call *dict

s.startswith(prefix[,start[,end]])

s.endswith(suffix[,start[,end]])

s.strip([chars])

s.count(sub[,start[,end]])

s.partition(sep) → (before, sep, after)

s.index(sub[,start[,end]])

s.find(sub[,start[,end]])

s.is_(...) tests on chars categories (ex. s.isalpha())

s.upper() s.lower() s.title() s.swapcase()

s.casefold() s.capitalize() s.center([width,fill])

s.ljust([width,fill]) s.rjust([width,fill]) s.zfill([width])

s.encode(encoding) s.split([sep]) s.join(seq)

formatting directives values to format Formating

"modele{} {} {}".format(x, y, r) → str

"{selection:formatting!conversion}"

Selection :

2 nom

0.nom

4[key]

0[2]

Formatting :

fill char alignment sign min width precision-maxwidth type

<> ^ = + - space 0 at start for filling with 0

integer: b binary, c char, d decimal (default), o octal, x or X hexa...

float: e or E exponential, f or F fixed point, g or G appropriate (default),

string: r represent

Examples: $\{:+2.3f\}.format(45.72793)$
→ '+45.728'
 $\{1:>10s\}.format(8, "toto")$
→ ' toto'
 $\{"x!\r\}.format(x="I'm")$
→ "I\\'m"

good habit : don't modify loop variable

In [3]:

```

num = int(input("Enter number: "))

if(num % 2) == 0:
    print("{0} is even".format(num))
else:
    print("{0} is odd".format(num))

```

Par ou Impar

Enter number: 100

100 is even

In [4]:

```

year = int(input("Enter a year: "))

if(year % 4) == 0:
    if(year % 100) == 0:
        if(year % 400) == 0:
            print("{0} is a leap year ".format(year))
        else:
            print("{0} is not a leap year ".format(year))
    else:
        print("{0} is a leap year ".format(year))
else:
    print("{0} is not a leap year ".format(year))

```

Enter a year: 100

100 is not a leap year

Ano Bíssesto

In [5]:

```

num1 = float(input(" Enter first number: "))
num2 = float(input(" Enter second number: "))
num3 = float(input(" Enter third number: "))

if(num1 >= num2) and (num1 >= num3):
    largest = num1
elif(num2 >= num1) and (num2 >= num3):
    largest = num2
else:
    largest = num3

print("The largest number between ", num1, " , ", num2, "and ", num3, "is ", largest)

```

Enter first number: 12

Enter second number: 14

Enter third number: 15

The largest number between 12.0 , 14.0 and 15.0 is 15.0

verifica se é primo

In [6]:

```

num = int(input("Enter a number: "))

#prime numbers are greater than 1
if num > 1:
    #check for factors
    for i in range(2,num):
        if(num % i)==0:
            print(num, "is not a prime number")
            print(i, "times", num//i,"is",num)# o que é esse // -> arredondando valor
            break
    else:
        print(num, "is a prime number")

#if input number is less than
#or equal to 1, it is not prime
else:
    print(num, "is not a prime number")

```

*Se for maior que 1 entra no condicão
if (num % i) == 0:
5.5 → 5*

Se for menor que 1 já sai e não é primo

Enter a number: 44
44 is not a prime number
2 times 22 is 44

primos intervalo

In [7]:

```

lower = int(input("Enter lower range: "))
upper = int(input("Enter lower range: "))

print("Primer numbers between", lower, "and", upper, "are: ")
for num in range(lower,upper +1):
    #prime number are greater than 1
    if num > 1:
        for i in range(2,num):
            if(num % i)==0:
                #print("entrou no primeiro for", num)
                break
        else:
            print(num)

```

esse mais 1 é para incluir o ultimo número

Enter lower range: 34
Enter lower range: 67
Primer numbers between 34 and 67 are:
37
41
43
47
53
59
61
67

factorial

In [1]:

```

num = int(input("Enter a number: "))

factorial = 1

#check if the number is negative, positive or zero

if num <0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1, num + 1):
        factorial = factorial * i
    print("The factorial of", num, "is", factorial)

```

Enter a number: 20
 The factorial of 20 is 2432902008176640000

In [2]:

```

num = int(input("Display multiplication table of? "))

#use for loop to iterate 10 times
for i in range(1,11):
    print(num, "x", i, "=", num * i)

```

single functions can +

Display multiplication table of? 10

10 x 1 = 10
 10 x 2 = 20
 10 x 3 = 30
 10 x 4 = 40
 10 x 5 = 50
 10 x 6 = 60
 10 x 7 = 70
 10 x 8 = 80
 10 x 9 = 90
 10 x 10 = 100

Fibonacci

In [1]:

```

nterms = int(input("How many terms? "))

#first two terms
n1 = 0
n2 = 1
count = 0

#check if the number of terms is valid
if nterms <= 0:
    print("Please enter a positive integer")
elif nterms == 1:
    print("Fibonacci sequence upto", nterms,":")
    print(n1)
else:
    print("Fibonacci sequence upto", nterms,":")
    while count < nterms:
        print(n1, end=' , ')# esse end=' , ' mantem na mesma linha o resultado do for
        nth = n1 + n2
        #update values
        n1 = n2
        n2 = nth
        count +=1

```

Se tirar o end="," para linha
 O print() é pula linha automaticamente

How many terms? 12
 Fibonacci sequence upto 12 :
 0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 , 55 , 89 ,

In [15]:

```

num = int(input("Enter a number: "))

#inicializa a soma
sum=0
Também é necessário iniciar variáveis

#find the sum of the cube of each digit
temp = num
digit=0
while temp > 0:
    digit = temp % 10
    print(str(digit))
    sum += digit ** 3
    temp //=10
convertir string

if num == sum:
    print(num,"is an Armstrong number")
else:
    print(num,"is no an Armstrong number")

```

Enter a number: 12
 2
 1
 12 is no an Armstrong number

In [14]:

```

num = int(input("Enter a number: "))

if num < 0:
    print("Enter a positive number")
else:
    sum = 0
    #use while loop to iterate until zero
    while(num > 0):
        sum += num
        num -= 1
    print("The sum is", sum)

```

Programa para somar os números de 16 até 0

Enter a number: 16
The sum is 136

Anonymous function

In [16]:

```

terms = int(input("how many terms? "))

# use anonymous function
result = list(map(lambda x: 2 ** x, range(terms)))# Lambda ??? 0 à 9
#display the result
print("The total terms is: ", terms)
for i in range(terms):
    print("2 raised to power ", i, "is", result[i])

```

how many terms? 10
The total terms is: 10
2 raised to power 0 is 1
2 raised to power 1 is 2
2 raised to power 2 is 4
2 raised to power 3 is 8
2 raised to power 4 is 16
2 raised to power 5 is 32
2 raised to power 6 is 64
2 raised to power 7 is 128
2 raised to power 8 is 256
2 raised to power 9 is 512

Cria um range de 0 ao valor
Passed
Lambda x : N°
Map. (N1, Anony)

Verificar

Ex Lambda: Função oculta

$x = \lambda a : a + 10$

Print(x(5))

In [18]:

```
# take a List of numbers
my_list = [12, 65, 54, 39, 102, 339, 221]

#use anonymous function to filter
result = list(filter(lambda x: (x % 13 == 0), my_list))# filter - Lambda ?????

#display the result
print("Numbers divisible by 13 are ", result)
```

Numbers divisible by 13 are [65, 39, 221]

In [19]:

```
# convert decimal to binary, octal and hexadecimal
# change this line for a different result

dec = 344

print("The decimal value of", dec, "is:")
print(bin(dec), "in binary.")
print(oct(dec), "in octal.")
print(hex(dec), "in Hexdecimal.")
```

The decimal value of 344 is:

0b101011000 in binary.

