

`print("Hello World")`

EU ODEIO O PYTHON

2^a Edição

Helder Guerreiro

2016

Deus seja Louvado.

Guerreiro, Helder

Eu odeio o Python 2º ed. / Helder Guerreiro – Manaus, 2016.

Bibliografia

Livro não catalogado e não institucional, o mesmo é amador.

SUMÁRIO

Conselhos de Um Amigo	4
Segunda Edição?	5
Nos Passos de Um Bebê	6
Algoritmos ou Programas?	8
1. O Início de Uma Novela	9
1.1 Primeiros Passos	9
1.2 Primeiras Funções de entrada	12
1.3 Módulos (bibliotecas)	14
1.4 Questões Resolvidas	16
Questão 1 – Custo da pintura de um muro	16
Questão 2 – Custo da pintura de um muro (reformulado)	18
Questão 3 – Caixa eletrônico	20
Questão 4 – Ordene três números inteiros	23
Questão 5 – Área do Círculo e Volume da Esfera	24
Questão 6 – Área do Triângulo	25
Questão 7 – Distância entre dois pontos na superfície da Terra	27
Questão 8 – Soma dos dígitos de um número inteiro	30
1.5 Avaliações	33
Avaliação A – A1	33
Avaliação B – A1	34
2. Decisões da Vida – As Condicionais	36
2.1 Comando Condicional	36
2.2 Operadores Condicionais	38
2.3 Questões Resolvidas	40
Questão 1 – Par ou ímpar?	40
Questão 2 – De volta ao problema do caixa eletrônico	42
Questão 3 – Ponto e reta	45
Questão 4 – Desconto	46
Questão 5 – Angry Birds	47
Questão 6 – Tanque de combustível	50
2.4 Avaliações	52
Avaliação A – A2	52
Avaliação B – A2	54

3. Sua Vida Mais Difícil – As Condicionais Encadeadas	56
3.1 Juntem os Dedos: Hora da Fusão!	56
3.2 Novas Funções	56
3.3 Um Novo Módulo (Random)	57
3.4 Questões Resolvidas	58
Questão 1 – De volta à área do triângulo	58
Questão 2 – Isósceles, equilátero ou escaleno?	62
Questão 3 – Animais nas cédulas do Real	65
Questão 4 – Conta de Energia	67
Questão 5 – Volume de combustível	70
Questão 6 – Sobreposição de Intervalos	74
3.5 Avaliações	77
Avaliação A – A3	77
Avaliação B – A3	79
4. Sua Vida um Inferno – Repetição por Condição	82
4.1 Variável Contadora e Variável Acumuladora	82
4.1.1 Variável Contadora	82
4.1.2 Variável Acumuladora	83
4.2 Questões Resolvidas	84
Questão 1 – Testando um Script	84
Questão 2 – Soma de uma coleção de números	86
Questão 3 – Somas de várias coleções de números	88
Questão 4 – Média de uma coleção de números	90
Questão 5 – Média de várias coleções de números	92
Questão 6 – Aproximação de π	94
4.3 Avaliações	97
Avaliação A – A4	97
Avaliação B – A4	99
5. Álgebra uma Garota Difícil de Conquistar – Vetores	103
5.1 Aprendendo a Invocar Demônios	103
5.2 Funções e Operações de Vetores	104
5.2.1 Selecionando Parte de um Veto	104
5.2.2 Operações com Vetores	105
5.3 Gráficos	106

5.3.1 A Invocação do Mal	106
5.3.2 Trabalhando como Doutores em T.I.	106
Exemplo Louco	107
5.4 Questões Resolvidas	112
Questão 1 – Vetor de zeros	112
Questão 2 – Vetor de uns	113
Questão 3 – Geração de vetores linearmente espaçados	114
Questão 4 – Informações básicas de um vetor	116
Questão 5 – Quantas ocorrências? (1)	118
Questão 6 – Quantas ocorrências? (2)	120
Questão 7 – Qual a porcentagem de acerto?	122
Questão 8 – Área de um polígono	124
Questão 9 – Raízes de um polinômio	128
Questão 10 – Derivada de um polinômio	130
5.5 Avaliações	134
Avaliação A – A5	134
Avaliação B – A5	136
6. Vetores no Modo Avançado - Estruturas de Repetição por Contagem	138
6.1 Uma Função Inovadora	138
6.2 Questões Resolvidas	140
Questão 1 – Quantos elementos são pares?	140
Questão 2 – Aprovado, mas não pode ser monitor	143
Questão 3 – Contar ocorrências (1): Time de Futebol	145
Questão 4 – Contar ocorrências (2): Aprovação em disciplina	148
Questão 5 – Contar ocorrências (3): Faltas ao trabalho	151
Questão 6 – Fração contínua (1): Raiz quadrada de dois	155
Questão 7 – Arte ASCII (1)	158
6.3 Avaliações	162
Avaliação A – A6	162
Avaliação B – A5	165
7. No Coração da Álgebra Linear – As Matrizes	167
7.1 A Construção do Império Matricial	167
7.2 Funções e Operações de Matrizes	168
7.2.1 As Dimensões	168

7.2.2 As Funções	169
7.2.3 Seleção de Valores	169
7.2.4 Operações Entre Matrizes	171
7.3 Questões Resolvidas	173
Questão 1 - SEL(1): Cesta de frutas	173
Questão 2 - SEL(2): Alimentando Bactérias	175
Questão 3 - SEL(3): Fluxo de Tráfego	177
Questão 4 - Impressão de padrão	180
Questão 5 - Horas de trabalho em uma empresa (1)	183
Questão 6 - Horas de trabalho em uma empresa (2)	185
Questão 7 - Distância entre duas cidades	188
8. Fechando Com Chave de Ouro - Strings	190
8.1 Um Comportamento Conhecido	190
8.2 Trabalhando com Strings	192
8.3 Testes Lógicos	195
8.3.1 Funções "is"	195
8.4 De Strings Para Números Ou Vice-Versa	197
8.5 Questões Resolvidas	199
Questão 1 - Data por extenso	199
Questão 2 - Formato de datas	202
Questão 3 - Operações básicas envolvendo strings	206
9. Arquivos	208
9.1 Pra que isso?	208
9.2 Abrindo e fechando	208
9.3 Modos	208
9.4 Escrevendo	209
9.5 Lendo	211
9.6 Dados numéricos	213
Um Futuro Não Tão Distante	215
O objetivo final	216

Conselhos de Um Amigo

Olá, meu nome é Helder Guerreiro aluno de Engenharia Química na Universidade Federal do Amazonas (UFAM) esses conselhos que lhe darei serão acerca do seu modo de aprendizagem com esta apostila, a qual foi criada para suprir as necessidades dos alunos que não se dão bem de forma alguma com a programação do Python. Um dia eu também cheguei a achar as aulas e laboratórios do Python inúteis e sem pé e sem cabeça, mas com o tempo percebi que se analisarmos com cuidado e carinho o Python se torna interessante e muito útil e infinitos casos como você verá nesta apostila e vendo a grande dificuldade da minha turma e dependência em confiar naqueles que sabia mais eu criei esta apostila, não pensando nas aulas do primeiro período mas pensando no futuro, existem disciplinas que irão exigir de você a técnica de programação, e es me aqui para lhe ajudar.

Quando fores estudar nesta apostila tenha muita paciência, o ponto forte para você aprender aqui são os exercícios resolvidos, eles são formados por textos que ajudam a entender a construção do script e como cria-lo também e no final é apresentado o resultado final, ao ler tenha calma e leia atentamente, o texto não é tão grande assim e quanto maior a dificuldade da questão maior é o texto. Eu não irei mostrar o resultado logo de cara, irei criar uma base de conhecimento e especulações que levarão você a entender como aquilo foi criado, por isso, não vá logo pulando para o resultado final, leia com calma. Ao estudar nesta apostila tenha sempre o seu programa Python aberto, para você fazer os testes das funções e comando que serão apresentados, também use o Python na hora de estudar as questões resolvidas para que você mesmo vá criando seus scripts com base na minha didática.

Esta apostila não fala tudo sobre o Python, ela é uma introdução a esse grande programa que tem muitas coisas a nos ensinar, existem dois assuntos que aparecem nesta apostila, mas não podem ser aprofundados que são os gráficos produzidos pelo Python e o assunto Strings, que apesar desse último parecer fácil ele é muito complicado e exige uma apostila só para ele, nesses dois assuntos você será introduzido nesta apostila e aprenderá fazer o básico e o suficiente, numa outra ocasião você aprenderá melhor como trabalhar com esses assuntos.

Desejo boa sorte aos seus estudos, de onde quer que você for e de qualquer curso que você seja eu desejo que aprenda a utilizar o Python e saia da dependência daqueles que são mais inteligentes nesses assuntos, eu fiz esta apostila para você, um dia eu também não sabia de muita coisa, mas eu estudei e aprendi e estou aqui para lhe passar esse ensinamento da melhor forma possível. Bons estudos!

Segunda Edição?

Nesta segunda edição eu irei trazer alguns assuntos novos que talvez você ache interessante, assuntos que possam lhe ajudar em tarefas simples do dia a dia, tenho certeza que valerá a pena você ter em mãos esta segunda edição.

Aqui também devo falar que todo meu entendimento e base para criar esta apostila veio dos meus estudos na matéria cedida ao meu curso e administrada pela ICOMP o Instituto de Computação da UFAM, os arquivos que usei para estudo de base vieram deles da mesma forma como alguns scripts produzidos pelo instituto ao qual usei para lhe ensinar a como resolver certas questões. Então, eu não sou um gênio e não tirei tudo isso aqui do bolso, eu cursei a matéria de Introdução a Programação de Computadores (IPC) da ICOMP e logo em seguida me dediquei em aprender de uma forma diferente e mais leiga a qual passarei para você em toda esta apostila.

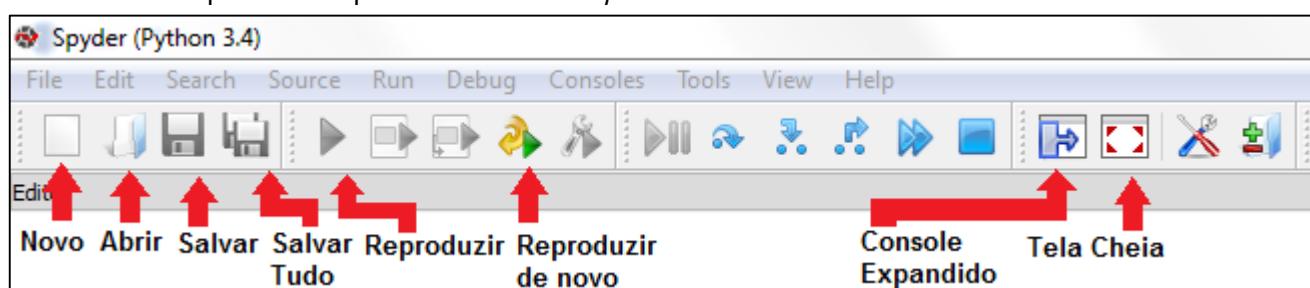
Devo falar um pouco de mim? Bom eu não sou um cara inteligente, conheço muita gente mil vezes melhor do que eu, então eu sou um cara burro que aprendeu a estudar programação de um jeito fácil e agora quero passar isso a você, bom na minha concepção os burros é que deveriam criar livros e apostilas, por que eles não conseguem aprender do jeito difícil e quando aprendem é de um jeito mais fácil, mas os gênios aprendem do modo difícil e quando criam um livro acham que todo mundo vai entender sua linguagem.

Boa sorte e bons estudos o bate papo acaba aqui.

Nos Passos de Um Bebê

A primeira coisa que devo falar a você é que você deve baixar o Python para o seu computador, se você quiser estudar sem ele, melhor desistir da vida amigo. Eu uso o programa chamado Anaconda Python e é o mais indicado pelos professores da área, baixe-o aqui: <https://www.continuum.io/downloads>, escolha as configurações que batem com a do seu computador e pronto, é grátis. A versão a qual me baseio para os estudos é a 3.4, se tiver outra versão, não tem problema, Python sempre será Python.

Vou lhe apresentar o painel do Anaconda Python:



Na apresentação dos ícones de ação do Anaconda Python, aqueles que não foram destacados não é necessário usar, em tudo que você aprender aqui não há necessidade das outras botões, eles são mais avançados e são para quem está bem mais avançado no estudo do Python.

Dos menus em cima dos ícones, só será necessário o uso do menu File e Edit, os outros podem ser desconsiderados no nosso estudo.

Este é o console, é a janela em que acontecerá todas as ações dos scripts, aonde se darão entradas de informações e aonde se darão os resultados dos scripts criados por você.

The screenshot shows the Spyder Console window titled 'Console'. It displays the Python 3.4.3 environment information: 'Python 3.4.3 |Anaconda 2.2.0 (64-bit)| (default, Mar 6 2015, 12:06:10) [MSC v.1600 64 bit (AMD64)] on win32'. It also shows the copyright message: 'Type "help", "copyright", "credits" or "license" for more information.' and the prompt '>>>'. At the bottom of the window, there are status bars for 'Permissions: RW', 'End-of-lines: CRLF', 'Encoding: UTF-8', 'Line: 1', 'Column: 1', and 'Memory: 72 %'.

Abaixo você estará vendo o campo de batalha, é aqui onde aconteceu todas as dores de cabeça que um dia eu tive e você também. Todas as suas linhas são contadas e sempre é bom criar um cabeçalho antes de criar um script qualquer.

```
1 # -*- coding: utf-8 -*-
2 """
3 04/02/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9
10
11
```

Com todas essas informações podemos começar os nossos estudos! Agora mantenha o seu Python aberto e estude e analise cada assunto e comandos apresentados aqui, neste caso a curiosidade é uma boa aprendizagem para você, não se acanhe em saber como realmente uma função ou comando trabalha, coloque no seu Python e a teste e continue sua rotina de estudos.

Algoritmos ou Programas?

Bom antes de começarmos a novela das oito cheia de ação, drama e suspense devo mostrar aqui para todos a crítica diferença entre esses dois termos.

Tudo que você for fazer aqui, exatamente tudo, serão **algoritmos**, que aqui eu os chamo de scripts, eles são informações que foram manuseadas pelo programador para que o Python trabalhasse de acordo com essas informações. Vou explicar melhor, o Python é um **programa** e os algoritmos são informações produzidas por ele, essas informações são construídas pelo programador que as constroem para que o programa venha executar as informações transcritas em algoritmos, veja um exemplo prático:

- Temos um robô, uma máquina.

O robô sem informações não pode fazer nada;

Pois ele tem os mecanismos já dentro de si, mas que os faz movimentar são as informações;

Um programador constrói um algoritmo que faz o robô se movimentar;

O algoritmo diz que se o robô chegar a 30 cm de uma parede ele deve parar e se virar;

O algoritmo diz também que o robô só irá andar se for acionado um comando por voz dizendo: "ande";

Da mesma forma, quando for a hora de parar, outro comando de voz será dito dizendo: "pare";

O algoritmo é inserido no robô e logo em seguida após o sistema reconhecer o algoritmo o comando é acionado;

O robô começa a andar tranquilamente e quando chega perto de uma parede ele se vira e começa a andar de novo;

O algoritmo só foi possível por que o robô tinha as ferramentas necessárias para que ele fosse executado;

Da mesma forma, só foi possível o robô andar por que ele tinha as informações necessárias para isso.

Moral da história, o robô é um programa, ele tem os mecanismos, as ferramentas e os aparatos estruturais, mas as informações ainda não foram inseridas nele, essas informações são algoritmos (scripts) que dão ordens e caminhos para que o programa possa funcionar à vontade do usuário.

Não adianta você ter um carro, mas ele não sabe a hora que seu tanque está cheio ou vazio; não adianta você ter um computador se o mesmo não tem um sistema operacional; não adianta você ter um jogo eletrônico de última geração se o jogo não faz nada, só fica parado.

Para você que é fã dos jogos, eles são movimentados por scripts, uma cidade construída aos mínimos detalhes num jogo enorme são estruturas de um programa, ou seja, a cidade do jogo é um campo de infinitas possibilidades, é a estrutura do jogo, mas o que faz os carros andarem, os personagens falarem, as pessoas andarem, o céu escurecer são algoritmos, scripts que ordenam o jogo o que ele deve fazer a cada situação. Nisso eu fecho a explicação e espero que entenda que programas são diferentes de algoritmos.

1. O Início de Uma Novela

1.1 Primeiros Passos

Aprendendo as partes mais básicas do Python vamos começar do zero!

Logo de início é muito importante criar um cabeçalho onde se colocará nome, data e alguma especificação do arquivo. Para colocar qualquer tipo de texto que não sejam comandos, ou seja, cabeçalhos e comentários usa-se estes comandos: ("""") três aspas para grandes comentários que ultrapassem mais de uma linha ou cabeçalhos; (#) Jogo da velha para comentários pequenos de uma só linha.

```
Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View Help
Editor - C:\Users\User\Documents\Python Scripts\untitled1.py
untitled1.py*
1 # -*- coding: utf-8 -*-
2 """
3 Data: 04/02/2016
4
5 Autor: Helder Guerreiro
6
7 Cabeçalho
8 """
9
10 #Comentário rápido
11
12
```

Figura 1: Cabeçalho e comentários

Fonte: Autoria própria

Toda função no Python fica roseado e deve ser acompanhado de parênteses, é como se a função fosse uma ferramenta e dentro dos parênteses é que está o objetivo dessa ferramenta. Se for colocar um texto dentro da função deve-se usar ("""") duas aspas.

A primeira função apresentada aqui será o **print**, essa função tem o objetivo de apresentar no console tudo que estiver inserido nessa função, seja texto ou o resultado de uma outra função.

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a file named 'untitled1.py' with the following content:

```

1 # -*- coding: utf-8 -*-
2 """
3 Data: 04/02/2016
4
5 Autor: Helder Guerreiro
6
7 Cabeçalho
8 """
9
10
11 print("Texto")
12
13
14
15

```

On the right, the 'Console' window shows the output of running the script:

```

Python 3.4.3 |Anaconda 2.2.0 (64-bit)| (default, Mar  6 2015, 12:06:1
Type "help", "copyright", "credits" or "license" for more information
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Fed
adores/Meus Programas/Teste.py', wdir='C:/Users/User/Documents/Helder
o a Progamação de Computadores/Meus Programas')
Texto
>>>

```

Figura 2: print modo 1

Fonte: Autoria própria

Figura 3: Resultado print modo 1

Fonte: Autoria própria

Ou até mesmo pode-se fazer de outro modo, usando uma variável e igualando essa variável ao comentário desejado, não esquecendo das aspas.

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a file named 'Teste.py' with the following content:

```

1 # -*- coding: utf-8 -*-
2 """
3 Data: 04/02/2016
4
5 Autor: Helder Guerreiro
6
7 Cabeçalho
8 """
9
10 x = "Texto"
11
12
13 print(x)
14
15
16
17

```

On the right, the 'Console' window shows the output of running the script:

```

>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Fed
adores/Meus Programas/Teste.py', wdir='C:/Users/User/Documents/Helder
o a Progamação de Computadores/Meus Programas')
Texto
>>>

```

Figura 4: print modo 2

Fonte: Autoria própria

Figura 5: Resultado print modo 2

Fonte: Autoria própria

□ Python também obedece aos comandos matemáticos que serão listados abaixo como devem ser feitos e o resultado de cada um.

```
07 print(123)
08 print(100 + 20 + 3)      # adicao
09 print(2015 - 1909)       # subtracao
10 print(6 * 7)             # multiplicacao
11 print(8 / 2)              # divisao
12 print(7 % 3)             # resto da divisao
13 print(5 ** 2)            # potenciacao
14 print(1 + 2 * 3 ** 4)    # expressao aritmetica
```

Figura 6: Comandos matemáticos

Fonte: ICOMP, UFAM

Pode se fazer de uma letra ou palavra uma variável, basta colocar qualquer letra ou palavra e igualá-la a um certo valor, e esse valor também pode ser uma função. Ao colocar uma variável igual a uma função o valor da variável será igual ao resultado da função. Para dar valores a qualquer variável se usa o sinal “=”.

The screenshot shows the Spyder Python 3.4 IDE interface. On the left, the code editor displays a script named 'Teste.py' with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Data: 04/02/2016
4
5 Autor: Helder Guerreiro
6
7 Cabeçalho
8 """
9
10 x = 5
11 variavel = 10
12
13 resultado = x + variavel
14
15 print(resultado)
16
17
18
19
```

On the right, the 'Console' window shows the output of running the script:

```
Python 3.4.3 |Anaconda 2.2.0 (64-bit)| (default, Mar 6 201
Type "help", "copyright", "credits" or "license" for more i
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Unive
dores/Meus Programas/Teste.py', wdir='C:/Users/User/Documen
o a Progamação de Computadores/Meus Programas')
15
>>>
```

Figura 7: Variáveis

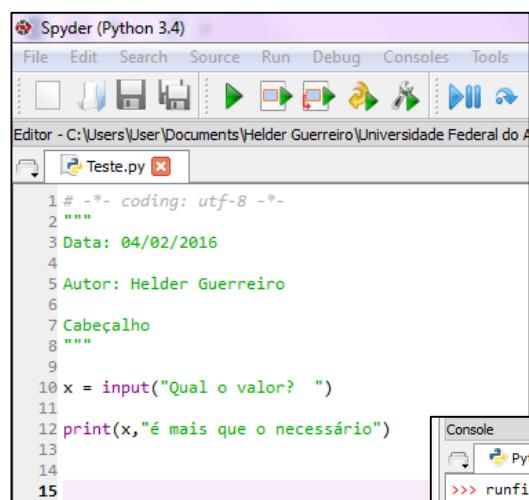
Fonte: Autoria própria

Figura 8: Resultado variáveis

Fonte: Autoria própria

Preste atenção que o Python diferencia letras maiúsculas das minúsculas, ou seja, se você for fazer uma variável e depois usá-la numa função, como o print, então não esqueça de usar o mesmo tamanho da letra que foi usada para fazer a função, isso quer dizer que “x” é diferente de “X” e isso é print(x) é diferente disso print(X). Outro caso é que as variáveis NÃO podem ter espaços, se quiser distanciar as letras use o underline “_”. Uma função não pode ter sinais de pontuação.

1.2 Primeiras Funções de entrada



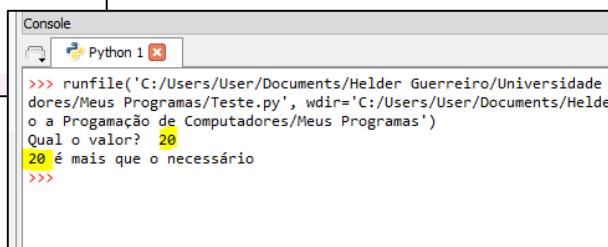
```
1 # -*- coding: utf-8 -*-
2 """
3 Data: 04/02/2016
4
5 Autor: Helder Guerreiro
6
7 Cabeçalho
8 """
9
10 x = input("Qual o valor? ")
11
12 print(x,"é mais que o necessário")
13
14
15
```

Figura 9: Função input

Fonte: Autoria própria

Três primeiras funções de entrada são muito importantes de se saber, `input`: é a função que permite a entrada de textos; `float`: é a função que permite a entrada de números reais (1.0, 2.0 ...); `int`: é a função que permite a entrada de números inteiros (1, 2, 3 ...).

O `input` é uma função que te faz uma pergunta e você tem que responder em forma de texto, mesmo que você coloque um número na resposta esse número ainda será considerado texto, ou seja, você não pode operá-lo somente representa-lo.



```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade F
dadores/Meus Programas/Teste.py', wdir='C:/Users/User/Documents/Helder
o a Programação de Computadores/Meus Programas')
Qual o valor? 20
20 é mais que o necessário
>>>
```

Figura 10: Resultado função input

Fonte: Autoria própria

O `float` é uma função que trabalha em conjunto com o `input`, isso quer dizer que o `input` pergunta e o `float` transforma, a função do `float` é transformar um valor em número real. O número real é aquele que vem seguido de casas decimais (mesmo que sejam zeros) como 2,0 e 3,0 a diferença é que no Python eles devem ser escritos e representados por pontos em vez de vírgulas. O `float` também trabalha sem o `input`, mas se torna algo meio que inviável, você pode usar o `float` para fazer operações dentro dele cujo resultado será dado em números reais, mas você pode fazer isso tranquilamente sem o uso da função. Mesmo que no `input` você coloque um número inteiro (como 1, 2, 3 ...) a resposta sempre será em reais.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View Help
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Introdução a Pro
Teste.py
1 # -*- coding: utf-8 -*-
2 """
3 Data: 04/02/2016
4
5 Autor: Helder Guerreiro
6
7 Cabeçalho
8 """
9
10 # float trabalhando com o input
11 x = float(input("Qual o valor? "))
12
13 print(x + 15, "é mais que o necessário")
14
15 # float trabalhando com operação, sem input
16 y = float(5+12)
17 print(y, "é um valor constante")
18
19 # float trabalhando com operação, sem input
20 z = float(x+y)
21 print(z, "é o resultado da variação de x")
22
23 # Operação feita sem o float
24 w = 5.0 + 12.153
25 print(w, "é um valor constante")
26
27

```

Figura II: Função float

Fonte: Autoria própria

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/1º Período/Introdução a Programação de Computadores/Meus Programas/Teste.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/1º Período/Introdução a Programação de Computadores/Meus Programas')
Qual o valor? 5
20.0 é mais que o necessário
17.0 é um valor constante
22.0 é o resultado da variação de x
17.153 é um valor constante
>>>

```

Figura I2: Resultado função float

Fonte: Autoria própria

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View Help
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Introdução a Programação de Computadores\Meus Programas\Teste.py
Teste.py
1 # -*- coding: utf-8 -*-
2 """
3 Data: 04/02/2016
4
5 Autor: Helder Guerreiro
6
7 Cabeçalho
8 """
9
10 # int trabalhando com o input
11 x = int(input("Qual o valor? "))
12
13 print(x + 15, "é mais que o necessário")
14
15 # int trabalhando com operação, sem input
16 y = int(5+12.52)
17 print(y, "é um valor constante")
18
19 # int trabalhando com operação, sem input
20 z = int(x+y)
21 print(z, "é o resultado da variação de x")
22
23 # Operação feita sem o int
24 w = 5 + 12
25 print(w, "é um valor constante")
26
27

```

Figura I3: Função int

Fonte: Autoria própria

A função int tem a mesma função da float, a única diferença é que a int só responde em números inteiros. Mesmo que você coloque números reais o resultado será inteiro, e o int pode trabalhar sem o input do mesmo jeito que o float usando operações matemáticas, só que as operações podem ser feitas sem o int, basta fazer operações com números inteiros que o resultado será inteiro.

Mesmo as funções float e int sendo de bom uso estando junto com o input, essas funções podem trabalhar em conjunto com outras funções, ou seja, funções compostas em que uma função dentro do float ou int dá um certo valor e o float transforma em número real ou o int transforma em número inteiro, logo serão mostradas outras formas de usar o float.

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federa-
dores/Meus Programas/Teste.py', wdir='C:/Users/User/Documents/Helder Guerr-
o a Progamação de Computadores/Meus Programas')
Qual o valor? 5
20 é mais que o necessário
17 é um valor constante
22 é o resultado da variação de x
17 é um valor constante
>>>

```

Figura 14: Resultado função int

Fonte: Autoria própria

1.3 Módulos (bibliotecas)

Esses módulos são as bibliotecas que o Python tem dentro de si, o Python não carrega todas as funções dentro de si então ele precisa importar algumas. A primeira a ser apresentada aqui será a math.

Você já deve ter percebido que a math tem alguma coisa haver com matemática, então você usará essa biblioteca para ter acesso ao comando de funções matemáticas do Python, se você não invocar o nome dessa biblioteca nenhuma função math irá funcionar. Para invocar qualquer biblioteca deve-se escrever: `from (módulo) import *`. A math tem as seguintes funções listadas abaixo:

exp (x)	• Calcula e^x	sin (x)	• Calcula o seno de x (em radianos)
log (x)	• Logaritmo natural de x (base e)	cos (x)	• Calcula o cosseno de x (em radianos)
log10 (x)	• Logaritmo de x na base 10	tan (x)	• Calcula a tangente de x (em radianos)
sqrt (x)	• Raiz quadrada de x	asin (x)	• Calcula o arco-seno de x
pi	• Valor da constante Pi	acos (x)	• Calcula o arco-cosseno de x
e	• Valor da constante de Euler	atan (x)	• Calcula o arco-tangente de x

Figura 15: Funções matemáticas e constantes

Fonte: ICOMP, UFAM

Figura 16: Funções Trigonométricas

Fonte: ICOMP, UFAM

Veja alguns exemplos do uso dessa biblioteca abaixo:

The screenshot shows the Spyder Python 3.4 IDE interface. On the left, the code editor window titled 'Teste.py' contains the following Python code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Data: 04/02/2016
4
5 Autor: Helder Guerreiro
6
7 Cabeçalho
8 """
9
10 from math import*
11
12 print("O valor do Pi é", pi)
13 print("O valor da constante de euler é", e)
14 print("A raiz quadrada de 2 é", sqrt(2))
15 print("O logarítmico de 10 na base e é", log(10))
16 print("O logarítmico de 10 na base 10 é", log10(10))
17 print("A exponecial de 2 é", exp(2))
18 print("Seno de 90° é", sin(pi/2))
19 print("Cosseno de 60° é", cos(pi/3))
20 print("Tangente de 60° é", tan(pi/3))
21
22
23

```

On the right, the 'Console' window titled 'Python 1' displays the output of running the script:

```

>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do
dores/Meus Programas/Teste.py', wdir='C:/Users/User/Documents/Helder Guerreiro
o a Progamação de Computadores/Meus Programas')
O valor do Pi é 3.141592653589793
O valor da constante de euler é 2.718281828459045
A raiz quadrada de 2 é 1.4142135623730951
O logarítmico de 10 na base e é 2.302585092994046
O logarítmico de 10 na base 10 é 1.0
A exponecial de 2 é 7.38905609893065
Seno de 90° é 1.0
Cosseno de 60° é 0.5000000000000001
Tangente de 60° é 1.7320508075688767
>>>

```

Figura 17: Exemplo de funções math

Fonte: Autoria própria

Figura 18: Resultado do exemplo de funções math

Fonte: Autoria própria

1.4 Questões Resolvidas

Estas questões foram desenvolvidas pela ICOMP um instituto da UFAM e certos algoritmos (scripts) apresentados aqui pertencem ao ICOMP.

