SCC 124 - Introdução à Programação para Engenharias

Entrada e Saída

Professor: André C. P. L. F. de Carvalho, ICMC-USP
Pos-doutorando: Isvani Frias-Blanco
Monitor: Henrique Bonini de Britto Menezes

Aula de hoje

- Introdução
- Métodos de entrada
- Métodos de saída
- Uso de arquivos
- Módulo *piclke*
- Interface com outras linguagens

© André de Carvalho - ICMC/USP

Introdução

- Em vários programas, é necessária a interação com o usuário
  - Entrar valores que serão utilizados em um programa
    - Entrada de valores
  - Gerar valores como resultados de um programa
    - Saída de valores

© André de Carvalho - ICMC/USP

Introdução

- Python tem vários métodos e funções para operações de entrada e saída
- Funções básicas de entrada e saída
  - Entrada
    - Função *input()*
  - Saída
    - Função *print()*

© André de Carvalho - ICMC/USP

Exemplos

- Para ler e imprimir *strings*

```
nome = input (“Qual eh o seu nome?\n”)
print (‘Ola’, nome)
```

- Para ler e imprimir valores inteiros

```
valor = int(input ("Digite o valor: "))
print ("Valor é ", valor)
```

© André de Carvalho - ICMC/USP

Exemplo

```
def reverso (texto):
   return texto[::-1]

def palindromo (texto):
   return texto == reverso (texto)

algo = input("Entre com um texto: ")
If palindromo (algo):
   print(algo + ”é um palindromo")
else:
   print(algo + ”não é um palindromo")
```

Saída:

Entre com um texto: ala
ala é um palindromo

© André de Carvalho - ICMC/USP

Formas alternativas

```
print ('Ola %s,  eu tenho %d biscoitos' %('Professor', 7))
```

Saida:
Ola Professor, Eu tenho 7 biscoitos

```
print ('Ola %s,  Eu tenho %f biscoitos' %('Professor', 7.2))
```

Saida:
Ola Professor, Eu tenho 7.200000 biscoitos

```
print ('Ola %s,  Eu tenho %4.2f biscoitos' %('Professor', 7.2))
```

Saida:
Ola Professor, Eu tenho 7.20 biscoitos

Introdução

- Operações de entrada e saída podem:
  - Entrar com valores a partir de
    - Teclado
    - Arquivo
  - Gerar valores para
    - Monitor de vídeo
    - Arquivo

Entrada e saída em arquivos

- Criar, ler de e escrever em arquivos é essencial em vários programas
  - Necessário criar um objeto da classe *file* (arquivo)
  - Classe *file* possui vários métodos

Classe file

- Alguns métodos
  - Métodos *read*
  - Métodos *readline*
  - Métodos *write*
  - Métodos *open*
  - Métodos *close*

Exemplo

Modo

```
texto = """Tem que estudar e
praticar bastante para conseguir
passar de ano"""
# Abrir arquivo para escrita – usa 'w'
f = open('texto-exemplo.txt', 'w')
# Escrever texto no arquivo
f.write(texto)
# Fechar o arquivo
f.close()
# Se modo não for especificado,
# 'r' é usado por default
f = open('texto-exemplo.txt')
```

```
while True:
    linha = f.readline()
    # Tamanho zero indica EOF
    if len(linha)==0:
        break
    # Variável linha já tem um \n
    # no final de cada linha, já
    # que lê de um arquivo.
    print(linha, end= ' ')
# Fechar o arquivo
f.close()
```


Exemplo

Modo

```
texto = """Tem que estudar e
praticar bastante para conseguir
passar de ano"""
# Abrir arquivo para escrita – usa 'w'
f = open('texto-exemplo.txt', 'w')
# Escrever texto no arquivo
f.write(texto)
# Fechar o arquivo
f.close()
# Se modo não for especificado,
# 'r' é usado por default
f = open('texto-exemplo.txt')
```

Armazena *string* na variável *texto*

Abre arquivo com nome *texto-exemplo.txt* para escrita e atribui ele ao arquivo *f*

Escreve conteúdo da variável *texto* na arquivo *f*

Fecha o arquivo *f*

Abre arquivo com nome *texto-exemplo.txt* para leitura e atribui ele ao arquivo *f*

Exemplo

```
while True:
    linha = f.readline()
    # Tamanho zero indica EOF
    if len(linha)==0:
        break
    # Variável linha já tem um \n
    # no final de cada linha, já
    # que lê de um arquivo.
    print(linha, end= ' ')
# Fechar o arquivo
f.close()
```

Atribui à variável *linha* o *string que está* na primeira linha do arquivo

Enquanto *linha* não for vazia

Escreve valor da variável *linha* na tela e pula para a próxima linha

Fecha o arquivo

Exemplo

- Método *open*
  - Abre o arquivo especificado para:
    - Escrever (w)
    - Ler (r) - usado por default
    - Ler e escrever (r+)
    - Anexar (a)
  - Em um dos seguintes modos
    - Texto (t) - usado por default
    - Binário (b)
    - Existem outros

Modo binário

- Dados são lidos e escritos em bytes
  - Deve ser usado para todos os arquivos que não contêm texto
    - Imagens
    - Códigos executáveis (*.exe*)

