Computação I - Python
Aula 12 - Teórica: Modularização

João C. P. da Silva

Carla A. D. M. Delgado

Ana Luisa Duboc

Dept. Ciência da Computação - UFRJ

Modularização

Faça um programa que dado o salário bruto calcula o salário líquido. O salário líquido é calculado a partir do salário bruto, primeiro descontando 11% referente ao INSS, e do resultado, descontando-se 15% de imposto de renda (IR).

Exemplo

- Salário Bruto = R$ 5000,00
- Desconto do INSS = R$ 550,00 (11% de R$ 5000,00)
- Desconto do IR = R$ 667,50 (15% de R$ 4450,00)
- Salário Líquido = 5000 - (550 + 667,50) = 3782,50

Modularização

Faça um programa que dado o salário bruto calcula o salário líquido. O salário líquido é calculado a partir do salário bruto, primeiro descontando 11% referente ao INSS, e do resultado, descontando-se 15% de imposto de renda (IR).

```
1 def salarioLiquido(salarioBruto):
2   """Dado o salario bruto calcula o salario liquido
3   Entrada: float
4   Saida: float"""
5
6   descontoINSS = (salarioBruto)*0.11
7   descontoIR = (salarioBruto - descontoINSS)*0.15
8   salarioLiq = (salarioBruto - descontoINSS - descontoIR)
9   return salarioLiq
```

Modularização

Faça um programa que dado o salário bruto calcula o salário líquido. O salário líquido é calculado a partir do salário bruto, primeiro descontando 11% referente ao INSS, e do resultado, descontando-se 15% de imposto de renda (IR).

```
1 def salarioLiquido(salarioBruto):
2   """Dado o salario bruto calcula o salario liquido
3   Entrada: float
4   Saida: float"""
5
6   descontoINSS = (salarioBruto)*0.11
7   descontoIR = (salarioBruto - descontoINSS)*0.15
8   salarioLiq = (salarioBruto - descontoINSS - descontoIR)
9   return salarioLiq
```

Note que o cálculo do desconto é sempre feito de uma mesma maneira.
Podemos então generalizar seu cálculo.

Modularização

Faça um programa que dado o salário bruto calcula o salário líquido. O salário líquido é calculado a partir do salário bruto, primeiro descontando 11% referente ao INSS, e do resultado, descontando-se 15% de imposto de renda (IR).

```
1 def salarioLiquido(salarioBruto):
2   """Dado o salario bruto calcula o salario liquido
3   Entrada: float
4   Saida: float"""
5
6   descontoINSS = (salarioBruto)*0.11
7   descontoIR = (salarioBruto - descontoINSS)*0.15
8   salarioLiq = (salarioBruto - descontoINSS - descontoIR)
9   return salarioLiq
```

Note que o cálculo do desconto é sempre feito de uma mesma maneira.
Podemos então generalizar seu cálculo.

```
1 def calculadesconto(valorBruto,percentual):
2   """Dado um valor bruto e um percentual de desconto
3   calcula o valor com o desconto aplicado
4   Entrada: float, float
5   Saida: float"""
6
7   desconto = valorBruto*percentual/100.0
8   valorLiquido = valorBruto-desconto
9   return valorLiquido
```

Modularização

Faça um programa que dado o salário bruto calcula o salário líquido. O salário líquido é calculado a partir do salário bruto, primeiro descontando 11% referente ao INSS, e do resultado, descontando-se 15% de imposto de renda (IR).

```
1 def calculadesconto(valorBruto,percentual):
2   """Dado um valor bruto e um percentual de desconto
3   calcula o valor com o desconto aplicado
4   Entrada: float, float
5   Saida: float"""
6
7   desconto = valorBruto*percentual/100.0
8   valorLiquido = valorBruto-desconto
9   return valorLiquido
```

Usando a função acima:

Modularização

Faça um programa que dado o salário bruto calcula o salário líquido. O salário líquido é calculado a partir do salário bruto, primeiro descontando 11% referente ao INSS, e do resultado, descontando-se 15% de imposto de renda (IR).

```
1 def calculadesconto(valorBruto,percentual):
2   """Dado um valor bruto e um percentual de desconto
3   calcula o valor com o desconto aplicado
4   Entrada: float, float
5   Saida: float"""
6
7   desconto = valorBruto*percentual/100.0
8   valorLiquido = valorBruto-desconto
9   return valorLiquido
```

Usando a função acima:

```
1 def salarioLiquido(salarioBruto):
2   """Dado o salario bruto calcula o salario liquido
3   Entrada: float
4   Saida: float"""
5
6   salariosemINSS = calculadesconto(salarioBruto,11.0)
7   salariosemIRsemINSS = calculadesconto(salariosemINSS,15.0)
8   return salariosemIRsemINSS
```

Modularização

- Torna o código mais legível.
- Permite que algumas funcionalidades sejam reaproveitadas.
- Permite que partes do código sejam testadas isoladamente.

Modularização

- Torna o código mais legível.
- Permite que algumas funcionalidades sejam reaproveitadas.
- Permite que partes do código sejam testadas isoladamente.

```
1 def calculadesconto(valorBruto,percentual):
2   """Dado um valor bruto e um percentual de desconto
3   calcula o valor com o desconto aplicado
4   Entrada: float, float
5   Saida: float"""
6
7   desconto = valorBruto*percentual/100.0
8   valorLiquido = valorBruto-desconto
9   return valorLiquido
```

```
1 >>> calculadesconto(5000.00,11.0)
2     4450.0
3
4 >>> calculadesconto(4450.00,15.0)
5     3782.5
```

Modularização

Faça um programa que permita o lançamento de notas de uma disciplina e a consulta a tais notas. O programa deve:

- possuir um menu principal com o seguinte layout:

```
Escolha uma opção :
1 - lançamento de notas de uma disciplina
2 - listar notas de uma disciplina
0 - terminar
Opção :
```

- possuir uma tela para lançamento das notas de uma disciplina com o seguinte layout:

```
Nome da Disciplina : computacao1
Ano-Periodo (XX-X) : 15-2
matrícula (0 para terminar) :106041177
Nota : 8.5
matrícula (0 para terminar) :114123723
Nota : 9.2
matrícula (0 para terminar) :123041173
Nota : 3.3
matrícula (0 para terminar) :  0
```

Modularização

Faça um programa que permita o lançamento de notas de uma disciplina e a consulta a tais notas. O programa deve:

- possuir um menu para escolher qual disciplina terá suas notas apresentadas com o seguinte layout:

```
Escolha uma disciplina:
1- computacao1-15-2
2- calculo1-15-2
0- terminar
Opção :
```

- possuir uma saída das notas de uma disciplina com o seguinte layout:

```
Disciplina - computacao1-15-2
Matrícula      Nota
106041177      8.5
114123723      9.2
123041173      3.3
```

Modularização

Faça um programa que permita o lançamento de notas de uma disciplina e a consulta a tais notas. O programa deve:

- possuir um dicionário cuja chave é uma string que representa o nome da disciplina e o período em que ela foi lecionada, e o seu conteúdo será também um dicionário cuja chave é uma string que representa o matrícula do aluno e seu conteúdo é a sua nota.

Exemplo

- Notas de Computação 1 de 2015-2 formam o dicionário {'106041177': 8.5, '114123723':9.2, '123041173':3.3}
- Notas de Cálculo 1 de 2015-2 formam o dicionário {'123041173': 1.4, '142123343':4.3, '187233954':6.7}
- Teremos então um dicionário com todas as disciplinas e suas notas
  {'computacao1-15-2' : {'106041177': 8.5, '114123723':9.2, '123041173':3.3} ,
  'calculo1-15-2' : {'123041173': 1.4, '142123343':4.3, '187233954':6.7}}

computacao1-15-2		**calculo 1-15-2**	
matrícula	**Nota**	**matrícula**	**Nota**
106041177	8.5	123041173	1.4
114123723	9.2	142123343	4.3
123041173	3.3	187233954	6.7

Modularização

Vamos resolver este problema passo a passo.