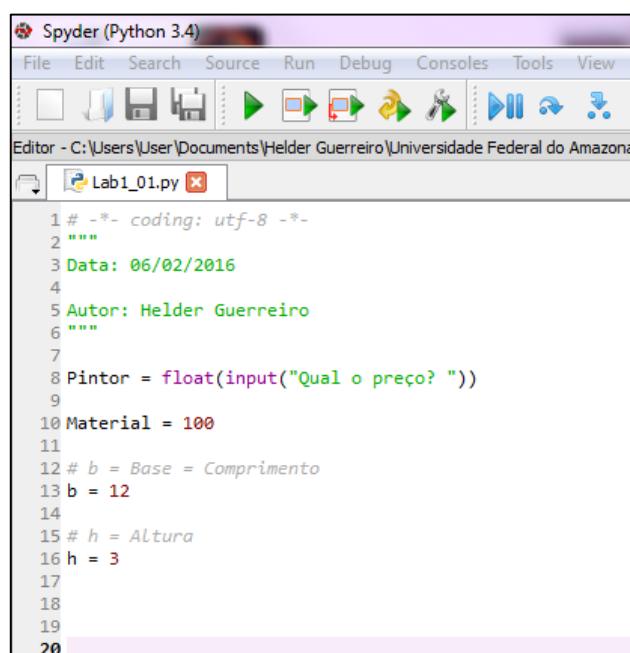
Questão 1 – Custo da pintura de um muro

Mário precisa pintar um muro, que tem 12m de comprimento e 3m de altura. O material de pintura (galão de tinta, lixa, rolo, etc.) custa R\$ 100. Cada pintor cobra um preço diferente por m^2 pelo serviço de pintura. Escreva um script Python que, dado o valor cobrado por um pintor, informe o custo total da pintura.

Primeiro passo: Identificar variáveis e constantes

Perceba que as dimensões do muro nunca vão mudar e o valor do material também, a única variável é quanto o pintor irá cobrar.

Lembre-se para valores constantes basta pegar uma variável no Python e igualá-la a um valor fixo, para valores variáveis basta pegar uma variável e igualá-la a uma função de acordo com o que se pede. No caso você deve colocar o valor variável no Python, ou seja, temos que usar uma função que coloque um número na programação. Pode-se usar `float` ou `int` como queira, o mais usado é o `float`.



The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, and View. Below the menu is a toolbar with various icons. The main area is titled 'Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas'. A file tab shows 'Lab1_01.py'. The code in the editor is:

```
1 # -*- coding: utf-8 -*-
2 """
3 Data: 06/02/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 Pintor = float(input("Qual o preço? "))
9
10 Material = 100
11
12 # b = Base = Comprimento
13 b = 12
14
15 # h = Altura
16 h = 3
17
18
19
20
```

Figura 19: Questão 1A Primeiro Passo

Fonte: Autoria própria

Segundo Passo: Identificar o que se pede

A questão pede para encontrar o valor total de acordo com o preço do metro quadrado pedido pelo pintor, perceba que temos base e altura e para encontrar a área basta multiplicá-los, e o resultado final é simplesmente o valor do metro quadrado mais o valor do material.

The screenshot shows the Spyder Python 3.4 IDE interface. The title bar says "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, and View. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\Lab1_01.py". The code in the editor is:

```
1 # -*- coding: utf-8 -*-
2 """
3 Data: 06/02/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 Pintor = float(input("Qual o preço? "))
9
10 Material = 100
11
12 # b = Base = Comprimento
13 b = 12
14
15 # h = Altura
16 h = 3
17
18 Area = b*h
19
20 Total = Area*Pintor + Material
21
22 print(Total)
23
24
```

Figura 20: Questão IA Segundo Passo

Fonte: Autoria própria

Veja o resultado:

The screenshot shows the Spyder Python 1 console. The title bar says "Console" and "Python 1". The command line shows: >>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab1_01.py', wdir='C:/'). The user then types "Qual o preço? 10" and the output is "460.0". The prompt ">>>" appears again.

Figura 21: Resultado a) questão IA

Fonte: Autoria própria

The screenshot shows the Spyder Python 1 console. The title bar says "Console" and "Python 1". The command line shows: >>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab1_01.py', wdir='C:/'). The user then types "Qual o preço? 20" and the output is "820.0". The prompt ">>>" appears again.

Figura 22: Resultado b) questão IA

Fonte: Autoria própria

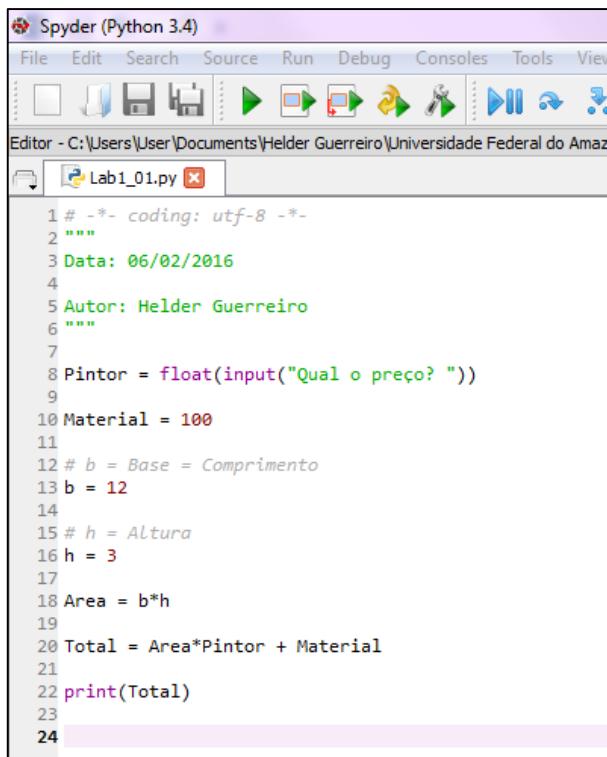
Questão 2 - Custo da pintura de um muro (reformulado)

Modifique a solução anterior de modo que o valor da altura e do comprimento do muro também sejam informados via teclado. Assuma a seguinte ordem de inserção de dados de entrada:

1. Altura do muro
2. Comprimento do muro
3. Custo do serviço de pintura por m^2

Primeiro passo: Analisar script anterior

Visualize o script anterior, e veja no que se tem de mudar para conseguir achar o que se pede.



The screenshot shows the Spyder Python 3.4 IDE interface. The title bar says "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, and View. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\Lab1_01.py". The code window contains the following Python script:

```
1 # -*- coding: utf-8 -*-
2 """
3 Data: 06/02/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 Pintor = float(input("Qual o preço? "))
9
10 Material = 100
11
12 # b = Base = Comprimento
13 b = 12
14
15 # h = Altura
16 h = 3
17
18 Area = b*h
19
20 Total = Area*Pintor + Material
21
22 print(Total)
23
24
```

Figura 23: Questão 2A Primeiro Passo

Fonte: Autoria própria

Perceba que você tem que fazer variar altura e comprimento, então para variar algum tipo de variável temos que igualar essa variável a uma função de acordo com o que se pede. Então vamos pegar as constantes e torná-las variável.

Para esse caso vamos usar, `input` para perguntar qual o valor para cada variável e `float` para transformar o valor adicionado em número real.

The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, and Help. Below the menu is a toolbar with various icons. The main area is titled 'Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal' and contains the file 'Lab1_01.py'. The code in the editor is:

```
1 # -*- coding: utf-8 -*-
2 """
3 Data: 06/02/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 Pintor = float(input("Qual o preço? "))
9
10 Material = 100
11
12 # b = Base = Comprimento
13 b = float(input("Valor do comprimento "))
14
15 # h = Altura
16 h = float(input("Valor da altura "))
17
18 Area = b*h
19
20 Total = Area*Pintor + Material
21
22 print(Total)
23
24
```

Figura 24: Questão 2A Segundo Passo

Fonte: Autoria própria

Com isso podemos testar colocando alguns dados e assim tendo um resultado final.

The screenshot shows the Python 1 console window. It displays the command `>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab1_01.py', wdir='C:/Users/Usaçao a Progamação de Computadores/Meus Programas')`. The program prompts for 'Qual o preço?' and 'Valor do comprimento'. The user inputs 5 and 10 respectively. The output shows the calculation: `350.0`.

Figura 25: Resultado a) questão 2A

Fonte: Autoria própria

The screenshot shows the Python 1 console window. It displays the command `>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab1_01.py', wdir='C:/Users/Uçao a Progamação de Computadores/Meus Programas')`. The program prompts for 'Qual o preço?' and 'Valor do comprimento'. The user inputs 10 and 5 respectively. The output shows the calculation: `600.0`.

Figura 26: Resultado b) questão 2A

Fonte: Autoria própria

Questão 3 - Caixa eletrônico

Um cliente de um banco deseja sacar uma quantia no caixa eletrônico. Este tem apenas notas de R\$50, R\$10 e R\$2 disponíveis. Escreva um programa Python que exiba quantas notas de cada tipo devem ser entregues ao cliente.

Considere que o cliente sempre insere um número par maior que zero como entrada, que o cliente tem saldo suficiente no banco, e que o caixa eletrônico tem sempre em estoque a quantidade de notas de cada tipo necessária para atender ao saque.

DICAS: Comece pelas notas de valor mais alto. Use o operador de resto da divisão (%) para determinar a quantidade de notas de valor imediatamente mais baixo.

Primeiro passo: Interpretação da questão

Vamos iniciando com calma, não se assuste logo de cara com o texto, depois de ler uma vez leia outra vez mais pausadamente e refletindo sobre o que se pede. A parte mais importante do texto da questão é somente o primeiro parágrafo, vamos entender com calma.

Todo caixa eletrônico de qualquer banco só tem certos tipos de cédulas para liberar para você, tem caixas que só funcionam com notas de 50 R\$ e 100 R\$ e tem outros que tem notas menores, então neste caso nós só vamos fazer a mesma situação que todo mundo passa ao tirar dinheiro no banco para o Python.

Se você vai tirar 300 R\$ no banco, quantas notas de 50 R\$ virão? É fácil de perceber que 6 notas de 50 R\$ dão 300 R\$, mas temos que ter uma forma matemática de achar esse valor, se você multiplica 6 por 50 o resultado é 300 então a forma de achar esse "6" é a divisão, se for dividir 300 por 50 você achará 6, e esse é o número de notas de 50 R\$.

Mas se fosse 330 R\$? Não tem como dividi-lo em notas de 50 R\$! Para isso serve a dica que foi dada no final da questão, o símbolo "%" no Python significa resto da divisão e ele funciona assim: Quando você divide 330 por 50, o resultado será um número quebrado o que não pode acontecer, por isso usaremos o símbolo de duas barras (//) para divisão, essa divisão tornará o resultado num número inteiro, em vez de resultar 6,6 o resultado será somente 6, se o resultado não for um número inteiro o script sairá errado. O resto da divisão (%) entra em ação depois da divisão, por exemplo, o resto de 330 por 50 é 30, pois 300 é divisível por 50 e 30 não, ou seja, da divisão de 330 por 50 somente 300 é divisível e o resto é 30. Esse cálculo é feito da seguinte forma no Python: $330 \% 50 = 30$.

Voltando à leitura da questão, o caixa eletrônico libera somente cédulas de 50, 10 e 2 reais. Perceba que o 30 que sobrou dos 330 pode ser dividido em 3 notas de 10 R\$, ou seja, é mesma coisa que: $30 // 10 = 3$, esse "3" é o número de notas de 10 R\$, nesse caso o seu resto com certeza é zero. Mas e se o valor fosse 334? O número de notas de 50 R\$ seria: $334 // 50 = 6$, seu resto seria $334 \% 50 = 34$; o número de notas de 10 R\$ seria: $34 // 10 = 3$, seu resto seria $34 \% 10 = 4$ (isso por que somente 30 é divisível por 10 e o 4 não); por fim temos a última nota que é a de 2 R\$, você já deve ter percebido que basta pegar: $4 // 2 = 2$, pronto acabamos aqui!

Segundo passo: Formulando o script

Você já percebeu bem como se deve fazer o script, então veja que o valor a ser tirado é uma variação, logo você deve usar uma função para variá-la, neste caso o input seguido do int (int por que estamos trabalhando com números inteiros).

Depois da variável ter sido variada, perceba que não há mais nenhuma outra variável, pois, todos os resultados saíram a partir da entrada da primeira variável. Então deve-se construir o sistema que entendemos no primeiro passo no Python, ou seja, o número de cédulas para cada caso.

A variável que você colocará o valor do caixa será primeiramente dividida (inteiramente) por 50, logo depois se acha o resto da divisão, e o valor do resto da divisão será dividido (inteiramente) por 10, logo depois se acha o resto da divisão, e o valor do resto da divisão será dividido (inteiramente) por 2, e no final não esqueça de usar a função print para exibir os resultados.

The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, and View. Below the menu is a toolbar with various icons. The main area is titled 'Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas' and contains the file 'Lab1_03.py'. The code is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 Data: 08/02/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 valor = int(input("Qual o valor do saque? "))
9
10 # Numero de notas de R$ 50
11 Notas50 = valor // 50
12
13 # Valor restante a ser sacado com notas menores que
14 Resto50 = valor % 50
15
16 # Numero de notas de R$ 10
17 Notas10 = Resto50 // 10
18
19 # Valor restante a ser sacado com notas menores que
20 Resto10 = Resto50 % 10
21
22 # Numero de notas de R$ 2
23 Notas2 = Resto10 // 2
24
25 print("Temos",Notas50,"cédulas de 50 R$")
26 print("Temos",Notas10,"cédulas de 10 R$")
27 print("Temos",Notas2,"cédulas de 2 R$")
28
29
```

Figura 27: Questão 3A Segundo Passo

Fonte: Autoria própria

Com isso podemos testar colocando alguns dados e assim tendo um resultado final.

```
Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab1_03.py', wdir='C:/Users/User/Documents/Helder Guerreiros/Meus Programas')
Qual o valor do saque? 334
Temos 6 cédulas de 50 R$
Temos 3 cédulas de 10 R$
Temos 2 cédulas de 2 R$
>>>
```

Figura 28: Resultado a) questão 3A

Fonte: Autoria própria

```
Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab1_03.py', wdir='C:/Users/User/Documents/Helder Guerreiros/Meus Programas')
Qual o valor do saque? 300
Temos 6 cédulas de 50 R$
Temos 0 cédulas de 10 R$
Temos 0 cédulas de 2 R$
>>>
```

Figura 29: Resultado b) questão 3A

Fonte: Autoria própria

Questão 4 – Ordene três números inteiros

Crie um programa que leia três inteiros a partir do teclado e os exiba em ordem crescente.

DICA 1: Aplique o processo de resolução de problemas algorítmicos.

DICA 2: Use as funções de mínimo (min) e máximo (max) para encontrar os valores menor e maior. O valor intermediário pode ser encontrado pelo cálculo da soma de todos os três valores, e em seguida subtraindo o valor mínimo e o valor máximo.

Primeiro passo: Preparando o conhecimento

Para este caso apresentar a você as funções `min` e `max`, essas funções tem por objetivo apresentar o menor valor numérico dentre uma certa quantidade de número (`min`) ou o maior valor numérico dentre uma certa quantidade de números (`max`). Para usá-los basta escrever a função e entre parênteses colocar os valores desejados. Assim: `max(a,b,c)`. Você também pode usar os valores diretamente na função, mas neste caso isso não convém.

Segundo passo: Formulando o script

Como temos três valores um deles é obrigatoriamente o maior, e outro obrigatoriamente o menor, são esses dois que serão apresentados pelo `max` e `min`, já o valor intermediário é muito simples de se achar, é a mesma coisa que você pegar um sanduiche e tirar o pão de cima e o pão de baixo e pronto. Você soma os três valores, logo em seguida subtraia o maior e o menor e sobrará o meio.

Bom, as variáveis são logicamente os números que você deve adicionar, e de novo eu digo: para variar uma variável, basta igualá-la a uma função de acordo com o que se pede. Então temos três variáveis e dessas três sairão os resultados.

```
# -*- coding: utf-8 -*-
"""
Data: 08/02/2016
Autor: Helder Guerreiro

a = int(input ("Primeiro número: "))
b = int(input ("Segundo número: "))
c = int(input ("Terceiro número: "))

minimo = min(a,b,c)
maximo = max(a,b,c)
medio = (a + b + c) - maximo - minimo

print("O menor número é",minimo)
print("O número intermediário é",medio)
print("O maior número é",maximo)
```

Figura 30: Questão 4A Segundo Passo

Fonte: Autoria própria

Com isso podemos testar colocando alguns dados e assim tendo um resultado final.

```
Console
Python 1

>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab1_04.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Meus Programas')
Primeiro número: 351
Segundo número: 631
Terceiro número: 21
O menor número é 21
O número intermediário é 351
O maior número é 631
>>>
```

Figura 31: Resultado a) questão 4A
Fonte: Autoria própria

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab1_04.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Meus Programas')
Primeiro número: 351
Segundo número: 654
Terceiro número: 684
O menor número é 351
O número intermediário é 654
O maior número é 684
>>>
```

Figura 32: Resultado b) questão 4A
Fonte: Autoria própria

Questão 5 - Área do Círculo e Volume da Esfera

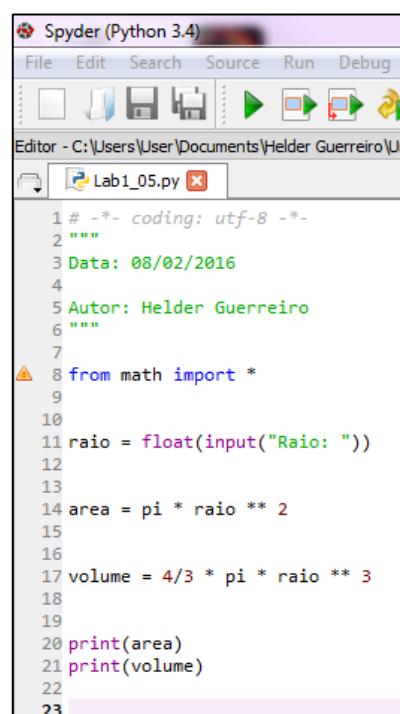
Escrever um programa que leia o valor de um raio r , inserido a partir do teclado. O programa deverá mostrar a área de um círculo com o raio r e o volume de uma esfera com raio r . Use a constante π do módulo `math` em seus cálculos.

DICA: A área de um círculo é dada pela fórmula $A = \pi r^2$. O volume de uma esfera é dada pela formula $V = \frac{4}{3} \pi r^3$.

Primeiro passo: Formulado o script

Não há muito o que discutir nessa questão, a variável será o raio e a partir dessa variável achar a área e o volume. Basta pegar a formula da área e volume e transpassar para o Python.

Como vamos usar o Pi então temos que usar a biblioteca `math` e assim inseri-lo normalmente.



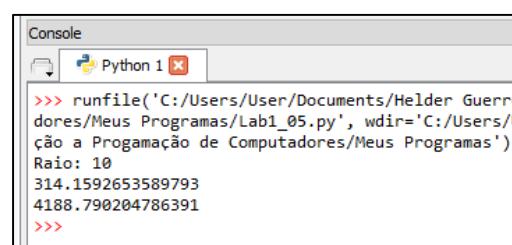
```
# -*- coding: utf-8 -*-
"""
Data: 08/02/2016
Autor: Helder Guerreiro
"""

from math import *
raio = float(input("Raio: "))
area = pi * raio ** 2
volume = 4/3 * pi * raio ** 3
print(area)
print(volume)
```

Figura 33: Questão 5A Primeiro Passo

Fonte: ICOMP, UFAM

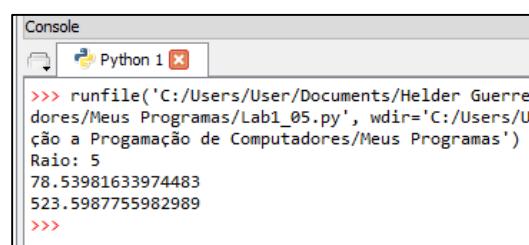
Com isso podemos testar colocando alguns dados e assim tendo um resultado final.



```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab1_05.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Meus Programas')
Raio: 10
314.1592653589793
4188.790204786391
>>>
```

Figura 34: Resultado a) questão 5A

Fonte: Autoria própria



```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab1_05.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Meus Programas')
Raio: 5
78.53981633974483
523.5987755982989
>>>
```

Figura 35: Resultado b) questão 5^a

Fonte: Autoria própria

Questão 6 - Área do Triângulo

Considere um triângulo cujos lados sejam designados por a , b e c . Considere ainda que $s = (a + b + c) / 2$. A área do triângulo pode ser calculada usando a seguinte fórmula:

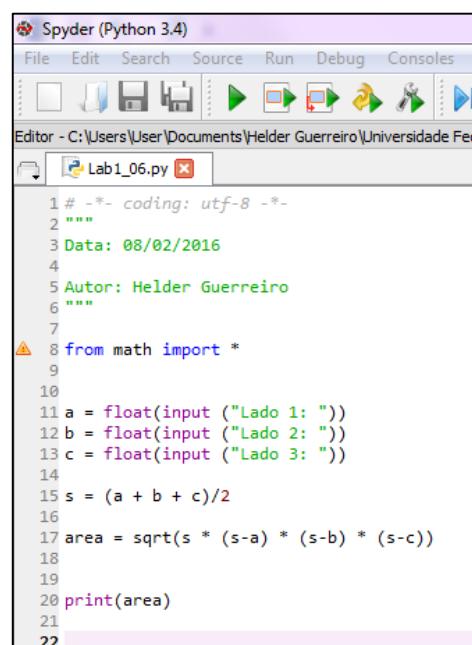
$$A = \sqrt{s(s - a)(s - b)(s - c)}$$

Escreva um programa que leia os comprimentos dos lados de um triângulo e mostre a sua área. Assuma que as entradas sempre correspondem às medidas validas de um triângulo.

Primeiro passo: Formulado o script

Essa questão não é muito diferente da anterior, a questão te dá algumas formulas e você só tem que transpassa-la para o Python de acordo com a variável que neste caso são os lados a , b e c .

Você já sabe que para variar uma variável deve-se igualar a variável a uma função de acordo como se pede e é isso que vamos fazer com os lados a , b e c . Perceba que como vamos usar a raiz quadrada precisaremos da biblioteca math a qual deve ser ativada no script.



The screenshot shows the Spyder Python 3.4 IDE interface. The title bar says "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, and Consoles. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Fed" and contains a file named "Lab1_06.py". The code in the editor is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 Data: 08/02/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 from math import *
9
10
11 a = float(input ("Lado 1: "))
12 b = float(input ("Lado 2: "))
13 c = float(input ("Lado 3: "))
14
15 s = (a + b + c)/2
16
17 area = sqrt(s * (s-a) * (s-b) * (s-c))
18
19
20 print(area)
21
22
```

Figura 36: Questão 6A Primeiro Passo

Fonte: ICOMP

Com isso podemos testar colocando alguns dados e assim tendo um resultado final.

```
Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab1_06.py', wdir='C:/Users/User/ação a Progamação de Computadores/Meus Programas')
Lado 1: 5
Lado 2: 10
Lado 3: 10
24.206145913796355
>>>
```

Figura 37: Resultado a) questão 6A

Fonte: Autoria própria

```
Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab1_06.py', wdir='C:/Users/User/ação a Progamação de Computadores/Meus Programas')
Lado 1: 5
Lado 2: 5
Lado 3: 5
10.825317547305483
>>>
```

Figura 38: Resultado b) questão 6A

Fonte: Autoria própria

Questão 7 - Distância entre dois pontos na superfície da Terra

Como o GoogleMaps calcula a distância entre dois pontos na superfície da Terra?

A superfície da Terra é curva, e a distância entre os graus de longitude varia com a latitude. Como resultado, encontrar a distância entre dois pontos na superfície da Terra é mais complicado do que simplesmente usando o teorema de Pitágoras. Sejam (t_1, g_1) e (t_2, g_2) a latitude e longitude de dois pontos 1 e 2 na superfície da Terra. A distância d entre esses pontos, na superfície da Terra, em km é dada por:

$$d = R \arccos(\sin(t_1)\sin(t_2) + \cos(t_1)\cos(t_2)\cos(g_1 - g_2))$$

Onde $R = 6371,01$ km é o raio médio da Terra.

Escreva um programa em que o usuário digite a latitude e longitude de dois pontos na Terra em graus. Seu programa deve exibir a distância entre os pontos, na superfície da Terra, em quilômetros, com duas casas decimais de precisão.

DICA: Use as funções de seno (`sin()`), cosseno (`cos()`) e arco cosseno (`acos()`) do módulo `math`. Use a função `round()`, nativa do Python, para fornecer a precisão desejada. Verifique seu código usando as coordenadas aproximadas de Manaus e de Presidente Figueiredo.

Funções trigonométricas do Python operam em radianos. Como resultado, você vai precisar converter a entrada do usuário de graus para radianos antes de calcular a distância com a fórmula discutida anteriormente. O módulo `math` contém uma função chamada `radians()`, que converte de graus em radianos.

Primeiro passo: Interpretação da questão

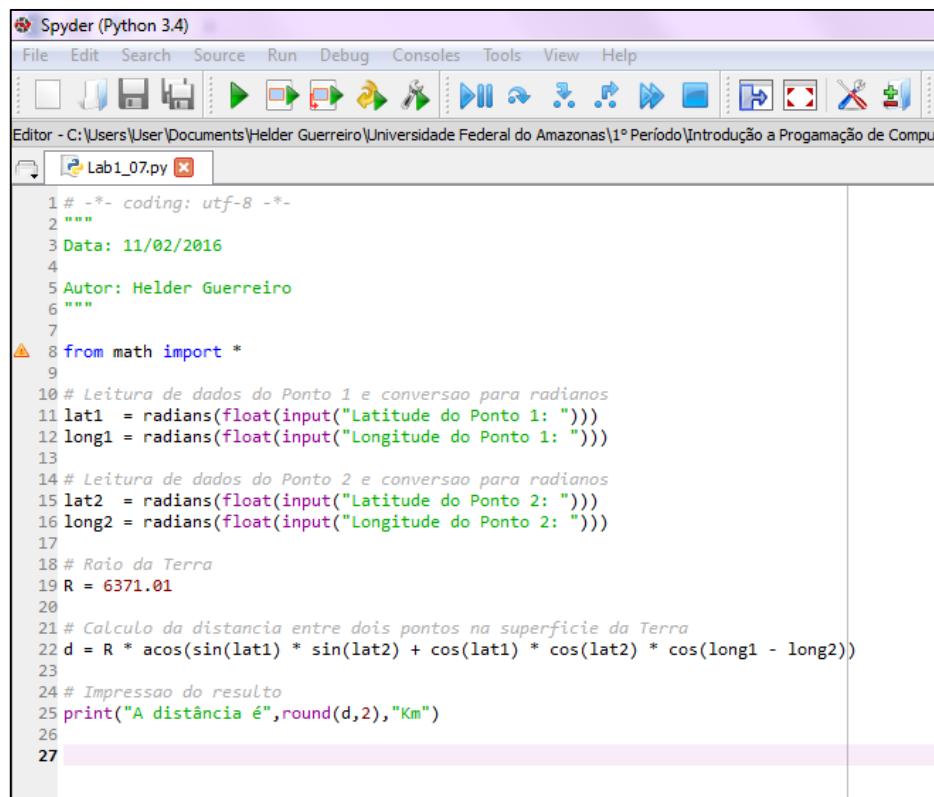
Calma! Não se assuste, tem gente que morre se ver um texto grande como o de cima numa questão. Simplificando todo o texto, temos que ter quatro entradas, pares de longitude e latitude, e o resultado final será a distância em quilômetros que será dado pela formula mostrada acima.

Esse tal `round` que foi falado acima é uma outra função do Python, essa função tem por objetivo arredondar um número real em uma certa quantidade de casas decimais que deve ser informado na própria função. A função tem a seguinte configuração `round(x, n)` onde “`x`” é o número a ser arredondado e “`n`” o número de casas decimais pretendido. A outra função a ser apresentada é o `radians`, isso por que o Python só trabalha com radianos e nunca com graus então ele pega o valor e transforma em radianos através dessa função, só assim deve-se usar as funções trigonométricas.

Segundo passo: Formulado o script

A primeira coisa a se fazer num script que envolve qualquer biblioteca é ativar a biblioteca desejada no início do script. Logo em seguida, construa as variáveis que são pares de longitude e latitude que serão variados com as seguintes

funções: `input`, para adicionar o valor desejado; `float`, para transformar o valor em número real; e `radians`, para transformar o número em radianos. Obs.: você usará todas as três funções como funções compostas, uma dentro da outra. Em seguida, não esqueça de colocar a constante do raio da terra e a formula da distância. E no final adicione a variável da distância na função `round` com duas casas decimais, se você quiser pode colocar a função `round` dentro da `print` ou, se não, pode usar outra variável para igualar a essa função `round`.



```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View Help
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Introdução a Programação de Computadores\Lab1_07.py
1 # -*- coding: utf-8 -*-
2 """
3 Data: 11/02/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 from math import *
9
10 # Leitura de dados do Ponto 1 e conversao para radianos
11 lat1 = radians(float(input("Latitude do Ponto 1: ")))
12 long1 = radians(float(input("Longitude do Ponto 1: ")))
13
14 # Leitura de dados do Ponto 2 e conversao para radianos
15 lat2 = radians(float(input("Latitude do Ponto 2: ")))
16 long2 = radians(float(input("Longitude do Ponto 2: ")))
17
18 # Raio da Terra
19 R = 6371.01
20
21 # Calculo da distancia entre dois pontos na superficie da Terra
22 d = R * acos(sin(lat1) * sin(lat2) + cos(lat1) * cos(lat2) * cos(long1 - long2))
23
24 # Impressao do resultado
25 print("A distancia é", round(d,2), "Km")
26
27

```

Figura 39: Questão 7A Primeiro Passo

Fonte: ICOMP, UFAM

Para colocar as coordenadas longitude e latitude no Python deve primeiro prestar atenção na unidade que essa coordenada está usando, se você usar o GoogleMaps ou o GoogleEarth você pode marcar um ponto no mapa e ver sua longitude e latitude, porém nesses programas é usado o sistema GMS (Graus, Minutos e Segundos) e o Python não consegue ler esse tipo de unidade então se queres saber a distância dos pontos da terra basta pegar essas coordenadas que estão em GMS e convertê-las para Graus decimais (na internet tem vários sites que fazem isso), com essa unidade se consegue facilmente fazer os cálculos pelo Python.

Pegando um ponto GoogleEarth selecionei a barreira da cidade antes da AM-010 e da BR-174 e outro ponto um pouco antes de chegar em Presidente Figueiredo, peguei suas latitudes e longitudes que foram: Manaus com latitude 2°58'18.66"S e longitude 60° 0'56.08"O; Presidente Figueiredo com latitude 2° 3'35.94"S e longitude 60° 1'11.74"O e depois de transformados para Graus decimais temos o resultado final do script.

The screenshot shows a Python console window titled "Python 1". The code run is a script named "labcod01_a07.py" located in the directory "C:/Users/User/Documents/Helder Guerreiro/Udores/Lab01/Questões". The output of the script is displayed below the code:

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Udores/Lab01/Questões/labcod01_a07.py', wdir='C:/Users/U  
trodução a Programação de Computadores/Lab01/Questões')  
Latitude do Ponto 1: 2.97185000  
Longitude do Ponto 1: 60.01557778  
Latitude do Ponto 2: 2.05998333  
Longitude do Ponto 2: 60.01992778  
A distância é 101.4 Km  
>>>
```

Figura 40: Resultado questão 7A

Fonte: Autoria própria

Questão 8 - Soma dos dígitos de um número inteiro

Desenvolva um programa que lê um número inteiro de quatro dígitos a partir do teclado e exibe a soma dos dígitos do número. Por exemplo, se o usuário digitar 3141, o programa deve exibir $3 + 1 + 4 + 1 = 9$.

