Python ‘Destilado’

Estruturas de Dados
Prof. Otávio Alcântara

Objetivos

- Revisar os principais recursos da linguagem Python
- Conhecer as funções e classes embutidas da linguagem
- Entender as estruturas de controle e repetição
- Utilizar os recursos de tratamento de exceções
- Manipular Iterators e Iterables

Visão Geral

- Python
  - Criada em 1990 por Guido Van Rossum
  - Utilizada na indústria, academia, sobretudo para ML e Data Science
  - Linguagem interpretada e OO
  - Dinamicamente tipada

Exemplo de Código Simples

```
print('Bem-vindo a calculadora de notas.')
print('Entre com o conceito de suas notas, um por linha')
print('Entre com uma linha em branco para finalizar')

pontos={'Excelente':10.0,'Otimo':9.0,'Bom':8.0, 'Medio':7.0, 'Ruim':6.0, 'Pessimo':5.0,
        'Terrivel':4.0, 'Desastre':3.0, 'Horripilante':2.0, 'Triste':1.0,'Pesadelo':0.0}

numero_disciplinas = 0
total_pontos = 0
done = False
while not done:
    grade = input()
    if grade == '':
        done = True
    elif grade not in pontos:
        print('Conceito desconhecido')
    else:
        numero_disciplinas +=1
        total_pontos += pontos[grade]
if numero_disciplinas > 0:
    print('Sua nota é {0:.3}'.format(total_pontos/numero_disciplinas))
```

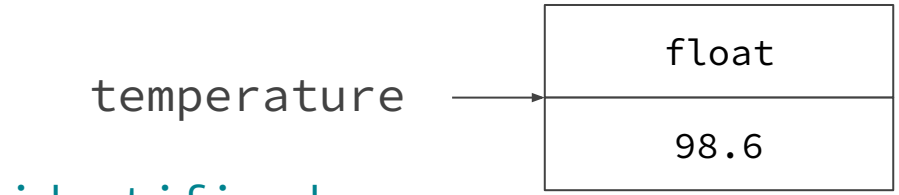
Objetos em Python

- Classes são a base de todos os tipos de dados em Python
- Nesta seção, vamos estudar
  - Identificadores, Objetos e Atribuições
  - Como criar Objetos e usar seus métodos
  - Classes built-in
    - int, float, list, tuple, str, dict

Identificadores, Objetos e Atribuições

```
temperature = 98.6
```

instância de float



identificador

- Endereço de memória
- Qualquer tipo de objeto
- Dinamicamente tipado

- Letras, Números, _ underscore
- Não pode iniciar com números
- 33 palavras reservadas

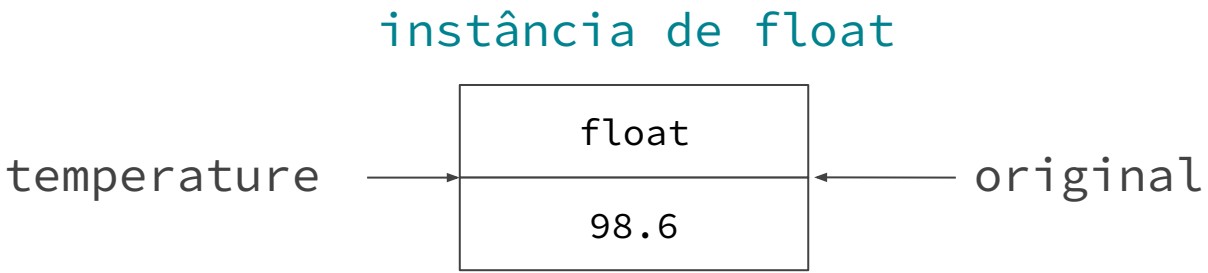
Identificadores, Objetos e Atribuições

- Palavras Reservadas

Reserved Words								
False	as	continue	else	from	in	not	return	yield
None	assert	def	except	global	is	or	try	
True	break	del	finally	if	lambda	pass	while	
and	class	elif	for	import	nonlocal	raise	with	