0o530 in octal.

0x158 in Hexdecimal.

In [20]:

```
c = input("Enter a character: ")

print("The ASCII value of '" + c + "' is", ord(c))# como representa aspas simples dentro de
```

Enter a character: 2

The ASCII value of '2' is 50

In [26]:

```
# define a function
def computeHCF(x,y):
    #choose the smaller number
    if x > y:
        smaller = y
    else:
        smaller = x
    for i in range(1, smaller +1):
        if((x % i ==0)and (y % i == 0)):
            hcf = i
    return hcf

#take input from user
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))

print("The H.C.F. of ", num1, "and", num2, "is", computeHCF(num1, num2))
```

Enter first number: 12
Enter second number: 87
The H.C.F. of 12 and 87 is 3

In [32]:

```
def lcm(x,y):  
    """This function takes two integers and results the L.C.M."""  
  
    #choose the greater number  
  
    if x > y:  
        greater = x  
    else:  
        greater = y  
  
    while(True):  
        if((greater % x == 0) and (greater % y == 0)):  
            lcm = greater  
            break  
        greater +=1  
  
    return lcm  
  
num1 = int(input("Enter first number :"))  
num2 = int(input("Enter second number :"))  
print("The L.C.M of ", num1, "and", num2, "is", lcm(num1,num2))
```

```
Enter first number :3  
Enter second number :6  
The L.C.M of  3 and 6 is 6
```

In [35]:

```
def print_factors(x):

    print("The factor of", x, "are: ")
    for i in range(1, x + 1):
        if x % i == 0:
            print(i)

num = int(input("Enter a number: "))

print_factors(num)
```

```
Enter a number: 320
The factor of 320 are:
1
2
4
5
8
10
16
20
32
40
64
80
160
320
```

In [37]:

```
def add(x,y):
    return x+y

def sub(x,y):
    return x-y

def mul(x,y):
    return x*y

def div(x,y):
    return x/y

print("select operator.")
print("1. ADD")
print("2. Subtract")
print("3. multiply")
print("4. Divide")

choice = input("Enter choice (1/2/3/4):")

num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))

if choice == '1':
    print(num1,"+", num2, "=", add(num1,num2))

elif choice =='2':
    print(num1,"-", num2, "=", sub(num1,num2))

elif choice =='3':
    print(num1,"*", num2, "=", mul(num1,num2))

elif choice =='4':
    print(num1,"/", num2, "=", div(num1,num2))

else:
    print("invalid input ")
```

```
select operator.
1. ADD
2. Subtract
3. multiply
4. Divide
Enter choice (1/2/3/4):2
Enter first number: 4
Enter second number: 5
4 - 5 = -1
```

In [40]:

```
import itertools, random (Parameter 1) (Parameter 2)
deck = list(itertools.product(range(1,14),['spade','Heart','Diamond','Club']))
random.shuffle(deck)
print("you got:")
for i in range(5):
    print(deck[i][0], "of", deck[i][1])!
```

you got:
 2 of Diamond
 8 of Diamond
 3 of Club
 1 of Club
 12 of spade

In [48]:

```
import calendar
import datetime

now = datetime.datetime.now()
print ("year - month - day - hour - minute - second", now.year, now.month, now.day, now.hour)

yy = int(input("Enter year: "))
mm = int(input("Enter month: "))

print(calendar.month(int(now.year),int(now.month)))
```

year - month - day - hour - minute - second 2018 9 28 10 42 32
 September 2018
 Mo Tu We Th Fr Sa Su
 1 2
 3 4 5 6 7 8 9
 10 11 12 13 14 15 16
 17 18 19 20 21 22 23
 24 25 26 27 28 29 30

In [54]:

```
def recur_fibo(n):
    if n <=1:
        return n
    else:
        return(recur_fibo(n-1) + recur_fibo(n-2))

nterms = int(input("how many terms?"))

if nterms <= 0:
    print("please, enter a positive integer" )

else:
    print("Fibonacci sequence:")
    for i in range(nterms):
        print("i =",i)
        print(recur_fibo(i))
```

```
how many terms?30
Fibonacci sequence:
```

```
i = 0
0
i = 1
1
i = 2
1
i = 3
2
i = 4
3
i = 5
5
i = 6
8
i = 7
13
i = 8
21
i = 9
34
i = 10
55
i = 11
89
i = 12
144
i = 13
233
i = 14
377
i = 15
610
i = 16
987
```

```
i = 17
1597
i = 18
2584
i = 19
4181
i = 20
6765
i = 21
10946
i = 22
17711
i = 23
28657
i = 24
46368
i = 25
75025
i = 26
121393
i = 27
196418
i = 28
317811
i = 29
514229
```

In [65]:

Recursividade.

```
def recur_sum(n):
    if n <= 1:
        return n
    else:
        return n + recur_sum(n-1)

num = int(input("Enter a number: "))

if num < 0:
    print("Enter a positive number")
else:
    print("the sum is", recur_sum(num))
```

Enter a number: 1000
 the sum is 500500

In [66]:

10.1.e

In [68]:

```
def recur_factorial(n):
    if n == 1:
        return n
    else:
        return n * recur_factorial(n-1)
num = int(input("Enter a number:"))

if num < 0:
    print("sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("the factorial of 0 is 1")

else:
    print("The factorial of", num, "is", recur_factorial(num))
```

In [81]:

```
print(int(10.5//2))# divisão com dois // arredonda o numero  
print(101.25//2)# 50.0  
print(101.25/2)# 50.625
```

5
50.0
50.625

In [87]:

```
def convertToBinary(n):  
    if n > 1:  
        convertToBinary(n//2)  
    print(n % 2, end = ' ')  
  
dec=10  
  
convertToBinary(dec)
```

1 0 1 0

In [88]:

```
x = [[12,7],  
     [4,5],  
     [3,8]]  
  
result = [[0,0,0],  
          [0,0,0]]  
  
for i in range(len(x)):  
    for j in range(len(x[0])):  
        result[j][i] = x[i][j]  
  
for r in result:  
    print(r)
```

```
[12, 4, 3]  
[7, 5, 8]
```

In []:

```
my_str = "A string with some text inside."  
  
def my_function():  
    print("My function")  
  
print("First output")  
print(my_function())  
print("Second output")  
  
# This is a multi-line  
# comment
```

Sempre comentários não devem ser

In [88]:

```
x = [[12,7],
      [4,5],
      [3,8]]

result = [[0,0,0],
          [0,0,0]]

for i in range(len(x)):
    for j in range(len(x[0])):
        result[j][i] = x[i][j]

for r in result:
    print(r)
```

[12, 4, 3]
[7, 5, 8]

In [89]:

```
my_str = 'aibohphobia'

my_str = my_str.casefold() ?

rev_str = reversed(my_str)

if list(my_str)==list(rev_str):
    print("it is a palindrome")

else:
    print(" it is not a palindrome")
```

it is a palindrome

retirar pontuações não desejadas

In [90]:

```
punctuations = '''!()-[]{};:'"\,;<>./?@#$%&*_-''''

my_str = "Hello!!!, he said ---- and went $##$$"

no_punct = ""

for char in my_str:
    if char not in punctuations:
        no_punct = no_punct + char

print(no_punct)
```

Hello he said and went

In [93]:

```
my_str = input("Enter a string ")
words = my_str.split() quebra a frase em um array
words.sort()
print("the sorted words are: ")
for word in words:    corre palavra por palavra no array[words]
    print(word)
```

Enter a string isso é um teste

the sorted words are:

isso
teste
um
é

In [102]:

E = {0,2,4,6,8};
N = {1,2,3,4,5};

print("union of E and N is ", E | N)

print("intersection E and N is ", E & N)

print("difference of E and N is ", E - N)

print("Symetric of E and N is ", E ^ N)