DICA: Use o operador de resto da divisão (%).

Primeiro passo: Interpretação da questão

O enunciado dessa questão é menor do que o da questão 7, mas o trabalho é maior do que o da questão 7, não julgue uma questão pelo tamanho do texto!

Esta questão não é difícil, só que tem que parar um pouco para pensar, não adianta ser apressado que não vai dar em nada. Quando se tem uma questão desse tipo o primeiro recurso que você deve recorrer é a matemática, pegue uma calculadora e comece a pensar sobre o que a questão está dizendo e como fazer isso na calculadora, se você conseguir pensar na questão pela calculadora você consegue facilmente transpassar isso para o Python.

Digite o número 1235 na sua calculadora, vamos usar este número como base do nosso pensamento, agora tente achar uma forma de encontrar cada um dos dígitos a partir do próprio número...

Lembre-se que o Python tem algumas operações matemáticas especiais que não temos em nossa calculadora. Pensa comigo, este número está dividido em unidade, dezena, centena e milhar, então 5 é a unidade, 3 a dezena, 2 a centena e 1 o milhar. Uma forma de colocar os algarismos de um número em destaque é quando existe vírgula; veja bem, se você tem um número 1.6546541651 qual é o algarismo mais importante? É claro que é o 1 né, então o mesmo podemos fazer com esses 4 dígitos, se você colocar cada um dos algarismos em destaque temos: 1,235 destacando o milhar; 12,35 destacando a centena; 123,5 destacando a dezena; 1235 destacando a unidade.

Aí você me pergunta....

- Eu até entendo que no 1,235 o milhar é destacado, mas o que os outros tem haver? 12,35 não tem sentido dizer que é centena!

Aí eu te respondo:

- Calma, isso aqui não tem caráter analítico e sim interpretativo! Ou seja, temos que interpretar o comportamento da vírgula, o último número antes dela é o destacado.

O que isso tem haver com o que estamos procurando? Calma, vamos devagar que a gente chega lá! Perceba que para conseguir esses números com vírgulas basta dividir o número com a unidade que se deseja, a milhar faça divisão por 1000, para a centena divisão por 100 e para dezena divisão por 10. Eu sei, ainda não conseguimos o que realmente queremos, como podemos tirar o 1, 2, 3 e o 5 daí? Bem, você se lembra da dica que apareceu lá no enunciado da questão? Preste muita atenção nelas, elas vão te ajudar demais!

Vamos relembrar o que significa resto de divisão (%), esse operador do Python tem por função resultar somente aquilo que o denominador não consegue dividir em inteiros, ou seja, $153 \% 10 = 3$, sacou? 150 é claramente divisível por 10 mas o 3 não.

Então voltando ao nosso número 1235, se você usar a divisão inteira (`//`) todos os números após a vírgula desapareceram e aparecerá somente o número da frente, ou seja, na divisão `1235 // 1000 = 1` em vez de `1235 / 1000 = 1,235`; agora usando a divisão de inteiros nas outras unidades temos: para a centena `1235 // 100 = 12` e para a dezena `1235 // 10 = 123`. Agora sim podemos usar o resto da divisão ao nosso favor! Veja, você não precisa fazer mais nada na unidade de milhar, mas no resto do pessoal precisamos fazer mais alguma coisa, pegue o resultado da centena `12` e veja que `12 % 10 = 2`, pegue o resultado da dezena `123` e veja que `123 % 10 = 3`. Percebeu agora? Acabamos de encontrar os algarismos do número a partir do próprio número! Ora, em `12 % 10` se você dividir 12 por 10 terá 1,2 mas se tirar o 2 e deixar somente o 10 você terá uma divisão inteira igual a 1; o mesmo acontece em `123 % 10` se você divide 123 por 10 terá 12,3 mas se tirar o 3 e deixar o 100 terá uma divisão inteira igual a 12.

Segundo passo: Formulado o script

Agora vamos transpassar todo o conhecimento adquirido na análise da questão para o script do Python. Perceba que não há motivo algum para ativar alguma biblioteca então podemos começar o script logo direto. A primeira coisa a se fazer é construir a variável que no caso será variada pela função `input`, para dar entrada ao valor, e `int`, para tornar o valor em número inteiro; esse número será primeiramente dividido inteiramente por `1000` e depois por `100` e por `10`, mas lembre-se do operador resto, a unidade de milhar não precisa então colocar o operador na centena, ou seja, ao lado da divisão inteira, na dezena e na unidade; para a unidade não precisa dividir nada basta colocar o operador do resto. Chegando no fim você tem que fazer uma formulazinha em que consistirá na soma dos quatro algarismos e por fim faça o `print` da soma. Se você gosta de deixar o seu script um pouco mais fácil de se entender e mais bonito, coloque textos dentro da função `print` usando as aspas (`""`) e também você pode apresentar quais valores estão sendo somados basta separar o texto e o valor por vírgulas.

The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. Below the menu is a toolbar with various icons. The main area is titled 'Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Per' and contains the Python code for question 8A. The code reads a 4-digit number from the user, extracts each digit, calculates their sum, and prints the result.

```
1 # -*- coding: utf-8 -*-
2 """
3 Data: 11/02/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 # Leitura de dados
9 num = int(input("Digite um numero de 4 digitos: "))
10
11 # Digito do milhar
12 a = num // 1000
13
14 # Digito da centena
15 b = (num // 100) % 10
16
17 # Digito da dezena
18 c = (num // 10) % 10
19
20 # Digito da unidade
21 d = num % 10
22
23 soma = a + b + c + d
24
25 print("Temos",a,"+",b,"+",c,"+",d,"=",soma)
26
27
28
29
```

Figura 41: Questão 8A Segundo Passo

Fonte: ICOMP, UFAM

Com isso podemos testar colocando alguns dados e assim tendo um resultado final.

The screenshot shows the Python 1 console window. It runs the code from Figura 41 and displays the output. The user inputs '1235' and the program outputs 'Temos 1 + 2 + 3 + 5 = 11'.

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab1_08.py', wdir='C:/Users/User/Documents/Helder Guerreiros/Meus Programas')
Digite um numero de 4 digitos: 1235
Temos 1 + 2 + 3 + 5 = 11
>>>
```

Figura 42: Resultado a) questão 8A

Fonte: Autoria própria

The screenshot shows the Python 1 console window. It runs the code from Figura 41 and displays the output. The user inputs '2414' and the program outputs 'Temos 2 + 4 + 1 + 4 = 11'.

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab1_08.py', wdir='C:/Users/User/Documents/Helder Guerreiros/Meus Programas')
Digite um numero de 4 digitos: 2414
Temos 2 + 4 + 1 + 4 = 11
>>>
```

Figura 43: Resultado b) questão 8A

Fonte: Autoria própria

1.5 Avaliações

Aqui teremos algumas avaliações para entender mais um pouco o assunto estudado neste primeiro capítulo. Aconselho você tentar fazer sozinho antes de ler a resolução.

Avaliação A - A1

Avaliação 1a - (A1) - Variáveis e Estrutura Sequencial de Programação

Informações

Questões

Notas

Questão 1

Uma lanchonete cobra 15% de gorjeta pelo serviço de atendimento sobre o valor consumido. Escreva um script Python que leia o valor consumido por um cliente e calcule o total a ser pago.

Dicas:

1. Teste o script no Spyder antes de submetê-lo ao Code Bench.
2. Verifique se você está submetendo o código correto à questão correspondente desta Avaliação Parcial.
3. Os valores em reais devem ser arredondados em **duas casas** decimais.
4. Por exemplo, para uma entrada 55, a saída deve ser 63.25.

Vamos ser mais diretos nas resoluções envolvendo avaliações, pois você já viu exemplos o suficiente para estar ciente de tudo que vamos fazer nas avaliações.

Nesta questão pede-se simplesmente para que se saiba o total gasto em uma lanchonete sabendo que a gorjeta é 15 % do valor gasto.

A variável é o valor de entrada que será variado com `input` e `float`, logo em seguida esse valor será multiplicado por 0,15 e no final tudo será somado. Use uma variável para iguala ao valor da gorjeta e outra variável para a soma do dois valores, não esquecendo de colocar a função `round` no final com 2 casas decimais por que a questão pede arredondamento.

```
# -*- coding: utf-8 -*-
"""
Data: 11/02/2016
"""
Autor: Helder Guerreiro

valor = float(input("Qual o valor de consumo? "))
gorjeta = valor*0.15
total = valor + gorjeta
print(round(total),2)
```

Figura 44: Resposta Avaliação A IA
Fonte: Autoria própria

Avaliação B - A1

Avaliação 1b (A1) - Variáveis e Estrutura Sequencial de Programação

Informações

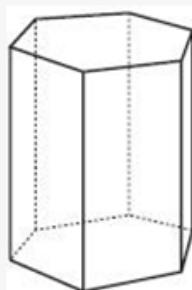
Questões

Notas

Questão 1

Zulmira vende um preparado à base de guaraná em um recipiente de formato de um prisma de base hexagonal (regular). Escreva um script Python que leia o preço de um litro do preparado, a altura do recipiente e a aresta da base hexagonal, e informe na saída o preço do líquido contido no recipiente.

A área de um hexágono regular de lado a é dada pela fórmula $\frac{3a^2}{2}\sqrt{3}$.



Dicas:

1. Teste o script no Spyder antes de submetê-lo ao Code Bench.
2. Verifique se você está submetendo o código correto à questão correspondente desta Avaliação Parcial.
3. Os valores em reais devem ser arredondados em **duas casas** decimais.
4. A leitura das grandezas no script deve ser a mesma do enunciado.
5. Por exemplo, para uma entrada preco = 5.5, altura = 10 e lado = 7, a saída deve ser 7001.82.

Perceba que o objetivo é saber o preço final do guaraná. Como o preço que será levado em consideração é o preço do litro então claramente você deve achar o volume do recipiente, se você tem pelo menos uma noção de geometria o volume de um prisma é base vezes altura.

As três variáveis a serem colocadas são, preço do litro, altura e aresta, que serão variadas com `input` e `float`, logo depois ache a área e o volume, a formula da área foi dada (perceba que você deve usar o módulo `math` por causa da raiz quadrada) e o volume pode ser encontrado multiplicando a área pela altura. Pelo volume se encontra o preço total multiplicando o volume pelo preço do litro.

The screenshot shows the Spyder Python 3.4 IDE interface. The title bar reads "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, Run, and Debug. The main area is the "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Introdução à Programação com Python\AV2.py" window. It displays the following Python code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Data: 11/02/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 from math import *
9
10 valor = float(input("Qual o preço do litro? "))
11 altura = float(input("Qual a altura do recipiente? "))
12 aresta = float(input("Qual o valor da aresta da base? "))
13
14 area = (3 * (aresta**2)/2)*sqrt(3)
15 volume = area*altura
16 resultado = volume*valor
17
18 print(round(resultado),2)
19
20
21
```

Figura 45: Resposta Avaliação A IB

Fonte: Autoria própria

2. Decisões da Vida – As Condicionais

2.1 Comando Condicional

Agora vou apresentar a você dois comandos que trabalham em conjunto, os quais vão nos ajudar na tomada de decisões de acordo com a situação do script.

Existem scripts que enquanto você está construindo você percebe que o ele não pode trabalhar diretamente somente num caminho, digamos que você esteja fazendo um script para calcular as raízes de uma equação quadrática, você constrói as variáveis a , b e c tudo tranquilo, depois constrói o delta ($\Delta = b^2 - 4ac$) seguido da busca pelas raízes x' e x'' ($x = \frac{-b \pm \sqrt{\Delta}}{2a}$). Só que tem um porem.... Será que você lembra de o que acontece com as raízes quando o delta é negativo? Isso mesmo, não tem raízes! E se você colocar mesmo assim, quando o Python for calcular a raiz negativa do delta ocorrerá um erro.

Então apresentarei oficialmente a você o `if` e o `else`, esses são comandos condicionais que permitirá o Python tomar decisões diferentes para cada situação, é como se fosse um script dentro de outro, você cria seu script normal até o ponto que você precisa colocar uma condição no script então após usar o `if` você terá outro caminho abaixo dele que será um script alternativo para o caso que aquela condição venha acontecer. O `else` é o comando que irá dar outro caminho para o caso em que a condição feita pelo `if` não venha ser obedecida então o Python pula a leitura do script para o `else`.

Me deixe explicar melhor, o `if` pode ser entendido como "Se isso acontecer então..." e o `else` é entendido como "Se isso não acontecer então...". É como se o seu script fosse um único caminho, mas chega uma hora que você deve decidir para onde deve ir para direita ou para esquerda, a direita é rua que passa pela lanchonete e a esquerda vai direto pra casa, "Se você estiver com fome então..." você pegará o caminho da direita e para na lanchonete para comer alguma coisa, "Se você não estiver com fome então..." você pega a esquerda e vai direto para casa descansar.

Entenda um script do Python como se fosse uma conversa, os comandos e funções dentro do script são o andamento da conversa e como o assunto está se desenvolvendo. Vamos a um exemplo maneiro abaixo:

Era uma vez...

Dois rapazes entram numa biblioteca de matemática, e o primeiro pergunta:

- Me diga os coeficientes a , b e c de uma equação quadrática.

E o segundo responde:

- 1,2 e 1

E o primeiro disse:

- Vou calcular o delta!

E o primeiro acrescenta:

- Se o delta der negativo então não temos raízes reais.

E o primeiro falou ainda mais:

- E se não for negativo teremos de calcular as raízes.

Então o primeiro respondeu o seguinte:

- Esses coeficientes dão um delta positivo, então calculando as raízes temos: $x' = -1$ e $x'' = -1$.

Fim!

Ta certo, esse tipo de conversa não é tão normal de se ouvir por aí... Mas do mesmo jeito é uma conversa, todo esse papo é a leitura do script abaixo.

The screenshot shows the Spyder Python 3.4 IDE interface. The title bar says "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, and Tools. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do A..." and contains a file named "Estudo.py". The code in the file is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb  4 13:52:04 2016
4
5 @author: Helder Guerreiro
6 """
7
8 from math import *
9
10 a = float(input("Coeficiente a: "))
11 b = float(input("Coeficiente b: "))
12 c = float(input("Coeficiente c: "))
13
14 delta = (b**2) - 4*a*c
15
16 if delta < 0:
17     print("Não há raízes reais")
18 else:
19     x1 = (-b + sqrt(delta))/2
20     x2 = (-b - sqrt(delta))/2
21     print(x1,x2)
22
23
```

Figura 46: Exemplo de condicional

Fonte: Autoria própria

The screenshot shows the Python 1 console window. The title bar says "Console" and "Python 1". The command line shows the execution of the script: ">>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Estudo.py', wdir='C:/Users/... a Programação de Computadores/Meus Programas')". The output shows the user input for coefficients and the resulting output: "Coeficiente a: 1", "Coeficiente b: 2", "Coeficiente c: 1", "-1.0 -1.0", and ">>>".

Figura 47: Resposta do exemplo de condicional

Fonte: Autoria própria

Agora vamos entender melhor como montar o sistema condicional. Perceba que no script feito acima o `if` e o `else` funcionam como uma espécie de mini script que funciona dentro do script maior. Quando você digita "if" lá no script ele fica azul, dê um espaço e digite a condição que você quer que o script obedeça terminando tudo com o sinal de dois pontos, logo em seguida tecle ENTER e será iniciado o novo mini script para você montar, esse mini script será espaçado à direta

abaixo da condição que você colocou, se não for espaçado automaticamente basta teclar TAB e pronto; terminando a sua condição aperte ENTER de novo, mas dessa vez sem espaço digite “else” e termine com dois pontos e aperte ENTER mais uma vez, será iniciado outro mini script para o caso da condição do if não ser respeitada, construa o seu script normalmente e pronto.

Lhe aviso de um seguinte, depois de feito tudo isso o seu script pode continuar seguindo normalmente e ainda ter mais outras condições lá para baixo. Você verá como fazer scripts como esses mais abaixo em outros assuntos.

Devo avisá-lo também que se você errar o posicionamento correto das linhas do script você estará cometendo um erro de identação, o script não roda se não estiver tudo corretamente alinhado.

2.2 Operadores Condicionais

Para escrever condições no if é necessário ter em mãos operadores para designar que tipo de condição é essa. No exemplo que foi dado acima foi-se usado “ $\delta < 0$ ”, o sinal de menor que é um operador condicional. Veja a lista deles abaixo.

Operador	Operação	Exemplos
$==$	Igual a	$3 == 3$ $20 == 18$
$>$	Maior que	$5 > 4$ $10 > 11$
$<$	Menor que	$3 < 6$ $9 < 7$
\geq	Maior ou igual a	$5 \geq 3$ $4 \geq 4$
\leq	Menor ou igual a	$3 \leq 5$ $7 \leq 7$
$!=$	Diferente de	$8 != 9$ $2 != 2$

Figura 48: Operadores condicionais

Fonte: ICOMP, UFAM

Perceba a diferença entre “=” e “==”, isso por que o Python usa o sinal de igualdade para dar valor a alguma variável, mas se o objetivo é usar igualdade matemática deve-se usar uma igualdade dupla para as expressões matemáticas.

Cada um desses operadores é como se fosse uma forma de comunicação dentro do Python vou dizer o real significado deles dentro do sistema: “==” quer dizer “é exatamente”; “>” quer dizer “está acima de”; “<” quer dizer “está abaixo de”; “ \geq ” quer dizer “é a partir de”; “ \leq ” quer dizer “é até”; “!=” quer dizer “não pode ser”.

Existem dois operadores condicionais que atuam nos comandos condicionais, eles são `and` e `or`, eles significam a mesma coisa que a própria palavra: `and` significa "e"; `or` significa "ou". São usados quando os comandos condicionais precisam estabelecer mais de uma condição. Por exemplo temos: para uma soma dar sempre números pares positivos, o número somado sempre deverá ser par `e` positivo.

2.3 Questões Resolvidas

Estas questões foram desenvolvidas pela ICOMP um instituto da UFAM e certos algoritmos (scripts) apresentados aqui pertencem ao ICOMP.

Questão 1 – Par ou ímpar?

Elabore um programa que leia um número inteiro e exiba na tela se ele é par ou ímpar.

Primeiro passo: Interpretação da questão

Bom como sempre vamos em busca de como fazer isso. Lembre-se o primeiro recurso que você deve recorrer é a matemática!

Para saber se um número é par ou ímpar nós temos o método mais simples que é contando de um por um que é par e quem é ímpar...

1 (ímpar), 2 (par), 3 (ímpar), 4 (par), 5 (ímpar), 6 (par), 7 (ímpar), 8 (par), 9 (ímpar), 10 (par)...

Então, qual é a característica em comum dos números pares? Bom todos eles são divisíveis por 2 né?! Ta bom bora lá, se você divide 2, 4, 6, 8, 10 ... pelo número 2 você terá resultados inteiros ou seja: $2/2 = 1$; $4/2 = 2$; $6/2 = 3$; $8/2 = 4$ e $10/2 = 5$ são todos resultados inteiros, mas se você divide 1, 3, 5, 7, 9 ... pelo número 2 você terá resultados de números quebrados, daí você pode diferenciar.

Vamos usar as ferramentas que temos do primeiro capítulo, divisão não ajuda muito por que o resultado será diferente dependendo do número inserido, mas existe outra forma de usar a divisão ao nosso favor, usando o resto! Lembra do resto de divisão (%)?! Ele pode parecer chato, mas nunca esqueça da grande utilidade dele.

O resto da divisão dos números pares por 2 é zero! Perceba nos resultados inteiros que a divisão dá, se a resposta é um número inteiro então o resto é zero.

Segundo passo: Formulado o script

Bom primeiramente devemos ter a entrada do número desejado, ou seja, é a nossa variável que deverá ser variada pelas funções `input` (para dar entrada ao valor) e `int` (para transformar em número inteiro) isso por que só estamos trabalhando com inteiros, agora que o número será inserido deveremos verificar se ele é par ou ímpar, para facilitar vamos usar o par como referência, se o número não for par obviamente ele será ímpar.

A condicional começa a existir a partir o "if", então digite o if e dê espaço e digite a condição que é seguinte: o número inserido deve ter resto de divisão zero; colocando os dois pontos e dando ENTER você irá montar um mini script dizendo o que acontece se o número for par, claramente devemos mostrar o resultado através do `print` para dizer se ele é par ou ímpar; abaixo do if você está escrevendo o que acontece se ele for par, então você deve usar a função `print` para mostrar que o número é par.

Então depois da condição ter sido feita você fará a exceção que é caso o número não seja par, ou seja, aperte ENTER e digite “else” no início da linha terminando com dois pontos e dando ENTER de novo; no else você monta a exceção que no caso é dizer que o número é ímpar e deverá ser feito da mesma forma que o par, ou seja, usando a função print.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal
Lab02_01.py

1 # -*- coding: utf-8 -*-
2 """
3 12/02/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 num = int(input("Digite um numero: "))
10
11 if (num % 2 == 0):
12     print("Par")
13 else:
14     print("Ímpar")
15
16

```

Figura 49: Questão 1B Segundo Passo

Fonte: ICOMP, UFAM

Com isso podemos testar colocando alguns dados e assim tendo um resultado final.

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab02_01.py', wdir='C:/Users/User/Documentação a Progamação de Computadores/Meus Programas')
Digite um numero: 2
Par
>>>

```

Figura 50: Resultado a) questão 1B

Fonte: Autoria própria

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab02_01.py', wdir='C:/Users/User/Documentação a Progamação de Computadores/Meus Programas')
Digite um numero: 135
Ímpar
>>>

```

Figura 51: Resultado b) questão 1B

Fonte: Autoria própria

Questão 2 - De volta ao problema do caixa eletrônico

Um cliente de um banco deseja sacar uma quantia no caixa eletrônico. Este tem apenas notas de R\$50, R\$10 e R\$2 disponíveis. Escreva um programa Python que exiba quantas notas de cada tipo devem ser entregues ao cliente.

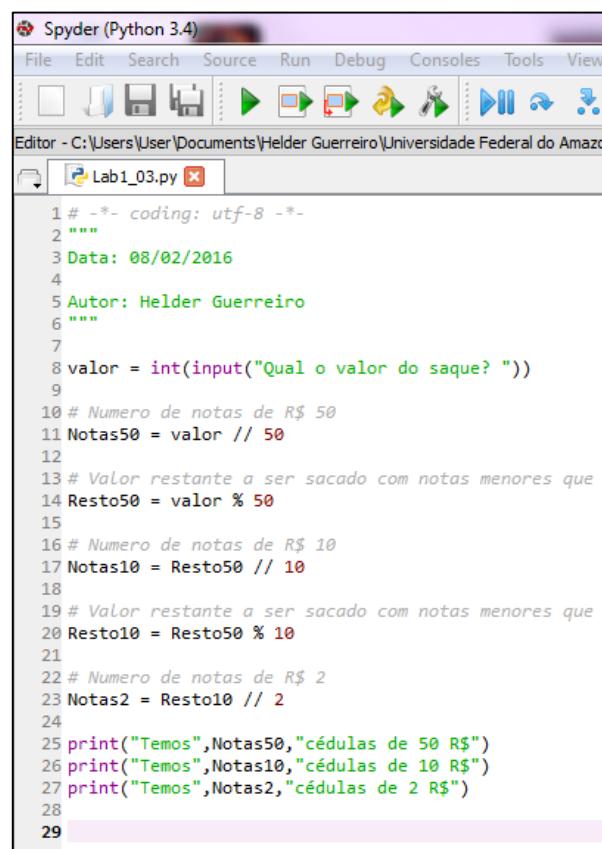
Desta vez, considere que o cliente pode inserir – intencionalmente ou não – um valor inválido, tal como um número negativo ou um número ímpar. Seu programa deve exibir a mensagem “valor inválido” nessas situações.

Mantenha as demais premissas de que o cliente tem saldo suficiente no banco, e que o caixa eletrônico tem sempre em estoque a quantidade de notas de cada tipo necessária para atender ao saque.

DICA: Adapte o código que você produziu na Questão 3 do Laboratório 01 para a estrutura de if-else da Questão anterior.

Primeiro passo: Interpretação da questão

Bom essa questão não tem tanta dificuldade assim não, vamos dar uma olhada no script da questão do banco lá do primeiro capítulo:



The screenshot shows the Spyder Python 3.4 IDE interface. The title bar says "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, and View. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas" and contains the file "Lab1_03.py". The code is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 Data: 08/02/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 valor = int(input("Qual o valor do saque? "))
9
10 # Número de notas de R$ 50
11 Notas50 = valor // 50
12
13 # Valor restante a ser sacado com notas menores que
14 Resto50 = valor % 50
15
16 # Número de notas de R$ 10
17 Notas10 = Resto50 // 10
18
19 # Valor restante a ser sacado com notas menores que
20 Resto10 = Resto50 % 10
21
22 # Número de notas de R$ 2
23 Notas2 = Resto10 // 2
24
25 print("Temos",Notas50,"cédulas de 50 R$")
26 print("Temos",Notas10,"cédulas de 10 R$")
27 print("Temos",Notas2,"cédulas de 2 R$")
28
29
```

Figura 52: Questão 2B Primeiro Passo

Fonte: ICOMP, UFAM

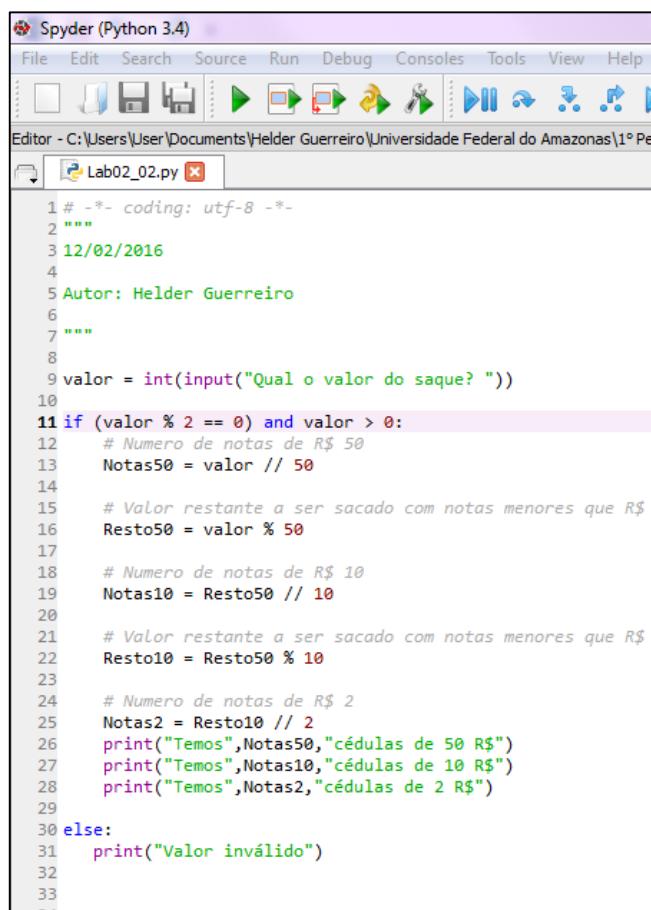
A única diferença desse script para o que nós queremos é só o fato de colocar uma condição para caso o usuário digitar um número inválido.

Tenha em mente que colocar um comando condicional é abrir outro caminho no script, olhando para o script acima temos somente um caminho e esse é o caminho do valor válido, isso quer dizer que um caminho já está pronto para você só falta o outro!

Veja bem, olhando o script anterior perceba que o primeiro caminho é pelo `if` e o segundo é pelo `else`, então a metodologia a ser usada aqui é a mesma.

Segundo passo: Formulado o script

Todo esse caminho feito abaixo da variável "valor" pertence ao primeiro caminho, ou seja, `if`. Não esqueça da indentação, isto é, colocar esse caminho espaçado para a direita abaixo do `if`. A condição do `if` é que o número não seja ímpar e nem negativo e para isso iremos usar o "`and`" por que a condição é ser um "e" outro. No final coloca-se o `else` no início da linha e dando ENTER coloque a exceção que no caso é para quando o número for ímpar ou negativo, coloque `print` com a mensagem "Valor inválido".