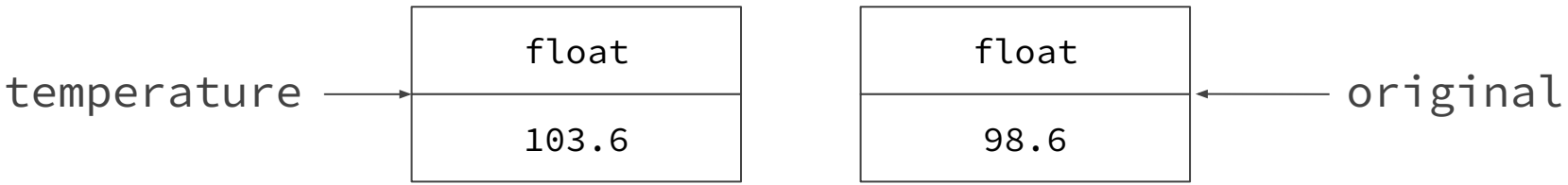
Identificadores, Objetos e Atribuições

```
original = temperature
```



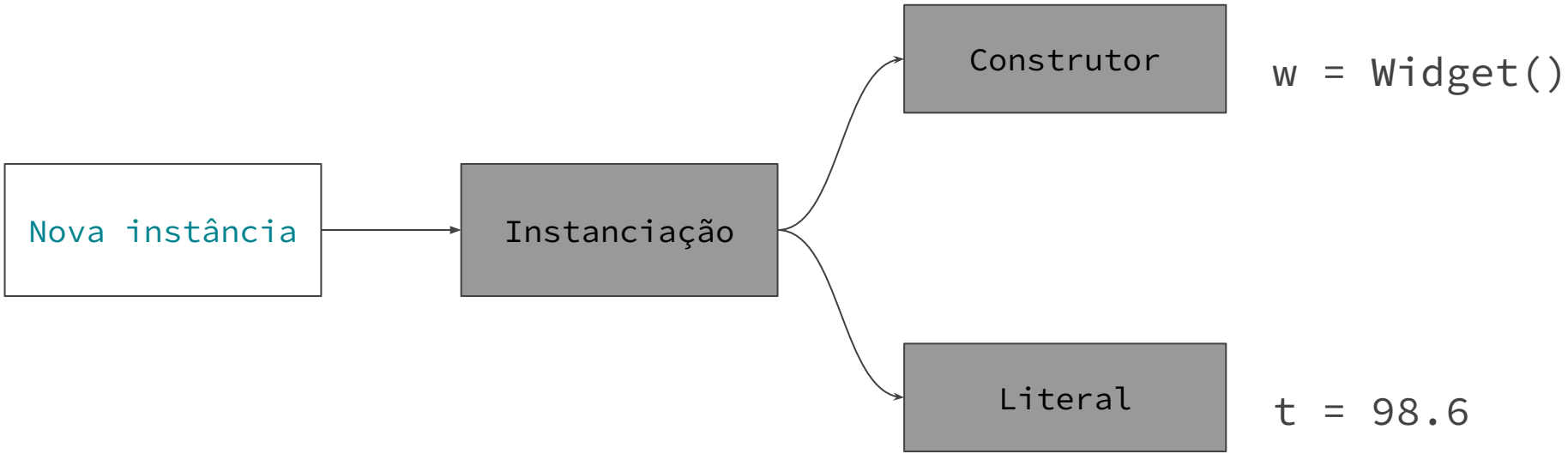
Identificadores, Objetos e Atribuições

```
temperature = temperature + 5.0
```