Bom! trabalho com conjuntos

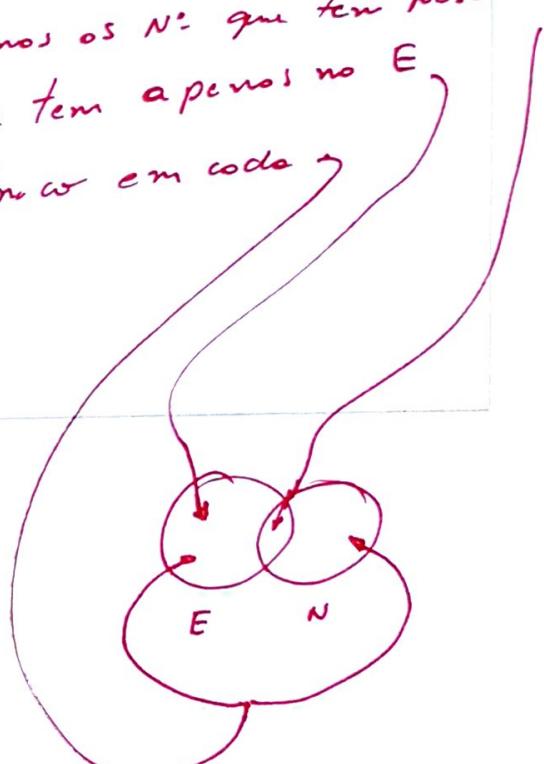
Junta os dois sem repetição

coloco apenas os N: que tem nos dois

tudo o que tem apenas no E

O que é só no em cima

union of E and N is {0, 1, 2, 3, 4, 5, 6, 8}
 intersection E and N is {2, 4}
 difference of E and N is {0, 8, 6}
 Symetric of E and N is {0, 1, 3, 5, 6, 8}



In [103]:

```

vowels = 'aeiou'

ip_str = input("Enter a string: ")      ↗ ONDA onoy

ip_str = ip_str.casefold()               ↗ count recebe vogais 'aeiou' !

count = {}.fromkeys(vowels, 0)           ↗ count[char] += 1

for char in ip_str:
    if char in count:
        count[char] += 1

print(count)

```

Enter a string: givaldo
 {'a': 1, 'e': 0, 'i': 1, 'o': 1, 'u': 0}

G está em count? N
 i está em count? S
 count[i] = +1
 V está em count? N
 A está em count? S
 count[A] = +1

D Va! de conocter em
 conocter GIVALDO

In [131]:

```

with open("names.txt", 'r', encoding = 'utf-8') as names_file:
    print(names_file)
    #name_file2 = names_file
    #names2 = name_file2.read()
    #print(names2)

    with open("body.txt", 'r', encoding = 'utf-8') as body_file:
        body = body_file.read()
        print("body:")
        print(body)
        print(names_file)

        for name in names_file:
            mail = "Hello" + name + body
            print("for ")

        with open(name.strip() + ".txt", 'w', encoding = 'utf-8') as mail_file:
            mail_file.write(mail)
            print("print mail_file ", mail_file)

```

2 arguments

```

<_io.TextIOWrapper name='names.txt' mode='r' encoding='utf-8'>
body:
ddssdfs
sd
fds
fds
fsd
fsd
fsd
fsd
ffgfdgd
g
fg
b

b
<_io.TextIOWrapper name='names.txt' mode='r' encoding='utf-8'>
for
print mail_file <_io.TextIOWrapper name='givado.txt' mode='w' encoding='utf-8'>
for
print mail_file <_io.TextIOWrapper name='francisco.txt' mode='w' encoding='utf-8'>
for
print mail_file <_io.TextIOWrapper name='silva.txt' mode='w' encoding='utf-8'>
for
print mail_file <_io.TextIOWrapper name='vivald.txt' mode='w' encoding='utf-8'>
for
print mail_file <_io.TextIOWrapper name='descartes.txt' mode='w' encoding='utf-8'>

```

```
'utf-8'
for
print mail_file <_io.TextIOWrapper name='galileu.txt' mode='w' encoding='ut
f-8'>
```

In [142]:

```
def jpeg_res(filename):
    print("função ")
    with open(filename, 'rb') as img_file:
        print(img_file)
        img_file.seek(163)
        a = img_file.read(2)
        print(a)
        height = (a[0]<<8) + a[1]
        a = img_file.read(2)
        width = (a[0]<<8) + a[1]
    print("The resolution of the image is", width, "x", height)

jpeg_res("img1.jpg")
```

```
função
<_io.BufferedReader name='img1.jpg'>
b'\x00\x00'
The resolution of the image is 0 x 0
```

In [146]:

```

import hashlib
def hash_file(filename):
    h = hashlib.sha1()
    print(h)
    with open(filename, 'rb') as file:
        chunk = 0
        while chunk != b'':
            chunk = file.read(1024)
            h.update(chunk)
    return h.hexdigest() → funções do tipo hashlib

message = hash_file("Pokerface.mp3")
print(message)

```

Funções

Argumentos

Retorno

do funções

o ?

funcões do tipo hashlib

```

<sha1 HASH object @ 0x000001EB413197B0>
bd68541ca2e534ea17438fa8c1dd272a1936a08d

```

In [158]:

```

string = input("Enter string: ")

count1=0
count2=0

for i in string:
    if(i.isdigit()):
        count1=count1+1
    count2 = count2+1

print("The number of digits is: ")
print(count1)
print("The number of characters is:")
print(count2)

```

Verifica se é um dígito

```

Enter string: hello123
The number of digits is:
3
The number of characters is:
8

```

In [159]:

```
string = input("Enter string: ")
count=0

for i in string:
    if(i.islower()):
        count=count+1

print("The number of lowercase characters is: ")

print(count)
```

?

Enter string: hello
 The number of lowercase characters is:
 5

In [160]:

```
string = input("Enter string: ")
sub_str= input("Enter word: ")

if(string.find(sub_str)==-1):
    print("substring not found in string!")
else:
    print("substring in string!")
```

Veifica se uma palavra está em
 uma frase.

substring retorna -1 ou + ?

Enter string: hello world
 Enter word: world
 substring in string!

In [161]:

```
def modify(string):
    final = ""
    for i in range(len(string)):
        if i % 2 == 0: → modulo de i == 0
            final = final + string[i]

    return final
string = input("Enter string: ")
print("modified string is:")
print(modify(string))
```

0,234

Enter string: hello
 modified string is:
 hlo
 o24

função

tamanho da string.

modulo de i == 0

reescreve a string apenas com
 as posições pares

In [162]:

```
string = input("enter string: ")
count=0
for i in string:
    count=count+1
new = string[0:2]+string[count-2:count]
print("newly formed string is: ")
print(new)
```

enter string: hello world
 newly formed string is:
 held

Posição da String + 2º posição da string

In [163]:

```
key = int(input("enter a key int to be added: "))

value = int(input("enter the value for the key to be added:"))

d={}
d.update({key:value})
print("updated dictionary is:")
print(d)
```

Dictionary

enter a key int to be added: 12
 enter the value for the key to be added:34
 updated dictionary is:
 {12: 34}

In [164]:

```
d1 = {'A':1,'B':2}
d2 = {'C':3}
d1.update(d2)

print("concatenated dictionary is: ")
print(d1)
```

concatenated dictionary is:
 {'A': 1, 'B': 2, 'C': 3}

In [165]:

```
d={'A':1,'B':2,'C':3}
key=input("enter key to check: ")
if key in d.keys():
    print("key is present and value of the key is")
    print(d[key])
else:
    print("key isn't present")
```