Esboço: A função **main()**

```
1  def main():
2    """Sistema academico de gerencia de notas e turmas para uso do professor"""
3
4    # Definir dicionario das notas
5    # Exibir o menu principal
6    while opcao != 0:
7      if opcao == 1:
8        # Lancar notas de uma disciplina
9        # acrescentar no dicionario as notas de uma disciplina
10     elif opcao == 2:
11       # Escolha de qual disciplina se quer saber as notas
12       # Mostrar as notas da disciplina escolhida
13     # Exibir o menu principal
14 if __name__ == "__main__":
15   main()
```

Modularização

Vamos resolver este problema passo a passo.

Esboço: A função **main()**

```
1  def main():
2    """Sistema academico de gerencia de notas e turmas para uso do professor"""
3
4    dicionarioDisciplinas = {} # Definir dicionario das notas
5    # Exibir o menu principal
6    while opcao != 0:
7      if opcao == 1:
8        # Lancar notas de uma disciplina
9        # acrescentar no dicionario as notas de uma disciplina
10     elif opcao == 2:
11       # Escolha de qual disciplina se quer saber as notas
12       # Mostrar as notas da disciplina escolhida
13     # Exibir o menu principal
14 if __name__ == "__main__":
15   main()
```

Modularização

Vamos resolver este problema passo a passo.

Passo 1: Fazendo o menu principal:

```
1  def menuPrincipal():
2    """Exibe o menu principal e retorna a opcao escolhida
3    Entrada: none
4    Saida: str"""
5
6    print("Escolha uma das opcoes: ")
7    print("1- lancamento de notas de uma disciplina")
8    print("2- listar notas de uma disciplina")
9    print("0- terminar")
10   opcao = input("Opcao : ")
11   return opcao
```

Para testar esta função, basta chamá-la no console digitando menuPrincipal().

Modularização

Vamos resolver este problema passo a passo.

Esboço: A função **main()**

```
1  def main():
2    """Sistema academico de gerencia de notas e turmas para uso do professor"""
3
4    dicionarioDisciplinas = {} # Definir dicionario das notas
5    opcao = menuPrincipal() # Exibir o menu principal
6    while opcao != 0:
7      if opcao == 1:
8        # Lancar notas de uma disciplina
9        # acrescentar no dicionario as notas de uma disciplina
10     elif opcao == 2:
11       # Escolha de qual disciplina se quer saber as notas
12       # Mostrar as notas da disciplina escolhida
13     opcao = menuPrincipal() # Exibir o menu principal
14 if __name__ == "__main__":
15   main()
```

Modularização

Vamos resolver este problema passo a passo.

Passo 2: Fazendo a tela para lançamento das notas de uma disciplina.

```
Nome da Disciplina : computacao1
Ano-Periodo (XX-X) : 15-2
matrícula (0 para terminar) :106041177
Nota : 8.5
matrícula (0 para terminar) :114123723
Nota : 9.2
matrícula (0 para terminar) :123041173
Nota : 3.3
matrícula (0 para terminar) :  0
```

Modularização

Vamos resolver este problema passo a passo.

Passo 2: Fazendo a tela para lançamento das notas de uma disciplina.

```
Nome da Disciplina : computacao1
Ano-Periodo (XX-X) : 15-2
matrícula (0 para terminar) :106041177
Nota : 8.5
matrícula (0 para terminar) :114123723
Nota : 9.2
matrícula (0 para terminar) :123041173
Nota : 3.3
matrícula (0 para terminar) :  0
```

Lembre que:

- Devemos perguntar o nome da disciplina e o ano-período que ela foi oferecida. Usaremos estas informações para construir a chave do dicionário.
- As notas devem ser armazenadas em um dicionário onde a chave deve ser o matrícula do aluno e o conteúdo deve ser a nota obtida na matéria.
- E se a disciplina-ano-periodo já estiver no dicionário?

Modularização

Vamos resolver este problema passo a passo.