nova instância

Criando e usando objetos



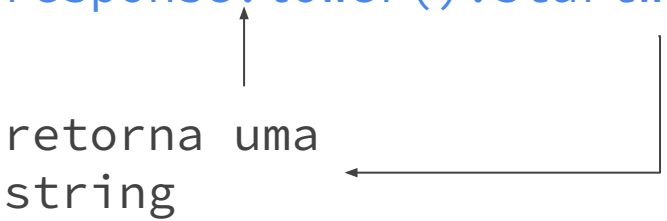
Chamando métodos

Funções Tradicionais

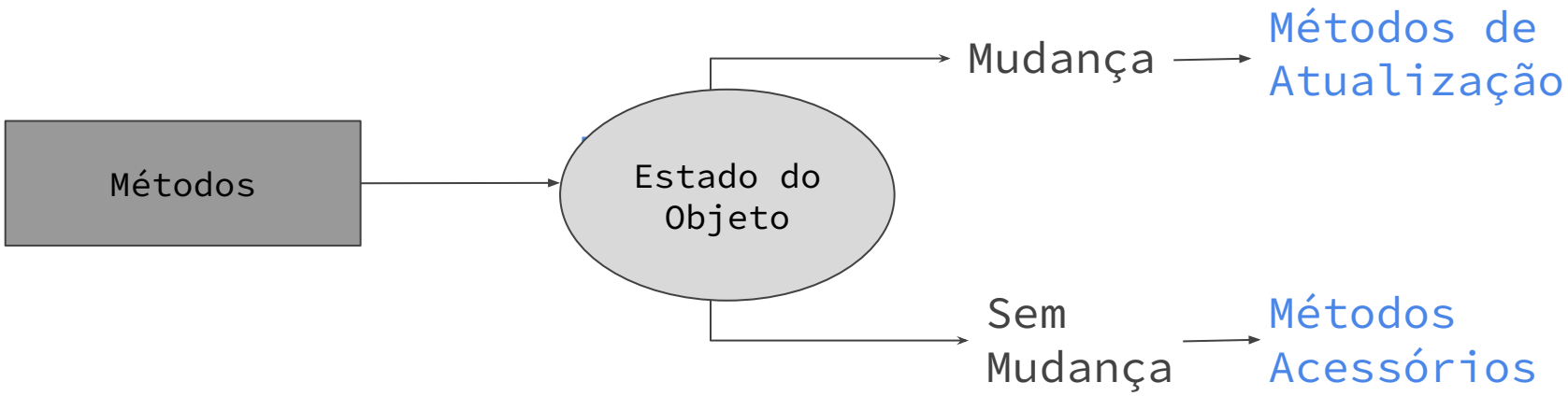
```
sorted()
```

Métodos (Funções Membro)

```
data.sort()
response.lower().startwith('y')
```



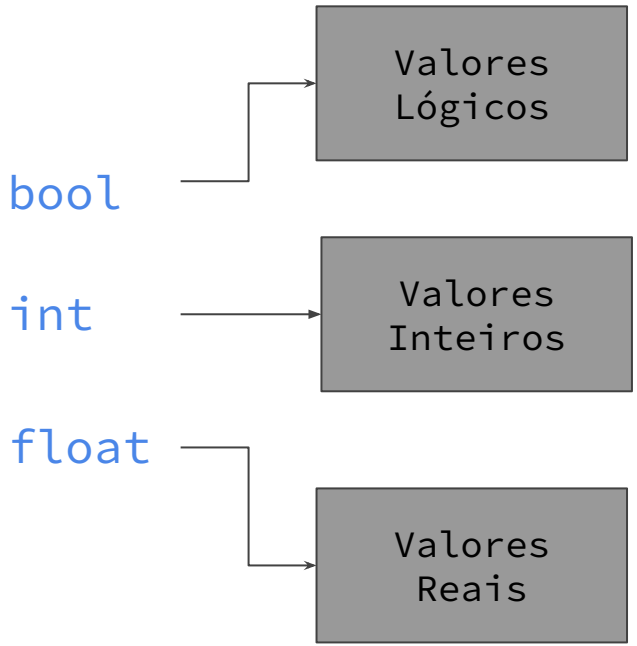
Chamando métodos



Classes Embutidas (Built-In)

Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	associative mapping (aka dictionary)	

Classes Embutidas (Built-In)




```
bool(), retorna False

bool(f), depende do
tipo de f

int(), retorna 0

int('3'), retorna 3

int(3.5), retorna 3

float(), retorna 0.0

float(2), retorna 2.0
float('3.14'),gera exceção
```

Classes Embutidas (Built-In)

Sequências

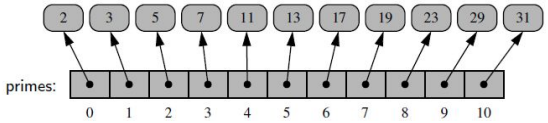
Coleção Ordenada de Itens

list → Sequência de Objetos

tuple → Sequência Imutável

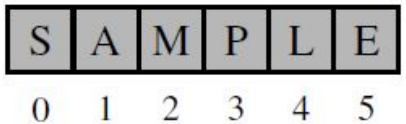
str → Sequência Imutável de Caracteres

```
prime = [2,3,5,..,31]
```




```
t= (2,3,6)
```

