Disciplina ILP-010: Linguagem de Programação

Comandos

Profª. Drª. Giovana Angélica Ros Miola
giovana.miola@fatec.sp.gov.br

Sintaxe e Semântica

Na ciência da computação, o termo **sintaxe** refere-se às regras que regem a composição de textos com significado em uma linguagem de programação, isto é, os textos para os quais faz sentido definir a semântica ou significado, ou fornecer uma interpretação.

Semântica formal é a área de estudo de ciência da computação que se preocupa em especificar o significado (ou comportamento) de programas de computador e partes de *hardware*.

A semântica é complementar à *sintaxe* de programas de computador, que se preocupa em descrever as estruturas de uma linguagem de programação.

Palavras Reservadas

As palavras reservadas definem as regras de sintaxe e estrutura da linguagem, e elas **não podem ser usadas como nomes de variáveis**.

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

Operadores Relacionais

Operador	Operação	Símbolo matemático
==	igualdade	=
>	maior que	>
<	menor que	<
!=	diferente	≠
>=	maior ou igual	≥
<=	menor ou igual	≤

Operadores Lógicos

Operador Python	Operação
not	não
and	e
or	ou

Operador not

```
>>> not True
False
>>> not False
True
```

Tipos de Variáveis

A função **type**(obj), com apenas um parâmetro, retorna o tipo (ou classe) de um objeto.
Alguns tipos de variáveis em Python:

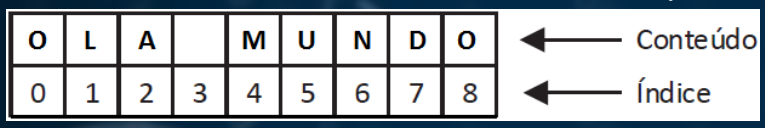
- inteiro (int)
- ponto flutuante (float)
- booleano (bool)
- complexo (complex)
- str (string)

```
>>> a = 23
>>> type(a)
<class 'int'>

>>> endereco = 'Rua Terezina, 75'
>>> type(endereco)
<class 'str'>
```

```
>>> a = 2.3
>>> type(a)
<class 'float'>

>>> n = 2+7j
>>> type(n)
<class 'complex'>

>>> reservista = True
>>> type(reservista)
<class 'bool'>

>>> nota = 8
>>> media = 7
>>> situacao = nota > média
>>> print(situacao)
True
```

String

Armazenamento de nomes e textos em geral, é realizado por variáveis do tipo string, em Python str, onde são consideradas cadeias de caracteres, ou seja uma sequência de símbolos como letras, números, sinais de pontuação etc.



Exemplo:

endereco = ‘Rua Terezina, 75, cep 19046-230’

>>> len(endereco)

31

String - Concatenação

O operador + concatena strings, ou seja, junta duas ou mais strings:

```
>>> texto1 = ‘Linguagem de Programação '
>>> texto2 = 'Python'
texto1 + texto2
Linguagem de Programação Python
```

String – Multiplicação

O operador * funciona com strings, multiplicando seu conteúdo por um inteiro, ou seja, replicará o conteúdo tantas vezes quanto for informado.

```
>>> texto1 = 'python'
>>> texto1 * 3
python python python
```

String - Composição

Juntar várias strings com valores de variáveis.

Exemplo:

```
>>> w = 31
>>> "João tem %d anos" %w
'João tem 31 anos'
```

Variável w

Local que apresentará o conteúdo da variável w

Marcador	Tipo
%d	Números inteiros
%s	Strings
%f	Números decimais

```
>>> idade = 22
>>> print("[%d]" % idade)
[22]
>>> print("[%03d]" % idade)
[022]
>>> print("[%3d]" % idade)
[ 22]
>>> print("[%-3d]" % idade)
[22 ]
>>> print("%5f" % 5)
5.000000
>>> print("%5.2f" % 5)
 5.00
>>> print("%10.5f" % 5)
   5.00000
```

String – Composição

```
>>> nome = "João"
>>> idade = 22
>>> grana = 51.34
>>> print("%s tem %d anos e R$%f no bolso." % (nome, idade, grana))
João tem 22 anos e R$51.340000 no bolso.
>>> print("%12s tem %3d anos e R$%5.2f no bolso." % (nome, idade, grana))
        João tem  22 anos e R$51.34 no bolso.
>>> print("%12s tem %03d anos e R$%5.2f no bolso." % (nome, idade, grana))
        João tem 022 anos e R$51.34 no bolso.
>>> print("%-12s tem %-3d anos e R$%-5.2f no bolso." % (nome, idade, grana))
João         tem 22  anos e R$51.34 no bolso.
```

String - Fatiamento

A	B	C	D	E	F	G	H	I	← Conteúdo
0	1	2	3	4	5	6	7	8	← Índice

Obtenção de parte de uma string.