Passo 2: Fazendo a tela para lançamento das notas de uma disciplina.

```
Nome da Disciplina : computacao1
Ano-Periodo (XX-X) : 15-2
matrícula (0 para terminar) :106041177
Nota : 8.5
matrícula (0 para terminar) :114123723
Nota : 9.2
matrícula (0 para terminar) :123041173
Nota : 3.3
matrícula (0 para terminar) :  0
```

```
1  def lancarNotas():
2    """Funcao para lancamento de notas de uma disciplina
3    Entrada: none
4    Saida: str,dict"""
5
6    dicionarioNotas = {}
7    disciplina = input("Nome da Disciplina : ")
8    periodo = input("Ano-Periodo (XX-X): ")
9    nomeDisciplina = disciplina+'-'+periodo
10   matricula = input('Matricula (0 para terminar) : ')
11   while matricula != '0':
12     nota = float(input('Nota : '))
13     dicionarioNotas[matricula] = nota
14     matricula = input('Matricula (0 para terminar) : ')
15   return (nomeDisciplina,dicionarioNotas)
```

Modularização

Vamos resolver este problema passo a passo.

Passo 2: Fazendo a tela para lançamento das notas de uma disciplina.

```
1  def lancarNotas():
2    """Funcao para lancamento de notas de uma disciplina
3    Entrada: none
4    Saida: str,dict"""
5
6    dicionarioNotas = {}
7    disciplina = input("Nome da Disciplina : ")
8    periodo = input("Ano-Periodo (XX-X): ")
9    nomeDisciplina = disciplina+'-'+periodo
10   matricula = input('Matricula (0 para terminar) : ')
11   while matricula != '0':
12     nota = float(input('Nota : '))
13     dicionarioNotas[matricula] = nota
14     matricula = input('Matricula (0 para terminar) : ')
15   return (nomeDisciplina,dicionarioNotas)
```

Teste esta função para os valores da seguinte tabela:

computacao1-15-2		calculo 1-15-2	
matrícula	**Nota**	**matrícula**	**Nota**
106041177	8.5	123041173	1.4
114123723	9.2	142123343	4.3
123041173	3.3	187233954	6.7

Modularização

Vamos resolver este problema passo a passo.

Passo 2: Fazendo a tela para lançamento das notas de uma disciplina.

```
1  def lancarNotas():
2    """Funcao para lancamento de notas de uma disciplina
3    Entrada: none
4    Saida: str,dict"""
5
6    dicionarioNotas = {}
7    disciplina = input("Nome da Disciplina : ")
8    periodo = input("Ano-Periodo (XX-X): ")
9    nomeDisciplina = disciplina+'-'+periodo
10   matricula = input('Matricula (0 para terminar) : ')
11   while matricula != '0':
12     nota = float(input('Nota : '))
13     dicionarioNotas[matricula] = nota
14     matricula = input('Matricula (0 para terminar) : ')
15   return (nomeDisciplina,dicionarioNotas)
```

Retorno da função para cada disciplina:

- ('computacao1-15-2', {'106041177': 8.5, '114123723': 9.2, '123041173': 3.3})
- ('calculo1-15-2', {'123041173': 1.4, '142123343': 4.3, '187233954': 6.7})

Vamos usar esta tupla para construir o dicionário com todas as disciplinas e suas respectivas notas.

Modularização

Vamos resolver este problema passo a passo.

Esboço: A função **main()**

```
1  def main():
2    """Sistema academico de gerencia de notas e turmas para uso do professor"""
3
4    dicionarioDisciplinas = {} # Definir dicionario das notas
5    opcao = menuPrincipal() # Exibir o menu principal
6    while opcao != 0:
7      if opcao == 1:
8        nomeDisciplina,dicionarioNotas = lancarNotas() # Lancar notas de uma disciplina
9        # acrescentar no dicionario as notas de uma disciplina
10     elif opcao == 2:
11       # Escolha de qual disciplina se quer saber as notas
12       # Mostrar as notas da disciplina escolhida
13     opcao = menuPrincipal() # Exibir o menu principal
14 if __name__ == "__main__":
15   main()
```

Modularização

Vamos resolver este problema passo a passo.