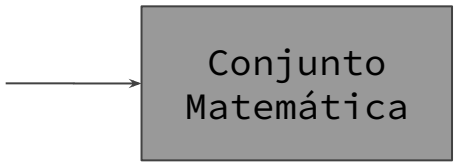
```
‘hello’ “hello”
```



Classes Embutidas (Built-In)

---

set
frozenset




```
s = {‘red’, ‘green’, ‘blue’ }
```

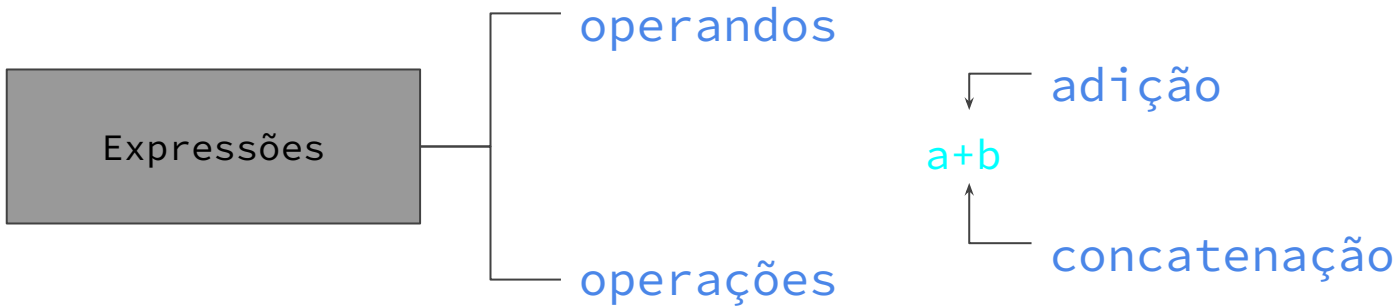
set(‘hello’) produz
{‘h’,’e’,’l’,’o’}

dict

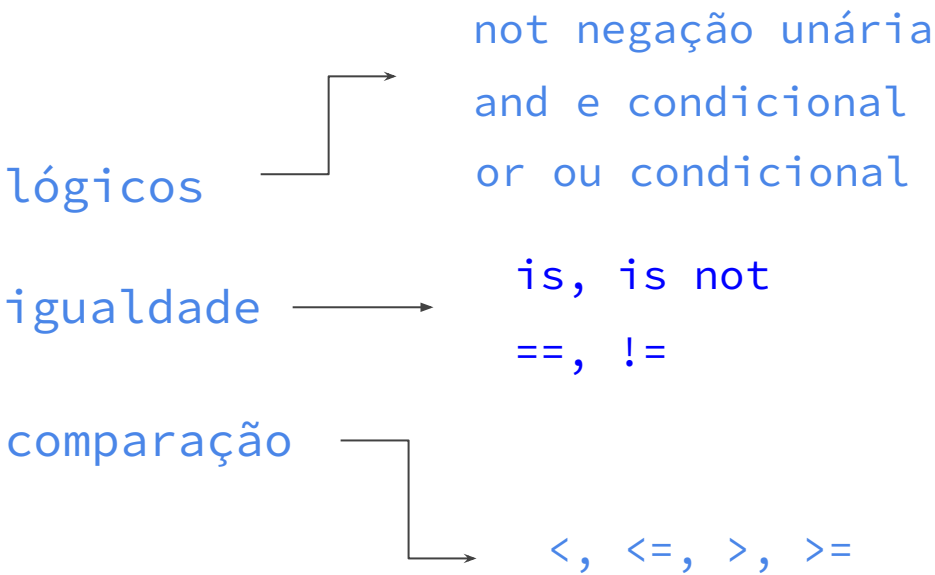

```
d = {‘ga’:’Irish’, ‘de’: ‘German}
```