The screenshot shows the Spyder Python 3.4 IDE interface. The title bar says "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Pe" and shows the file "Lab02_02.py". The code is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 12/02/2016
4 Autor: Helder Guerreiro
5 """
6
7 valor = int(input("Qual o valor do saque? "))
8
9 if (valor % 2 == 0) and valor > 0:
10     # Número de notas de R$ 50
11     Notas50 = valor // 50
12
13     # Valor restante a ser sacado com notas menores que R$
14     Resto50 = valor % 50
15
16     # Número de notas de R$ 10
17     Notas10 = Resto50 // 10
18
19     # Valor restante a ser sacado com notas menores que R$
20     Resto10 = Resto50 % 10
21
22     # Número de notas de R$ 2
23     Notas2 = Resto10 // 2
24     print("Temos",Notas50,"cédulas de 50 R$")
25     print("Temos",Notas10,"cédulas de 10 R$")
26     print("Temos",Notas2,"cédulas de 2 R$")
27
28 else:
29     print("Valor inválido")
30
31
32
33
34
```

Figura 53: Questão 2B Segundo Passo

Fonte: ICOMP, UFAM

Com isso podemos testar colocando alguns dados e assim tendo um resultado final.

A screenshot of a Python 1 console window. The title bar says "Console" and "Python 1". The window contains the following text:
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Unidades/Lab02/Questões/labcod02_a02.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Unidades/Lab02/Questões')
Qual o valor do saque? 552
Temos 11 cédulas de 50 R\$
Temos 0 cédulas de 10 R\$
Temos 1 cédulas de 2 R\$
>>>

Figura 54: Resultado a) questão 2B

Fonte: Autoria própria

A screenshot of a Python 1 console window. The title bar says "Console" and "Python 1". The window contains the following text:
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Unidades/Lab02/Questões/labcod02_a02.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Unidades/Lab02/Questões')
Qual o valor do saque? 351
Valor inválido
>>>

Figura 55: Resultado b) questão 2B

Fonte: Autoria própria

Questão 3 - Ponto e reta

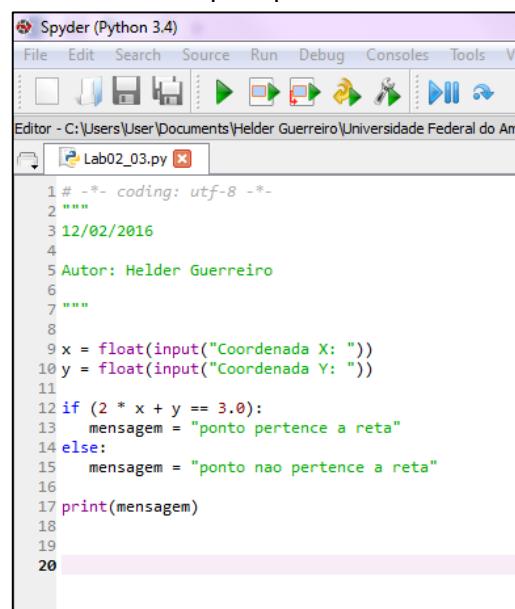
Faça um algoritmo para indicar se um ponto $P(x,y)$ do plano cartesiano pertence à reta $2x + y = 3$.

Primeiro passo: Interpretação da questão

Acho que você deve saber que para um ponto pertencer a uma reta a soma das suas variáveis deve bater com a igualdade, ou seja, o ponto que pertence a essa reta é o ponto em que quando substituído pelas variáveis resulta no número 3.

Segundo passo: Formulado o script

Simplesmente a primeira coisa a se fazer é variar o x e y com as funções `input` e `float`, esses dois devem variar para que a condição seja feita e a condição é que essa função, quando substituída os valores que foram dados entrada, resulte no número 3. O primeiro caminho é dizer que este ponto está na reta, é só isso que a questão pede e o segundo caminho é dizer que o ponto não está na reta.

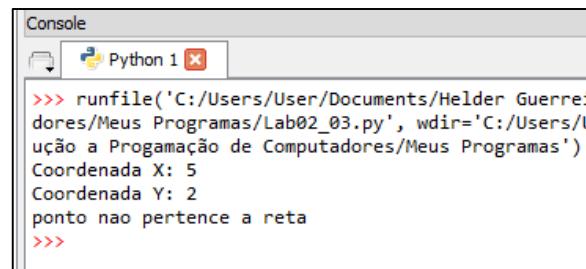


```
Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools V
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do A
Lab02_03.py
1 # -*- coding: utf-8 -*-
2 """
3 12/02/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 x = float(input("Coordenada X: "))
10 y = float(input("Coordenada Y: "))
11
12 if (2 * x + y == 3.0):
13     mensagem = "ponto pertence a reta"
14 else:
15     mensagem = "ponto nao pertence a reta"
16
17 print(mensagem)
18
19
20
```

Figura 56: Questão 3B Segundo Passo

Fonte: ICOMP, UFAM

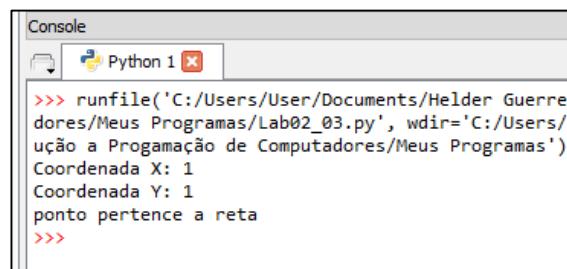
Com isso podemos testar colocando alguns dados e assim tendo um resultado final.



```
Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab02_03.py', wdir='C:/Users/User/Documents/Helder Guerreiros/Meus Programas')
Coordenada X: 5
Coordenada Y: 2
ponto nao pertence a reta
>>>
```

Figura 57: Resultado a) questão 3B

Fonte: Autoria própria



```
Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab02_03.py', wdir='C:/Users/User/Documents/Helder Guerreiros/Meus Programas')
Coordenada X: 1
Coordenada Y: 1
ponto pertence a reta
>>>
```

Figura 58: Resultado b) questão 3B

Fonte: Autoria própria

Questão 4 - Desconto

Para atrair mais clientes, uma loja de roupas oferece um desconto de 5% em compras de R\$200,00 ou mais. Escreva um programa que lê o preço sem desconto de uma compra e imprime o valor a ser pago pelo cliente.

DICA: Como o resultado deve ser dado em reais, então deve ter duas casas decimais de precisão.

Primeiro passo: Interpretação da questão

Não tem muito mistério aqui não, é só você fazer o Python ler uma entrada de valor e "Se o valor for maior ou igual a 200,00 R\$ então..." terá de ter um desconto de 5 %, "Se o valor não for maior ou igual a 200,00 R\$ então..." não tem desconto.

Segundo passo: Formulado o script

A sua variável será o valor gasto, que será variado pelo `input` e `float` (por causa dos centavos). A condição é ser maior ou igual a 200,00 para que haja um desconto e caso a condição não seja obedecida não terá desconto.

The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, and Tools. Below the menu is a toolbar with various icons. The main area is titled 'Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal' and contains the file 'Lab02_04.py'. The code is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 12/02/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9
10 preco = float(input("Valor da compra: "))
11
12 if (preco >= 200.00):
13     preco = preco * 0.95
14
15 print(round(preco,2))
16
17
18 |
```

Figura 59: Questão 4B Segundo Passo

Fonte: ICOMP, UFAM

Com isso podemos testar colocando alguns dados e assim tendo um resultado final.

The screenshot shows the Spyder Python 1 console. The command `>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab02_04.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Meus Programas')` was run. The user then typed `Valor da compra: 50`. The output was `50.0`.

Figura 60: Resultado a) questão 4B

Fonte: Autoria própria

The screenshot shows the Spyder Python 1 console. The command `>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab02_04.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Meus Programas')` was run. The user then typed `Valor da compra: 250`. The output was `237.5`.

Figura 61: Resultado b) questão 4B

Fonte: Autoria própria

Questão 5 - Angry Birds

Considere o cenário do jogo Angry Birds (Figura 62), onde um pássaro é lançado com uma velocidade inicial v_0 a partir do estilingue, cujo elástico faz um ângulo α com o solo. Considere ainda que o pássaro e o porco-alvo estão na mesma altura em relação ao solo. O alcance máximo (R) do pássaro horizontalmente é dado pela seguinte equação:

$$R = \frac{|v_0|^2 \sin 2\alpha}{g}$$

Onde $|v_0|$ é o valor em módulo da velocidade inicial (sem considerar os componentes vertical ou horizontal) e $g = 9,8 \text{ m/s}^2$ é a aceleração da gravidade.

Escreva um programa que leia a velocidade inicial v_0 , o ângulo α (em graus), e a distância horizontal D entre o pássaro e o porco, e informe se o pássaro atingirá (saída 1) ou não o porco (saída 0). Admita uma tolerância de 0,1. Ou seja, se $D = 20$ e $R = 19,9$, então podemos considerar que o pássaro acerta o porco.

DICAS: Importe o módulo math. Não se esqueça que a função seno (sin) trabalha com ângulos em radianos.

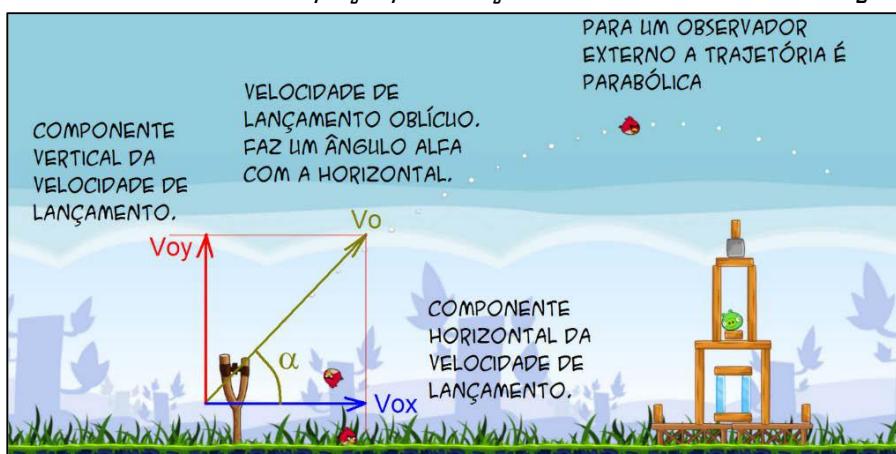


Figura 62: Exemplo de trajetória

Fonte: polibentinhofisica.blogspot.com apud ICOMP, UFAM

Primeiro passo: Interpretação da questão

Calma, eu sei que a primeira coisa que vem a cabeça é o desespero de não sabermos o que vamos fazer. Você não precisa dificultar mais as coisas, comece pelo básico e por onde você sabe, que tal?

Sempre que uma questão venha parecer difícil comece montando-a aos poucos. A própria questão já está lhe falando para convocar a biblioteca math, então a primeira coisa a se fazer é isso. Logo em seguida, é bem óbvio que essa questão tem variáveis, ela mesma já diz que são o ângulo, distância e a velocidade inicial então vamos construindo essas variáveis; perceba, estamos trabalhando com ângulos, nunca esqueça que o Python trabalha somente com radianos! Use a função radians encaixada na função float com o input para receber o valor do ângulo. A gravidade é a constante e claramente você deve escrever aquela equação do alcance máximo no Python.

Perceba que tudo que você fez até agora você já sabe fazer muito bem, não é nenhuma novidade. Agora que vem a parte que devemos analisar com cuidado, a condição. Vamos com calma, a condição é: que o valor do alcance máximo seja

igual ou próximo ao valor da distância horizontal. Vamos entender com calma, você se lembra como devemos escrever as condições no "if"?

- Operadores condicionais....

Sim são esses caras que sempre escrevem qual a condição do script para cada tomada de caminho, se você consegue representar a condição imposta pela questão por meio dos operadores condicionais você já consegue acabar com toda a questão. Vamos simplificar a condição da questão aos poucos, o alcance máximo é aquela equação que você deve escrever no Python no caso R e a distância horizontal é D, ou se você usou outras letras ou palavras para representar cada um tudo bem é só mudar, então vamos ler a condição imposta pela questão de novo só que dessa vez abreviado: R seja igual ou próximo de D, melhorou? Vamos seguir em frente...

A questão fala que o R pode ser próximo a uma tolerância de 0,1 de D, como representar isso?

- Se você tirar a tolerância como fica a condição?

R igual a D ou $R = D$.

- Percebeu como a tolerância vai mudar as coisas?

Vamos recorrer a matemática...

Se $R = D$ então $R - D = 0$, não é mesmo? Você se lembre qual é o outro nome da subtração?

- A subtração é formalmente chamada de "diferença". Ex.: A diferença entre 5 e 5 é 0, isso quer dizer que $5 - 5 = 0$.

Então se R e D são iguais a sua diferença é zero! Mas e se não forem iguais?

- Preste atenção, o resultado de uma diferença entre dois números é a quantidade de unidades necessárias para que o segundo número chegue no primeiro. Veja só: $5 - 4 = 1$, o número 1 é a quantidade de unidade necessária para que o 4 chegue no 5.

Se você relacionar tudo o que estamos analisando aqui com essa tolerância verá que R e D podem ser iguais ou podem ter uma diferença de no máximo 0,1, então podemos ter $R - D = 0$ ou $R - D = 0,1$.

Ok, achamos a solução o problema que devemos representa-la do jeito certo, não adianta achar a resposta mas não saber como demonstrá-la através do Python. Se você fizer o cálculo da equação na sua calculadora usando qualquer valor você chegará num resultado com muitas casas decimais! Qual o problema? Há é só o fato da nossa tolerância ser só de uma casa decimal...

Se colocarmos $R - D == 0$ or $R - D == 0,1$ no Python estaremos especificando que somente os valores que forem de diferença 0 e 0,1 serão válidos, o que não é verdade! A tolerância é até aonde um certo valor pode chegar e esse valor é 0,1. Então hora de colocar a cuca para funcionar e pensar, se é até 0,1 então todos os valores abaixo de 0,1 são válidos! Como por exemplo 0,0999999 ainda continua sendo válido. Então claramente estamos falando do sinal de menor ou igual a, que na linguagem Python quer dizer "é até".

$R - D \leq 0,1$, ou seja, a diferença entre R e D é até 0,1.

Agora com tudo pronto não esqueça de uma coisa, você está trabalhando com subtração, ou seja, diferença e o resultado de uma diferença pode resultar em um número negativo que para este caso é inviável. Na lista de funções do Python temos uma que se chama "**abs**" você já deve ter a visto pela sua calculadora, ela significa "absoluto", ou seja, é a função módulo e você já sabe o que ela faz, se o resultado for negativo a função não permitirá que o número saia negativo.

Segundo passo: Formulado o script

A primeira coisa a se fazer é convocar a biblioteca math, lembre-se as bibliotecas vem primeiro no seu script, e em segundo lugar em qualquer script vem as variáveis. As variáveis são, velocidade inicial, ângulo e distância, em todas serão usadas float e input e a do ângulo envolverá a função radians. Em terceiro lugar em todo script deve-se colocar as constantes da questão. Logo, basta fazer a equação dada na questão usando os operadores matemáticos do Python.

Depois de tudo isso agora sim vem a condicional. Logo depois de colocar o if utilize a função abs e dentro dela coloque a subtração do alcance máximo pela distância e em seguida use o operador condicional “é até” 0.1, isto é, ≤ 0.1 , então com isso sua condição já está pronta, basta dar ENTER colocar a função print para o primeiro caminho que é o fato da condição ser obedecida cuja resposta de saída é 1 e o segundo caminho fica no else que é o fato da condição não ser obedecida cuja saída é 0.

```
Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View Help
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Lab02_05.py
1 # -*- coding: utf-8 -*-
2 """
3 17/02/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from math import *
10
11 v = float(input("Velocidade inicial: "))
12
13 a = radians(float(input("Angulo com o solo: ")))
14
15 D = float(input("Distancia entre passaro e porco: "))
16
17 g = 9.8
18
19 R = ((v ** 2) * sin(2 * a)) / g
20
21
22 if (abs(R - D) <= 0.1):
23     print("1")
24 else:
25     print("0")
26
27
```

Figura 63: Questão 5B Segundo Passo

Fonte: Autoria própria

```
Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab02_05.py', wdir='C:/Users/lução a Progamação de Computadores/Meus Programas')
Velocidade inicial: 16
Angulo com o solo: 25
Distancia entre passaro e porco: 20
1
>>>
```

Figura 64: Resultado a) questão 5B

Fonte: Autoria própria

```
Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab02_05.py', wdir='C:/Users/lução a Progamação de Computadores/Meus Programas')
Velocidade inicial: 16
Angulo com o solo: 30
Distancia entre passaro e porco: 25
0
>>>
```

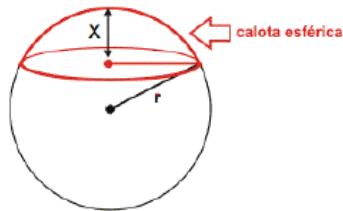
Figura 65: Resultado b) questão 5B

Fonte: Autoria própria

Questão 6 – Tanque de combustível

Um tanque de combustível tem o formato esférico. Escreva um programa que leia o valor do raio (r) do tanque, o valor da altura do ar (x) e pergunte ao usuário se deseja calcular o volume de ar (opção 1) ou o volume de combustível no tanque (opção 2).

A saída do programa deverá ser o volume conforme a opção selecionada, com quatro casas decimais de precisão. Não é necessário verificar situações inválidas (opções diferentes de 1 ou de 2, ou valores de altura maiores que o raio da esfera).



Volume da esfera de raio r	$V = \frac{4}{3}\pi r^3$
Volume da calota esférica de raio r e altura x	$V = \frac{\pi}{3}x^2(3r - x)$

DICA: O volume de ar corresponde à fórmula da calota esférica. Já o volume do combustível corresponde ao complemento do volume da calota esférica em relação ao volume total da esfera.

Primeiro passo: Interpretação da questão

Essa questão não é tão complicada assim, é simplesmente uma tomada de decisão que se deve tomar e pronto. Veja, a questão já está lhe dando tudo o que você precisa para fazer a questão, basta ler com calma. Você já sabe que deve usar a biblioteca math para usar o pi, e você logo de cara vê que precisará de três variáveis que são: raio, altura e opção.

Toda vez que você tiver uma equação matemática envolvendo a questão, você deve primeiramente colocar as variáveis necessárias, as constantes e depois escrever as equações. Neste caso, veja que temos duas equações, uma o Python calculará o volume da calota e a outra o Python calculará o volume restante, você já deve saber que se você calcular o volume total menos a calota você tem o volume só da parte abaixo da calota que nesse caso é o nível do combustível.

Vamos dar uma olhada na condição da questão: se eu escolher a opção 1 eu tenho o volume da calota, se eu escolher opção 2 eu tenho o volume abaixo da calota; vamos passar isso para a linguagem Python. A opção é uma variável no Python que será variada com a função `input` e o `int`, essa variável servirá de base para a condição; lembre-se que para escrevermos condições precisamos dos operadores condicionais. Vejamos que se a opção for igual a 1 teremos que calcular a calota, se não for igual a 1 teremos que calcular abaixo da calota, sacou? Vamos ver isso de outra forma, se `opção = 1` então temos o volume da calota, se não for então temos o volume abaixo da calota. Lembre-se que o sinal matemático de igual no Python é assim: `==`. Então temos que criar uma variável para a opção, lembre-se que o Python não

aceita variáveis com acentuação. Depois da variável deve-se criar a condição começando com o if e a condição é que a variável da opção seja 1, se for 1 então calculamos o volume da calota, se não for 1 calculamos o volume abaixo da calota.

Segundo passo: Formulado o script

Agora vamos construí-lo. A biblioteca é a primeira coisa que se digita depois do cabeçalho no script, em seguida coloque as três variáveis: raio (variando com float e input), altura do ar (variando com float e input) e a opção (variando com int e input). A condição é a próxima parte, no caso ela deve ter a configuração de que se a opção igual a 1 então temos o primeiro caminho que é a equação do volume da calota e claramente o resultado do print é o resultado dessa equação, o segundo caminho é se a opção não for igual a 1 então o Python terá de calcular a equação do volume da esfera e subtraí-la pelo volume da calota e o resultado do print será o resultado dessa diferença. Para facilitar sua vida crie uma variável igual a equação da calota antes de construir a condição, isso por que você terá de usar a equação duas vezes no script, então para evitar de ficar digitando formula toda hora então é melhor fazer somente uma variável e repeti-la quando for necessário. Por fim não esqueça de arredondar quatro casas decimais do resultado usando a função **round**.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View Help
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Lab02_06.py
1 # -*- coding: utf-8 -*-
2 """
3 18/02/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from math import *
10
11 raio = float(input("Raio do tanque: "))
12 x     = float(input("Altura do ar: "))
13
14 opcao = int(input("Volume do ar (1) ou do liquido (2)? "))
15
16 calota = pi/3 * x**2 * (3 * raio - x)
17
18 if (opcao == 1):
19     volume = calota
20 else:
21     volume = 4/3 * pi * raio**3 - calota
22
23 print(round(volume, 4))
24
25
26
27

```

Figura 66: Questão 6B Segundo Passo

Fonte: ICOMP, UFAM

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab02_06.py', wdir='C:/Users/User/Documentos/Helder Guerreiro/Meus Programas')
Raio do tanque: 5
Altura do ar: 4
Volume do ar (1) ou do liquido (2)? 1
184.3068
>>>

```

Figura 67: Resultado a) questão 6B

Fonte: Autoria própria

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab02_06.py', wdir='C:/Users/User/Documentos/Helder Guerreiro/Meus Programas')
Raio do tanque: 5
Altura do ar: 4
Volume do ar (1) ou do liquido (2)? 2
339.292
>>>

```

Figura 68: Resultado b) questão 6B

Fonte: Autoria própria

2.4 Avaliações

Aqui teremos algumas avaliações para entender mais um pouco o assunto estudado neste primeiro capítulo. Aconselho você tentar fazer sozinho antes de ler a resolução.

Avaliação A – A2

Avaliação Parcial 02 (A2) – Estrutura Condicional Composta (sem Aninhamento) - 2a

Informações

Questões

Notas

Questão 1

Escreva um programa que tome as **quatro notas** parciais de um aluno e mostre, além do valor da média, a mensagem "Aprovacao", caso a média seja **igual ou superior a 5**, ou a mensagem "Reprovacao", caso contrário.

Dicas:

1. Teste o script no Spyder antes de submetê-lo ao Code Bench.
2. Verifique se você está submetendo o código correto à questão correspondente desta Avaliação Parcial.
3. Os valores devem ser arredondados em até **duas casas** decimais.
4. Atenção ao uso de letras maiúsculas e minúsculas, e à acentuação.
5. Por exemplo, para as entradas **3.0 3.5 4.0 4.5**, a saída deve ser **3.75 Aprovacao**.

Nesta prova você percebe que não há tanta dificuldade assim, logo de cara você já sabe que tem quatro variáveis em que todas são variadas por **float** e **input**. Depois se tem a média que é pedida na questão, acho que você sabe muito bem como fazer uma média né? É só fazer uma média das quatro notas pelo Python, lembre-se de que a média será como uma espécie de equação matemática nesse script. Já a condição pede que a nota seja maior ou igual a 5, então você deve saber muito bem que a condição começa com **if** e depois usá-se os operadores condicionais para escrever as condições a serem impostas; logo abaixo da condição a ser obedecida o primeiro caminho será mostrar o resultado da média e a mensagem "Aprovacão"; caso a condição não seja obedecida, o segundo caminho é mostrar a média e a mensagem "Reprovacão".

The screenshot shows the Spyder Python 3.4 IDE interface. The title bar reads "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, and Run. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal de Minas Gerais\AV_A.py". The code in the editor is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 19/02/2016
4 
5 Autor: Helder Guerreiro
6 """
7 
8 nota1 = float(input("Primeira nota: "))
9 nota2 = float(input("Segunda nota: "))
10 nota3 = float(input("Terceira nota: "))
11 nota4 = float(input("Quarta nota: "))
12 
13 media = (nota1 + nota2 + nota3 + nota4)/4
14 
15 if media >= 5:
16     print(media)
17     print("Aprovação")
18 else:
19     print(media)
20     print("Reprovação")
21 
22
```

Figura 69: Resposta Avaliação A 2A

Fonte: Autoria própria

Avaliação B - A2

Avaliação 02 (A2) – Estrutura Condicional Composta (sem Aninhamento) - 2b

Informações

Questões

Notas

Questão 1

O número 99980001 possui a seguinte característica:

$$99980001 = (9998 + 0001)^2$$

Elabore um programa que imprima a mensagem “sim” se um número fornecido pelo usuário satisfaz essa característica, ou o valor da diferença das partes elevada ao quadrado, caso contrário.

Dicas:

1. Utilize o operador de resto da divisão (%).
2. Por exemplo, para a entrada 49980002, a saída deve ser 25000000.

Muito cuidado com a pegadinha nesta prova, você deve se lembrar na aula de algarismos significativos que o zero à esquerda vale nada não é? Então por que tem um “0001” ali? Só para enganar quem não entende nada de algarismos significativos ou já está bastante enferrujado....

Pegue sua calculadora e coloque essa mesma conta do quadrado da soma que aparece lá em cima só que dessa vez não coloque 0001 coloque somente 1... Voilá! O mesmo resultado.

Agora é hora de pensar o seguinte: como transformar o número 99980001 em 9998? Usando os operadores matemáticos do Python, pegue sua calculadora e divida esse grande número por 10000, o resultado será 9998,0001 e então como retirar esse 0001? Use a divisão por inteiro cujo resultado não mostra os decimais por maiores que eles sejam. Agora, você se lembra da dica que aparece lá em baixo na questão? O resto da divisão serve para você encontrar somente o 0001, isso por que a divisão do grande número por 10000 traz o resultado 9998,0001, ou seja, 9998 foram divididos inteiramente enquanto 0001 não, se tornado o resto da divisão.

Na produção do script faça uma variável para a entrada do número e depois faça outra variável para escrever a fórmula do quadrado da soma que acabamos de desvendar. A condição será que o número seja igual a essa fórmula, se a condição for respeitada então faça um print mostrando a resposta “sim” e se não for respeitada (`else`) faça outro print com o resultado da fórmula que esse número vai dar.

The screenshot shows the Spyder Python 3.4 IDE interface. The title bar reads "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, and Tools. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Rio de Janeiro\AV_B.py". The code in the editor is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 16/03/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 Numero = int(input("Qual o número? "))
9 Formula = (Numero//10000 + Numero%10000)**2
10
11 if Numero == Formula:
12     print("SIM")
13 else:
14     print(Formula)
15
16
```

Figura 70: Resposta Avaliação B 2A

Fonte: Autoria própria

3. Sua Vida Mais Difícil – As Condicionais Encadeadas

3.1 Juntem os Dedos: Hora da Fusão!

Você sabia que dá para criar outros `if's` e `else's` um dentro do outro? Ou seja, você tem uma condição a satisfazer (`if`) mas ela não é satisfeita então temos a exceção (`else`) só que essa exceção leva a uma outra condição que também tem uma exceção... Isso quer dizer que tudo pode virar um belo mingau bem misturado. É por isso que agora apresento um tal de “`elif`”, esse cara é a fusão (sim é uma inderata aos fãs de dragonball) do `if` e do `else`, ou seja, em vez de fazer uma outra condição dentro de uma exceção é melhor usar esse cara, mas preste atenção: o `elif` substitue o `else` de cima com o `if` de baixo, ou seja, quando se faz um `if` obrigatoriamente deve-se ter um `else` então a primeira condição é montada e em seguida a exceção dessa condição só que dentro dessa exceção tem outra condição que também tem uma exceção,

```
if (delta < 0):
    print("Nao tem raiz real")
else:
    if (delta == 0):
        r1 = -b/(2 * a)
    else:
        r1 = (-b+delta**0.5)/(2*a)
        r2 = (-b-delta**0.5)/(2*a)
```