Verifica algumas informações pela
chave. Key

```
enter key to check: A
key is present and value of the key is
1
```

In [166]:

```
d={'A':10,'B':10,'C':239}
tot=1
for i in d:
    tot = tot*d[i]
print(tot)
```

Multiplica todos os itens do dicionário
uns pelos outros

23900

In [167]:

```
d = {'a':1,'b':2,'c':3,'d':4}

print("initial dictionary ")
print(d)

key = input("Enter the key to delete(a-d): ")

if key in d:
    del d[key]
else:
    print("key not found!")
    exit(0)
print("updated dictionary")
print(d)
```

Deleta determinado item
diferente da pop e del
exclui permanentemente

```
initial dictionary
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
Enter the key to delete(a-d): c
updated dictionary
{'a': 1, 'b': 2, 'd': 4}
```

In [168]:

```
s=input("enter string: ")
count=0
vowels = set("aeiou")
for letter in s:
    if letter in vowels:
        count +=1
print("count of the vowels is:")
print(count)
```

enter string: hello world
 count of the vowels is:
 3

In [30]:

```
s1 = input("enter first string ")
s2 = input("enter second string ")
a=list(set(s1) & set(s2))# envia para lista 'a' o que é em comum entre s1 e s2
print("the common letter are: ")
for i in a: #
    print(i)
```

enter first string isso é um teste
 enter second string para ver a letras que coincidem nas duas frases
 the common letter are:

t
 m
 s
 i
 e
 u
 o

Verafica se essa letras (vowels) estão na frase passada.

Para verificar palavras em comum entre duas frases

In [14]:

```
a=str(input("enter the name of the file with .txt extension: "))

file2 = open(a,'r') → Abre arquivo, apenas leitura?
print(file2)
line = file2.readline() → lê todas as linhas e joga p/ line
print(line)
#while(line != ""):
for i in line:
    print(line)
    line = file2.readline()

file2.close() → Fecha o arquivo
```

```
enter the name of the file with .txt extension: givado.txt
<_io.TextIOWrapper name='givado.txt' mode='r' encoding='cp1252'>
Hello givado
```

```
Hello givado
```

```
ddssdfs
```

```
sd
```

```
fds
```

```
fds
```

```
fsd
```

```
fsd
```

```
fds
```

```
ffgfdfgd
```

```
g
```

```
fg
```

In [16]:

```
with open("givado.txt") as f:
    with open("out.txt", "w") as f1:
        for line in f:
            f1.write(line)

print("arquivo out criado! ")
```

In [29]:

```
from timeit import Timer

def blank_spaces():

    fname = input("enter file name: ")
    k = 0

    with open(fname, 'r') as f:
        for line in f:
            words = line.split()
            for i in words:
                for letter in i:
                    if(letter.isspace()):
                        k=k+1

    return k

print(" occurrences of blank spaces: ")
#total_time= Timer(blank_spaces())
print(str(blank_spaces()))
```

```
occurrences of blank spaces:
enter file name: shakespeare.txt
4040422
```

In [32]:

```
fname = input("Enter file name: ")  
  
with open(fname,'r') as f:  
    for line in f:  
        l=line.title()  
        print(l)
```

Enter file name: names.txt

Givado

Francisco

Silva

Vivald

Descartes

Galileu

In [40]:

```
class rectangle():

    def __init__(self, breadth, lenght):#self é o construtor, o objeto tipo rectangle
        self.breadth = breadth
        self.lenght = lenght
        #print(self, breadth, lenght)

    def area(self):
        #print(self)
        return self.breadth * self.lenght

    def retorna_breadth(self):
        return self.breadth

    def retorna_lenght(self):
        return self.lenght

a=int(input("Enter lenght of rectangle: "))
b=int(input("Enter breadth of rectangle: "))

obj=rectangle(a,b)

print("Area of rectangle: ", obj.area())

print("Lenght = ", obj.retorna_lenght())
print("Breadth = ", obj.retorna_breadth())

print()
```

Enter lenght of rectangle: 10

Enter breadth of rectangle: 20

Area of rectangle: 200

Lenght = 20

Breadth = 10

In [41]:

```
class check():
    def __init__(self):
        self.n=[]
    def add(self,a):
        return self.n.append(a)
    def remove(self,b):
        self.n.remove(b)
    def dis(self):
        return(self.n)

obj=check()

choice=1

while choice!=0:
    print("0. Exit")
    print("1. Add")
    print("2. Delete")
    print("3. Display")
    choice=int(input("Enter choice: "))

    if choice ==1:
        n=int(input("Enter number to append: "))
        obj.add(n)
        print("List: ", obj.dis())

    elif choice==2:
        n=int(input("enter number to remove:"))
        obj.remove(n)
        print("List: ", obj.dis())

    elif choice==3:
        print("List: ", obj.dis())

    elif choice==0:
        print("Exiting!")
    else:
        print("Invalid choice! try again.")
```

```
0. Exit
1. Add
2. Delete
3. Display
Enter choice: 1
Enter number to append: 123
List: [123]
0. Exit
1. Add
2. Delete
3. Display
Enter choice: 1
Enter number to append: 234
List: [123, 234]
0. Exit
1. Add
```

```
2. Delete
3. Display
Enter choice: 1
Enter number to append: 345
List: [123, 234, 345]
0. Exit
1. Add
2. Delete
3. Display
Enter choice: 2
enter number to remove:123
List: [234, 345]
0. Exit
1. Add
2. Delete
3. Display
Enter choice: 3
List: [234, 345]
0. Exit
1. Add
2. Delete
3. Display
Enter choice: 0
Exiting!
```

In [43]:

```
import math

class circle():
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * (self.radius ** 2)

    def perimeter(self):
        return 2 * math.pi * self.radius

r = int(input("enter radius of circle: "))

obj = circle(r)

print("Area of cicle: ", round(obj.area(), 2))
print("Perimeter of circle: ", round(obj.perimeter(), 2))
```

```
enter radius of circle: 23
Area of cicle: 1661.9
Perimeter of circle: 144.51
```

In [45]:

```
class cal():

    def __init__(self,a,b):
        self.a=a
        self.b=b

    def add(self):
        return self.a+self.b

    def mul(self):
        return self.a*self.b

    def div(self):
        return self.a/self.b

    def sub(self):
        return self.a-self.b

a=int(input("Enter first number: "))

b=int(input("Enter second number: "))

obj= cal(a,b)

choice=1

while choice!=0:

    print("0. Exit")
    print("1. Add")
    print("2. Subtraction")
    print("3. multiplication")
    print("4. Division")
    choice=int(input("Enter choice: "))

    if choice==1:
        print("Result: ", obj.add())

    elif choice==2:
        print("Result: ", obj.sub())

    elif choice==3:
        print("Result: ", obj.mul())

    elif choice==4:
        print("Result: ", round(obj.div(),2))#round arredonda, o segundo argumento é a quant

    elif choice==0:
        print("Exiting! ")

    else:
        print("Invalid choice!!")

print()
```

Enter first number: 2

```

Enter second number: 3
0. Exit
1. Add
2. Subtraction
3. multiplication
4. Division
Enter choice: 1
Result: 5
0. Exit
1. Add
2. Subtraction
3. multiplication
4. Division
Enter choice: 0
Exiting!

