Tipo de dados abstratos

Jacqueline

Tipo de dados

Um tipo de dados deve caracterizar o **conjunto de valores** a que uma constante pertence ou o conjunto de valores que pode ser assumido por uma variável ou gerado por uma expressão/função.

Inteiros

Um espaço de memória de 1 Byte (8 bits) permite 2^8 (256) valores representados.

Inteiro sem signal: 0 a 255 Inteiro

Sem sinal: 10110101_2 =

$(1.2^7 + 0.2^6 + 1.2^5 + 1.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0)_{10} = 181_{10}$

Com sinal (primeiro bit é sinal): −128 a 127

Com sinal: 10110101_2 =

$-1.(0.2^6 + 1.2^5 + 1.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0) = -53_{10}$

Caracteres

Padrão ASCII: 8 bits (1 Byte)

ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Tipo de dados abstratos

Modelo de dados que encapsula com conjunto de dados de tipos primitivos e um **conjunto de procedimentos** que atuam sobre os dados encapsulados.

Exemplos: representar um ponto no espaço bidimensional; representar informações e histórico do aluno na universidade;...

```
ponto = { float x,
          float y}
```

Como representar dados abstratos?

Usando registros (em C)

Usando classes (linguagens orientadas a objetos: C++, python)

Registros

Declarando:

```
struct nome_registro {
     tipo nome_campo_1;
     tipo nome_campo_2;
     …
     tipo nome_campo_n;
} ;
```

Exemplificando:

```
struct ponto {
     int x;
     int y;
};

struct aluno{
     char nome[100];
     float cr; //coeficiente de rendimento
};
```

Dica!

- Declarar as estruturas no cabeçalho do código fonte, desta forma, será acessível para toda função que você implementar

```
#include <stdio.h>
/* Declare tipos registro aqui */

int main () {
/* Construa seu programa aqui */
}

/* demais funções que poderá usar o tipo abstrato declarado (registro)*/
```

Como usar?

Criando variáveis com tipo abstrato:

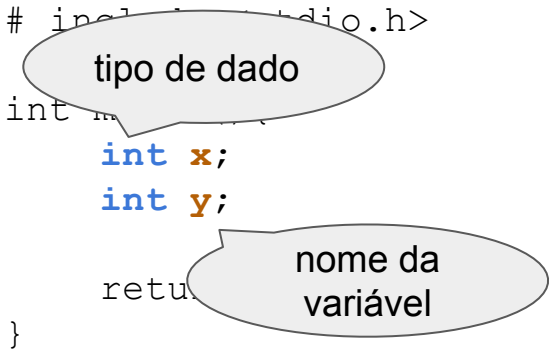
```
# include <stdio.h>

int main (){
    int x;
    int y;

    return 0;
}
```

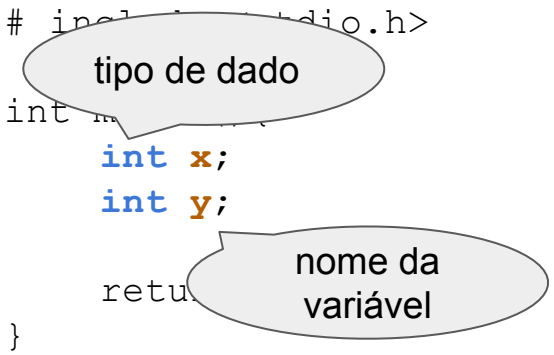
Como usar?

Criando variáveis com tipo abstrato:



Como usar?

Criando variáveis com tipo abstrato:




```c
# include <stdio.h>

struct ponto{
     int x;
     int y;
};


int main (){
     struct ponto a;

     return 0;
}
```

definição de um tipo

declaração de variável, igual!

Como usar?

Acesso aos campos:

```
nome_da_variavel.nome_do_campo
```

```
int main (){
     struct ponto a, b;
     a.x=10;
     a.y=0;
     b.x=a.x*3+a.y;

     return 0;
}
```

```
# include <stdio.h>

struct ponto{
     int x;
     int y;
};


int main (){
     struct ponto a;
     a.x=10;
     a.y=0;

     return 0;
}
```

Classes

Em linguagem orientadas a objetos, uma classe representa um dado semelhante a uma estrutura, com o detalhe de que além dos dados internos da estrutura, também associamos funções aos elementos da estrutura

```
class ponto {

     float x, y;



     float distance(ponto a, ponto b) {...}

}
```

Classes - Python

Declarando:

```
class nome_classe:

     # construtor básico
     def __init__(self, nome_campo_1, nome_campo_2, …nome_campo n):
          self.nome_campo_1=nome_campo_1
          self.nome_campo_2=nome_campo_2
          …
          self.nome_campo_n=nome_campo_n
```

Exemplificando:

```
class ponto:

    def __init__(self, x, y):
        self.x = x
        self.y = y
```

Indentação é extremamente importante, python não usa chaves!!

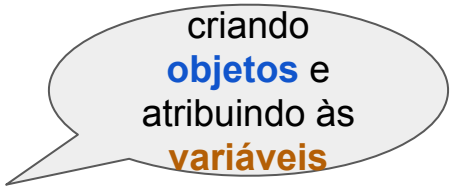
Python não é uma linguagem tipada

Variáveis não precisa ser declaradas antes do uso, mas podem, desde que atribua valor inicial

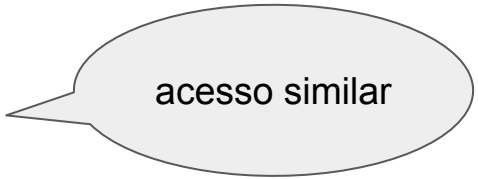
Classes – Python

```
ponto_a = ponto(10,0)
ponto_b = ponto(1,5)

print(ponto_a.x)
soma_xs=ponto_a.x+ponto_b.x
print(soma_xs)
```



a instrução ponto(10, 0) chama o construtor da classe (a função __init__)



Vamos exercitar um pouco de classes em python?

1. execute o código anterior

2. mostre a distância entre os pontos a e b (sem criar funções)

```
import math
math.sqrt(10)  ## para calcular a raiz quadrada de 10
```

3. agora sim, crie uma função para calcular a distância:
Estrutura de funções em python:

```
def nome_func (param_1: tipo, param_2: tipo):
      ##…operações…##
      return valor
```

Cabeçalho que devem usar:

```
def distancia (p1: ponto, p2:ponto):
```

4. Crie uma estrutura para guardar alunos e notas, peça para o usuário valores para o nome de nota de 3 alunos, informe quem é o melhor e pior aluno da turma

Como criar a mesma função de distância como operação dentro da classe ponto?

```
import math

class Point(object):
    """A 2D point in the cartesian plane"""

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance(self, Point):
        dist = math.sqrt((self.x - Point.x)**2 + (self.y - Point.y)**2)
        return dist


p1 = Point(4,9)
p2 = Point(10,5)
print(p1.distance(p2))
```