Esboço: A função **main()**

```
1  def main():
2    """Sistema academico de gerencia de notas e turmas para uso do professor"""
3
4    dicionarioDisciplinas = {} # Definir dicionario das notas
5    opcao = menuPrincipal() # Exibir o menu principal
6    while opcao != 0:
7      if opcao == 1:
8        # Lancar notas de uma disciplina
9        nomeDisciplina,dicionarioNotas = lancarNotas()
10       # Acrescentar no dicionario as notas de uma disciplina
11       dicionarioDisciplinas[nomeDisciplina]=dicionarioNotas
12     elif opcao == 2:
13       # Escolha de qual disciplina se quer saber as notas
14       # Mostrar as notas da disciplina escolhida
15     opcao = menuPrincipal() # Exibir o menu principal
16 if __name__ == "__main__":
17   main()
```

Modularização

Vamos resolver este problema passo a passo.
Passo 3: Vamos fazer um menu para escolher qual disciplina terá suas notas apresentadas.

```
Escolha uma disciplina:
1- computacao1-15-2
2- calculo1-15-2
0- terminar
Opção :
```

Lembre que:

- Os nomes das disciplinas serão as chaves do dicionário *dicionarioDisciplinas*.

Modularização

Vamos resolver este problema passo a passo.
Passo 3: Vamos fazer um menu para escolher qual disciplina terá suas notas apresentadas.

```
Escolha uma disciplina:
1- computacao1-15-2
2- calculo1-15-2
0- terminar
Opção :
```

Lembre que:

- Os nomes das disciplinas serão as chaves do dicionário *dicionarioDisciplinas*.

```
1  def menuDisciplinas(Disciplinas):
2    """Mostra o menu para escolher as disciplinas
3    Entrada: list
4    Saida: string"""
5
6    print("Escolha uma disciplina:")
7    for indice in range(len(Disciplinas)):
8      print(indice+1,"- ",Disciplinas[indice])
9      print("0- terminar")
10     opcao = input("Opcao : ")
11   return Disciplinas[opcao-1]
```

Teste a função com a lista ['computacao1-15-2','calculo1-15-2']

Modularização

Vamos resolver este problema passo a passo.

Esboço: A função **main()**

```
1  def main():
2    """Sistema academico de gerencia de notas e turmas para uso do professor"""
3
4    dicionarioDisciplinas = {} # Definir dicionario das notas
5    opcao = menuPrincipal() # Exibir o menu principal
6    while opcao != 0:
7      if opcao == 1:
8        # Lancar notas de uma disciplina
9        nomeDisciplina,dicionarioNotas = lancarNotas()
10       # Acrescentar no dicionario as notas de uma disciplina
11       dicionarioDisciplinas[nomeDisciplina]=dicionarioNotas
12     elif opcao == 2:
13       # Constroi uma lista com os nomes das disciplinas
14       Disciplinas = dict.keys(dicionarioDisciplinas)
15       # Escolha de qual disciplina se quer saber as notas
16       DisciplinaSelecionada = menuDisciplinas(Disciplinas)
17       # Mostrar as notas da disciplina escolhida
18     opcao = menuPrincipal() # Exibir o menu principal
19 if __name__ == "__main__":
20   main()
```

Modularização

Vamos resolver este problema passo a passo.

Passo 4: Vamos fazer uma função para mostrar as notas de uma disciplina. Lembre que:

- A entrada desta função vai ser o dicionário com as notas de uma certa disciplina.

Modularização

Vamos resolver este problema passo a passo.

Passo 4: Vamos fazer uma função para mostrar as notas de uma disciplina. Lembre que:

- A entrada desta função vai ser o dicionário com as notas de uma certa disciplina.

```
1  def imprimeNotas(DisciplinaSelecionada,notasDisciplina):
2    """imprime as notas de uma disciplina
3    Entrada: string (nome da disciplina), dict (aluno:nota)
4    Saida: none"""
5
6    print("Disciplina - "+DisciplinaSelecionada)
7    print("Matricula NOTA")
8    for aluno in notasDisciplina:
9      print(str.format("{0} {1:2.1f}",aluno,notasDisciplina[aluno]))
10   return
```

Modularização

Vamos resolver este problema passo a passo.