```

In [47]:

```

class print1():
    def __init__(self):
        self.string=""

    def get(self):
        self.string=input("Enter string: ")# self.string receive a frase

    def put(self):
        print("String is: ")
        print(self.string) # metodo put printa a frase dentro de self.string

obj=print1()

obj.get()

obj.put()

```

```

Enter string: teste ello world
String is:
teste ello world

```

In [48]:

```

l_range=int(input("Enter the lower range: "))
u_range=int(input("Enter the upper range: "))

a=[(x,x**2) for x in range(l_range, u_range +1)]
print(a)

```

```

Enter the lower range: 5
Enter the upper range: 10
[(5, 25), (6, 36), (7, 49), (8, 64), (9, 81), (10, 100)]

```

In [*]:

```
def last(n):
    return n[-1]

def sort(tuples):
    return sorted(tuples, key=last)

a=input("Enter a list of tuples: ")

print("Sorted: ")
print(sort(a))
```

Enter a list of tuples: [

In []:

In []:

Data Science Academy - Python Fundamentos - Capítulo 3

Download: <http://github.com/dsacademybr>
<http://github.com/dsacademybr>

Condisional If

In [1]:

```
# Condisional If ✓  
if 5 > 2:  
    print("Python funciona!")
```

Python funciona!

In [2]:

```
# Statement If...Else ✓  
if 5 < 2:  
    →print("Python funciona!")  
else:  
    →print("Algo está errado!")
```

Algo está errado!

In [3]:

```
6 > 3 ✓
```

Out[3]:

True

In [4]:

```
3 > 7 ✓
```

Out[4]:

False

In [5]:

```
4 < 8 ✓
```

Out[5]:

True

In [6]:

4 >= 4 ✓

Out[6]:

True

In [7]:

```
if 5 == 5:
    print("Testando Python!") ✓
```

Testando Python!

In [8]:

```
if True:
    print('Parece que Python funciona!') ✓
```

Parece que Python funciona!

In [9]:

```
# Atenção com a sintaxe
if 4 > 3
    print("Tudo funciona!")
```

File "<ipython-input-9-da7e3854cb73>", line 2
 if 4 > 3 :
 ^

Faltou :

SyntaxError: invalid syntax

(C:\Users\Guilherme\OneDrive - UFGO\Documentos\Python\Exercícios\Exercício 03\Exercício 03.ipynb)
 print("Olá mundo") está comentado.

In [10]:

```
# Atenção com a sintaxe
if 4 > 3:
    print("Tudo funciona!")
```

File "<ipython-input-10-92620333c5a9>", line 3
 print("Tudo funcional!")
 ^

IndentationError: expected an indented block

Condicionais Aninhados

In [11]:

```
idade = 18
if idade > 17:
    print("Você pode dirigir!")
```

Você pode dirigir!

In [12]:

```
Nome = "Bob"
if idade > 13:
    if Nome == "Bob":
        print("Ok Bob, você está autorizado a entrar!")
    else:
        print("Desculpe, mas você não pode entrar!")
```

Ok Bob, você está autorizado a entrar!

In [13]:

```
idade = 13
Nome = "Bob"
if idade >= 13 and Nome == "Bob":
    print("Ok Bob, você está autorizado a entrar!")
```

Ok Bob, você está autorizado a entrar!

In [14]:

```
idade = 12
Nome = "Bob"
if (idade >= 13) or (Nome == "Bob"):
    print("Ok Bob, você está autorizado a entrar!")
```

Ok Bob, você está autorizado a entrar!

Elif

In [15]:

```
dia = "Terça"
if dia == "Segunda":
    print("Hoje fará sol!")
else:
    print("Hoje vai chover!")
```

Hoje vai chover!

In [16]:

```
if dia == "Segunda":
    print("Hoje fará sol!")
elif dia == "Terça":
    print("Hoje vai chover!")
else:
    print("Sem previsão do tempo para o dia selecionado")
```

Hoje vai chover!

Operadores Lógicos

In [17]:

```
idade = 18
nome = "Bob"
if idade > 17:
    print("Você pode dirigir!")
```

Você pode dirigir!

In [18]:

```
idade = 18
if idade > 17 and nome == "Bob":
    print("Autorizado!")
```

Autorizado!

In [19]:

```
# Usando mais de uma condição na cláusula if
disciplina = input('Digite o nome da disciplina: ')
nota_final = input('Digite a nota final (entre 0 e 100): ')

if disciplina == 'Geografia' and nota_final >= '70':
    print('Você foi aprovado!')
else:
    print('Lamento, acho que você precisa estudar mais!')
```

Digite o nome da disciplina: Geografia
 Digite a nota final (entre 0 e 100): 76
 Você foi aprovado!

In [20]:

```
# Usando mais de uma condição na cláusula if e introduzindo Placeholders

disciplina = input('Digite o nome da disciplina: ')
nota_final = input('Digite a nota final (entre 0 e 100): ')
semestre = input('Digite o semestre (1 a 4): ')

if disciplina == 'Geografia' and nota_final >= '50' and int(semestre) != 1:
    print('Você foi aprovado em %s com média final %r!' %(disciplina, nota_final))
else:
    print('Lamento, acho que você precisa estudar mais!')
```

Digite o nome da disciplina: Geografia
Digite a nota final (entre 0 e 100): 87
Digite o semestre (1 a 4): 2
Você foi aprovado em Geografia com média final '87'!

--> Fique atento aos espaços entre a margem e cada um dos seus comandos.
Falaremos mais sobre indentação ao longo do curso. A indentação faz parte da sintaxe da linguagem Python.

Fim

Obrigado - Data Science Academy - facebook.com/dsacademybr
<http://facebook.com/dsacademybr>

Data Science Academy - Python Fundamentos -

Capítulo 3

Download: <http://github.com/dsacademybr>
<http://github.com/dsacademybr>)

Loop For

In [3]:

```
# Criando uma tupla e imprimindo cada um dos valores
tp = (2,3,4) → Lista
for i in tp:           i assume valores de tp
    print(i)
```

2

3

4

In [4]:

```
# Criando uma lista e imprimindo cada um dos valores
ListaDoMercado = ["Leite", "Frutas", "Carne"]
for i in ListaDoMercado:
    print(i)           i assume valores mesmo sendo string
```

Leite

Frutas

Carne

In [5]:

```
# Imprimindo os valores no intervalo entre 0 e 5 (exclusive)
for contador in range(0,5):
    print(contador)      Range on 0 to 5
```

0

1

2

3

4

In [6]:

```
# Imprimindo na tela os números pares da lista de números
lista = [1,2,3,4,5,6,7,8,9,10]
for num in lista:
    if num % 2 == 0: → 2 em 2
        print (num)
```

```
2
4
6
8
10
```

In [7]:

```
# Listando os números no intervalo entre 0 e 101, com incremento em 2
for i in range(0,101,2):
    print(i) → de 0 a 100 indo de 2 em 2
```

```
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
...
```

In [8]:

```
# Strings também são sequências
for caracter in 'Python é uma linguagem de programação divertida!':
    print (caracter)
```

Python é uma linguagem legal!

Legal !

Punto oodo caracter

Loops Aninhados

In [9]:

```
# Loops aninhados
for i in range(0,5):
    for a in range(0,5):
        print(a)
```

In [10]:

```
# Operando os valores de uma Lista com loop for
listaB = [32,53,85,10,15,17,19]
soma = 0
for i in listaB:
    double_i = i * 2           → multiplica cada item por 2
    soma += double_i           += significa (soma + double_i)
print(soma)
```

462 ✓

In [11]:

```
# Loops em Lista de Listas 2
listas = [[1,2,3], [10,15,14], [10.1,8.7,2.3]]
for valor in listas:
    print(valor)
```

[1, 2, 3] 0
[10, 15, 14] 1
[10.1, 8.7, 2.3] 2

$$\begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}_1 \quad \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}_2$$

In [12]:

```
# Contando os itens de uma lista
lista = [5,6,10,13,17]
count = 0
for item in lista:
    count += 1

print(count)
```

5

In [13]:

```
# Contando o número de colunas
lst = [[1,2,3],[3,4,5],[5,6,7]]
primeira_linha = lst[0]
count = 0
for column in primeira_linha:
    count = count + 1

print(count)
```

3

In [14]:

```
# Pesquisando em Listas
listaC = [5, 6, 7, 10, 50]

# Loop através da Lista
for item in listaC:
    if item == 5:
        print("Número encontrado na lista!")
```

Número encontrado na lista!

