Aula de revisão

Tipos das variáveis

Tipos de variáveis

Para cada tipo de variável, as funções tem comportamento diferente.

- Texto (str)
- Número inteiro (int)
- Número real (float)
- Booleano (bool), True ou False

Tipos de variáveis

Ao declarar uma variável, o Python determina seu tipo dinamicamente, uma variável pode mudar de tipo apenas declarando-a novamente:

```
idade = “Trinta e dois” # tipo texto
idade = 32 # tipo inteiro
```

Listas

Declarando listas

Uma lista pode ter vários argumentos de diferentes tipos.

```
minhaLista = [‘eu’, ‘amo’, ‘Python’]
minhaLista[0]
```

Declarando listas

Modificando elementos de uma lista

```
minhaLista = [‘eu’, ‘amo’, ‘Python’]
minhaLista[2] = ‘software livre’
```

idp

Dicionários

Dicionários

Declarando um dicionário com duas entradas e acessando cada uma delas:

```
meuDict = {‘linguagem’: ’Python’, ’idade’: ‘28 anos’}
meuDict[‘linguagem’]
meuDict[‘idade’]
```

Funções e bibliotecas

Funções

Algumas funções importantes para Listas:

- append()
- insert()
- del
- pop()
- remove()
- count()
- reverse()
- sort()
- index()

Funções

Funções importantes para Dicionários:

- clear()
- copy()
- items()
- keys()
- pop()
- update()
- values()

Controle de fluxo

Laços

Em português, chamamos “loop” de “laço”

- Palavra-chave **for**

```
for cliente in clientes:
    print(cliente)
    if cliente == “Álvaro”:
        print(“Te peguei!”)
```

Laços for ou “para cada”

Com **for**, o bloco irá executar para cada elemento do objeto.

```
for cliente in clientes:
    print(cliente)
```

idp

Laços while ou “enquanto”

O **while** executará o bloco “enquanto” a condição a direita dele resultar em verdadeiro, ou booleano True.

```
while nome == “Álvaro”:
    print(“Olá, professor!”)
```

idp

Blocos if... else ou “se... senão”

O **if** ou “se”, executa o bloco se a condição for satisfeita.

```
if nome == “Álvaro”:
    print(“Bom dia, professor”)
else:
    print(“Bom dia, querido colega”)
```

idp

Funções

Funções

Define-se funções com a palavra-chave **de**
da palavra definir:

```
def somar_numeros(a,b):
    return a + b
```

Retornando valores

Uma função pode retornar um ou mais valores após a execução.

```
def somar_numeros(a,b):
 """ Soma dois números.
 """

     return a + b
```

Módulos

Módulos de funções

```
def main():
    print(“Meu módulo é o máximo.”)
if __name__ == ‘__main__’:
    main()
```

Encapsulando funcionalidades

Importando bibliotecas

Importando bibliotecas

A função random() da biblioteca random gera um número aleatório entre 0 e 1.

```
import random
random.random() # será entre 0 e 1
```

Importando bibliotecas

A função randint() da biblioteca random gera um número inteiro aleatório.

```
import random
random.ranint(0,2) # será 0, 1 ou 2
```

Tempo de processamento em operações matriciais com e sem Numpy

INSTITUTO BRASILEIRO DE ENSINO,
DESENVOLVIMENTO E PESQUISA