```
if (delta < 0):
    print("Nao tem raiz real")
elif (delta == 0):
    r1 = -b/(2 * a)
else:
    r1 = (-b + delta**0.5) / (2*a)
    r2 = (-b - delta**0.5) / (2*a)
```

Figura 71: Substituição por `elif`
Fonte: ICOMP, UFAM

O pessoal que será substituído pelo `elif` será a exceção que tem a condição, não substitua pela exceção da outra condição. Não entendeu nada? Veja a imagem ao lado.

Uma observação a se tomar é que esse `elif` pode ser repetida quantas vezes você quiser, ou seja, você pode criar enúmeras condições só usando o `elif`, mas lembre-se: o final sempre termina com `else`.

3.2 Novas Funções

Devo apresentar a vocês agora uma função do próprio python, a função `abs(x)`. Suponho que você já tenha visto esse `abs` em algum lugar... é isso mesmo a sua calculadora também tem um desse. Abs quer dizer absoluto, ela é a função módulo da matemática, ou seja, a distância desse número até o zero. Para que vou querer usar essa função no Python? Nunca subestime nenhuma função, essa função ajuda muito em operações matemáticas, por exemplo: você queira fazer a diferença entre dois números mas não quer que de for alguma o resultado saia negativo, então é hora do `abs` entrar em ação.

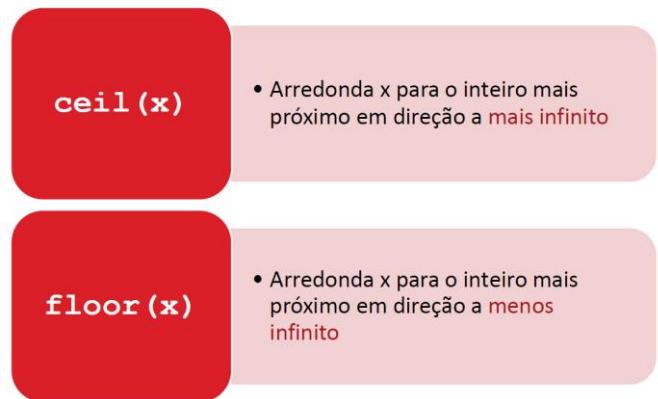


Figura 72: Funções `ceil` e `floor`
Fonte: ICOMP, UFAM

Temos duas novas funções a serem apresentadas do módulo math, sim apesar de ter tantas funções ele ainda tem algumas mais, as funções `ceil(x)` e `floor(x)`, esses dois trabalham de forma semelhante à função `round`, lembrando mais atrás a função `round` faz o arredondamento de um número real á quantas casas decimais você desejar. As funções `ceil` (que significa teto) e `floor` (que significa piso) fazem o arredondamento até que o número fique inteiro, a função `ceil` arredonda para cima e a `floor` para baixo, vou dar um exemplo para não ficar muito complicado: tenho 4.4, usando a função `ceil` o resultado fica 5 e usando a função `floor` fica 4.

3.3 Um Nova Módulo (Random)

Esta será uma apresentação rápida, vamos só dar entrada a essa nova biblioteca que se chama random. A função que irá representá-la será a função `randint(x,y)`, essa função tem por objetivo fazer um jogo aleatório de x á y e marcar um resultado, ou seja, será tipo um sorteio: você informa á função que ela deve sortear de x á y e o resultado irá sair por entre esses limites estabelecidos. Não esqueça que sempre em que você for ativar uma biblioteca deve escrever "from (módulo) import *".

3.4 Questões Resolvidas

Estas questões foram desenvolvidas pela ICOMP um instituto da UFAM e certos algoritmos (scripts) apresentados aqui pertencem ao ICOMP.

Questão 1 – De volta à área do triângulo

Escreva um algoritmo para calcular a área de um triângulo, a partir das medidas dos três lados, fornecidas pelo usuário, em qualquer ordem. O algoritmo não pode permitir a entrada de dados inválidos, ou seja, medidas menores ou iguais a zero, ou medidas que não correspondam às de um triângulo.

Primeiro passo: Interpretação da questão

De uma coisa você já sabe, as questões estão se complicando cada vez mais. Nesta questão iremos trabalhar com o `if` e o `else` só que agora eles aparecerão mais de uma vez, não se apresse em querer fazer tudo logo de uma vez e entrar em desespero por que você não saber fazer, faça tudo com calma e pensando em cada passo dado no seu script.

Como eu sei que essa questão irá usar o `if` e o `else` mais de uma vez? É só ler a questão e perceber que ela está te dando mais de uma condição: a de não negatividade e das medidas serem correspondente a de um triângulo (já explico isso), neste caso você irá criar uma condição primeiramente para a não negatividade (ou vice versa) e a condição sendo respeitada fará o script seguir para mais outra condição que deve ser respeitada, se os dados não obedecerem a primeira condição toda a parte do script relacionada a segunda condição será ignorada e o script irá pular diretamente para o `else` da primeira condição; se a primeira condição for respeitada e na segunda condição o mesmo não acontecer o script irá pular para o `else` da segunda condição. Tenha algo em mente e é muito importante que você se lembre: todo `else` deve ficar na mesma direção da sua respectiva condição (`if`).

Agora vamos á construção das condições, você se lembra como montamos condições? Sim isso mesmo, com os operadores condicionais (aposto que você já se esqueceu deles), então vamos lá: a primeira condição (`if`) será a não negatividade, é bem claro como água do rio solimões que devemos usar o sinal maior que (`>`) e colocar um zero do lado não? É parece ser isso mesmo.... no início do script você irá colocar a entrada dos lados, digamos que você coloque `a`, `b` e `c`, então nossa condição será `a > 0` e `b > 0` e `c > 0`. Voilá! Primeira condição pronta! "peraí" aposto que você esqueceu de uma coisa né.... além desse operadores (`>`) devemos usar outro para podermos juntar essas três condições em uma só! Lembre-se que o Python é um programa cuja programação é inglesa então... eu duvido que você se lembre do "and", sim meu querido grilo você esqueceu do `and`... tá vendo onde eu coloque "e" lá em cima? Então é só você trocar por `and`.

- "Mas eu não sei como fasso para colocar uma condição dentro de outra (emoticon de choro do Whatsapp)"

Calma bebê, você deve saber colocar um pequeno script abaixo do `if` não é mês? ... (por favor me diz que você lembra disso!) então é a mesma coisa! Você coloca o `if` e escreve a condição, depois coloca dois ponto e dá `ENTER` e embaixo dele coloque um pequeno script dizendo o que acontece se essa condição for obedecida, nessa parte você irá

colocar outro if que terá outro mini script abaixo dele (sentiu a pressão?), ou seja, abaixo do primeiro if (condição) você irá colocar somente outro if, só isso! Não tem como errar! É só você escrever seu primeiro if, colocar sua condição (nesse caso a não negatividade), colocar dois pontos, dá ENTER, e embaixo desse if colocar mais outro if que também tem uma condição, que também tem dois pontos, que também tem que dá ENTER e que também ter um else!

Então... no segundo if temos que colocar a segunda condição, a que os lados colocados devem corresponder aos lados de um triângulo... (sei, você não sabe o que é isso...), então vamos a explicação: Use sua régua e faça um triângulo em que os lados são 1, 1 e 2 em centímetros; (sei que você tem preguiça até para desenhar com uma régua ... como você quer aprender assim?) perceba que você não consegue fechar o triângulo perfeitamente os lados 1 e 1 não dão espaço para que tenha um lado com 2; mas se você tiver um triângulo com 2, 2 e 1 você consegue fechar.

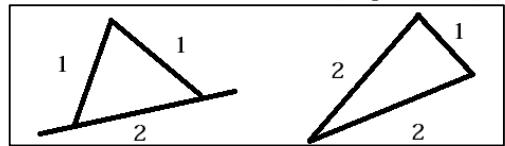


Figura 73: Exemplo de triângulos

Fonte: Autoria própria

Como representar isso matematicamente para colocarmos isso como condição? Temos que testar nossos conhecimentos em operadores condicionais, perceba que o primeiro triângulo deu errado por que temos dois números que comprometem o último, então veja que o 2 é maior que todo mundo, nesse caso podemos usar o sinal ($>$) para indicar isso, vamos representar que o 2 é maior que todos os lado, ou seja, $2 > 1$ e $2 > 1$, mas isso está certo? De certa forma sim, mas é necessário juntar esse pessoal para ficar correto e organizado, para isso use a soma: $2 > 1+1$, mas com a soma o sinal ($>$) se torna inválido então $2 = 1+1$; fazendo isso com o segundo triângulo chega-se ao resultado que $2+1 > 2$ e isso realmente é verdade! Se ficarmos mudando os valores dessa operação de lugar temos:

Primeiro triângulo:

$$\begin{aligned} - |+| &= 2 \\ - 2+| &> 1 \\ - |+2 &> 1 \end{aligned}$$

Segundo triângulo:

$$\begin{aligned} - 2+| &> 2 \\ - |+2 &> 2 \\ - 2+2 &> 1 \end{aligned}$$

Veja como aquele sinal de igual (=) torna o resultado diferente de um triângulo para outro, o segundo triângulo tem todas as somas de lados maior que seus próprios lados mas o primeiro triângulo tem uma soma de lados igual a um dos seus lados, isso torna o triângulo inviável! As condições a serem obedecidas é claramente as do segundo triângulo, basta trocar esses número por a, b e c. Esta é a nossa condição.

Se essa segunda condição também for obedecida estamos nos dando com um real triângulo e então é hora de fazer o que a questão pede: "calcular a área de um triângulo, a partir das medidas dos três lados, fornecidas pelo usuário, em qualquer ordem." Essa parte já é muito fácil, basta você escrever a formula para calcular a área de qualquer triângulo e pronto, há! Lembre-se que essa formula deve estar debaixo da segunda condição. A formula a ser usada aqui é a mesma da questão 6 do primeiro laboratório, que é a seguinte:

Considere um triângulo cujos lados sejam designados por a , b e c . Considere ainda que $s = (a + b + c) / 2$. A área do triângulo pode ser calculada usando a seguinte fórmula:

$$A = \sqrt{s(s - a)(s - b)(s - c)}$$

Depois de você escrever essas formulas abaixo da segunda condição coloque o print abaixo delas para apresentar qual foi o resultado obtido da área.

Agora acabamos com todas as condições, agora é hora das exceções. O primeiro else a ser construído é o da segunda condição, simplesmente por que você já está abaixo dela, não esqueça de manter o else alinhado com o seu respectivo if. O else é simplesmente o fato de que os lados que você inseriu no script não são de um triângulo, então simplesmente coloque abaixo do else um print dizendo "Não é um triângulo" ou alguma coisa assim e pronto. Agora o próximo else que é o da primeira condição tem o mesmo caso que o outro, esse else é para quando o resultado não der um triângulo, faça a mesma coisa que foi feita no outro else: coloque um print dizendo que aquilo que você inseriu não dá um triângulo.

Segundo passo: Formulado o script

Finalmente a construção do script. Comece claramente ativando a biblioteca math, isso por que a formula da área do triângulo tem uma raiz quadrada; depois é só escrever as entradas dos três lados do triângulos. Depois disso começa a construção das condicionais, a primeira condicional é a não negatividade: $a > 0$ and $b > 0$ and $c > 0$, abaixo dessa condição coloque a segunda condição que é a de que os lados devem corresponder a um triângulo: $a + b > c$ and $a + c > b$ and $b + c > a$, com a segunda condição feita coloque o que acontecerá com as duas condições respeitadas que é o cálculo da área, primeiro faça o "s" da formula e em seguida a formula de Heron, aí é só colocar o print para mostrar o resultado. Ainda no alinhamento da segunda condição faça a exceção dela, o else terá como objetivo mostrar a mensagem através do print dizendo que "Não é um triângulo", em seguida faça a mesma coisa com o else da segunda condição colocando-o alinhado com o primeiro if.

The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, and Run. The main window displays a code editor titled 'Lab03_01.py'. The code is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 17/03/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from math import *
10
11 # Leitura dos lados do triangulo a, b, & c
12 a = float(input ("Lado 1: "))
13 b = float(input ("Lado 2: "))
14 c = float(input ("Lado 3: "))
15
16 # Condição da não negatividade
17 if (a > 0 and b > 0 and c > 0):
18     # Testa se medidas correspondem aas de um triangulo
19     if (a + b > c and a + c > b and c + b > a):
20         s = (a + b + c) / 2.0
21         area = sqrt(s * (s-a) * (s-b) * (s-c))
22         print(area)
23     else:
24         print("Não é um triangulo!")
25 else:
26     print("Não é um triângulo!")
27
28
```

Figura 74: Questão IC Segundo Passo

Fonte: Autoria própria

The screenshot shows the Spyder Python 1 console. The user runs the script with the command `>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab03_01.py', wdir='C:/Users/User/Documents/Helder Guerreiros/Meus Programas')`. The output shows the input of three sides (2, 2, 1) and the calculated area (0.9682458365518543).

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab03_01.py', wdir='C:/Users/User/Documents/Helder Guerreiros/Meus Programas')
Lado 1: 2
Lado 2: 2
Lado 3: 1
0.9682458365518543
>>>
```

Figura 76: Resultado b) questão IC

Fonte: Autoria própria

The screenshot shows the Spyder Python 1 console. The user runs the script with the command `>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab03_01.py', wdir='C:/Users/User/Documents/Helder Guerreiros/Meus Programas')`. The output shows the input of three sides (1, 1, 2), which results in an error message: "Não é um triangulo!" (Not a triangle!).

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab03_01.py', wdir='C:/Users/User/Documents/Helder Guerreiros/Meus Programas')
Lado 1: 1
Lado 2: 1
Lado 3: 2
Não é um triangulo!
>>>
```

Figura 75: Resultado a) questão IC

Fonte: Autoria própria

Questão 2 – Isósceles, equilátero ou escaleno?

Escreva um programa que leia as medidas dos três lados de um triângulo, fornecidas pelo usuário, em qualquer ordem, e imprima o nome do tipo de triângulo: equilátero (todos os três lados iguais), isósceles (apenas dois lados iguais), ou escaleno (nenhum par de lados iguais).

Se pelo menos um dos lados for negativo ou os três lados não formarem um triângulo, o programa deverá imprimir a mensagem "invalido" na tela.

DICA: Adapte o código da Questão 1 anterior. No lugar do cálculo da área, você deverá inserir um teste de condições, com relação à igualdade dos lados fornecidos. Atenção para os caracteres acentuados. Todas as mensagens devem estar em letras minúsculas.

Primeiro passo: Interpretação da questão

Como já fizemos a primeira questão e a segunda é só um complemento dela então nós já temos meio caminho andado, tudo que fizemos no primeiro vamos usar aqui no segundo com uma diferença: temos mais condições para adicionar ao script.

As condições a mais é o fato do triângulo ser ou equilátero ou isósceles ou escaleno e veja só: essa condição aparece depois de todas as outras, pois para ser um desses três tipos o cara tem que ser obrigatoriamente um triângulo. Como essa questão não está pedindo a área do triângulo então apague as formulas e deixe as duas condições com seus else's. É abaixo da condição em que os lados devem corresponder a um triângulo que vai acontecer toda a mágica, isso quer dizer ... Traramramran! Um novo **if!** (Ai você me diz: que droga!). A nova condição é justamente o fato de qual o tipo do triângulo inserido e essa nova condição deve ser carregada de três condições diferentes: equilátero, isósceles e escaleno.

Você deve saber muito bem que para ser equilátero todos os lados devem ser iguais, só tome cuidado que você não pode escrever no Python simplesmente assim: `a==b==c`, o Python não aceita isso, mas então use o fato de que se "a" é igual a "b" e "b" é igual a "c" então "a" é igual a "c", logo: `a == b and b == c`. Após escrever a condição claramente é hora de escrever o mini script abaixo dela, ou seja, o acontece se os lados forem iguais. Coloque um `print` com a mensagem "Equilátero" e pronto. Agora e se os lados não forem iguais? Ou ele é isósceles ou escaleno, mas se colocarmos o `else` só podemos colocar uma das opções, não duas! Isso quer dizer.... Uma nova condição, ou seja, um novo `if!` Por que `if de novo?`? Simplesmente por que faremos um `else` que irá levar a um `if` e esse é o fato dele ser isósceles e esse `if` terá o seu próprio `else` que será o fato dele ser escaleno!

Isso vai virar uma bagunça eu sei, mas foi por isso que foi criado o `elif`, esse cara vai substituir o `else` e o `if` ao mesmo tempo! Ele funciona assim: Usando o `elif` você irá ficar alinhado com o `if` acima dele, o recuo usado para o `else` não precisará mais, agora você coloca somente o `elif` e ele será a mesma coisa que você estiver escrevendo `else` e depois `if`, a condição que deveria ficar no `if` agora ficará ao lado do `elif` e pronto. Com o `elif` a condição a ser colocada é o fato do triângulo ser isósceles, ou seja, um dos lados iguais e então para escrever em operadores condicionais podemos fazer

simplesmente assim: `a == b or b == c or a == c`. Abaixo do elif escreva o que acontece se essa condição for obedecida, coloque um print com a mensagem “Isósceles” e pronto.

Nunca esqueça de uma coisa: mesmo que você tenha colocado o elif não esqueça que todo if tem seu else, e o else que foi englobado pelo elif não é o else do if que também foi englobado pelo elif. Isso quer dizer que depois de escrever elif você deve escrever um else e nesse caso o else é o fato dos lados serem diferentes, abaixo desse else é só escrever um print dizendo “Escaleno”.

Não esqueça que além desse else tem outro que pertence à segunda condição e mais outro que pertence à primeira condição, eles já foram construídos na primeira questão e não devem ser mexidos, só faça mudar a mensagem do else que antes era “Não é um triângulo! ” E agora deve ser “Inválido”.

Segundo passo: Formulado o script

Neste caso pegue o script da questão anterior e apague as formulas da área do triângulo, nesse mesmo local coloque a condição do triângulo ser equilátero: `a == b and b == c`, seguido do print dizendo “Equilátero”; logo depois alinhado com essa condição crie um elif com a condição do isósceles: `a == b or b == c or a == c`, seguido do print dizendo “Isósceles”; por fim coloque o else com a mensagem “Escaleno”. Não esqueça de mudar as mensagens dos else’s das duas primeiras condições para “Inválido”.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View Help
Editor - C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/1º Período
Lab03_02.py

1 # -*- coding: utf-8 -*-
2 """
3 17/03/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from math import *
10
11 # Leitura dos lados do triangulo a, b, & c
12 a = float(input("Lado 1: "))
13 b = float(input("Lado 2: "))
14 c = float(input("Lado 3: "))
15
16 # Condição da não negatividade
17 if (a > 0 and b > 0 and c > 0):
18     # Testa se medidas correspondem aas de um triangulo
19     if (a + b > c and a + c > b and c + b > a):
20         if (a == b and b == c ):
21             print("Equilátero")
22         elif (a == b or b == c or a == c):
23             print("Isósceles")
24         else:
25             print("Escaleno")
26     else:
27         print("Inválido")
28 else:
29     print("Inválido")
30
31
32

```

Figura 77: Questão 2C Segundo Passo

Fonte: ICOMP, UFAM

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab03_02.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Meus Programas')
Lado 1: 10
Lado 2: 11
Lado 3: 12
Escaleno
>>>

```

Figura 78: Resultado a) questão 2C

Fonte: Autoria própria

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab03_02.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Meus Programas')
Lado 1: 10
Lado 2: 10
Lado 3: 10
Equilátero
>>>

```

Figura 79: Resultado b) questão 2C

Fonte: Autoria própria

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab03_02.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Meus Programas')
Lado 1: 10
Lado 2: 10
Lado 3: 11
Isósceles
>>>

```

Figura 80: Resultado c) questão 2C

Fonte: Autoria própria

Questão 3 – Animais nas cédulas do Real

As cédulas do real começaram a entrar em circulação no Brasil em 1994. Diferentemente das moedas que haviam circulado anteriormente, o real não traz na sua nota personalidades da história nacional, mas sim animais da fauna brasileira, conforme mostra a tabela abaixo.

Animal	Valor da cédula
Tartaruga	R\$ 2
Garça	R\$ 5
Arara	R\$ 10
Mico-leão-dourado	R\$ 20
Onça-pintada	R\$ 50
Garoupa	R\$ 100

Escreva um programa que leia o valor de uma cédula e apresente na tela o nome do animal representado no verso da nota. Se não existir uma cédula no valor inserido, o programa deverá informar “erro”.

DICA: use a estrutura if-elif-else. Não use acentos. Atenção para os hifens e iniciais maiúsculas.

Primeiro passo: Interpretação da questão

Você não terá muita dificuldade em entender essa questão ela é muito simples, só de ler o enunciado você já percebe que o programa deve simplesmente encaminhar o valor da nota inserida a um respectivo animal, e caso o valor da nota não tenha nenhum animal o resultado seria simplesmente “erro”.

A dica da questão já entrega tudo, temos de usar o `if`, `elif` e o `else`, sei que para você não deve estar muito claro como o script será montado, mas vamos com calma construindo todo o entendimento passo a passo.

A primeira coisa a se fazer neste script é sem dúvida e entrada do valor da cédula, ou seja, o uso do `input` com a função `int`, vamos usar a função `int` por que as cédulas com animais são claramente números inteiros.

Agora vamos construir usando a lógica.... Tendo a entrada agora posso colocar qualquer número... SE eu colocar o 2 o que acontece? E SE eu colocar o 5? E SE eu colocar o 10? E SE eu colocar o 20? E SE eu colocar o 50? E SE eu colocar o 100? OU SE eu colocar qualquer número menos esses?

Você percebeu? Toda vez que você for fazer um script e não sabe quem exatamente vai usar transforme seu script num show de perguntas, se ao colocar qualquer valor o caminho tomado para chegar ao resultado for o mesmo então não temos a participação de `if` e `else`, por exemplo: Se eu coloco 20, tenho como resultado: 400 m^2 ; se eu coloco 30, tenho como resultado: 900 m^2 ; perceba que nos dois casos não importando o valor inserido o script sempre irá calcular A^2 , agora veja este outro exemplo: Se eu coloco -22, tenho como resultado: Erro!; se eu coloco 5, tenho como resultado: 25 m^2 ; perceba que agora há uma diferença de caminhos, SE o valor for positivo o resultado irá dar a área normal, SE o valor for negativo o resultado irá dar Erro!, isso já lhe diz algo não é: SE = if.

Agora já posso lhe falar que **SE = if**, **E SE = elif**, **OU SE = else**, guarde isso em sua mente e não erre mais as questões de condicionais. Para resolver essa questão já não me resta dizer mais nada, não é?

Segundo passo: Formulado o script

Construa a entrada para os valores usando o input e o int, agora use as condicionais (faça em ordem para ficar organizado), o if é a primeira condicional que irá carregar a condição do valor ser igual a 2 (não esqueça de usar os operadores condicionais, o sinal “igual à” no Python é “==”), agora construa o mini script que irá fazer o caminho caso a condição seja respeitada: é só um print com o nome do animal da nota de 2 (Tartaruga), ou se você quiser pode fazer uma variável chamada “Animal” e igualá-la ao nome do animal e no final do script só colocar um print da variável Animal, nesse caso a variável Animal apareceria em todas as condições, é uma forma de se fazer. A segunda condição será feita pelo elif que irá carregar a condição do valor inserido ser igual a 5, agora você já sabe o que fazer depois de escrever a condição não é mesmo? A terceira condição também será elif (lembre-se a lista de elif's somente irá mudar quando você chegar na última condição que será feita pelo else) que irá carregar a condição do valor inserido ser igual a 10. Agora você já sabe o que fazer com o resto, vamos pular para o final: a última condição será feita pelo else que irá carregar a condição do valor inserido ser um valor inválido com qualquer uma das notas, obviamente no min script abaixo do else você irá colocar print ou a variável Animal com a mensagem “Erro”.

```
# -*- coding: utf-8 -*-
"""
2 21/03/2016
4
5 Autor: Helder Guerreiro
6 """
8
9 # Leitura do valor da cedula
10 cedula = int(input ("Valor da cedula: "))
11
12 # Testa cada possibilidade de cedula
13 if (cedula == 2):
14     Animal = "Tartaruga"
15 elif (cedula == 5):
16     Animal = "Garca"
17 elif (cedula == 10):
18     Animal = "Arara"
19 elif (cedula == 20):
20     Animal = "Mico-leao-dourado"
21 elif (cedula == 50):
22     Animal = "Onca-pintada"
23 elif (cedula == 100):
24     Animal = "Garoupa"
25 else:
26     Animal = "erro"
27
28 # Impressao do resultado
29 print(Animal)
30
```

Figura 81: Questão 3C Segundo Passo

Fonte: ICOMP, UFAM

```
Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab03_03.py', wdir='C:/Users/User')
Valor da cedula: 20
Mico-leao-dourado
>>>
```

Figura 82: Resultado a) questão 3C

Fonte: Autoria própria

```
Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab03_03.py', wdir='C:/Users/User')
Valor da cedula: 15
Erro
>>>
```

Figura 83: Resultado b) questão 3C

Fonte: Autoria própria

Questão 4 – Conta de Energia

Escreva um programa que determine o preço a pagar pelo fornecimento de energia elétrica. Como entrada, ele deve receber o consumo de energia (em kWh) e o tipo de instalação (R para residências, I para indústrias, e C para comércios). Use a tabela a seguir para calcular o preço devido:

Preço por tipo e faixa de consumo		
Tipo	Faixa (kWh)	Preço
Residencial	Até 500	R\$ 0,40
	Acima de 500	R\$ 0,65
Comercial	Até 1000	R\$ 0,55
	Acima de 1000	R\$ 0,60
Industrial	Até 5000	R\$ 0,55
	Acima de 5000	R\$ 0,60

DICA: use a estrutura if-elif-else. Atenção para as letras maiúsculas e o uso de aspas quando os caracteres são considerados como valor, e não como nome de variável. Se valores inválidos forem inseridos, o resultado deve ser -1.

Primeiro passo: Interpretação da questão

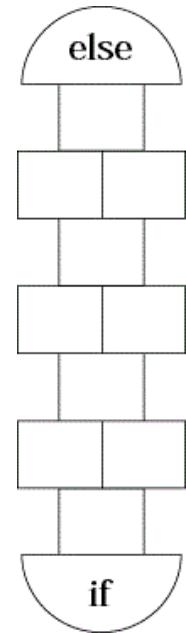
Essa questão não perdeu a mesma visão que a anterior tem, MAS... ela é um só um pouquinho mais complicada (é ironia, essa questão é ferrada mesmo), bom correr da sala para a cozinha não ajuda muito, então vamos passo a passo para você não acaba chorando aí.

O que tem de diferente dessa vez? Bom digamos que agora temos mais `if's` e `elif's` e ainda um dentro do outro (sentiu a pressão?), é como se fosse um quebra cabeça, a sua cabeça quebrando no teclado. Mas não é o fim do mundo, eu to aqui para te ensinar (pelo menos eu estou tentando).

Vamos lá no começo mesmo... "No princípio era o verbo..." E que verbo é esse? Isso mesmo é o verbo entrar! Todo script (que se preste) deve começar com uma entrada, então a entrada já está sendo dita lá no enunciado da questão: Consumo de energia e Tipo de instalação, bom você já sabe que o Consumo de energia é uma entrada do tipo `float-input` e a Tipo de instalação é do tipo `input` não é mesmo? (Se você não souber: parabéns você é um burro [sem ofensa]). Bom esse é o primeiro passo e o mais fácil, pronto seu script já está pegando um bom rumo (Tu achas mesmo que fazer só a entrada do script é um bom rumo?).

Agora não tem outra é claro que está na hora das condicionais! Mas não se desespere TODA condicional começa com um `if` e termina com um `else`, então você já sabe né: o `if` é o primeiro passo obrigatório da condicional. Se você tem retardamento mental a imagem ao lado irá lhe ajudar a entender (brincadeira).

Veja a tabela dada no enunciado da questão, (tá que é que tem?) é uma pista! (nossa sério?), perceba que a tabela te faz ler primeiro o tipo de instalação, depois quanto se gastou e depois o quanto é aplicado em cada caso, adivinhe só: o computador ler do mesmo jeito que você! É por isso que os scripts do Python devem ser organizados de forma coerente, do mesmo jeito que você não entende o quarto de um moleque de 13 anos o computador não entende um script todo mal



formulado, então sempre quando você estiver em dúvida num script de por onde começar, lembre-se do seguinte: comece o script pelo mesmo lugar que você mesmo começaria se estivesse lendo o processo, exemplo: Você lê que primeiro deve botar os ovos (de galinha) e depois mexer, do mesmo jeito você colocará lá no seu script do jeito que se deve colocar. Eu falei isso tudo só para você entender que devemos começar colocando a primeira condição usando o Tipo de instalação.

Mas antes de criar a primeira condição tenha cuidado com o seguinte: iremos usar essa primeira condição para um tipo de instalação ou para um tipo de instalação seguido de um limite de consumo de energia? Essa resposta depende de uma coisa tão pequena que você nem pensava, você se lembra que na dica diz para você colocar o -1 caso os dados forem inválidos? Isso mesmo, esse cara fará a primeira condição ser somente o fato do Tipo de instalação ser R, ou seja, Residencial.

Com o if sendo o Tipo de instalação igual a R devemos olhar o fato do valor ser negativo (esse é um dado inválido), então vamos fazer aquelas famosas perguntas: Meu tipo de instalação é R, mas SE o meu consumo for negativo? E SE meu consumo for até 500 KW/h? OU SE meu consumo não for nem negativo nem menor que 500 KW/h? Acho que com essas perguntas você já deve ter sacado como iremos construir essa primeira condicional, não é? Relembrando: SE = if, E SE = elif, OU SE = else. Com isso já sabemos como será nossa primeira condição: A primeira condição é o fato do Tipo de instalação for igual a R, sendo respeitada essa condição deve-se ver SE o valor do consumo é negativo, sendo respeitada a condição deve-se sair um resultado dizendo "-1", não sendo respeitada haverá outra condição em que o valor seja até 500 KW/h, essa sendo respeitada deve-se sair um resultado que irá multiplicar o valor do consumo por 0.40, não sendo respeitada também resta a última que é a exceção que é o fato do valor do consumo ser maior que 500 KW/h, sendo a exceção a única opção a se seguir deve-se sair um resultado que irá multiplicar o valor do consumo por 0.65.

Depois de você ler esse passo a passo você já deve estar quase pedindo para sair, calma ta acabando (tá nem na metade). Essa parte da condição em que o Tipo de instalação é igual a R é a mesma coisa a ser feita no C e no I, é simplesmente a mesma coisa! Quer ver? Vamos ao show de perguntas que tudo irá se esclarecer: Meu tipo de instalação é C, mas SE o meu consumo for negativo? E SE meu consumo for até 1000 KW/h? OU SE meu consumo não for nem negativo nem menor que 1000 KW/h? Agora deu para esclarecer legal como todas as três condições são iguais. Só uma coisa: na hora de sair o resultado fica a seu critério, pois sabemos que temos que dar o resultado do preço, você pode usar uma variável e igualar aos resultados que as condições dão (igual eu fiz na última questão) ou se quiser pode usar o print normal como você já sabe, e na hora de multiplicar o valor do consumo pela taxa você pode fazer uma formula e igualar a uma variável ou simplesmente na hora de dar o resultado pegue o valor do consumo e multiplique pela taxa.

Segundo passo: Formulado o script

Comece com as duas entradas, a do Consumo é do tipo float-input e a do Tipo de ligação é só o input mesmo. Nas condicionais a primeira etapa é a condição do Tipo de ligação ser R, sendo respeitada o seu mini script terá a condição do Consumo ser não negativo, sendo respeitada o mini script abaixo dela deverá mostrar o resultado "-1", depois dessa condição há a condição do Consumo ter o valor de até 500 KW/h, sendo respeitada o mini script abaixo dela deverá mostrar o resultado da multiplicação do Consumo pela taxa de 0.40, a última é a exceção que deverá mostrar no seu mini

script o resultado da multiplicação do Consumo pela taxa de 0.65. Repita o mesmo processo na segunda etapa em que o Tipo de ligação é C e na terceira em que o Tipo de ligação é I.