Associa chaves a valores

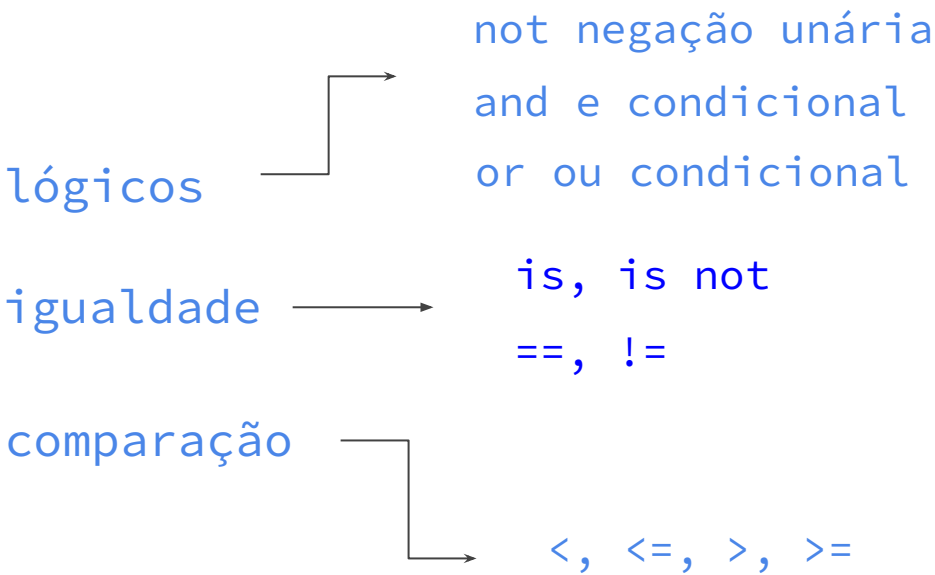
Expressões, Operadores e Precedência



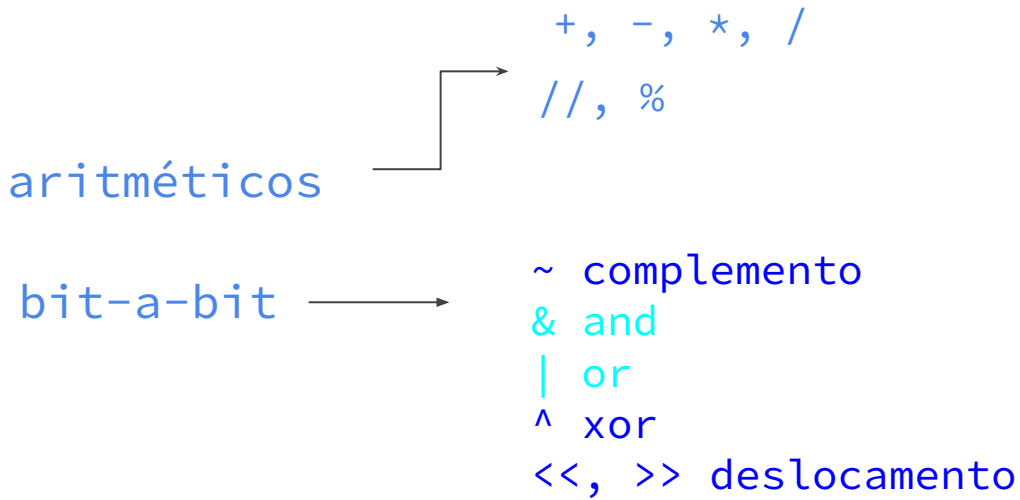
Expressões, Operadores e Precedência



Expressões, Operadores e Precedência



Expressões, Operadores e Precedência



Expressões, Operadores e Precedência

sequência ⟶

- Iniciam no zero
- Índices negativos

```
s[j]
s[start:stop]
s[start:stop:step]
s+t
k*s
val in s
val not in s
del s[j]
```

Expressões, Operadores e Precedência

Comparação de sequências

- Ordem lexicográfica

```
s == t
s != t
s < t
s <= t
s > t
s >= t
```

Expressões, Operadores e Precedência

Conjuntos e Dicionários →

```
key in s
key not in s
s1 == s2
s1 != s2
s1 <= s2 subconjunto
s1 < s2 subconjunto próprio
s1 >= s2 superconjunto
s1 > s2 superconjunto próprio
s1 | s2 união
s1 & s2 interseção
s1-s2 elementos de s1 mas não de s2
s1^s2 (s1|s2) - (s1&s2)
```