In [15]:

```
# Listando as chaves de um dicionário
dict = {'k1': 'Python', 'k2': 'R', 'k3': 'Scala'}
for item in dict:
    print(item)
```

k1
k2
k3

In [16]:

Imprimindo chave e valor do dicionário. Usando o método items() para retornar os itens de

```
for k,v in dict.items():
    print(k,v)
```

k1 Python
k2 R
k3 Scala

$K = \text{dict}$
 $v = \text{items}$

Fim

Obrigado - Data Science Academy - facebook.com/dsacademybr
[\(http://facebook.com/dsacademybr\)](http://facebook.com/dsacademybr)

Data Science Academy - Python Fundamentos -

Capítulo 3

Download: <http://github.com/dsacademybr>
[\(http://github.com/dsacademybr\)](http://github.com/dsacademybr)

While

In [1]:

```
# Usando o Loop while para imprimir os valores de 0 a 9
counter = 0
while counter < 10:
    print(counter)
    counter = counter + 1
```

```
0
1
2
3
4
5
6
7
8
9
```

In [2]:

```
# Também é possível usar a cláusula else para encerrar o loop while
x = 0

while x < 10:
    print ('O valor de x nesta iteração é: ', x)
    print ('x ainda é menor que 10, somando 1 a x')
    x += 1
    — Ou x = x + 1
else:
    print ('Loop concluído!')
```

O valor de x nesta iteração é: 0
 x ainda é menor que 10, somando 1 a x
 O valor de x nesta iteração é: 1
 x ainda é menor que 10, somando 1 a x
 O valor de x nesta iteração é: 2
 x ainda é menor que 10, somando 1 a x
 O valor de x nesta iteração é: 3
 x ainda é menor que 10, somando 1 a x
 O valor de x nesta iteração é: 4
 x ainda é menor que 10, somando 1 a x
 O valor de x nesta iteração é: 5
 x ainda é menor que 10, somando 1 a x
 O valor de x nesta iteração é: 6
 x ainda é menor que 10, somando 1 a x
 O valor de x nesta iteração é: 7
 x ainda é menor que 10, somando 1 a x
 O valor de x nesta iteração é: 8
 x ainda é menor que 10, somando 1 a x
 O valor de x nesta iteração é: 9
 x ainda é menor que 10, somando 1 a x
 Loop concluído!

Pass, Break, Continue

In [3]:

```
counter = 0
while counter < 100:
    if counter == 4:
        break
    else:
        pass
    print(counter)
    counter = counter + 1
```

Pauso o processo quando counter = 4

? → ?

0
1
2
3

In [4]:

```
for verificador in "Python":
    if verificador == "h":
        continue
    print(verificador)
```

Pulo as instruções abaixo de continue

P
y
t
o
n

While e For juntos

In [5]:

```
for i in range(2,30):
    j = 2
    counter = 0
    while j < i:
        if i % j == 0:
            counter = 1
            j = j + 1
        else:
            j = j + 1

    if counter == 0:
        print(str(i) + " é um número primo")
        counter = 0
    else:
        counter = 0
```

2 é um número primo
 3 é um número primo
 5 é um número primo
 7 é um número primo
 11 é um número primo
 13 é um número primo
 17 é um número primo
 19 é um número primo
 23 é um número primo
 29 é um número primo

Fim

Obrigado - Data Science Academy - facebook.com/dsacademybr
 (<http://facebook.com/dsacademybr>)

Data Science Academy - Python Fundamentos - Capítulo 3

Download: <http://github.com/dsacademybr>
<http://github.com/dsacademybr>

Range

In [1]:

```
# Imprimindo números pares entre 50 e 101
for i in range(50, 101, 2):
    print(i)
```

50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80
82
84
86
88

Comeca com 50
contando de 2 em 2
Vai ate 101

In [2]:

```
for i in range(3, 6):
    print (i)
```

3
4
5
Comeca no 3
Pare no 5

In [3]:

```
for i in range(0, -20, -2):
    print(i)
```

0
-2
-4
-6
-8
-10
-12
-14
-16
-18

notar o -2^o

*Se colocar um número negativo
ele decrementa*

In [4]:

```
lista = ['Morango', 'Banana', 'Abacaxi', 'Uva']
lista_tamanho = len(lista)
for i in range(0, lista_tamanho):
    print(lista[i])
```

0 1 2 3

lista_tamanho = 4

0 Morango
1 Banana
2 Abacaxi
3 Uva

0, 4

In [5]:

```
# Tudo em Python é um objeto
type(range(0,3))
```

Out[5]:

range

Fim

OBS: Pode ir ate o ultimo item tem que
colocar +1

muito importante:

tudo em Python é um objeto

Obrigado - Data Science Academy - facebook.com/dsacademybr
[\(http://facebook.com/dsacademybr\)](http://facebook.com/dsacademybr)

Data Science Academy - Python Fundamentos - Capítulo 3

Download: <http://github.com/dsacademybr>
<http://github.com/dsacademybr>)

Métodos

In [1]:

```
# Criando uma lista  
lst = [100, -2, 12, 65, 0]
```

In [2]:

```
# Usando um método do objeto Lista  
lst.append(10)
```

In [3]:

```
# Imprimindo a Lista  
lst
```

Out[3]:

```
[100, -2, 12, 65, 0, 10]
```

In [4]:

```
# Usando um método do objeto Lista  
lst.count(10)
```

Out[4]:

```
1
```

In [5]:

```
# A função help() explica como utilizar cada método de um objeto  
help(lst.count)
```

Help on built-in function count:

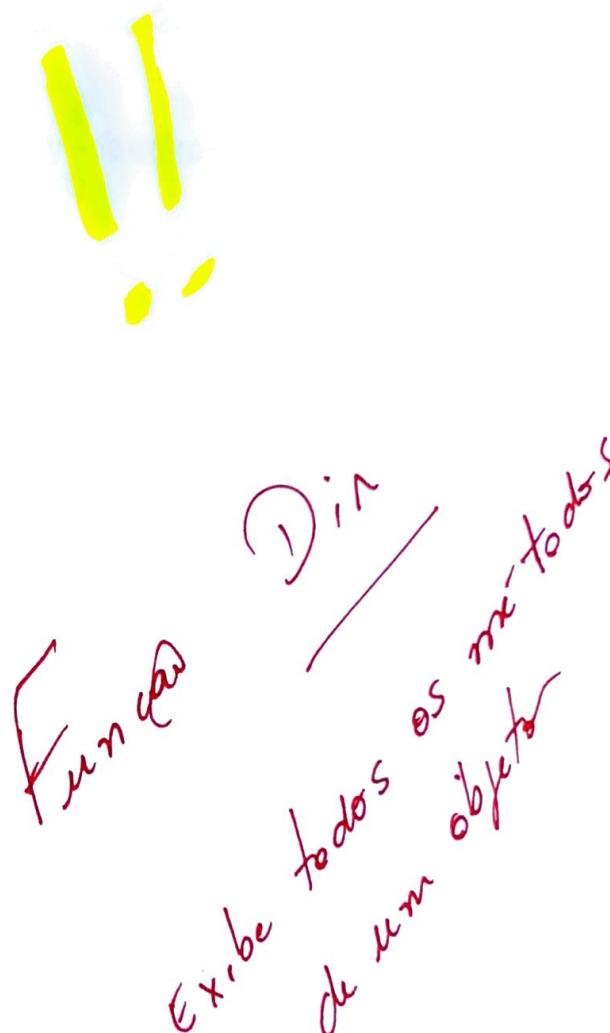
```
count(...) method of builtins.list instance  
L.count(value) -> integer -- return number of occurrences of value
```