Lembre-se as etapas do Tipo de ligação começaram com um if, obrigatoriamente deve-se terminar com um else, o else será dado quando a entrada do tipo de ligação não for igual a R, C ou I, o else no seu min script deverá mostrar o resultado “-1”.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View Help
Editor - C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/
Lab03_04.py

1 # -*- coding: utf-8 -*-
2 """
3 21/03/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 # Leitura do valor da cedula
10 tipo = input("Tipo de instalacao (R, I ou C): ")
11 faixa = int(input ("Faixa de consumo: "))

13 # Testa cada possibilidade de cedula
14 if (tipo == "R"):
15     if (faixa < 0):
16         preco = -1
17     elif (faixa <= 500):
18         preco = faixa * 0.40
19     else:
20         preco = faixa * 0.65
21 elif (tipo == "I"):
22     if (faixa < 0):
23         preco = -1
24     elif (faixa <= 1000):
25         preco = faixa * 0.55
26     else:
27         preco = faixa * 0.60
28 elif (tipo == "C"):
29     if (faixa < 0):
30         preco = -1
31     elif (faixa <= 5000):
32         preco = faixa * 0.55
33     else:
34         preco = faixa * 0.60
35 else:
36     preco = -1
37
38 # Impressao do resultado
39 print(preco)
40

```

Figura 84: Questão 4C Segundo Passo

Fonte: ICOMP, UFAM

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/Lab03_04.py', wdir='C:/Users/User/Universidade Federal do Amazonas/Lab03_04')
Tipo de instalacao (R, I ou C): R
Faixa de consumo: 336
134.4
>>>

```

Figura 85: Resultado a) questão 4C

Fonte: Autoria própria

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/Lab03_04.py', wdir='C:/Users/User/Universidade Federal do Amazonas/Lab03_04')
Tipo de instalacao (R, I ou C): I
Faixa de consumo: -65
-1
>>>

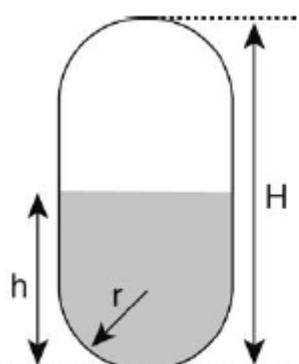
```

Figura 86: Resultado b) questão 4C

Fonte: Autoria própria

Questão 5 – Volume de combustível

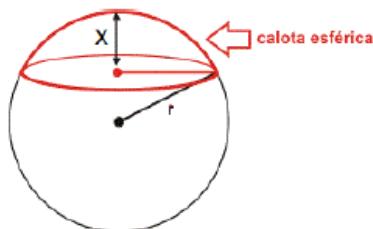
Considere um tanque de combustível com o formato descrito na figura abaixo. Utilize o método de resolução de problemas para escrever um script Python que calcule o volume de combustível (em litros), com três casas decimais de precisão, dadas as seguintes medidas (em metros):



H – Altura total do tanque
h – nível de combustível no tanque
r – raio dos bojos semiesféricos inferior e superior

CONVENÇÃO: as entradas são fornecidas na ordem $H \ h \ r$. A saída deve ter três casas decimais de precisão, se as entradas forem válidas. Caso contrário, a saída deve ser -1 .

FÓRMULAS ÚTEIS:



Volume do cilindro de raio r e altura x	$V = \pi r^2 x$
Volume da esfera de raio r	$V = \frac{4}{3} \pi r^3$
Volume da calota esférica de raio r e altura x	$V = \frac{\pi}{3} x^2 (3r - x)$

DICA: Verifique diversas possibilidades do nível de combustível h em relação à altura do tanque H .

Primeiro passo: Interpretação da questão

Essa questão parece assustar, mas nunca julgue uma questão por causa do tamanho do seu enunciado, essa questão não é difícil e é bem mais fácil que a anterior. Qual é o nosso trabalho aqui? A construção do script é simples, mas

o bicho pega aqui na matemática, ou seja, temos que organizar as formulas de tal forma que possamos calcular o tanto de volume para cada caso dependendo da posição do combustível. Mas é por isso que estamos aqui, vou lhe ensinar a fazer isso aqui passo a passo, então presta atenção.

É óbvio que temos de usar a biblioteca math não? (Olha o Pi ali) E outra, é bem óbvio que temos três entradas a inserir neste script como o próprio enunciado diz. Então sem enrolação vamos entender a distribuição das formulas nessa questão.

Você já sabe que essa questão envolve estrutura condicional, mas a condição irá depender de que? Bom você já deve saber, o combustível é que mudará toda a história. Antes de pensarmos em como montarmos essas formulas devemos primeiro entender que a base de tudo será o tamanho do combustível, ou seja, temos que relacionar o tamanho do combustível com todo o tanque, dependendo de onde o combustível está iremos usar formulas diferentes. Pensando agora nas formulas, temos a do volume do cilindro: claramente é o corpo do tanque que fica no meio; o volume da esfera: pense assim, se você tirar o cilindro que está no meio o tanque virará uma esfera, não?; por fim o volume da calota da esfera: a calota é a fatia da esfera.

Olhe para as formulas e para o tanque, nós só temos três casos diferentes que é o fato do combustível está acabando que é quando ele está abaixo do cilindro, quando ele está entre o cilindro e quando ele está quase cheio (ou cheio mesmo) acima do cilindro, vamos distribuir as formulas entre essas situações.

A questão nos mandou que se o valor inserido fosse inválido o script deveria mostrar o resultado -1, claro estamos falando do número negativo, SE o valor inserido a qualquer um dos três for negativo então o resultado é -1. Você pode fazer essa condição no `if` usando os operadores condicionais `or`, para colocar `h < 0 or H < 0 or r < 0`, sacou?

E SE o combustível ficar no finalzinho abaixo do cilindro? Bom nós percebemos que aí não temos mais contato com o cilindro e nem com a metade da esfera lá em cima, nesse caso o combustível pode pegar parte dessa meia esfera ou a meia esfera completa. Você como um bom entendedor da matemática deve perceber que estamos falando da calota esférica e já tem uma formula prontinha para você lá em cima. Isso quer dizer que o `elif` será responsável para quando o combustível estiver na calota e o seu script deverá calcular a formula da calota, mas como representar a condição do combustível estiver na calota através dos operadores condicionais? Calma vamos pensar, olhe melhor a imagem do tanque completo, percebe que esse tal bojo é a calota e ele tem um raio, o raio é o tamanho exato da calota, então para que o combustível esteja dentro da calota ele deve estar menor ou igual ao tamanho da calota que no caso é o raio dela.

E SE o combustível ficar entre o meio do cilindro? Nesse caso o combustível vai pegar o bojo de baixo completamente e uma parte ou todo o cilindro do meio, nesse caso você deve entender que iremos somar formulas, por que o cilindro tem a sua formula e o bojo de baixo também tem a sua. Há então é só somar a formula do cilindro e a da esfera e acabou-se? Não calma aí, entenda que como o bojo de cima não está entrando nessa soma então nós temos a metade de uma esfera por que só a parte de baixo está contando, então veja a formula da esfera, como seria a metade da esfera? Será que não é a metade da formula? Bom essa é a lógica, a formula matemática é representação real do elemento então a metade da esfera é a metade da formula; então eu tenho que colocar a formula dividida por 2 lá no Python? Não precisa, coloca logo dividido por 2 lá no script para evitar bagunça, a única coisa que muda na formula da esfera dividida por 2 é que em vez de um 4 vai ter um 2 ali. Mas ainda não acabou, na formula do cilindro está dizendo que x é igual à altura do cilindro, perceba que a altura não é a mesma coisa que a altura do combustível, por que? Simplesmente

por que o x é a altura somente do cilindro nada mais. Pegue a altura do combustível e subtraia pelo raio do bojo e tudo ocorrerá bem.

E SE o combustível ficar quase ou totalmente cheio acima do cilindro? Acho que você já sabe o que acontece aqui, teremos que juntar todas as formulas para pegar o cilindro e os bojos, só que dessa vez tem uma pequena diferença, não é somente somar tudo teremos que subtrair também: a formula da calota; por que? Por que a formula que irá pegar os dois bojos é a formula da esfera e a formula do cilindro pega o meio, mas como iremos demarcar onde está à altura do combustível? Perceba-a que a calota é só uma parte de cima do bojo e como o combustível está quase no tanque todo (senão no tanque todo) a calota irá demarcar onde está o combustível e iremos subtrair pelo volume total que é a soma da formula do cilindro mais a da esfera (dessa vez a formula da esfera é completa mesmo já que pegamos os dois bojos). A calota deve ter sua altura de acordo com a posição do combustível, ou seja, na parte da formula da calota onde fica a altura deve-se colocar onde o combustível está que você pode perceber que é simplesmente a altura total menos a altura do combustível. Não esqueça que a altura do cilindro agora muda de novo, agora você deve subtrair os dois bojos ou seja subtrair duas vezes o raio pela altura total do tanque.

OU SE os dados inseridos não tiverem nada ver com tudo acima? Bom essa é a parte do else em que você deve colocar para aparecer a mensagem -1. E no final de tudo não esqueça que a questão pediu para você arredondar o resultado em 3 casas decimais e é claro que você irá usar a função `round`.

Segundo passo: Formulado o script

Dê entrada na biblioteca math, escreva as três entradas requeridas na questão: Altura do tanque, Raio dos bojos e Altura do combustível. A primeira condição é o fato do valor inserido ser inválido, ou seja, negativo para as três entradas. A segunda é quando o combustível está abaixo do cilindro, ou seja, no bojo inferior que pelos operadores condicionais pode ser representado assim: $h < r$; pois o combustível está entre o bojo inferior, no seu mini script coloque a formula da calota esférica. A terceira condição é quando o combustível está no meio do cilindro ou o cilindro cheio, abaixo do bojo superior, a representação dessa condição através dos operadores condicionais pode ser descrita a altura está entre o cilindro, ou seja, a altura total menos o raio do bojo: $h < H - r$, no seu mini script coloque a formula da esfera dividida pela metade mais a formula do cilindro. A quarta condição é do tanque quase ou totalmente cheio que pode ser representado por: $h \leq H$, no seu mini script coloca-se a formula da esfera mais o cilindro menos a calota esférica.

The screenshot shows the Spyder IDE interface with the following details:

- Title Bar:** Spyder (Python 3.4)
- Menu Bar:** File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, Help
- Toolbar:** Includes icons for file operations like Open, Save, Run, Stop, and Help.
- Editor Area:** Shows the code for `Lab03_05.py`. The code calculates the volume of a cylinder given its height (`H`) and radius (`r`). It handles various cases including negative dimensions and zero values. The code uses mathematical formulas for different sections of the cylinder based on the relative positions of the top and bottom surfaces.

```

1 # -*- coding: utf-8 -*-
2 """
3 22/03/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from math import *
10
11 # Calculo do volume
12 H = float(input('Digite a altura: '))
13 r = float(input('Digite o raio: '))
14 h = float(input('Digite o nível de combustível: '))
15
16 if (h < 0 or H < 0 or r < 0):
17     vol = -1
18 elif (h < r):
19     vol = (1./3) * pi * h**2 * (3 * r - h)
20 elif (h < H - r):
21     vol = (2./3) * pi * r**3 + pi * r**2 * (h - r)
22 elif (h <= H):
23     vol = (4./3) * pi * r**3 + pi * r**2 * (H - 2 * r) - (1/3) * pi * (H - h)**2 * (3 * r - H + h)
24 else:
25     vol = -1
26
27 print(round(vol, 3))
28

```

Figura 87: Questão 5C Segundo Passo

Fonte: ICOMP, UFAM

The screenshot shows the Spyder IDE Console window with the following output:

```

Console
Python 1

>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab03_05.py', wdir='C:/Users/User/Documents/Helder Guerreiros/Meus Programas')
Digite a altura: 10
Digite o raio: 5
Digite o nível de combustível: 10
523.599
>>>

```

Figura 88: Resultado a) questão 5C

Fonte: Autoria própria

The screenshot shows the Spyder IDE Console window with the following output:

```

Console
Python 1

>>> runfile('C:/Users/User/Documents/Helder Guerreiros/Meus Programas/Lab03_05.py', wdir='C:/Users/User/Documents/Helder Guerreiros/Meus Programas')
Digite a altura: 5
Digite o raio: 10
Digite o nível de combustível: 13
-1
>>>

```

Figura 89: Resultado b) questão 5C

Fonte: Autoria própria

Questão 6 - Sobreposição de Intervalos

Sejam dois intervalos numéricos sobre a reta inteira, representados por quatro números, $[a, b]$ e $[c, d]$. Escreva um programa que verifique se existe sobreposição (pelo menos um ponto em comum) entre os intervalos. Se houver sobreposição, a saída será 1. Caso contrário, 0.

Note que:

1. *Seu programa deve verificar se os intervalos são válidos, ou seja, se $b > a$ e $d > c$. Caso contrário, a saída do programa deverá ser -1.*
2. *Seu programa não deve partir de nenhum pressuposto sobre a posição relativa entre os intervalos $[a, b]$ e $[c, d]$. Ou seja, eles podem vir ante ou depois um do outro.*

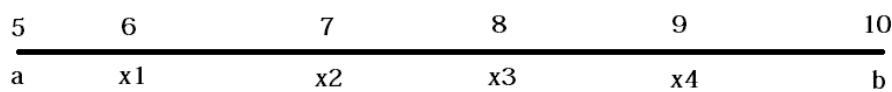
DICA: Tente resolver um problema mais simples, que é determinar se um ponto pertence ou não a um intervalo numérico. Depois generalize. Alguns dos operadores booleanos (and, or e not) serão úteis.

Primeiro passo: Interpretação da questão

Essa é um dos tipos questões inúteis que você faz uma vez na vida e depois fica se perguntando se isso vai ajudar em alguma coisa na sua vida, essa questão não tem nenhuma novidade em questão ao uso do Python e ainda por cima tem um assunto que você deve nunca ter visto (ou você não lembra se já viu) em sua vida, tudo isso sabe para que? Isso mesmo para lhe ferrar... mesmo não querendo vou lhe explicar como resolver esse quebra cabeça que no final vai ser jogado fora como lixo.

Esse assunto não é tão complicado assim como você pensa, vamos pensar esses intervalos como se fossem retas, uma reta que começa em "a" e termina em "b" e outra que começa em "c" e termina em "d". Você que é bom em português deve saber que a palavra sobreposição quer dizer que uma coisa está em cima de outra, nesse caso: uma reta em cima de outra, e para isso acontecer pelo menos um ponto de uma das retas deve estar em sobreposição.

Você percebe que a questão te dá uma dica dizendo que os intervalos devem ser válidos, ou seja, $b > a$ e $d > c$, mas como assim? Lembre-se que estamos considerando esses intervalos como retas, então na primeira reta "a" é o primeiro ponto e "b" é o último, é óbvio que o último ponto deve ser maior que o primeiro, por que se lembre que essas letras são na verdade números, então afirmar que $b < a$ é verdade é a mesma coisa que afirmar que $10 < 5$, sacou?



Então você já sabe de uma coisa, a primeira condição é que os números colocados como intervalos sejam válidos. Sim isso está certo, agora temos que ver a próxima condição que é saber se os intervalos têm alguma sobreposição entre eles, como ver isso? Vamos pensar na onda dos números que é mais fácil, veja que estamos numa reta e isso quer dizer

que qualquer número entre "a" e "b" ou "d" e "c" está em sobreposição, isso quer dizer que para um está em cima de outro pelo menos um ponto da reta deve estar entre o intervalo de outra ou exatamente em cima. Vamos tentar passar isso para operadores condicionais? Vamos usar o intervalo (reta) $[a,b]$ como referência, digamos que o outro intervalo (reta) $[c,d]$ tenha um de seus pontos cujo valor seja o mesmo ou menor que um dos valores do intervalo $[a,b]$, isso quer dizer que essa reta está passando exatamente onde a outra está, mas quais pontos que temos como referência para dizer que a reta está passando antes ou depois da outra? Nós só temos os extremos dos dois intervalos, então vamos pensar, qualquer ponto depois de "a" está dentro do intervalo, se o ponto estiver antes de "a" com certeza está fora, por que se esse "a" fosse um 5 e passasse um ponto com valor 4, certamente o 4 não está dentro do intervalo por que o 5 é o início do intervalo e tudo que vier antes dele não está dentro dele, esse mesmo pensamento vale o mesmo para o "b" só que dessa vez qualquer ponto depois dele está fora do intervalo por que ele é o último. Todo o pensamento que passei foi usando o intervalo $[a,b]$, mas a mesma coisa acontece no intervalo $[c,d]$.

Você já deve ter sacado que para fazer a condição da sobreposição de intervalos você tem que fazer o ponto inicial de um ser maior ou igual ao inicial e menor ou igual ao final do outro (se for menor que o inicial não faz parte e se for maior que o final também) ou o ponto final de um deve ser maior ou igual ao inicial e menor ou igual ao final.

$$c \geq a, c \leq b$$

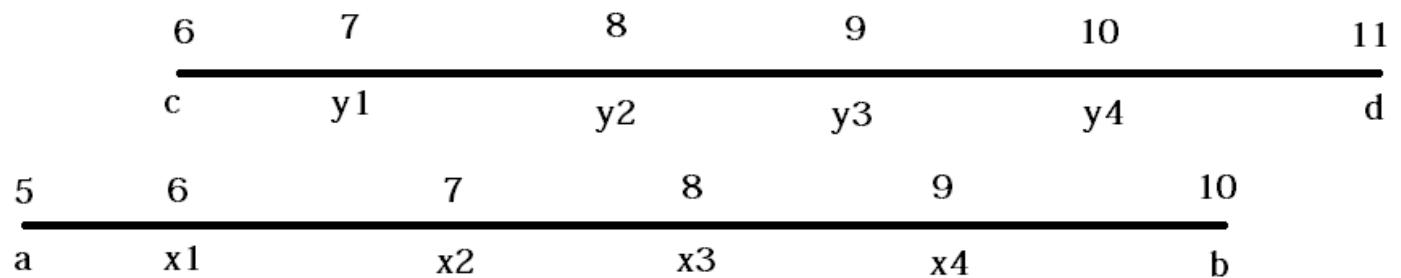


Figura 90: Exemplo a) de intervalos

Fonte: Autoria própria

$$d \geq a, d \leq b$$

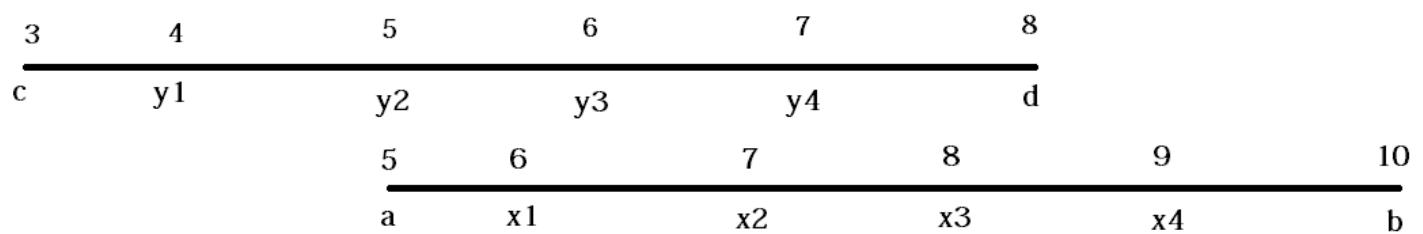


Figura 91: Exemplo b) de intervalos

Fonte: Autoria própria

Segundo passo: Formulado o script

Claramente você já sabe que deverá colocar quatro entradas para os intervalos no script, a primeira condição é que os dados inseridos sejam válidos, ou seja, que nos dois intervalos o último ponto não seja menor que o primeiro ($b \geq a$ and $d \geq c$) usamos **and** por que é necessário que essa regra seja estabelecida aos dois intervalos, dentro dessa condição há outra condição e como ela é a última então iremos usar o **if** normalmente aqui, e essa condição é saber se os intervalos estão sobrepostos independente quem ou o quê, nós já refletimos aqui que para haver uma sobreposição um ponto de um dos intervalos deve estar dentro do outro chegando ao resultado que a extremidade inicial de um (nesse caso vamos usar "c") deve ser maior ou igual à extremidade inicial E menor ou igual à extremidade final de outra OU a extremidade final de um (nesse caso vamos usar "d") deve ser maior ou igual à extremidade inicial E menor ou igual à extremidade final de outra. Não esqueça que para isso você deve usar os operadores condicionais **and** e **or**.

O mini script da segunda condição é mostrar o resultado "1" se houver sobreposição OU SE não houver sobreposição deverá mostrar o resultado "0". Na primeira condição caso os dados sejam inválidos o **else** deverá ter um min script que mostre o resultado "-1", como sempre avisando que você pode usar o **print** toda vez que precisar imprimir um resultado ou use uma variável e coloque o **print** lá no final.

```
File Edit Search Source Run Debug Consoles Tools View
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas
Lab03_06.py

1 # -*- coding: utf-8 -*-
2 """
3 23/03/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 # Intervalo 1
10 a = int(input("a: "))
11 b = int(input("b: "))
12
13 # Intervalo 2
14 c = int(input("c: "))
15 d = int(input("d: "))
16
17 if b >= a and d >= c:
18     if (c >= a and c <= b) or (d >= a and d <= b):
19         print(1)
20     else:
21         print(0)
22 else:
23     print(-1)
24
```

Figura 92: Questão 6C Segundo Passo

Fonte: ICOMP, UFAM

```
Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab03_06.py', wdir='C:/Users/User')
a: 5
b: 10
c: 6
d: 11
1
>>>
```

Figura 93: Resultado a) questão 6C

Fonte: Autoria própria

```
Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab03_06.py', wdir='C:/Users/User')
a: 10
b: 5
c: 6
d: 8
-1
>>>
```

Figura 94: Resultado a) questão 6C

Fonte: Autoria própria

3.5 Avaliações

Aqui teremos algumas avaliações para entender mais um pouco o assunto estudado neste primeiro capítulo. Aconselho você tentar fazer sozinho antes de ler a resolução.

Avaliação A – A3

Avaliação Parcial 3a (A3) – Estrutura Condisional Aninhada

Informações

Questões

Notas

Questão 1

Intervalos

Em química, a acidez de uma solução aquosa é medida pelo pH, em uma escala que varia entre 0 e 14. Uma solução com pH igual a 7 é dita neutra. Uma solução com pH maior que 7 é básica, e com pH menor que 7 é ácida. Escreva um script que leia a medida do pH de uma solução e imprima o tipo dela, conforme tabela abaixo.

pH	Mensagem
< 0	invalido
[0; 7[acido
7	neutro
]7; 14]	basico
> 14	invalido

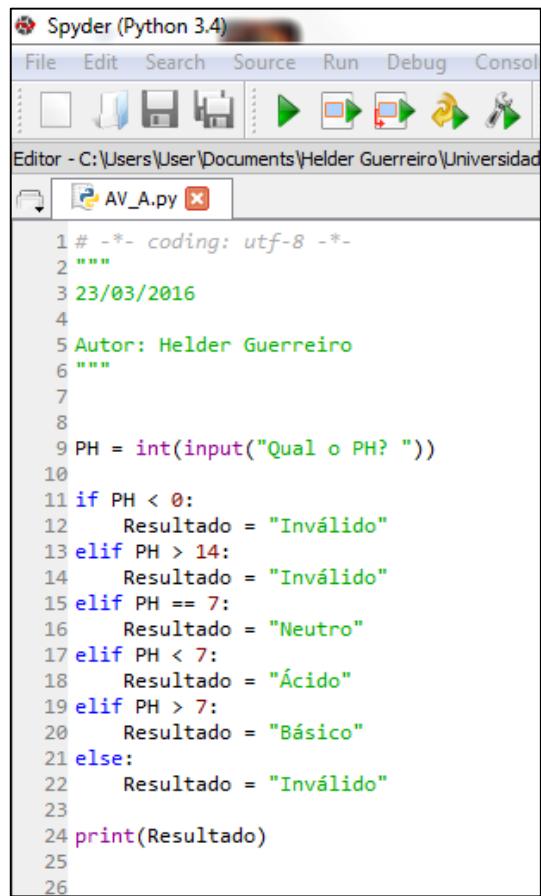
Dicas:

1. Teste o script no Spyder antes de submetê-lo ao Code Bench.
2. Verifique se você está submetendo o código correto à questão correspondente desta Avaliação Parcial.
3. Por exemplo, para a entrada **14**, a saída deve ser "**basico**" (sem aspas). Para a entrada **16**, a saída deve ser "**invalido**" (sem aspas).

Essa questão é bem simples, leia com calma que você já mata toda a charada. Obviamente temos várias condições e todas as condições do script estão listadas na tabela acima, ou seja, um `if`, três `elif's` e um `else`, simplesmente por que a entrada a ser analisada é o PH, então temos a análise: SE o PH for menor que zero temos o resultado inválido, E SE o PH for maior que 14 então temos o resultado inválido, E SE o PH for 7 então temos o resultado neutro, E SE o PH for menor que 7 então temos o resultado ácido, E SE o PH for maior que 7 então temos o resultado básico.

Acho que você já deve ter percebido como o script vai ficar, então simplesmente faça a lista de `if`, `elif's` e o `else` e pronto coloque o `print` no final e acabou-se. Você deve estar se questionando por que eu não falei do "OU SE" (`else`) lá em cima, bem nessa parte você pode simplesmente fazer o `else` e colocar no seu script o resultado inválido e pronto,

isso por que todas as condições necessárias já foram feitas e o que vier que não seja nada disso será com certeza inválido.



The screenshot shows the Spyder Python 3.4 IDE interface. The title bar says "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, and Console. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidad". A tab labeled "AV_A.py" is active. The code in the editor is:

```
1 # -*- coding: utf-8 -*-
2 """
3 23/03/2016
4
5 Autor: Helder Guerreiro
6 """
7
8
9 PH = int(input("Qual o PH? "))
10
11 if PH < 0:
12     Resultado = "Inválido"
13 elif PH > 14:
14     Resultado = "Inválido"
15 elif PH == 7:
16     Resultado = "Neutro"
17 elif PH < 7:
18     Resultado = "Ácido"
19 elif PH > 7:
20     Resultado = "Básico"
21 else:
22     Resultado = "Inválido"
23
24 print(Resultado)
25
26
```

Figura 95: Resposta Avaliação A 3A

Fonte: Autoria própria

Avaliação B - A3

Avaliação Parcial 3b (A3) – Estrutura Condicional Aninhada

Informações

Questões

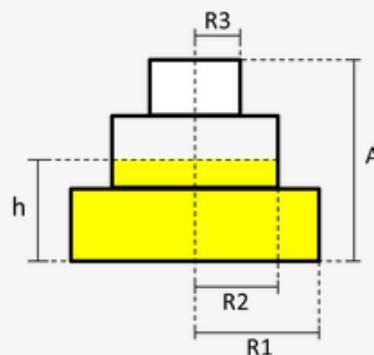
Notas

Questão 1

Volume de água

Uma caixa d'água de altura A é composta por três cilindros concêntricos sobrepostos e de mesma altura, mas raios diferentes (R_1 , R_2 e R_3), conforme mostra a figura abaixo em perfil. O nível de água no reservatório é dado por h . Escreva um script que leia como entrada os valores de h , de A e de R_1 , R_2 e R_3 , nessa ordem, medidos em centímetros, e informe o volume em litros do líquido, com até quatro casas decimais de precisão.

Lembre-se que o usuário pode informar qualquer valor para o nível de água h , inclusive valores negativos ou maiores que a altura da caixa d'água. Cabe ao seu script verificar a validade dos dados de entrada. Se valores inválidos forem inseridos, a saída do script deve ser **-1**.



Dicas:

1. O volume de 1000cm^3 corresponde a 1 litro.
2. O volume V de um cilindro de raio X e altura H é dado por $V = \pi X^2 H$.
3. A ordem de leitura das entradas deve ser $h A R_1 R_2 R_3$. Por exemplo, para a entrada **100 600 300 200 100**, a saída deve ser **28274.3339**. Já para a entrada **100 600 100 300 200**, a saída deve ser **-1**, pois o raio do cilindro maior $R_1=100\text{cm}$ é menor que o raio do cilindro intermediário $R_2=300\text{cm}$.

Bom essa questão é um pouco complicada para quem não está muito bem no Python, eu me lembro que quando eu fiz essa prova no dia eu não consegui fazê-la e então eu fiquei somente com a nota da primeira prova (nem colar deu certo), claro que hoje eu sei fazer essa prova e eu percebi que não consegui fazê-la por causa da falta de treino e da falta de familiaridade com os problemas do Python. Mas nada de medo, vamos fazer essa prova com calma.

Primeiramente no seu script vá logo adiantando as coisas fazendo as entradas que nesse caso são 5 entradas. Antes de você ir com sede nas condicionais (já que você acha que já sabe tudo) pare um pouco para pensar, quando a água estiver pela metade ou quase cheia dentro da caixa, como você irá diferenciar aonde está cada cilindro? Com os raios você pode diferenciar os tamanhos, mas não as suas respectivas alturas. A pista já foi dada logo no início da questão, os três cilindros têm as mesmas alturas, isso quer dizer que se você dividir "A" (altura do tanque) por 3 você irá encontrar a

altura de cada cilindro, com o resultado de $A/3$ você já sabe que essa é a altura de um cilindro e como nós estamos falando do nível da água então o primeiro cilindro será o de baixo, claramente a altura do chão até o topo do primeiro cilindro é o resultado de $A/3$, mas como fica a altura do chão até o topo do segundo cilindro? Bom nesse caso já teremos dois cilindros, um em cima do outro, ou seja, duas vezes o tamanho de um cilindro que nesse caso é $A/3$, isso significa que a altura do segundo cilindro até a base é $A/3 * 2$; agora chegando no topo claramente o último cilindro compõe todo o tanque fazendo com que a altura da base até o topo do último cilindro seja a mesma altura do tanque. Agora que nós descobrimos as alturas de cada cilindro em relação a base, coloque essas informações em forma de variáveis como por exemplo: RA, RB e RC sendo o RA o primeiro (o maior) o RB o segundo (o intermediário) e o RC o terceiro (o menor).

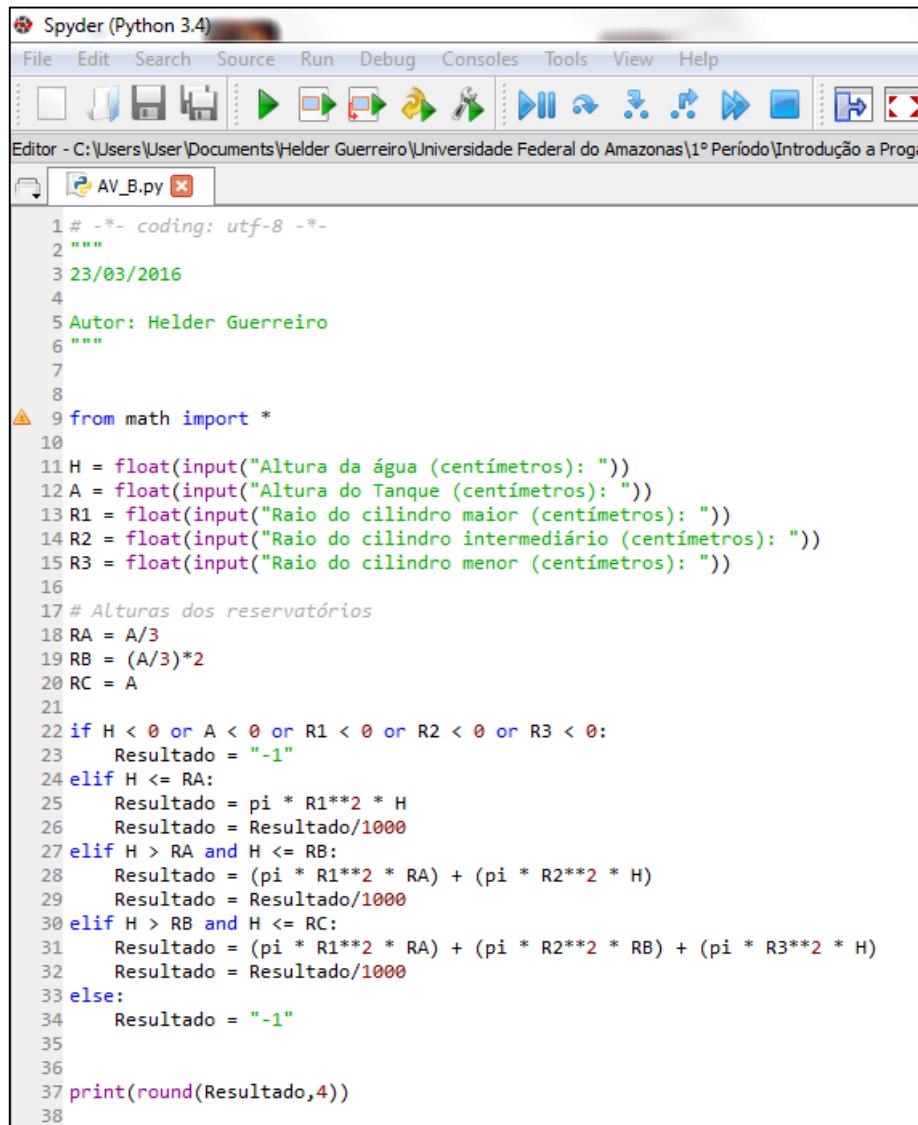
Agora sim chegamos nas condicionais, logo de cara você já sabe que tem que fazer uma condicional para os valores inválidos que dessa vez estamos falando dos negativos, então o if será o encarregado disso. Você percebeu que o if vai ter que colocar as condições para todas as 5 entradas não é mesmo? Bem isso vai ficar um pouco grande, mas essa condição será escrita como qualquer outra usando o operador `or`. Você já deve saber que toda vez que tiver mais de duas condições nós iremos usar o `elif`, não é mesmo? Então o próximo passo é usar o elif, mas qual será a condição? Bem lembre-se que nós diferenciamos cada cilindro pela sua altura em relação a base, fizemos isso justamente por que quando a água está no primeiro ou no segundo ou no terceiro o cálculo irá mudar totalmente, então as condições que seguirão a frente serão basicamente o fato da água está no ou no primeiro, ou no segundo ou no terceiro cilindro.

A questão já te deu uma dica lá no enunciado, vamos usar a formula do volume do cilindro. No primeiro elif estaremos falando da condição em que a água está no primeiro cilindro, então o volume será baseado somente na formula do volume de um cilindro. Temos que escrever a condição usando os operadores condicionais, o nível da água é representado por "H" e é nessa parte que iremos precisar das alturas de cada cilindro em relação a base, o nível da água deve estar abaixo ou exatamente igual à altura do primeiro cilindro (RA). No seu mine script estará a formula do volume do cilindro, aqui nós vamos usar o Pi então lembre-se de ativar o módulo math, o raio (R1) será elevado ao quadrado e a altura do cilindro será a altura da água, Depois do Python calcular a formula do volume o resultado estará pronto, mas só uma coisa! Não se esqueça do mínimo detalhe em que a resposta deve ser em litros! Mas como assim em litros? O raio está em centímetros que depois de elevado ao quadrado fica centímetros quadrados e depois ele se multiplica com o centímetro da altura do cilindro e por fica centímetros cúbicos, o enunciado deu a dica de como converter centímetros cúbicos para litros, basta você dividir por 1000, então antes de imprimir o resultado com o `print` fala a divisão da resposta por 1000.

A próxima condição é a de que o nível da água está no segundo cilindro, ou seja, no meio do tanque. A condição deve ser escrita com mais cuidado aqui, a água deve estar acima do primeiro cilindro e também abaixo ou exatamente em cima do segundo cilindro. O seu mine script também terá a formula do volume e a conversão do resultado, a formula do volume não é diferente da primeira condição, dessa vez você irá somar o volume do cilindro de baixo (só que dessa vez a altura dele não é mais "H" agora será a sua própria altura "RA") com o volume do cilindro do meio, a altura do cilindro do meio será o "H".

A última condição é exatamente igual as outras, a sua condição é escrita dizendo que a água deve estar acima do cilindro do meio e abaixo ou exatamente em cima do último cilindro. O cálculo do volume será basicamente a soma do primeiro cilindro (sua própria altura) com o segundo cilindro (sua própria altura) com a o terceiro cilindro (altura da água) e não esqueça de converter a água no final do mine script.

Por fim o `else` mostrando o resultado “-1” para qualquer outro valor sem contexto com qualquer uma das condições acima. E no final não esqueça de colocar o `round` nos resultados finais com o limite de 4 casas decimais.