Expressões, Operadores e Precedência

Operator Precedence		
	Type	**Symbols**
1	member access	expr.member
2	function/method calls container subscripts/slices	expr(...) expr[...]
3	exponentiation	**
4	unary operators	+expr, −expr, ~expr
5	multiplication, division	*, /, //, %
6	addition, subtraction	+, −
7	bitwise shifting	<<, >>
8	bitwise-and	&
9	bitwise-xor	^
10	bitwise-or	|
11	comparisons containment	**is**, **is not**, ==, !=, <, <=, >, >= **in**, **not in**
12	logical-not	**not** expr
13	logical-and	**and**
14	logical-or	**or**
15	conditional	val1 **if** cond **else** val2
16	assignments	=, +=, −=, *=, etc.

Controle de Fluxo

- Condicionais

```
if first_condition:
    first_body
elif second_condition:
    second_body
elif third_condition:
    third_body
else:
    fourth_body
```

- Laços While

```
while condition:
    body
```

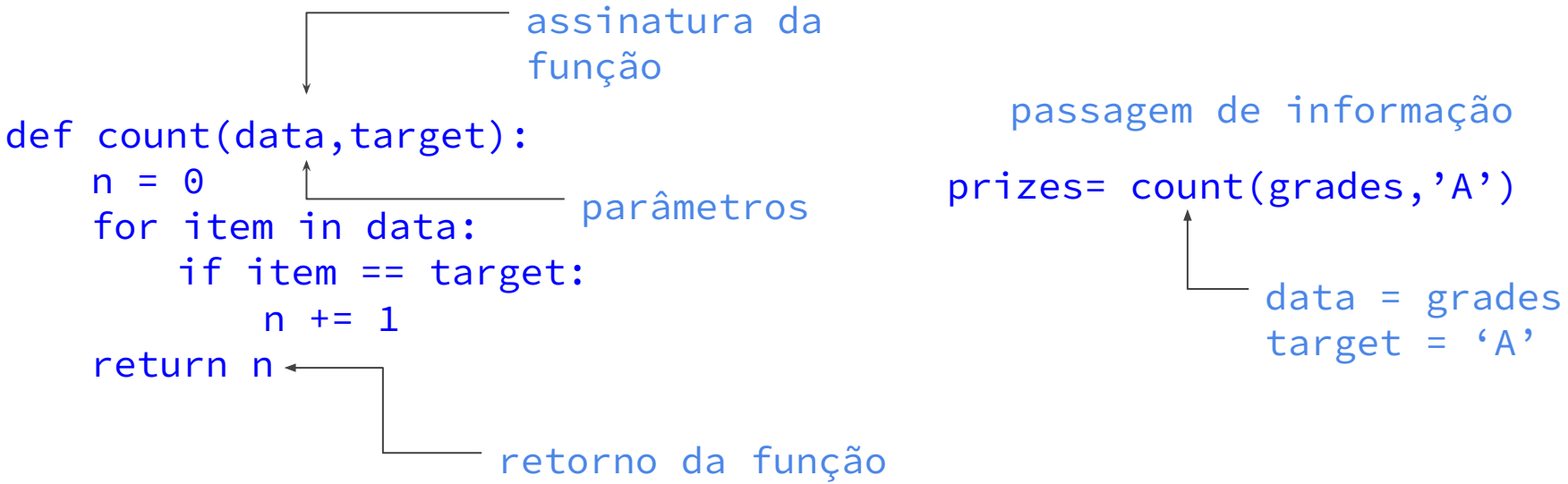
```
j = 0
while j < len(data) and data[j] != 'X':
    j += 1
```