In [6]:

A função dir() mostra todos os métodos e atributos de um objeto
 dir(lst)

Out[6]:

```
[ '__add__',
  '__class__',
  '__contains__',
  '__delattr__',
  '__delitem__',
  '__dir__',
  '__doc__',
  '__eq__',
  '__format__',
  '__ge__',
  '__getattribute__',
  '__getitem__',
  '__gt__',
  '__hash__',
  '__iadd__',
  '__imul__',
  '__init__',
  '__init_subclass__',
  '__iter__',
  '__le__',
  '__len__',
  '__lt__',
  '__mul__',
  '__ne__',
  '__new__',
  '__reduce__',
  '__reduce_ex__',
  '__repr__',
  '__reversed__',
  '__rmul__',
  '__setattr__',
  '__setitem__',
  '__sizeof__',
  '__str__',
  '__subclasshook__',
  'append',
  'clear',
  'copy',
  'count',
  'extend',
  'index',
  'insert',
  'pop',
  'remove',
  'reverse',
  'sort']
```



dir / Função
 exibe todos os métodos de um objeto

In [7]:

```
a = 'Isso é uma string'
```

In [8]:

O método de um objeto pode ser chamado dentro de uma função, como print()
print (a.split())

['Isso', 'é', 'uma', 'string']

a = "isso é uma string"

Fim

→ Cria uma lista de palavras

Obrigado - Data Science Academy - facebook.com/dsacademybr
[\(http://facebook.com/dsacademybr\)](http://facebook.com/dsacademybr)

Data Science Academy - Python Fundamentos -

Capítulo 3

Download: <http://github.com/dsacademybr>
[\(http://github.com/dsacademybr\)](http://github.com/dsacademybr)

Funções

In [1]:

```
# Definindo uma função
def primeiraFunc():
    print('Hello World')
```

In [2]:

```
primeiraFunc()
```

Hello World

In [3]:

```
# Definindo uma função com parâmetro
def primeiraFunc(nome):
    print('Hello %s' %(nome))
```

In [4]:

```
primeiraFunc('Aluno')
```

Hello Aluno

In [5]:

```
def funcLeitura():
    for i in range(0, 5):
        print("Número " + str(i))
```

In [6]:

```
funcLeitura()
```

Número 0
Número 1
Número 2
Número 3
Número 4

In [7]:

```
# Função para somar números
def addNum(firstnum, secondnum):
    print("Primeiro número: " + str(firstnum))
    print("Segundo número: " + str(secondnum))
    print("Soma: " + firstnum + secondnum)
```

firstnum e secondnum são iguais

In [8]:

```
# Chamando a função e passando parâmetros
addNum(45, 3)
```

Primeiro número: 45
 Segundo número: 3
 Soma: 48

Variáveis locais e globais

In [9]:

```
# Variável Global
var_global = 10 # Esta é uma variável global

def multiply(num1, num2):
    var_global = num1 * num2 # Esta é uma variável Local
    print(var_global)
```



In [10]:

```
multiply(5, 25)
```

125

In [11]:

```
print(var_global)
```

Valor dentro do função apenas

10

In [12]:

```
# Variável Local
var_global = 10 # Esta é uma variável global
def multiply(num1, num2):
    var_local = num1 * num2 # Esta é uma variável Local
    print(var_local)
```

In [13]:

```
multiply(5, 25)
```

125

In [14]:

```
print(var_local)
```

```
NameError                                 Traceback (most recent call last)
<ipython-input-14-5e93b2b588ab> in <module>()
----> 1 print(var_local)
```

```
NameError: name 'var_local' is not defined
```

Funções Built-in

In [15]:

```
abs(-56)
```

Out[15]:

```
56
```

In [16]:

```
abs(23)
```

Out[16]:

```
23
```

In [17]:

```
bool(0)
```

Out[17]:

```
False
```

In [18]:

```
bool(1)
```

Out[18]:

```
True
```

Funções str, int, float

In [19]:

```
# Erro ao executar por causa da conversão
idade = input("Digite sua idade: ")
if idade > 13:
    print("Você pode acessar o Facebook")
```

Digite sua idade: 14

```
-----
TypeError Traceback (most recent call last)
<ipython-input-19-0e8453c6c058> in <module>()
      1 # Erro ao executar por causa da conversão
      2 idade = input("Digite sua idade: ")
----> 3 if idade > 13:
      4     print("Você pode acessar o Facebook")
```

TypeError: '>' not supported between instances of 'str' and 'int'

In [20]:

```
# Usando a função int para converter o valor digitado
idade = int(input("Digite sua idade: "))
if idade > 13:
    print("Você pode acessar o Facebook")
```

Digite sua idade: 14

Você pode acessar o Facebook

In [21]:

int("26")

Out[21]:

26

In [22]:

float("123.345")

Out[22]:

123.345

In [23]:

str(14)

Out[23]:

'14'

In [24]:

```
len([23,34,45,46])
```

Out[24]:

4

In [25]:

```
array = ['a', 'b', 'c']
```

In [26]:

```
max(array)
```

Out[26]:

'c'

In [27]:

```
min(array)
```

Out[27]:

```
'a'  
def primo(num):  
    if num < 2:  
        return "Este número não é primo"  
    for i in range(2, int(math.sqrt(num)) + 1, 1):  
        if (num % i) == 0:  
            return "Este número não é primo"
```

In [28]:

```
array = ['a', 'b', 'c', 'd', 'A', 'B', 'C', 'D']
```

In [29]:

```
array
```

Out[29]:

```
['a', 'b', 'c', 'd', 'A', 'B', 'C', 'D']
```

In [30]:

```
max(array)
```

Out[30]:

'd'

In [31]:

```
min(array)
```

Out[31]:

'A'

16/11/2018

In [32]:

```
list1 = [23, 23, 34, 45]
```

In [33]:

```
sum(list1)
```

Out[33]:

125

Criando funções usando outras funções

In [34]:

```
import math

def numPrimo(num):
    """
    Verificando se um número
    é primo.
    """
    if (num % 2) == 0 and num > 2:
        return "Este número não é primo"
    for i in range(3, int(math.sqrt(num)) + 1, 2):
        if (num % i) == 0:
            return "Este número não é primo"
    return "Este número é primo"
```

In [35]:

```
numPrimo(541)
```

Out[35]:

'Este número é primo'

Fazendo **split** dos dados

In [36]:

```
# Fazendo split dos dados
def split_string(text):
    return text.split(" ")
```

In [37]:

texto = "Esta função será bastante útil para separar grandes volumes de dados."