Leitura de dados em arquivos

- Método *read()*
  - *Formato: arq.read (size)*
    - Lê trecho de tamanho *size* de um arquivo *arq*, a partir da posição atual no arquivo
      - Retorna um *string* (modo texto) ou um objeto bytes (modo binário)
    - Parâmetro *size* é opcional
      - Se omitido ou for um número negativo todo o restante arquivo, a partir da posição atual, será lido

Leitura de dados em arquivos

- Método *readline()*
  - Utilizado para arquivos de texto
  - Lê uma linha de um arquivo por vez
    - Ao final de cada linha vai ter um comendo para pular de linha (\n), a menos da última linha
    - Quando retorna um *string* vazio
      - Sinaliza que chegou ao final do arquivo

Outros métodos

- Para arquivos de texto
- Para ler todas as linhas de um arquivo *arq* em uma lista, podem ser usados os métodos:
  - *list(arq)*
  - *arq.readlines()*

Procura de dados em arquivo

- Método *seek()*
  - Ex.: *arq.seek(n,m)*
    - Se desloca n bytes (characteres) no arquivo *arq a partir da posição m*
    - Arquivos de texto não usam parâmetro *m (ou m=0)*
    - Em arquivos binários, *m* pode ter os seguintes valores:
      - 0: a partir do início do arquivo
        - Se não tiver valor, assume valor 0 (default)
      - 1: a partir da posição atual no arquivo
      - 2: a partir do final do arquivo, para trás *(n < 0)*

Posição no arquivo

- Método *f.tell()*
  - Retorna um valor inteiro, posição atual no arquivo
    - Modo binário: número de bytes desde o início do arquivo
    - Modo texto: número de caracteres desde o início do arquivo

Procura de dados em arquivo

```
arqcaracteres = 'arquivo.dat'
arq = open(arqcaracteres, 'rb+')
arq.write(b'0123456789abcdef')
a = arq.seek(11) # Vai para o 11o caracter no arquivo
print ("a = ", a)
print ("posição = ", arq.tell())
b = arq.read(3)
print("b = ", b)
```

Saída:

```
a = 11
posição = 11
b = b'bcd'
```


Módulo *pickle*

- Facilita armazenamento de um objeto em um arquivo
  - E recuperá-lo para uso futuro
    - Armazenar o objeto persistentemente
- Arquivo deve ser aberto em modo binário

Módulo *pickle*

```
import pickle

# Definir nome do arquivo que
# armazenará o objeto
arquivolista = 'listadecompras.dat'
# Definir o que será comprado
compras = ['maçã', 'manga', 'cenoura']

# Escrever no arquivo
arq = open(arquivolista, 'wb')
# Escreve os objetos (bytes) no arquivo
pickle.dump(compras, arq)
arq.close()
```

```
# Destruir a variável compras
del compras

# Ler de volta do arquivo
arq= open(arquivolista, 'rb')
# Carregar o objeto do arquivo
listaguardada = pickle.load(arq)
print(listaguardada)
```

Saída:

```
['maçã', 'manga', 'cenoura']
```


Interfaces de Python

- XML
  - DOM, expat
  - XMLRPC, SOAP, Web Services
- Bancos de dados relacionais
  - MySQL, PostgreSQL, Oracle , ODBC, Sybase, Informix
- Java (via Jython)
- Objective C
- COM, DCOM e .NET

Conclusão

- Python possui várias bibliotecas
- Entrada e saída
- Ler e escrever de arquivo
- Métodos
- Módulo *piclke*
- Interface com outras linguagens

Perguntas

Exercício

- Criar arquivo que armazena lista com dados de 5 alunos
- Para cada aluno, armazenar:
  - Nome
  - Número USP
  - Idade
- Abrir arquivo e somar idade dos alunos

Formatos para impressão

%d %hd %ld	Estes formatos mostram o valor como um número decimal dos tipos *int*, *short* e *long*, respectivamente. O sinal % pode ser seguido por um número especificando a largura mínima a ser usada na impressão do número. Se o número é pequeno demais para preencher toda a largura, espaço extra é adicionado à esquerda. De forma que os números se alinhem à direita. Ex.: *printf ("val = %4d\n", val);* /* reserva 4 espaços para imprimir o valor da variável val */
%f	Utilizado para valores dos tipos *float* ou *double*. Exibe valores desses tipos com um ponto decimal. A precisão pode ser especificada indicando quantos dígitos devem ser exibidos à direita do ponto decimal. Ex.: *printf ("val = %8.3f\n", val);* /* imprime um número real com 8 caracteres, 3 deles após o ponto */

Formatos para impressão

%g	Também é usado para valores dos tipos *float* ou *double* Semelhante ao formato %f quando o número a ser impresso cabe em um espaço pequeno. Representa de forma mais compacta números cuja magnitude é ou muito grande ou muito pequena Ex. 27000000.0 ou .00000000006) usando notação científica (ex.: 2.7e+7 ou 6.0e-11). Pode também incluir um campo largura e precisão Especifica o número de dígitos significativos à esquerda do ponto (ao invés dos números a direita do ponto)
%c	Utilizado para valores do tipo char. Exibe apenas um caracter Ex.: *printf ("letra= %c\n, var");* /* imprime um caracter */