- Laços For

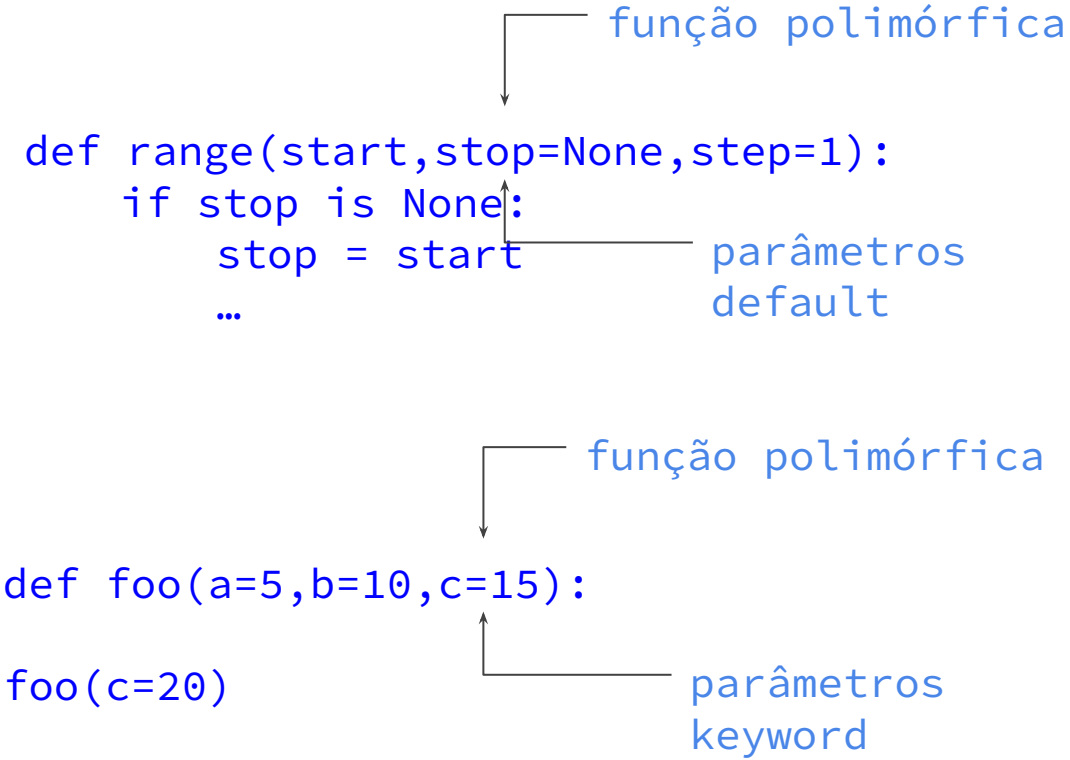
```
for element in iterable:
    body
```

```
found = False
for item in data:
    if item == target:
        found = True
        break
```

Funções


assinatura da função
def count(data,target):
n = 0
for item in data:
if item == target:
n += 1
return n
parâmetros
retorno da função
passagem de informação
prizes= count(grades,'A')
data = grades
target = 'A'

Funções


função polimórfica
def range(start,stop=None,step=1):
if stop is None:
stop = start
…
parâmetros
default
função polimórfica
def foo(a=5,b=10,c=15):
foo(c=20)
parâmetros
keyword

Funções

Common Built-In Functions	
Calling Syntax	**Description**
abs(x)	Return the absolute value of a number.
all(iterable)	Return True if bool(e) is True for each element e.
any(iterable)	Return True if bool(e) is True for at least one element e.
chr(integer)	Return a one-character string with the given Unicode code point.
divmod(x, y)	Return (x // y, x % y) as tuple, if x and y are integers.
hash(obj)	Return an integer hash value for the object (see Chapter 10).
id(obj)	Return the unique integer serving as an "identity" for the object.
input(prompt)	Return a string from standard input; the prompt is optional.
isinstance(obj, cls)	Determine if obj is an instance of the class (or a subclass).
iter(iterable)	Return a new iterator object for the parameter (see Section 1.8).
len(iterable)	Return the number of elements in the given iteration.
map(f, iter1, iter2, ...)	Return an iterator yielding the result of function calls f(e1, e2, ...) for respective elements e1 ∈ iter1, e2 ∈ iter2, ...
max(iterable)	Return the largest element of the given iteration.
max(a, b, c, ...)	Return the largest of the arguments.
min(iterable)	Return the smallest element of the given iteration.
min(a, b, c, ...)	Return the smallest of the arguments.
next(iterator)	Return the next element reported by the iterator (see Section 1.8)