In [38]:

```
# Isso divide a string em uma lista.  
print(split_string(texto))
```

```
['Esta', 'função', 'será', 'bastante', 'útil', 'para', 'separar', 'grandes',  
'volumes', 'de', 'dados. ']
```

In [39]:

```
# Podemos atribuir o output de uma função, para uma variável  
token = split_string(texto)
```

In [40]:

token

Out[40]:

```
['Esta',  
'função',  
'será',  
'bastante',  
'útil',  
'para',  
'separar',  
'grandes',  
'volumes',  
'de',  
'dados. ']
```

In [41]:

```
caixa_baixa = "Este Texto Deveria Estar Todo Em LowerCase"
```

In [42]:

```
def lowercase(text):  
    return text.lower()
```

In [43]:

```
lowercased_string = lowercase(caixa_baixa)
```

In [44]:

lowercased_string

Out[44]:

```
'este texto deveria estar todo em lowercase'  
'este texto deveria estar todo em lowercase'
```

In [45]:

```
# Funções com número variável de argumentos
def printVarInfo( arg1, *vartuple ):
    # Imprimindo o valor do primeiro argumento
    print ("O parâmetro passado foi: ", arg1)

    # Imprimindo o valor do segundo argumento
    for item in vartuple:
        print ("O parâmetro passado foi: ", item)
    return;
```

In [46]:

```
# Fazendo chamada à função usando apenas 1 argumento
printVarInfo(10)
```

O parâmetro passado foi: 10

In [47]:

```
printVarInfo('Chocolate', 'Morango', 'Banana')
```

O parâmetro passado foi: Chocolate
O parâmetro passado foi: Morango
O parâmetro passado foi: Banana

Fim

Obrigado - Data Science Academy - facebook.com/dsacademybr
<http://facebook.com/dsacademybr>)

Data Science Academy - Python Fundamentos - Capítulo 4

Download: <http://github.com/dsacademybr>
<http://github.com/dsacademybr>

Expressões Lambda

importante

In [1]:

```
# Definindo uma função - 3 Linhas de código
def potencia(num):
    result = num**2
    return result
```

In [2]:

```
potencia(5)
```

Out[2]:

25

In [3]:

```
# Definindo uma função - 2 Linhas de código
def potencia(num):
    return num**2
```

In [4]:

```
potencia(5)
```

Out[4]:

25

In [5]:

```
# Definindo uma função - 1 Linha de código
def potencia(num): return num**2
```

In [6]:

```
potencia(5)
```

Out[6]:

25

In [7]:

```
# Definindo uma expressão Lambda  
potencia = lambda num: num**2
```



In [8]:

```
potencia(5)
```

Out[8]:

25

In [9]:

```
# Lembre: operadores de comparação retornam boolean, true or false  
Par = lambda x: x%2==0
```

In [10]:

```
Par(3)
```

Out[10]:

False

In [11]:

```
Par(4)
```

Out[11]:

True

In [12]:

```
first = lambda s: s[0]
```

In [13]:

```
first('Python')
```

Out[13]:

'P'

In [14]:

```
reverso = lambda s: s[::-1]
```

In [15]:

```
reverso('Python')
```

Out[15]:

'nohtyP'

In [16]:

```
addNum = lambda x,y : x+y
```

In [17]:

```
addNum(2,3)
```

Out[17]:

```
5
```

Fim

Obrigado - Data Science Academy - facebook.com/dsacademybr
[\(http://facebook.com/dsacademybr\)](http://facebook.com/dsacademybr)

Data Science Academy - Python Fundamentos - Capítulo 3

Download: <http://github.com/dsacademybr>
<http://github.com/dsacademybr>

Exercícios - Métodos e Funções

In [1]:

```
# Exercício 1 - Crie uma função que imprima a sequência de números pares entre 1 e 20 (a função deve depois fazer uma chamada à função para listar os números)
def listaPar():
    for i in range(2, 21, 2):
        print(i)

listaPar()
```

```
2
4
6
8
10
12
14
16
18
20
```

In [2]:

```
# Exercício 2 - Crie uma função que receba uma string como argumento e retorne a mesma string com todas as letras maiúsculas. Faça uma chamada à função, passando como parâmetro uma string
def listaString(texto):
    print(texto.upper())
    return

listaString('Rumo à Análise de Dados')
```

RUMO À ANÁLISE DE DADOS

In [3]:

```
# Exercício 3 - Crie uma função que receba como parâmetro uma lista de 4 elementos, adicione 5 e 6 ao final e imprima a lista
def novaLista(lista):
    print(lista.append(5))
    print(lista.append(6))

lista1 = [1, 2, 3, 4]
novaLista(lista1)
print(lista1)
```

None

None

[1, 2, 3, 4, 5, 6]

In [4]:

```
# Exercício 4 - Crie uma função que receba um argumento formal e uma possível lista de elementos à função, com apenas 1 elemento e na segunda chamada com 4 elementos
def printNum( arg1, *lista ):
    print (arg1)
    for i in lista:
        print (i)
    return;

# Chamada à função
printNum( 100 )
printNum( 'A', 'B', 'C' )
```

100

A

B

C

In [5]:

```
# Exercício 5 - Crie uma função anônima e atribua seu retorno a uma variável chamada soma. # números como parâmetro e retornar a soma deles
soma = lambda arg1, arg2: arg1 + arg2
print ("A soma é : ", soma( 452, 298 ))
```

A soma é : 750

In [6]:

Exercício 6 - Execute o código abaixo e certifique-se que compreende a diferença entre va

```
total = 0
def soma( arg1, arg2 ):
    total = arg1 + arg2;
    print ("Dentro da função o total é: ", total)
    return total;

soma( 10, 20 );
print ("Fora da função o total é: ", total)
```

Dentro da função o total é: 30
 Fora da função o total é: 0

In [7]:

Exercício 7 - Abaixo você encontra uma lista com temperaturas em graus Celsius
 # Crie uma função anônima que converta cada temperatura para Fahrenheit
 # Dica: para conseguir realizar este exercício, você deve criar sua função lambda, dentro da
 # (que será estudada no próximo capítulo). Isso permite aplicar sua função a cada elemento
 # Como descobrir a fórmula matemática que converte de Celsius para Fahrenheit? Pesquise!!!
 Celsius = [39.2, 36.5, 37.3, 37.8]
 Fahrenheit = map(lambda x: (float(9)/5)*x + 32, Celsius)
 print (list(Fahrenheit))

[102.56, 97.7, 99.14, 100.03999999999999]

In [8]:

```
# Exercício 8
# Crie um dicionário e Liste todos os métodos e atributos do dicionário
dic = {'k1': 'Natal', 'k2': 'Recife'}
dir(dic)
```

Out[8]:

```
['__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'clear',
 'copy',
 'fromkeys',
 'get',
 'items',
 'keys',
 'pop',
 'popitem',
 'setdefault',
 'update',
 'values']
```

In [9]:

```
# Exercício 9
# Abaixo você encontra a importação do Pandas, um dos principais pacotes Python para análise
# Analise atentamente todos os métodos disponíveis. Um deles você vai usar no próximo exercício
import pandas as pd
dir(pd)
```

Out[9]:

```
['Categorical',
 'CategoricalIndex',
 'DataFrame',
 'DateOffset',
 'DatetimeIndex',
 'ExcelFile',
 'ExcelWriter',
 'Expr',
 'Float64Index',
 'Grouper',
 'HDFStore',
 'Index',
 'IndexSlice',
 'Int64Index',
 'Interval',
 'IntervalIndex',
 'MultiIndex',
 'NaT']
```

In [10]:

```
# ***** Desafio ***** (pesquise na documentação Python)

# Exercicio 10 - Crie uma função que receba o arquivo abaixo como argumento e retorne um re
# do arquivo. Dica, use Pandas e um de seus métodos, describe()
# Arquivo: "binary.csv"
import pandas as pd
file_name = "binary.csv"

def retornaArq(file_name):
    df = pd.read_csv(file_name)
    return df.describe()

retornaArq(file_name)
```

Out[10]:

	admit	gre	gpa	rank
count	5.000000	5.000000	5.000000	5.000000
mean	0.600000	616.000000	3.480000	3.000000
std	0.547723	185.148589	0.421307	1.224745
min	0.000000	380.000000	2.930000	1.000000
25%	0.000000	520.000000	3.190000	3.000000
50%	1.000000	640.000000	3.610000	3.000000
75%	1.000000	660.000000	3.670000	4.000000
max	1.000000	880.000000	4.000000	4.000000

Fim

Obrigado - Data Science Academy - facebook.com/dsacademybr
[\(http://facebook.com/dsacademybr\)](http://facebook.com/dsacademybr)