Passo 4: Vamos fazer uma função para mostrar as notas de uma disciplina. Lembre que:

- A entrada desta função vai ser o dicionário com as notas de uma certa disciplina.

```
1  def imprimeNotas(DisciplinaSelecionada,notasDisciplina):
2    """imprime as notas de uma disciplina
3    Entrada: string (nome da disciplina), dict (aluno:nota)
4    Saida: none"""
5
6    print("Disciplina - "+DisciplinaSelecionada)
7    print("Matricula NOTA")
8    for aluno in notasDisciplina:
9      print(str.format("{0} {1:2.1f}",aluno,notasDisciplina[aluno]))
10   return
```

Teste esta função para as seguintes chamadas:

- imprimeNotas("computacao1-15-2",{'106041177': 8.5, '114123723':9.2, '123041173':3.3})
- imprimeNotas("calculo1-15-2",{'123041173': 1.4, '142123343':4.3, '187233954':6.7})

Modularização

Vamos resolver este problema passo a passo.

A função **main()**

```
1  def main():
2    """Sistema academico de gerencia de notas e turmas para uso do professor"""
3
4    dicionarioDisciplinas = {} # Definir dicionario das notas
5    opcao = menuPrincipal() # Exibir o menu principal
6    while opcao != 0:
7      if opcao == 1:
8        # Lancar notas de uma disciplina
9        nomeDisciplina,dicionarioNotas = lancarNotas()
10       # acrescentar no dicionario as notas de uma disciplina
11       dicionarioDisciplinas[nomeDisciplina]=dicionarioNotas
12     elif opcao == 2:
13       # Constroi uma lista com os nomes das disciplinas
14       Disciplinas = dict.keys(dicionarioDisciplinas)
15       # Escolha de qual disciplina se quer saber as notas
16       DisciplinaSelecionada = menuDisciplinas(Disciplinas)
17       # Mostrar as notas da disciplina escolhida
18       imprimeNotas(DisciplinaSelecionada,dicionarioDisciplinas[DisciplinaSelecionada])
19     opcao = menuPrincipal() # Exibir o menu principal
20 if __name__ == "__main__":
21   main()
```

Modularização

- A programação permeia muitas áreas atualmente. O nível de sofisticação esperado é elevado.
- Para justificar o investimento, um trecho de código deve ser reutilizável. Várias pessoas reutilizam e compartilham seus códigos, criando comunidades.
- A **organização e a legibilidade** do código são requisitos essenciais, tão importantes quanto eficiência e eficácia.
- Ter seu código bem documentado também é importante para que você mesmo consiga utilizá-lo futuramente. É muito fácil esquecer o que um código faz, e gasta-se tempo para tentar entendê-lo novamente.

Modularização

- Para atender às demandas atuais, cada vez mais complexas e sofisticadas, o programador deve pesquisar sobre bibliotecas disponíveis que o auxiliem.
- Tão importante quanto saber programar tudo o que precisa, é saber aproveitar o que está disponível, ou não é possível atender às demandas em tempo aceitável.
- É importante ser capaz de ler e entender a documentação de tais bibliotecas. Um bom conhecimento da linguagem de programação em que a biblioteca foi escrita ajuda muito.
- Uma vez que se produza código reutilizável e de boa qualidade, é de bom tom compartilhar! A comunidade agradece e reconhece o esforço.

Autores

- João C. P. da Silva ▸ Lattes
- Carla Delgado ▸ Lattes
- Ana Luisa Duboc ▸ Lattes

Colaboradores

- Anamaria Martins Moreira ▸ Lattes
- Fabio Mascarenhas ▸ Lattes
- Leonardo de Oliveira Carvalho ▸ Lattes
- Charles Figueiredo de Barros ▸ Lattes
- Fabrício Firmino de Faria ▸ Lattes

Computação I - Python
Aula 12 - Teórica: Modularização

João C. P. da Silva

Carla A. D. M. Delgado

Ana Luisa Duboc

Dept. Ciência da Computação - UFRJ