O número à esquerda dos dois pontos indica a posição de início da fatia; e o à direita, do fim.

```
>>> s="ABCDEFGHI"
>>> print (s[0:2])
AB
>>> print (s[1:2])
B
>>> print (s[8])
I
>>> print (s[2:4])
CD
>>> print (s[0:5])
ABCDE
>>> print (s[1:8])
BCDEFGH
```

String - Fatiamento

-9	-8	-7	-6	-5	-4	-3	-2	-1	← Índice
A	B	C	D	E	F	G	H	I	← Conteúdo
0	1	2	3	4	5	6	7	8	← Índice

Exemplo de **fatiamento com omissão de valores e com índices negativos**:

```
>>> s="ABCDEFGHI"
>>> print (s[:2])
AB
>>> print (s[1:])
BCDEFGHI
>>> print (s[0:-2])
ABCDEFG
>>> print (s[:])
ABCDEFGHI
>>> print (s[-1:])
I
>>> print (s[-5:7])
EFG
>>> print (s[-2:-1])
H
```

String - Funções

Possuem várias funcionalidades prontas chamadas de métodos (funções).

- upper(), converte um texto para letras maiúsculas
  texto1 = 'python'
  >>> texto1.upper()
  'PYTHON'

- lower(), retorna o texto em letras minúsculas
  >>> "ALFA".lower()
  'alfa'

- capitalize() retorna o texto capitalizado (com a primeira letra em maiúscula)
  >>> texto1.capitalize()
  'Python'

- str(), converte número em string
  num = 123
  type(num) # <class 'int'>
  type(str(num)) # <class 'str'>

String - Funções

- **in** operador permite a pesquisa em uma string

```
>>>"b" in "abc“
True
>>>"d" in "abc“
False
>>>"d" not in "abc“
True
>>>"b" not in "abc“
False
```

- **strip()**, retira espaços em branco no começo e no fim

```
>>>" sobrando espaços ".strip()
‘sobrando espaços’
>>>" sobrando espaços ".strip()
'sobrando espaços’
```

- **join()**, junta cada item da string com um delimitador especificado. É o inverso do split().

```
>>>", ".join("abc")
'a, b, c’
```

- split(), separa uma string conforme um delimitador. É o inverso do join().

```
>>> e='Rua Terezina'
>>>e.split()   # ou e.split(‘ ‘)
['Rua', 'Terezina']
>>>s.split(“e")
['Rua T', 'r', 'zina’]
>>>s.split("") #sem caracter da erro
ValueError: empty separator
```

String - Funções

- replace(), troca o conteúdo

```
versao = ‘python 2’
>>> versao.replace(‘2’,’3’)
‘python 3’
```

- startswith() começa com um caracter

```
>>> versao.startswith(‘p’)
True
```

- endswith(), termina com um caracter

```
>>> versao.endswith(‘2’)
True
```

- count(), quantidade de vezes que um caracter é encontrado

```
>>> "maracana".count("a")
4
```

- isalpha(), retorna False se a string contiver algum caracter que não seja letra

```
"abc".isalpha() # True
"1fg".isalpha() # False
"123".isalpha() # False
"/+)".isalpha() # False
```

String - Funções

- isdecimal(), retorna True se s não for vazia e cada caractere de s é um numérico Unicode.

  >>>s = “descobrimento1500"
  False
  >>>s = "23443434"
  True

- isdigit() retorna True se s não for vazia e cada caractere de s é um numérico ASCII.

  >>> "21".isdigit()
  True

O computador não entende um texto. Os códigos são transformadores de texto em dados numéricos, compreensíveis ao PC. Podendo usar alguns padrões:

- **ASCII** (American Standard Code for Information Interchange), padrão americano
- **UNICODE**, representa **todos os caracteres específicos** de diversos idiomas, com a desvantagem de usar **2 bytes por** caracter
- **UTF-8** (PEP-393) é simples, é feito completamente em ASCII e, quando precisa de um caractere do UNICODE, usa-se um caractere especial, Exemplo: Ele é alto
  Ele Ã© alto

print()

print(), comando de apresentação de dados

O método *format()* cria strings que contem campos entre chaves a serem substituídos pelos argumentos do format.

>>>print(‘Hello World’)

>>>nome = input('Entre com o nome do aluno: ‘)

>>>media = (nota1 + nota2)/2

>>>print('{0} teve media igual a: {1:4.2f}'.format(nome, media))

Ou
print(nome, 'teve media igual a:', media)

print()

Método format com strings

>>>s = 'print do python' #15 caracteres
Alinha a direita ocupando n espaços
>>>print("{0:>20}".format(s))
  print do python

Alinha a direita ocupando n espaços e inserindo símbolo
>>>print("{0:#>20}".format(s))
#####print do python

Alinha a esquerda ocupando n espaços
>>>print("{0:<30}".format(s))
print do python

Centraliza, usando n espaços em branco a esquerda e n a direita
>>>print("{0:^20}".format(s))
  print do python
>>>print("{0:.^20}".format(s))
..print do python...

Imprime apenas as primeiras três letras
>>>print("{0:.3}".format(s))
pri

Comando de Entrada de Dados

input()

Exemplo:

no = input('Informe seu nome: ')
print("Você digitou %s" % no)
print("Olá, %s!" % no)

Resultado:

Digite seu nome: Paula
Você digitou Paula
Olá, Paula!