Funções

next(iterator)	Return the next element reported by the iterator (see Section 1.8).
open(filename, mode)	Open a file with the given name and access mode.
ord(char)	Return the Unicode code point of the given character.
pow(x, y)	Return the value x^y (as an integer if x and y are integers); equivalent to x ** y.
pow(x, y, z)	Return the value (x^y mod z) as an integer.
print(obj1, obj2, ...)	Print the arguments, with separating spaces and trailing newline.
range(stop)	Construct an iteration of values 0, 1, ..., stop − 1.
range(start, stop)	Construct an iteration of values start, start + 1, ..., stop − 1.
range(start, stop, step)	Construct an iteration of values start, start + step, start + 2*step, ...
reversed(sequence)	Return an iteration of the sequence in reverse.
round(x)	Return the nearest int value (a tie is broken toward the even value).
round(x, k)	Return the value rounded to the nearest 10^{-k} (return-type matches x).
sorted(iterable)	Return a list containing elements of the iterable in sorted order.
sum(iterable)	Return the sum of the elements in the iterable (must be numeric).
type(obj)	Return the class to which the instance obj belongs.

Entrada e Saída Simplificada

- Função Print

```
print(a,b,c,sep=':')
print(a,b,c,end='')
```

  - Pode ser redirecionada para arquivo

- Função Input

```
year =
int(input("ano"))
```

- Arquivos

```
fp = open('s.txt')
fp.read()
fp.readline()
fp.readlines()
for line in fp:
fp.seek(k)
fp.tell(), posição
atual
fp.write(str)
fp.writelines(seq)
print(...,file=fp)
```

Tratamento de Exceções

- Eventos inesperados
- Levantadas pelo código, capturadas para serem tratadas
- Útil para I/O

Levantando uma Exceção

```
def sqrt(x):
  if not isinstance(x, (int, float)):
    raise TypeError('x must be numeric')
  elif x < 0:
    raise ValueError('x cannot be negative')
```

Capturando uma Exceção

```
try:
  ratio = x / y
except ZeroDivisionError:
  ... do something else ...
```

Iterators & Iterables

- Iterator
  - produz uma sequência de valores

```
data = [1,2,4,8]
i = iter(data)
next(i) #produz próximo valor da sequência
```

- Iterable
  - produz um Iterator via iter(obj)

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

Generators

- Sintaxe similar às funções, mas usa o yield para gerar a sequência de valores

```
def factors(n):
    for k in range(1,n+1):
        if n % k == 0:
            yield k


for f in factors(100):
```

Conveniências adicionais

- Expressão Condicional

```
param = n if n>=0
else -n
```

- Sintaxe de Compressão

```
squares = [k*k for k
in range(1,n+1)]

set: {k*k for k in
range(1,n+1)}
generator: (k**k for
k in range(1,n+1))
dic: {k:k*k for k in
range(1,n+1)}
```

Conveniências adicionais

- Packing & Unpacking

Packing

```
data = 2,4,6,8
return x,y
```

Troca

```
x,y = y,x
```

Unpacking

```
a,b,c,d = range(7,11)
for x,y in [ (7,2), (5,8), (6,4)]:
```

Problemas e Soluções

- Escreva uma função em Python que receba uma sequência de um ou mais números, e retorne o menor e o maior número na forma de uma tupla de tamanho 2.

Problemas e Soluções

- Escreva uma função em Python que receba um inteiro positivo n e retorne a soma dos quadrados de todos os inteiros positivos menores do que n.

Problemas e Soluções

- Escreva uma função em Python que receba um inteiro positivo n e retorne a soma dos quadrados de todos os inteiros positivos pares menores do que n.

Problemas e Soluções

- Escreva uma função em Python que receba uma sequência de números e determine se todos os números são diferentes um dos outros.

Problemas e Soluções

- O paradoxo do aniversário diz que a probabilidade de duas pessoas em uma mesma sala terem o mesmo dia de aniversário é maior do que 50%, dado n, o número de pessoas na sala for maior do que 23. Projete um programa em Python que teste este paradoxo por meio de uma série de experimentos em datas de aniversário aleatórias, que testam o paradoxo para n = 5,10,15,20,...,100.