```
Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View Help
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Introdução a Prog
AV_B.py
1 # -*- coding: utf-8 -*-
2 """
3 23/03/2016
4
5 Autor: Helder Guerreiro
6 """
7
8
9 from math import *
10
11 H = float(input("Altura da água (centímetros): "))
12 A = float(input("Altura do Tanque (centímetros): "))
13 R1 = float(input("Raio do cilindro maior (centímetros): "))
14 R2 = float(input("Raio do cilindro intermediário (centímetros): "))
15 R3 = float(input("Raio do cilindro menor (centímetros): "))
16
17 # Alturas dos reservatórios
18 RA = A/3
19 RB = (A/3)**2
20 RC = A
21
22 if H < 0 or A < 0 or R1 < 0 or R2 < 0 or R3 < 0:
23     Resultado = "-1"
24 elif H <= RA:
25     Resultado = pi * R1**2 * H
26     Resultado = Resultado/1000
27 elif H > RA and H <= RB:
28     Resultado = (pi * R1**2 * RA) + (pi * R2**2 * H)
29     Resultado = Resultado/1000
30 elif H > RB and H <= RC:
31     Resultado = (pi * R1**2 * RA) + (pi * R2**2 * RB) + (pi * R3**2 * H)
32     Resultado = Resultado/1000
33 else:
34     Resultado = "-1"
35
36
37 print(round(Resultado,4))
38
```

Figura 96: Resposta Avaliação B 3A

Fonte: Autoria própria

4. Sua Vida um Inferno – Repetição por Condição

Chegou a hora da sua cruz ficar mais pesada, a repetição por condição é uma habilidade do Python que devemos reconhecer que realmente é muito boa, pode parecer difícil assim olhando de primeira todas as grandes estruturas que estamos fazendo e que ficam cada vez maiores, mas lembre-se de uma coisa: tudo que estamos aprendendo aqui é na calma e as grandes estruturas que fazemos você já sabe como fazer, a repetição por condição é só uma coisa a mais para ser colocada no seu conhecimento sobre o Python.

Eu lhe apresento o comando condicional `while`, ele é parecido com o `if`, `elif` e o `else`, a sua diferença é que a sua condição é voltada para a repetição do seu mine script, isso quer dizer que: o `while` tem sua condição igual ao `if` e também tem o seu próprio mine script abaixo dele. A palavra `while` em português significa “enquanto”, ou seja, ENQUANTO uma certa condição ainda está valendo o mine script continuará rodando.

4.1 Variável Contadora e Variável Acumuladora

Essas são as últimas coisas que você deve saber sobre o `while`, essas duas variáveis você terá de aprender a usá-las para que você consiga usar com êxito toda a habilidade do `while`.

4.1.1 Variável Contadora

A variável contadora serve para o `while` saber quantas vezes ele está repetindo o seu mine script, essa variável também será a base de sua condição, isso quer dizer que quando você for escrever a condição do `while` deve-se colocar a variável contadora como referência. Mas como é que é essa variável? Ela é simplesmente uma variável que você coloca lá em cima no script e iguala a um certo número, no mine script do `while` ela aparece novamente só que dessa vez ela será somada com o número 1, o `while` irá analisar a variável acumuladora novamente e se mesmo depois de somada a variável acumuladora continuar na condição dele o mine script executará de novo e a variável somada de novo e assim vai... Veja esse exemplo abaixo para entender melhor.

```
x = 1
while (x <= 5):
    print(x)
    x = x + 1
```

O “`x`” é a variável acumuladora que começa com o número 1, a condição do `while` é que `x` seja menor ou igual a 5, então temos ENQUANTO `x <= 5` o mine script continuará executando. O mine script irá apresentar o valor da variável acumuladora e depois será somada a 1. Como a variável valia 1, agora ela vale 2 e ainda assim está dentro da condição. E assim vai até o 6.

Figura 97: Variável Contadora

Fonte: ICOMP, UFAM

Todas as condições do `while` sempre serão nesse estilo, maior, maior igual, menor, menor igual etc.... Veja que essa variável está aí só para ajudar e ela faz jus ao seu nome de contadora.

4.1.2 Variável Acumuladora

A variável acumuladora serve para ajudar quem está programa a chegar a algum resultado que queira, digamos que você esteja querendo saber a média da nota de uma turma de 40 alunos, deveremos ter uma variável acumuladora para permitir o while trabalhando até os 40 alunos, a função da variável acumuladora é justamente pegar as notas dos alunos e somar a cada vez que uma nota for colocada.

```
# Pede o numero de alunos
num_alunos = int(input("Digite o no. de alunos: "))

i = 1      # Variavel contadora
soma = 0    # Variavel acumuladora

while (i <= num_alunos):
    print("Digite a nota do aluno", i, ":")
    nota = float(input())    # Le nota
    soma = soma + nota       # Acumula nota
    i = i + 1                 # Atualiza contador

print("Media:", round(soma/num_alunos, 2))
```

Esse script muito simples serve de exemplo, o número de alunos é colocado e a variável contadora irá fazer o while repetir até o último aluno, e enquanto o mine script se repete a nota será requerida e cada vez que a nota for inserida a variável acumuladora irá somar as notas e no final o print irá mostrar a média da sala.

Figura 97: Variável Acumuladora

Fonte: ICOMP, UFAM

4.2 Questões Resolvidas

Estas questões foram desenvolvidas pela ICOMP um instituto da UFAM e os algoritmos (scripts) apresentados aqui pertencem ao ICOMP.

Neste quarto capítulo não vou mais mostrar o segundo passo que é o "Formulando o script" (Com exceção da primeira questão), somente a interpretação da questão será apresentada.

Questão 1 – Testando um Script

Escreva um script que leia repetidamente um número inteiro digitado e exiba na tela se ele é par ou ímpar, até que o valor -1 seja lido, quando o programa deve ser encerrado.

Primeiro passo: Interpretação da questão

O enunciado é bem pequeno, mas tem um pouquinho de trabalho, esse script pegue um pouco do seu conhecimento lá do início do seu estudo do Python que é a verificação de um número se ele é par ou ímpar. Nesse caso teremos de usar o `if` para saber SE ele é par OU SE ele é ímpar. Mas lembre-se de uma coisa, o processo para saber se um número é par ou ímpar será contínuo, ou seja, ENQUANTO você não inserir o número -1 o script continuará perguntando qual o próximo número, é claro que devemos usar o `while` antes de usar o `if`.

Preste atenção em uma coisa, a função que irá perguntar qual o número não deve ser colocado somente depois do comando `while` ter começado, ela deve ser colocada também antes do comando `começar`, por que a pergunta será a referência da condição do `while` por que simplesmente ENQUANTO o número -1 não for inserido o programa continuará funcionando. A variável que vai perguntar deve ser colocada antes para que o `while` tenha uma referência e depois deve ser colocada também para que a variável venha ser atualizada. Perceba que essa variável está se comportando como uma variável contadora, só que ela não está sendo somada e sim inserida.

Refrescando a sua memória um número é par SE ele é divisível por 2 OU SE ele é ímpar mesmo. Lembre-se do nosso amigo resto da divisão (%) não esqueça as coisas passadas, pois um dia você acaba precisando delas, quando você usa o resto da divisão ele mostra somente aquilo que sobrou de uma divisão, se você divide 10 por 2 você consegue tranquilamente dividi-lo só usando o 2 e claramente o resto é 0, mas se você divide 15 por 2 você não consegue fazer essa divisão sem quebrar o número, então nesse caso essa divisão terá um resto.

Segundo passo: Formulado o script

O início começará com a variável contadora, que nesse caso não será somada e sim será requerida pelo usuário, crie a variável com o `input` e o `int`; logo depois da variável pronta hora de criar o comando condicional com o `while`, a sua condição é que ENQUANTO o número -1 não for inserido na variável contadora o `while` continuará repetindo o mine script,

mas temos que escrever essa condição com os operadores condicionais, podemos usar o operador "diferente" (\neq) na condição mostrando que para o while continuar funcionando a variável contadora (vamos chama-la de "num") deve ser diferente de -1. O mine script é a condição do número ser par que será feita pelo if, para escrever essa condição use o operador % mostrando que se o resultado do resto da divisão for 0 então temos um par, o mine script do if deverá mostrar a mensagem Par. Não precisamos criar outra condição para o ímpar, se o número não for par então claramente ele é ímpar, então usamos o else e no seu mine script deve estar a mensagem Ímpar. No final de tudo não esqueça de colocar a variável "num" de novo para que ela seja atualizada e o usuário insira outro número.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles To
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal
Lab04_01.py X
1 # -*- coding: utf-8 -*-
2 """
3 24/03/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 # Primeiro input
10 num = int(input("Digite um numero: "))
11
12 # Laco de repeticao
13 while (num != -1):
14     # Verificacao de divisibilidade por 2
15     if (num % 2 == 0):
16         mensagem = "par"
17     else:
18         mensagem = "ímpar"
19
20     print(mensagem)
21
22     # Inputs seguintes
23     num = int(input("Digite um numero: "))
24

```

Figura 99: Questão 10 Segundo Passo

Fonte: ICOMP, UFAM

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal/Meus Programas/Lab04_01.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Universidade Federal/Meus Programas')
Digite um numero: 5
ímpar
Digite um numero: 2
par
Digite um numero: 12
par
Digite um numero: 3
ímpar
Digite um numero: 16
par
Digite um numero: -1
>>>

```

Figura 100: Resultado questão 10

Fonte: Autoria própria

Questão 2 – Soma de uma coleção de números

Escreva um programa que calcula a soma de uma coleção de valores digitados pelo usuário. O usuário irá inserir -1 para indicar que não há mais valores a serem fornecidos.

DICAS:

1. Utilize uma variável acumuladora para guardar o valor da soma.
2. Como o valor -1 marca o final de digitação da sequência, seu código não pode incluí-lo na soma.
3. Para a sequência de entrada 1 2 3 4 -1, a saída deverá ser 10.

Primeiro passo: Interpretação da questão

Essa questão é bem simples também, não tem muito o que discutir aqui, só vamos entendê-la melhor para fazermos bonito no script.

Você já deve saber que esse script começa com uma variável, essa variável é a variável que irá requerer um número do usuário (também vamos chama-la de "num"), perceba que ela é a contadora e de novo o -1 será usado para terminar a repetição do `while`. Dessa vez iremos usar a variável acumuladora que vamos chama-la de "soma" que deverá começar com o valor 0, por que se colocarmos qualquer outro número o resultado será interferido por que essa variável será somada pelo número que for inserido pelo usuário e depois de ser somada a variável soma passa a ter outro valor e esse valor é o resultado da soma, cada vez mais que o `mine script` do `while` for repetido o valor da variável será somado com o número inserido.

O próximo passo é a condição de repetição comandada pelo `while`, a condição é claramente que a "num" seja diferente de -1 para que o `while` continue repetindo seu `mine script`. O `mine script` do `while` é simplesmente acumular em forma de soma cada número inserido, essa soma deve ser feita usando a variável acumuladora, a variável soma será igual a ela mesma somada a variável num, por que a variável soma deve ser igual a ela mesma mais a outra variável? Não poderia ser outra variável somando os dois? Não, por que quando a variável soma é igualada a outro valor (que nesse caso é a soma dela mesma com a variável num) a variável soma muda de valor, antes era 0 agora é o resultado da soma e assim ela vai acumulando o resultado. No final do `mine script` do `while` deve estar novamente a variável num perguntando outro número.

No final nós temos que mostrar o resultado final que nesse caso é a soma de todos os números inseridos, não coloque o `print` no `mine script` do `while`, o `print` deve estar fora do `while`. Por que? Isso por que o `print` mostrará o resultado da soma, se você coloca no caminho do `while` toda vez que o `while` repetir seu `mine script` ele irá acionar o `print`, ou seja, o `print` vai mostrar a soma cada vez que o usuário inserir um número, não que seja errado isso acontecer, mas não é isso que a questão pede, temos de mostrar o resultado final e não o resultado somado aos poucos.

The screenshot shows the Spyder Python 3.4 IDE interface. The title bar says "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Ama..." and shows the Python code for Question 20. The code is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 24/03/2016
4
5 Autor: Helder Guerreiro
6
7 """
8 # Primeiro input
9 num = int(input("Digite um numero: "))
10
11 # Zera variavel acumuladora
12 soma = 0
13
14 # Laco de repeticao
15 while (num != -1):
16     # Acumula valor
17     soma = soma + num
18
19     # Inputs seguintes
20     num = int(input("Digite outro numero: "))
21
22 # Imprime resultado
23 print(soma)
24
```

Figura 101: Questão 20 Primeiro Passo

Fonte: ICOMP, UFAM

The screenshot shows the Python 1 console window. The command `runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab04_02.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Meus Programas')` is run. The output shows the user inputting numbers 1 through 5, followed by -1 to exit the loop, and the final sum being printed.

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab04_02.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Meus Programas')
Digite um numero: 1
Digite outro numero: 2
Digite outro numero: 3
Digite outro numero: 4
Digite outro numero: 5
Digite outro numero: -1
15
>>>
```

Figura 102: Resultado questão 20

Fonte: Autoria própria

Questão 3 - Somas de várias coleções de números

Escreva um programa que calcula a soma de várias coleções de valores digitados pelo usuário. O usuário irá inserir -1 para indicar o fim de uma coleção e o início da próxima, e 666 para indicar que não há mais valores a serem fornecidos.

DICAS:

1. Use o comando while aninhado em outro comando while. O laço mais externo verifica se o valor 666 foi digitado e inicia a variável acumuladora antes de uma nova soma. O mais interno realiza a soma de uma coleção de números enquanto o valor -1 não foi inserido.

2. Como o valor -1 marca o final de digitação de cada sequência, seu código não pode incluí-lo na soma.

3. Para a sequência de entrada 123 -11234 -112345 -1666, a saída deverá ser 61015.

Primeiro passo: Interpretação da questão

Agora eu tenho uma novidade para você que apostou que você não irá ficar tão feliz assim: podemos colocar um **while** dentro de outro **while**. Aí você me pergunta: por que eu iria querer fazer isso? Bom você já vai saber.

Essa questão ela quer que se faça a soma de várias coleções de números, como assim? Você comece o seu script e ele pede os números, você coloca tanto e aperta -1, e depois mostra o resultado dessa soma de número que você inseriu. E ele pede mais número mesmo depois do -1 só que dessa vez é outra sequência, e quando você apertar -1 de novo ele mostrar a soma dessa sequência e partirá para outra até que você digite o 666 (é digamos que o pessoal do ICOMP são bem engraçadinhos), só que aqui vamos mudar esse número para 0 em vez de 666 (quero ninguém invocando o mal aqui).

Vou dar uma dica valiosa, o script da questão anterior é a metade do script desta questão então pegue o script da questão anterior, vamos fazer as modificações nela. A diferença aqui é que antes da variável soma temos um **while**, e aquele **while** que fica um pouco mais embaixo continua lá. Não esqueça que como você colocou um **while** antes da variável soma você deve organizar tudo que está embaixo do **while** como se fosse o **mine script** dele, ou seja, dê aquele espaço avançando para ficar debaixo da ponta do primeiro **while**, até o segundo **while** deverá recuado também, não esqueça que o **mine script** do segundo **while** deve ficar alinhado com o seu próprio **while** não com o primeiro. Resumindo o que eu disse acima: Alinhe todo o script da questão anterior (exceção da variável num) com o primeiro **while** que fica acima da variável soma, só não misture o **mine script** do segundo **while**.

```
12 soma = 0
13
14 # Laco de repetição
15 while (num != -1):
16     # Acumula valor
17     soma = soma +
18
19     # Inputs seguinte
20     num = int(input())
21
22 # Imprime resultado
23 print(soma)
24
```



```
11 while (num != 0):
12     # Zera variável
13     soma = 0
14
15 # Laco de repetição
16 while (num != -1):
17     # Acumula valor
18     soma = soma +
19
20     # Inputs seguinte
21     num = int(input())
22
23 # Imprime resultado
24 print(soma)
```

Figura 103: Questão 3D Primeiro Passo

Fonte: Autoria própria

A num (a variável que irá requerer o número do usuário) está acima do primeiro while, justamente porque o primeiro while usa essa variável como referência, assim como o segundo while. Você percebeu que agora o print está dentro do primeiro while? É dessa vez o print deve estar lá mesmo, toda vez que o primeiro while for repetir seu mine script ele apresentará a soma que foi acumulada no primeiro while, ou seja, a função do primeiro while é repetir de forma automática aquilo que foi feito na questão anterior, você pode somar uma certa quantia de números e depois começar outra quantia do zero sem precisar iniciar o script de novo.

Quando você termina de somar uma certa quantia de números ele sai do segundo while e corre para o que vem abaixo dele, abaixo dele está o **print** para mostrar a soma que foi feita dele, e depois devemos colocar a variável num novamente, lembre-se a num está se comportando como uma variável contadora e todo while precisa dessa variável, mas o que acontece nessa outra variável num? Se você colocar 0 o programa parará, mas se você colocar qualquer outro número o primeiro while vai recomeçar seu mine script, e dentro do seu mine script está o outro while que irá acumular esse número e pedir mais outro de você até que você insira -1 e termine essa sequência de número e continue a reproduzir o mine script do primeiro while que irá mostrar a soma e pedir novamente outro número para iniciar uma nova sequência ou para terminar o script por ali mesmo.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View Help
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Introdução
Lab04_03.py

1 # -*- coding: utf-8 -*-
2 """
3 24/03/2016
4
5 Autor: Helder Guerreiro
6
7 """
8 # Primeiro input
9 num = int(input("Primeira sequencia. Digite o primeiro valor: "))
10
11 while (num != 0):
12     # Zera variavel acumuladora
13     soma = 0
14
15     # Laco de repeticao
16     while (num != -1):
17         # Acumula valor
18         soma = soma + num
19
20         # Inputs seguintes da mesma sequencia
21         num = int(input("Digite outro valor: "))
22
23     # Imprime resultado
24     print(soma)
25
26     # Primeiro input da sequencia seguinte
27     num = int(input("Proxima sequencia. Digite o primeiro valor: "))
28

```

Figura 104: Questão 3D Primeiro Passo

Fonte: ICOMP, UFAM

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab04_03.py', wdir='C:/Users/User/Documentos/Helder Guerreiro/Meus Programas')
Primeira sequencia. Digite o primeiro valor: 1
Digite outro valor: 2
Digite outro valor: 3
Digite outro valor: 4
Digite outro valor: 5
Digite outro valor: -1
15
Proxima sequencia. Digite o primeiro valor: 5
Digite outro valor: 5
Digite outro valor: 5
Digite outro valor: -1
15
Proxima sequencia. Digite o primeiro valor: 0
>>>

```

Figura 105: Resultado questão 3D

Fonte: Autoria própria

Questão 4 - Média de uma coleção de números

Escreva um programa que calcula a média de uma coleção de valores digitados pelo usuário, com precisão de até duas casas decimais. O usuário irá inserir 0 para indicar que não há mais valores serem fornecidos.

DICAS:

1. Modifique o código da Questão 2, acrescentando uma variável contadora para determinar quantos valores foram inseridos.

2. Como o valor 0 marca o final de digitação da sequência, seu código não pode incluí-lo no cálculo da média.

3. Para a sequência de entrada 1 2 3 4 0, a saída deverá ser 2.5.

Primeiro passo: Interpretação da questão

Você já viu a dica no enunciado né, pegue o script da questão 2 de novo que vamos usa-la para facilitar nosso trabalho, dessa vez não precisamos de outro `while`, só precisamos fazer umas modificações simples.

O script da questão ficará intacto só mudará a condição do `while` que agora a `num` deverá ser diferente de 0 não mais do -1. Vamos pensar com calma, o script da segunda questão já faz a soma, do que precisamos para fazer a média? Primeiro devemos pensar o seguinte: como fazer uma média? Claro, pegamos o todo (que é a soma) e divide pela quantidade, mas é aí que está: como saber quantos números nós inserimos script? A resposta é simples, precisamos de uma variável contadora. Sim eu me lembro que eu disse que a `num` se comporta como uma variável contadora, mas na verdade ela não é por que ela não está contando nada ela está sendo modificada pelos números que nós inserimos no script.

Vamos chamar a variável contadora de "cont" e você sabe que esse tipo de variável sempre deve estar lá no início do script, e você já deve saber também que essa variável contadora começa igualada ao número 0. Dentro do meu script do `while` deveremos colocar essa variável contadora igualando a ela mesma mais 1, isso por que a cada vez que o `while` repetir seu meu script a `cont` irá somar mais 1 no seu valor total, e no final de tudo a `cont` terá a quantidade de vezes que inserimos um número.

Agora temos que construir a variável para calcular a média, essa variável deve estar fora do `while` por que ela só irá calcular após todos os números serem inseridos, a variável `soma` terá o resultado final da soma e a variável `cont` terá a quantidade de números inseridos, basta dividir os dois para encontrar o resultado final. Por fim não esqueça de colocar a função `round` no `print` com o arredondamento para 2 casas decimais.

The screenshot shows the Spyder Python 3.4 IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, and View. Below the menu is a toolbar with various icons. The main area displays a code editor for a file named 'Lab04_04.py'. The code is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 24/03/2016
4
5 Autor: Helder Guerreiro
6
7 """
8 # Primeiro input
9 num = int(input("Digite um numero: "))
10
11 # Zera variavel acumuladora
12 soma = 0
13
14 # Zera variavel contadora
15 cont = 0
16
17 # Laco de repeticao
18 while (num != 0):
19     # Acumula valor
20     soma = soma + num
21
22     # Atualiza contador
23     cont = cont + 1
24
25     # Inputs seguintes
26     num = int(input("Digite outro numero: "))
27
28 # Calcula media
29 media = soma/cont
30
31 # Imprime resultado
32 print(round(media, 2))
33
```

Figura 106: Questão 4D Primeiro Passo

Fonte: ICOMP, UFAM

The screenshot shows the Python 1 console window. The command `>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab04_04.py', wdir='C:/Users/Helder Guerreiro/Meus Programas')` was run. The console then prompts for user input and displays the results:

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab04_04.py', wdir='C:/Users/Helder Guerreiro/Meus Programas')
Digite um numero: 2
Digite outro numero: 2
Digite outro numero: 2
Digite outro numero: 2
Digite outro numero: 0
2.0
>>>
```

Figura 107: Resultado questão 4D

Fonte: Autoria própria

Questão 5 – Média de várias coleções de números

Escreva um programa que calcula a média de várias coleções de valores digitados pelo usuário, com precisão de até três casas decimais. O usuário irá inserir 0 para indicar o fim de uma coleção e o início da próxima, e 9999 para indicar que não há mais valores a serem fornecidos.

Dicas:

1. Modifique o código da Questão 3, acrescentando uma variável contadora para determinar quantos valores foram inseridos em cada sequência.
2. Como o valor 0 marca o final de digitação da sequência, seu código não pode incluí-lo no cálculo da média.
3. Para a sequência de entrada 9 10 7 0 12 3 4 0 1 3 7 1 3 1 9 0 9999, a saída deverá ser 8.667 2.5 8.6.

Primeiro passo: Interpretação da questão

Essa questão é a cópia da anterior, é muito simples, a dica fala para você pegar a questão 3 e modificar, mas eu prefiro que você pegue a questão anterior e modifique. Em vez de usar aquele 9999 vamos usar o -1 mesmo.

Na questão anterior você deixe ela do jeito que estar não precisa modificar nada. Basta você adicionar aquele primeiro **while** acima da variável soma e abaixo da variável num, não esqueça de alinhar tudo de acordo com o mine script do primeiro while e o segundo while, não misture os dois mine scripts.

Dessa vez o **round** deve arredondar até 3 casas decimais, e abaixo dele estar a variável num novamente para que o primeiro while venha reiniciar ou parar o seu mine script.

The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. Below the menu is a toolbar with various icons. The main area shows a file named 'Lab04_05.py' with the following code:

```
1 # -*- coding: utf-8 -*-
2 """
3 24/03/2016
4
5 Autor: Helder Guerreiro
6
7 """
8 # Primeiro input
9 num = int(input("Primeira sequencia. Digite o primeiro valor: "))
10
11 while (num != -1):
12     # Zera variavel acumuladora
13     soma = 0
14
15     # Zera variavel contadora
16     cont = 0
17
18     # Laco de repeticao
19     while (num != 0):
20         # Acumula valor
21         soma = soma + num
22
23         # Atualiza contador
24         cont = cont + 1
25
26         # Inputs seguintes da mesma sequencia
27         num = int(input("Digite outro valor: "))
28
29     # Calcula media
30     media = soma/cont
31
32     # Imprime resultado
33     print(round(media, 3))
34
35     # Primeiro input da sequencia seguinte
36     num = int(input("Proxima sequencia. Digite o primeiro valor: "))
37
```

Figura 108: Questão 50 Primeiro Passo

Fonte: ICOMP, UFAM

The screenshot shows the Python 1 console window. The command `runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab04_05.py', wdir='C:/Users/Us')` is run, followed by user inputs and the program's output:

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Meus Programas/Lab04_05.py', wdir='C:/Users/Us')
Primeira sequencia. Digite o primeiro valor: 5
Digite outro valor: 5
Digite outro valor: 5
Digite outro valor: 0
5.0
Proxima sequencia. Digite o primeiro valor: 2
Digite outro valor: 2
Digite outro valor: 2
Digite outro valor: 2
Digite outro valor: 0
2.0
Proxima sequencia. Digite o primeiro valor: -1
>>>
```

Figura 109: Resultado questão 50

Fonte: Autoria própria

Questão 6 – Aproximação de π

O valor de π pode ser aproximado pela seguinte série infinita:

$$\pi \approx 3 + \frac{4}{2 \times 3 \times 4} - \frac{4}{4 \times 5 \times 6} + \frac{4}{6 \times 7 \times 8} - \frac{4}{8 \times 9 \times 10} + \frac{4}{10 \times 11 \times 12} - \dots$$

Escreva um programa que exibe N aproximações de π , sendo N um número natural inserido pelo usuário. A primeira aproximação deve fazer uso de apenas o primeiro termo da série infinita (3). Cada aproximação adicional indicada pelo seu programa deve incluir mais um termo na série, fazendo uma melhor aproximação de π cada vez que um termo é incluído na soma. Utilize até seis casas decimais de precisão.

DICAS

1. Determine o termo geral da série antes de começar a programar.
2. Por exemplo, para a entrada 1, a saída deve ser 3. Para a entrada 2, a saída deve ser 3.166667. Já para a entrada 5, a saída deve ser 3.166667 3.133333 3.145238 3.139683.

Primeiro passo: Interpretação da questão

É isso mesmo, acabamos de invocar o demônio, você se assustou com essa questão? É eu também, pensei dez vezes antes de colocar essa questão aqui, mas fazer o que.

Nesse tipo de questão preste muita, mas muita atenção mesmo no enunciado! Olhe tudo o que tiver de olhar por que a resposta está lá mesmo. Toda vez que o enunciado de uma questão trouxer para você uma formula, você terá de passar essa formula para o seu script do Python.

Nesta questão o número que você inserir será a quantidade de vezes que o script irá aproximar um valor para o Pi. Então ele será a base do `while` nessa questão, devemos ter uma variável contadora também para marcar quantas vezes o `while` está repetindo seu `mine script` e assim o `while` possa saber quando parar, ele vai parar quando a quantidade de vezes que o usuário inseriu ser atingida. Você percebeu na formula que independente do denominador e numerador ele sempre começa com 3, então vamos criar uma variável e deixar esse 3 guardado, mas por que guardar o 3 como variável? Por que esse 3 ele já é uma aproximação do Pi, mas depois de calcular a formula a aproximação já será outra, então vamos fazer uma variável chamada "Ap" (aproximação Pi) que começará com o número 3. Na dica do enunciado da questão mostra que quando damos entrada em um número o 3 sempre aparece na resposta, isso quer dizer que antes de começarmos com o `while` devemos colocar um `print` para o Ap para que o 3 seja mostrado antes dele ser modificado no `while`.

Agora começando com o `while`, a condição dele é que o `mine script` seja repetido até que a quantidade de vezes que o usuário inseriu seja alcançada, vamos chamar a variável em que o usuário insere a quantidade de aproximações de "num". Então se caso o usuário insere 5, o `while` irá repetir seu `mine script` 5 vezes isso quer dizer que a variável contadora "cont" não pode passar de 5, ou seja, `cont < num`.