Conversão da entrada de dados

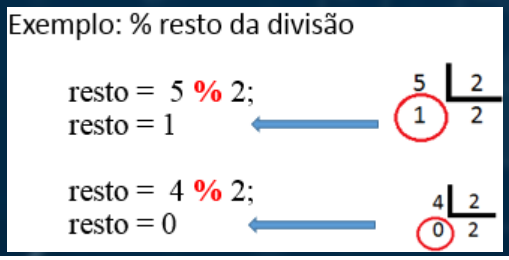
```
>>> anos = int(input("Anos de serviço: "))
>>> type(anos)
<class 'int'>

>>> valor_por_ano = float(input("Valor por ano: "))
>>> type(valor_por_ano )
<class 'float'>
```

Operadores aritméticos

- x + y (adição)
- x - y (subtração)
- x * y (multiplicação)
- x / y (divisão)
- x // y (divisão inteira, descarta a parte fracionária)
- x % y (módulo, obtêm o resto de uma divisão)
- x ** y (exponenciação)

Exemplos:

Operação	Resultado
>>> 1 + 2	3
>>> 3 – 1	2
>>> 10 / 2	5.0
>>> 10 // 3	3
>>> 10 * 2 + 1	21
>>> 10 % 3	1
>>> 2 ** 8	256



Estrutura Condicional

- A estrutura condional, ou de decisão, auxilia a decidir que partes do programa devem ser executadas com base no resultado de uma condição.
- A base dessas decisões consistirá em expressões lógicas que permitam representar escolhas em programas.

```
if <condição>:
        bloco verdadeiro

if <condição>:
        bloco verdadeiro
else:
        bloco falso
```

Estrutura Condicional - Exemplos

```
a = int(input("Primeiro valor: "))
b = int(input("Segundo valor: "))
if a > b:
    print ("O primeiro número é o maior!")
```

```
idade = int(input("Digite a idade de seu carro: "))
if idade <= 3:
    print("Seu carro é novo")
else:
    print("Seu carro é velho")
```

Estrutura Condicional - Aninhada

```
# Preço de telefone com três faixas de preço
minutos = int( input("Quantos minutos você utilizou este mês: ") )
if minutos < 200:
    preco = 0.20
elif minutos < 400:
        preco = 0.18
else:
        preco = 0.15
print("Você vai pagar este mês: R$%6.2f" % (minutos * preco))
```

Expressão Condicional

Também conhecida como **OPERADOR TERNÁRIO** em outras linguagens de programação

Sintaxe:

<expressao1> if <condicao> else <expressao2>

Entendendo, valorVerdadeiro if condicao else valorFalso

Exemplo:

```
>>>x = 10
>>>print ("par" if x % 2 == 0 else "impar")
par
```

Estrutura de Repetição

Utilizada para executar a mesma parte de um programa várias vezes, dependendo de uma condição

```
while <condição>:
   Bloco
```

x=0

Exemplo:

```
while x <= 3:
    print(x)
    x = x + 1
```

```
for <variável> in range (qtde):
    Bloco
```

Exemplo:

```
for  x in range (3):
    print(x)
```

Estrutura de Repetição - Contador

```
aprovados = 0
while x<=5:
    m = input(float('Informe a média: '))
    if m >= 6:
        aprovados = aprovados + 1
    x = x + 1
print('Quantidade de aprovados: ',
  aprovados)
```

```
aprovados = 0
for x in range (5):
    m = input(float('Informe a média: '))
    if m >= 6:
        aprovados = aprovados + 1
print('Quantidade de aprovados: ',
aprovados)
```

Estrutura de Repetição - Acumulador

```
n = 1
soma = 0
while n <= 10:
    x = int(input("Digite o %d número:“ %n))
    soma = soma + x
    n = n + 1
print("Soma: %d“ %soma)
```

```
soma = 0
for n in range(1,11):
    x = int(input("Digite o %d número:“ %n))
    soma = soma + x
print("Soma: %d“ %soma)
```

Estrutura de Repetição - Aninhada

```
tabuada=1
while tabuada <= 10:
    numero = 1
    while numero <= 10:
        print("%d x %d = %d" % (tabuada, numero, tabuada * numero))
        numero+=1
    tabuada+=1
.....................................................
for tabuada in range (1,11):
    for numero in range (1,11):
        print("%d x %d = %d" % (tabuada, numero, tabuada * numero))
```

Lista / Vetor

Uma lista é uma **estrutura de dados** composta por itens organizados de forma linear, na qual cada um pode ser acessado a partir de um índice, que representa sua posição na coleção (iniciando em zero).

lista	16	2	77	40	12071
índice	0	1	2	3	4

```
vet = []
for x in range(5):
    vet.append( input('Informe um nome: ') )
```

Matriz

Uma matriz é uma variável composta homogênea bidimensional (multidimensional) formada por uma sequência de variáveis, todas do mesmo tipo, com o mesmo identificador (mesmo nome) e alocadas sequencialmente na memória.

```
import numpy
mat = numpy.empty([2,4])
for l in range(2):
    for c in range(4):
        mat[l,c]=int( input('Informe um valor: ') )
print(mat)
```