Dentro do while temos que desenvolver a formula apresentada no enunciado, ela é digamos grande para colocarmos de uma vez só, podemos separar a formula em denominador e o numerador com a soma dos outros termos, mas tenha em mente uma coisa, essa formula apresentada no enunciado não é uma formula padrão, ou seja, uma formula geral, ela é só um exemplo, devemos encontrar sua formula geral para podermos passarmos para Python.

$$\pi \approx 3 + \frac{4}{2 \times 3 \times 4} - \frac{4}{4 \times 5 \times 6} + \frac{4}{6 \times 7 \times 8} - \frac{4}{8 \times 9 \times 10} + \frac{4}{10 \times 11 \times 12} - \dots$$

Veja que o denominador vai aumentando de um em um começando pelo 2, perceba uma coisa olhando para esse exemplo acima: cada fração é uma aproximação diferente, ou seja, primeiro nosso script fará o 3 depois vai para a fração seguinte, em seguida vai para outra fração, o número de frações que o script irá calcular será o número de vezes que o usuário inseriu no início do script. Veja que se o usuário escolher o número 3, o script irá apresentar o 3 e a fração do lado dele que é $\frac{4}{2 \times 3 \times 4}$, e em seguida a fração $\frac{4}{4 \times 5 \times 6}$, você consegue ver o que essas duas frações tem em comum? No script antes de entrar no while ele já apresenta o 3, dentro do while a variável contadora estará com 1, e o denominador dessa fração resulta: $2 \times 3 \times 4$, depois a cont estará com 2, e o denominador dessa fração resulta: $4 \times 5 \times 6$. Isso mesmo, a variável contadora está dentro da formula e posso apresentar a vocês que o denominador pode ser calculado assim:

$$Den = (cont2) \times (cont2 + 1) \times (cont2 + 2)$$

O cálculo do denominador já começa com variável cont multiplicada por 2 por que no início a cont vale 1 e 1×2 é igual a 2 que é como inicia o denominador dessa formula. Por que separar o denominador como formula? Para evitar que a formula fique grande demais e confusa, perceba que o 4 nunca muda, então podemos simplesmente fazer $4/Den$, só que agora temos que nos preocupar com outra coisa: você percebeu que o sinal está mudando em positivo e negativo? Perceba que na primeira fração ela está positiva, na segunda ela está negativa, na terceira positiva, na quarta negativa... Temos uma oscilação de sinais e podemos mudar o sinal usando a mágica do expoente ímpar e par, você se lembra das aulas de exponencial? Um número negativo com expoente par fica positivo, se o expoente for ímpar ele continua negativo, você se lembra do Ap? Ela é a variável que começa com o número 3 e ela é a nossa variável acumuladora, ela será a próxima formula do meu script. De primeira a formula deve ser igual a ela mesma, isso por que Ap é igual a 3 no início e você vê no exemplo que o cálculo começa com 3, agora devemos oscilar os sinais usando a exponencial, vamos usar um número negativo (-1) e eleva-lo a um expoente, esse expoente que terá o cont somado a um, toda vez que o cont for atualizado o expoente irá mudar, na primeira o expoente será 2 e a fração será positiva, na segunda o expoente será 3 e a fração será negativa. Agora você viu né como realmente isso funciona, depois de ter o Ap e somado ao -1 elevado ao expoente é hora de multiplicar esse -1 pelo $4/Den$, devemos multiplicar simplesmente por que para o sinal do -1 afetar a fração ele deve estar multiplicando. Veja a formula abaixo.

$$Ap = Ap + (-1)^{(cont+1)} \times \frac{4}{Den}$$

Pronto, abaixo dessas duas formula colocamos a cont somado a 1 para que ela venha fazer o while se repetir até que ele atinja o número de vezes que o usuário deseja, você deve estar se perguntando por que a condição do while é "cont < num" e não "cont <= num"? Por que quando o print antes do while apresenta o 3 já devemos contar isso como uma aproximação, se a condição do while for "cont <= num" quando o usuário inserir 2 aproximações aparecerá três

aproximações na tela. Não esqueça que o print final deve estar dentro do while para que ele apresente cada aproximação feita e a aproximação deve ser arredondada até 6 casas decimais.

The screenshot shows the Spyder Python 3.4 IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. Below the menu is a toolbar with various icons. The main area displays a Python script named 'Lab04_06.py'. The code is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 24/03/2016
4
5 Autor: Helder Guerreiro
6
7 """
8 # Primeiro input
9 # Le numero de aproximacoes a serem impressas
10 num = int(input("No. de aproximacoes: "))
11
12 # Inicia variavel contadora
13 cont = 1
14
15 # Inicia variavel acumuladora (primeiro termo da serie do PI)
16 Ap = 3
17
18 # Imprime primeira aproximacao
19 print(Ap)
20
21 # Determina e imprime demais aproximacoes
22 while (cont < num):
23     # Determina o denominador
24     den = (cont*2) * (cont*2 + 1) * (cont*2 + 2)
25
26     # Computa novo termo da serie do PI
27     Ap = Ap + (-1)**(cont+1) * 4. / den
28
29     # Incrementa contador
30     cont = cont + 1
31
32     # Imprime resultado
33     print(round(Ap, 6))
34
```

The screenshot shows the Python 1 console window. It displays the command runfile followed by the path to the script and the number of approximations requested. The output shows the first five approximations of pi, each rounded to 6 decimal places:

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/1º Período/Introdução a Programação de Computadores/Meus Programas/Lab04_06.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/1º Período/Introdução a Programação de Computadores/Meus Programas')
No. de aproximacoes: 5
3
3.166667
3.133333
3.145238
3.139683
>>>
```

Figura III: Resultado questão 60

Fonte: Autoria própria

Figura II0: Questão 60 Primeiro Passo

Fonte: ICOMP, UFAM

4.3 Avaliações

Aqui teremos algumas avaliações para entender mais um pouco o assunto estudado neste primeiro capítulo. Aconselho você tentar fazer sozinho antes de ler a resolução.

Avaliação A – A4

Avaliação Parcial 04 (A4) a – Estrutura de Repetição por Condição (while)

Informações

Questões

Notas

Questão 1

Uma pessoa investe uma quantia numa aplicação que rende **7%** ao mês. Escreva um script que leia a partir do teclado o valor investido e a quantidade de meses da aplicação, e informe o saldo do investimento mês a mês, considerando apenas a quantia inicial.

Dicas:

1. Teste o script no Spyder antes de submetê-lo ao Code Bench.
2. Verifique se você está submetendo o código correto à questão correspondente desta Avaliação Parcial.
3. Utilize uma variável **acumuladora** para guardar o saldo atualizado e uma variável **contadora** para registrar os meses.
4. Lembre-se de arredondar os resultados com até **duas casas** decimais, pois este problema envolve quantias em real.
5. Alguns exemplos (não exaustivos): para a entrada **1000 e 3**, indicando uma aplicação de mil reais em três meses, a saída deve ser **1070.0 1144.9 1225.04**. Já para as entradas **200 e 5**, a saída deverá ser **214.0 228.98 245.01 262.16 280.51**.

Logo de primeira já se percebe que essa questão não exige muito assim de você, mas claro você deve estar em dia com o seu entendimento sobre o Python.

Num script a primeira coisa a se ter em mente é o uso de bibliotecas e quais as entradas que serão inseridas nele, neste caso não precisaremos de nenhuma biblioteca e as entradas você pode ver que somente duas: o valor investido e a quantidade de meses que esse valor será investido.

Guarde essa dica: toda vez que um problema do Python envolver tempo numa questão em que vamos usar o **while**, o tempo sempre será controlado através de uma variável contadora, ou seja, o tempo é aquela variável que começa com o número "0" e no meu script do while ela é somada a mais um de tal forma que a cada repetição do while o tempo irá ser acrescentado (ou seja, vai passando) logicamente.

Com isso, faça uma variável contadora chamada **Tempo** e iguale-a zero, neste caso não precisaremos colocar uma variável acumuladora dela por que o valor investido que o usuário insere já faz esse papel, pois o valor será modificado de acordo com os juros então o valor inserido será acumulado de juros ao passar de cada repetição do while.

O while tem uma condição, certo? Então de acordo com a questão o valor vai depender da quantidade de meses, isso quer dizer que a cada mês passado a quantia se modificará, a entrada da quantidade de meses que o usuário irá

inserir será o limite de vezes que o while irá repetir seu mine script; logo, a condição deve ser que a contagem do tempo (variável contadora) não ultrapasse o limite da quantidade de meses inserida pelo usuário, mas preste atenção numa coisa: devemos usar o sinal “menor que” ou “menor ou igual que”? Se a variável contadora for igual a zero (nossa caso) devemos usar “menor que”, mas se você optar por usar a variável contadora igual a 1, deve-se usar o sinal “menor ou igual que”. Por que? Isso se dá pelo fato de que quando a variável está em zero, o while começará a trabalhar com a variável no número “0”, ou seja, é como se fosse o “mês zero” e se colocarmos o sinal “menor que” a contagem irá parar um número antes do número inserido pelo usuário, porém o zero já foi contado então se parar em 4 será a mesma coisa que 5 ou se parar em 9 será a mesma coisa que 10 e assim vai... Por que eu disse para você colocar a variável contadora igual a “0” se colocando igual a “1” seria mais simples? Só para você entender que o número que você escolhe nas suas variáveis têm grande importância,

Abaixo do while tem o seu mine script, faça uma variável que irá calcular o lucro obtido (chame a variável de lucro mesmo) que é simplesmente pegar o valor investido e multiplicar por 7% (0,07); logo abaixo terá a mesma variável usada para o usuário inserir o valor investido, só que ela dessa vez será igualada a ela mesma mais o lucro, ou seja, ela está fazendo papel de variável acumuladora, toda vez que o while for repetido a variável do valor investido será atualizada com a soma dela mesma mais o lucro; não esqueça de fazer com que a variável Tempo seja contada a cada repetição do while; no final deve-se pensar: onde ficará o print? Preste bem atenção no que o enunciado diz, ele quer que o valor de cada mês seja apresentado, isso quer dizer que o print deve estar dentro do mine script do while, pois a cada repetição dele o print será acionado e o print vai mostrar justamente a situação no atual momento da variável do valor investido. Não esqueça da função round com aproximação de duas casas decimais dentro da função print.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do A
AV_A.py
1 # -*- coding: utf-8 -*-
2 """
3 05/04/2016
4
5 Helder Guerreiro
6 """
7
8 Valor = float(input("Valor investido: "))
9 Mes = int(input("Quantidade de meses: "))
10
11 Tempo = 0
12
13 while Tempo < Mes:
14     Lucro = Valor*0.07
15     Valor = Valor + Lucro
16     Tempo = Tempo + 1
17     print(round(Valor,2))
18
19
20

```

Figura II2: Resposta Avaliação A 4A

Fonte: Autoria própria

Avaliação B - A4

Avaliação Parcial 04 (A4) b – Estrutura de Repetição por Condição (while)

Informações

Questões

Notas

Questão 1

Escreva um script que determine os valores parciais da seguinte série para N termos, sendo N informado pelo usuário via teclado. Utilize até três casas decimais de precisão.

$$S_D = \frac{1}{3 \cdot 3} - \frac{2}{3 \cdot 5} + \frac{3}{3 \cdot 7} - \frac{4}{3 \cdot 9} + \frac{5}{3 \cdot 11} - \frac{6}{3 \cdot 13} + \dots$$

Dicas:

1. Determine o termo geral da série antes de começar a programar.
2. Verifique quais os valores mais apropriados para inicializar a variável **contadora** e a variável **acumuladora**.
3. Observe se a divisão resulta em um número real, e não um inteiro.
4. Alguns exemplos (não exaustivos): para a entrada 1, a saída deve ser 0.111. Já para a entrada 2, a saída deve ser -0.022. Para a entrada 84, a saída deve ser -0.035.

Vamos ser realistas: essa prova foi difícil..., mas ela é a mesma que a sexta questão que resolvemos acima, a diferença é só a série em que devemos fazer uma forma de representá-la no Python.

Você já deve saber que para esse tipo de script deveremos ter uma variável contadora e outra acumuladora. Mas antes de tudo você já percebeu que não irá usar nenhuma biblioteca neste script então vamos ao primeiro passo: a entrada (que nesse caso é o número de termos, vamos chamar essa entrada de N mesmo).

Antes de começarmos a montar o **while** e seu mine script devemos pensar em como representar aquela coisa lá em cima (a série) o Python. Vamos usar a mesma estratégia que fizemos na sexta questão resolvida, o denominador será formulado e depois juntamos tudo; veja bem os denominadores dessa série... todos os denominadores vem acompanhado de um 3 multiplicando outro número, o número que multiplica com esse 3 tem a seguinte sequência: 3, 5, 7, 9, 11, 13... perceba que o intervalo entre os números é 2, veja: 5 - 3 = 2; 9 - 7 = 2; 13 - 11 = 2. Então pensa o seguinte, devemos fazer com que a cada repetição o segundo número seja adicionado a 2, sendo que o primeiro número é o 3 e tem outro 3 que sempre se permanecerá constante... perceba que a variável contadora entrará em cena aqui, isso por que o denominador irá mudar a cada repetição do while e quem é responsável por contar cada repetição? Isso mesmo, a contadora; já sabemos que é uma multiplicação, vamos o primeiro denominador como base:

3.3

A pergunta é: qual a diferença entre esse denominador e os outros? A diferença entre eles não é a adição de duas unidades no segundo número? Mas como colocar essa diferença nos outros denominadores? Você se lembra da variável contadora? Então, vamos usar ela como a mudança entre os denominadores.

Vamos continuar pensado: como ficaria o segundo denominador? E se fizermos como uma soma? Olha como fica:

3. $(3 + 2)$

Vamos concordar que $3 + 2 = 5$ né? Então, veja como seria o terceiro denominador:

3. $(3 + 2 + 2)$

Acho que você deve saber que $3 + 2 + 2 = 7$, não? Então veja para haver uma mudança de um denominador para outro deve haver o acréscimo de duas unidades ao segundo número, você associar esse acréscimo à variável contadora? Por que devemos usá-la? Digamos que essa série é infinita (toda série é) então devemos dar um jeito de não escrever o script infinito não? (Claro que isso é impossível). Você se lembra de quando aprendeu a multiplicar? Eu acho que você sabe que a multiplicação pode ser simplificada em soma (digamos que eu estou escrevendo para um universitário [ou não], então você deve saber disso), veja que podemos mudar aquela visão dos denominadores lá em cima e tirar aquela soma e colocar uma multiplicação no lugar:

3. $(3 + 1.2) \& 3. (3 + 2.2)$

É a mesma coisa né? Mas como ficaria o primeiro denominador, já que não vamos acrescentar nenhum 2? Você sabia que existe um número que anula os outros? Acho que você deve conhecê-lo, ele se chama "zero"...

3. $(3 + 0.2)$

Se você é esperto acho que você deve ter percebido que está havendo um crescimento linear (de um em um) do número que está multiplicando o 2, é isso mesmo esse cara é a variável contadora que começará com o número 0. Perceba que você não pode ir logo colocando a variável contadora como 0 ou 1, devemos analisar o que estamos trabalhando dentro do while, a lógica é começar a contagem do 1, mas se fizermos isso não conseguiremos fazer com que o primeiro denominador apareça no cálculo. Então chamando essa formula de "Den" e chamando a variável contadora "Vc" temos o seguinte resultado:

$$Den = 3. (3 + Vc. 2)$$

Calma não fique logo pulando de alegria, isso é só o denominador (estraga prazer). Agora vamos ao principal, ou seja, a série em si. Uma coisa é importante você ter em mente, quando o assunto é séries a formula principal dela sempre uma variável acumuladora, isso quer dizer que a variável acumuladora será a encarregada de ter a formula principal nas costas, e para ela por se acumular ela deve ser igual a ela mesma mais a formula. Por que tudo isso? Toda vez que o while repetir seu script a variável acumuladora será igual à soma daquilo ela tinha acumulado antes com a formula e desse jeito o resultado será acumulado a cada repetição.

Na montagem da variável acumuladora (Va) deve-se olhar atentamente a construção da série apresentada no enunciado, perceba que está havendo uma troca de sinais: primeiro positivo, segundo negativo, terceiro positivo, quarto negativo... ou seja, teremos que fazer com que os sinais se troquem a cada repetição do while, fizemos isso na questão 6 lembra? Naquele caso fizemos o número -1 elevado à variável acumuladora mais 1, mas como ficaria neste caso? Lembre-se que no caso da questão 6 a primeira repetição ia resultar no -1 elevado a 2, por que lá a variável contadora começa com o número 1, sendo -1 elevado a 2 o resultado ia ser positivo e na próxima o expoente seria 3 e então ficaria negativo, para a nossa prova aqui a única coisa que muda é que a variável contadora no expoente será somada ao número 2 e não mais ao 1, isso por que nossa variável contadora começa com o número 0 e dessa forma na primeira repetição ia resultar em -1, ou seja, já ia começar com a fração negativa, fato que não pode acontecer nesta série.

Perceba a diferença entre esta série acima e a série da questão 6, na questão 6 existe o número 3 que soma no início antes de qualquer outra fração, nesse caso a variável acumuladora começaria com o número 3, isso por que a variável acumuladora deve ser somada à formula da série e como no caso desta prova temos somente frações sendo somadas entre si então a variável começará simplesmente com o número zero, isso por que a variável deve ser somada à formula de qualquer forma para que aja acumulação, como não temo nenhum número para somar então vamos simplesmente colocar a variável acumuladora igual a zero (isso acima do while).

Já providenciamos o denominador, mudança de sinal e a variável acumuladora, está faltando os numeradores dessa série, veja que eles são bem simples e seguem em ordem ordinária: 1, 2, 3, 4, 5, 6... então para que aja uma mudança de número a cada repetição que o while der é claro que temos que colocar a variável contadora no meio e eu acho que você já deve ter sacado como é que isso vai funcionar né, a variável contadora não é igual a zero antes de começar toda a conta? Então nesse caso na primeira contagem o numerador deve ser 1, na segunda contagem deve ser 2, e na terceira 3 e assim por diante, ou seja, você já deve ter se tocado que estamos falando de uma soma, se a variável contadora é 0 então na primeira se ela for somada a 1 o resultado será 1, e na segunda repetição a variável contadora já estará com o seu valor contado como 1 e somado a 1 o resultado será 2 e assim por diante... ou seja temos que o numerador pode ser expresso por: $Vc + 1$.

Depois de tudo isso você ainda fica com dúvida de como organizar tudo... é simples, a variável acumuladora (Va) será igual a ela mesma somada com o numerador ($Vc + 1$) que será multiplicado pelo -1 elevado a variável contadora mais 2 (para que aja a mudança de sinal) que será dividido pelo denominador, pronto.

$$Va = Va + \frac{(Vc + 1) \cdot (-1)^{Vc+2}}{Den}$$

Logo abaixo da variável acumuladora não esqueça de fazer a variável contadora ser acrescentada a 1 cada vez que o while for repetido e o `print` neste caso estará fora do mine script do while e terá a função `round` embutida dentro dele com três casas decimais.

The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, Run, and Stop. The main area is titled 'Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º' and contains the code for 'AV_B.py'. The code is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 05/04/2016
4
5 Helder Guerreiro
6 """
7
8
9 N = int(input("Insira o número de termos da série: "))
10
11 # Variável acumuladora
12 Va = 0
13
14 # Variável contadora
15 Vc = 0
16
17
18 while Vc < N:
19     Den = 3 * (3 + Vc*2)
20     Va = Va + ((Vc + 1 )* (-1)**(Vc + 2) )/ Den
21     Vc = Vc + 1
22 print(round(Va,3))
23
```

Figura 113: Resposta Avaliação B 4A

Fonte: Autoria própria

5. Álgebra uma Garota Difícil de Conquistar – Vetores

Você não gosta de álgebra? Que pena, você escolheu exatas no vestibular... ou você faz biológicas e foi amaldiçoado.

Vetores é uma oportunidade do Python poder carregar informações de forma cada vez mais simplificada, (engraçado a simplificação deveria melhorar nossa vida, não piorar) então os vetores vão nos ajudar a fazer aquilo que seriam scripts enormes em scripts não tão grandes (não te ilude não).

5.1 Aprendendo a Invocar Demônios

Então para você ativar os vetores no seu script, iremos precisar da biblioteca "numpy" basta escrever como se fosse uma biblioteca qualquer no início do script.

A álgebra dentro do Python é cheia de frescura, você deve usar vários comandos para que consiga trabalhar com os vetores e etc. Veja as seguintes formas de se construir vetores:

a) Construir um vetor já com os dados embutidos:

- Usa-se um comando chamado "array", esse comando possibilita a criação de um vetor e para colocar os dados dentro dele faz-se o seguinte: `vetor = array [(x, y, z ...)]`

b) Construir um vetor em que os dados são inseridos pelo usuário:

- Usa-se além do array, outro comando chamado "eval" que fica dentro do array e além desses dois deve-se usar o `input` também, já que estamos pedindo do usuário para colocar um vetor temos que usar o `input`.

Ex.: `vetor = array (eval (input ("Informe um vetor: ")))`

Os dados deverão ser colocados separados por vírgula,

c) Construir um vetor automático que carrega uma quantidade determinada de zeros:

- Neste caso não importa quantos elementos tenha o vetor todos eles serão zeros (para que vou usar isso? - Não te interessa, depois explicarei), esse vetor seria a mesma coisa que você criasse um array com um bocado de zeros dentro, para você criar esse vetor deve usar um comando chamado "zeros" e dentro desse cara em parênteses você irá colocar um número que será o tamanho do vetor, Ex.: se você colocar 5, o vetor terá a quantidade de 5 zeros dentro dele, só que não é só isso do lado do número separado por vírgula deve-se colocar um comando chamado "dtype=int", esse comando serve para que o vetor reconheça esses zeros como números inteiros. Então para construir vetores nulos fica assim: `vetor = zeros (N, dtype=int)`

d) Construir um vetor automático que carrega uma quantidade determinada de um's:

- É a mesma coisa que lá em cima, em vez de usar o comando "zeros" vamos usar o comando "ones" e pronto, coloca o número da quantidade de valores dentro do vetor e coloque o comando `dtype=int`.

Ex.: vetor = ones (N, dtype=int)

e) Construir um vetor automático com uma certa quantidade de números:

- Aqui a frescura aumenta a um nível alarmante, devemos usar um comando chamado "linspace" e dentro dele devemos colocar o intervalo de números que queremos que ele adicione dentro do vetor, assim: de 5 a 10 devemos colocar linspace (5, 10), o comando dtype=int também entra aí dentro, o negócio é que agora inventarão uma chatice que nem sei como isso foi para aí, se chama "espaçamento linear dos valores do vetor", isso aí é um terceiro número que será colocado dentro do linspace ele funciona para que você diga a ele quantos valores devem estar dentro do vetor, veja: linspace (5, 10, 6, dtype=int) o intervalo é de 5 a 10 e o espaçamento é 6, por que 6? Veja que se você contar o 5 até o 10 você terá 6 unidades, você terá 6 unidades por que estará incluindo o 5 na conta também (o 5 deve ficar dentro do vetor também).

Mas o que acontece se esse valor for maior ou menor que o intervalo de números?

Bom esse número indica quantos valores estarão dentro do vetor, se o número for 6, independente do intervalo, haverá 6 números lá e se for 4, independente do intervalo, haverá 4 números lá, lembre-se de uma coisa: o linspace não quebra os números (lembre-se do dtype=int) então se o intervalo for maior que o número de vetores o linspace irá excluir alguns números, mas se o intervalo for menor que o número de vetores o linspace irá repetir alguns números.

Então: vetor = linspace (x, y, N, dtype=int)

5.2 Funções e Operações de Vetores

Podemos trabalhar com os vetores utilizando de várias funções e operações que podemos usufruir. São essas funções que iremos mais utilizar em nossas questões.

5.2.1 Selecionando Parte de um Vtor

Esta forma de seleção se é muito útil quando se vai trabalhar com os valores do vetor.

Código	Objetivo
<code>vec[i]</code>	Seleciona o elemento de índice <code>i</code> do vetor <code>vec</code>
<code>vec[i:j]</code>	Seleciona os elementos do vetor <code>vec</code> cujos índices estão compreendidos entre <code>i</code> e <code>j-1</code>
<code>vec[i:]</code>	Seleciona os elementos do vetor <code>vec</code> do índice <code>i</code> até o final do vetor.

Figura II4: Comandos de seleção de parte de um vetor

Fonte: ICOMP, UFAM

O primeiro comando seleciona somente um dos valores do vetor, o vetor tem a quantidade de seus valores contadas a partir do 0 até o último valor, o vetor também pode ser contado de trás para frente por forma de número negativo, o número -1 representa o último valor do vetor e a sua sequência é: -1, -2, -3, -4... quanto mais negativo o número, mais perto do início do vetor, na contagem normal (0, 1, 2, 3, 4...) o 0 representa o primeiro valor e quanto maior o número mais perto do fim do vetor. Esse “:” mostrado na imagem representa um índice de sua escolha para selecionar o valor correspondente a esse índice. Se por exemplo você usar esse comando dentro do print o valor escolhido será apresentado a você.

O segundo comando selecionará um intervalo dentro do vetor estabelecido por você, só que dessa vez o último índice será o limite do intervalo só que não será contado dentro do intervalo, é como se você fizesse um intervalo em que o primeiro índice é um intervalo aberto (faz parte da contagem) e o último é fechado (não faz parte).

O terceiro é bem simples, você seleciona um índice como referência para que todo o resto do vetor a partir do índice escolhido seja selecionado também.

5.2.2 Operações com Vetores

Você sabia que um vetor pode carregar também outros vetores? É meu amigo a coisa ta ficando séria essa álgebra está com brincadeira não. É a mesma coisa que construir um vetor normal, só que dessa vez dentro dele haverá outros vetores, claro que você deve construir esses vetores antes de adicionar dentro de outro vetor. Deixa eu explicar, você não vai colocar um bocado de número (até dá, mas que tal sermos mais organizados?) Você fará o vetor por fora e depois colocará só a sua variável dentro do outro vetor. Se quiser pode até inserir vetores e números num só vetor, que loucura. Não precisa usar o array no vetor principal.

Ex.: `V1 = array [(x, y, z)]`

`V2 = array [(a, b, c)]`

`V3 = [V1, V2, i, j, k]`

Agora irei lhe apresentar algumas operações maneiras que se pode fazer com o vetor em mãos, saca só o esquema logo mais abaixo:

- Encontrar o **menor** elemento de um vetor:
`>> min(vetor)`
- Encontrar o **maior** elemento de um vetor:
`>> max(vetor)`
- Encontrar o **tamanho** de um vetor:
`>> size(vetor)`
- Encontrar a **soma** dos elementos de um vetor:
`>> sum(vetor)`

Figura II5: Operações com vetores

Fonte: ICOMP, UFAM

Você se lembra daquelas questões chatas sobre tirar a médias de não sei lá o que? Então aqui com os vetores fica muito mais simples mesmo. Usando a função sum você já encontrará a soma de todos os valores e como a média é a soma do todo pela quantidade você deve prestar atenção para o seguinte: o vetor começa a contagem de seus elementos a partir do 0 então se o tamanho do vetor for 5, na verdade será 6, pois não devemos colocar o 0 num cálculo de média. Então para tirar média temos: `sum(vetor)/n`, sendo o "n" o número real do tamanho do vetor. Mas se você não souber o tamanho do tal vetor pode-se usar a função size, essa função vai fornecer o número real do tamanho do vetor. Então resumindo: $\text{média} = \text{sum}(\text{vetor})/\text{size}(\text{vetor})$.

5.3 Gráficos

Então, chegamos numas das partes mais odiadas pelos estudantes do Python, os gráficos são uma boa para quem faz laboratórios ou para quem quer se achar por saber fazer um gráfico no Python, mas agora a brincadeira acabou: todo mundo vai saber fazer gráficos agora. A pergunta ressoa... por que a parte mais odiada? É.... digamos que você saberá logo.

5.3.1 A Invocação do Mal

Prepare-se você vai falar uma língua nova, para trabalharmos com gráficos e funções e tudo mais é necessário invocar o nome desse exu: `from matplotlib.pyplot import *`, entendeu? Eu também não, essa coisa é o nome da biblioteca que fará com que o Python se comporte como MatLab, querendo ou não meu caro leitor isso é importante demais e não duvide nada que essa beleza possa te pegar em Cálculo Numérico.

Além de escrever aquela coisa feia lá em cima use a biblioteca numpy também por que iremos trabalhar com vetores.

5.3.2 Trabalhando como Doutores em T.I.

Acredita que tem gente que faz graduação, mestrado e doutorado nisso? O que será que essa pessoa tem para querer sofrer desse jeito? Lembre-se de que você odeia o Python, não queira isso para sua vida.

Então, para trabalharmos com o Python você deve aprender umas novas funções, essas funções são relacionadas as funções matemáticas coisa e tal.

Função	Objetivo
<code>poly1d(vet)</code>	Define um polinômio a partir de um vetor <code>vet</code> contendo seus coeficientes
<code>roots(p)</code>	Determina as raízes de um polinômio <code>p</code> . A saída desta função também é um vetor.
<code>polyval(p, x)</code>	Calcula o valor de um polinômio <code>p</code> no ponto <code>x</code> .
<code>polyder(p)</code>	Determina a 1ª derivada de um polinômio <code>p</code> .

Figura 116: Comandos de seleção de parte de um vetor.

Fonte: ICOMP, UFAM

Tem outras funções também, mas você as entenderá melhor com o exemplo muito louco que darei a seguir, presta atenção mano.

Exemplo Louco

Desenhar o gráfico da função $y = x^2 - 10x + 15$ para valores de x entre 0 e 10.

Que loucura! O cara fala para desenhar o gráfico assim como se fosse a coisa mais fácil do mundo, mas eu estou aqui para lhe prova que isso é difícil demais, só que você vai aprender e se tornará mais prático! Se você não aprender, me desculpe a culpa não é minha.

Primeiro Passo: Definir um polinômio a partir do vetor de coeficientes

Vou explicar o que é esse primeiro passo, todo polinômio tem seus coeficientes né? Se você não sabe (acho meio difícil você não saber) os coeficientes são todos os números da função em exceção do X, ou seja, se o X estiver sozinho então o coeficiente é 1 já que X vezes 1 é ele mesmo e caso não tenha nenhum tipo de X então o coeficiente é 0 já que X vezes 0 é nada. Com isso você já deve saber que os coeficientes desse polinômio são: 1, -10, 15. O primeiro passo que dizer que devemos criar um vetor só dos coeficientes e usando esse vetor iremos aplicar a função poly1d.

(Deixa eu lhe dar um aviso: a função poly1d NÂO se escreve com dois L's! Fui fazer isso e o Python estava dando erro toda hora... aí eu percebi que esse que parece um segundo L é na verdade o número 1, ou seja, a função se escreve P O L Y 1 D). Mostrarei todo aquilo que expliquei acima para você,

```
2 """
3 14/04/2016
4
5 Helder Guerreiro
6 """
7
8 from numpy import *
9 from matplotlib.pyplot import *
10
11
12 f = poly1d([1,-10,15])
13
14
```

Você sacou o que eu fiz aqui do lado? Fiz o chamado das duas bibliotecas e fiz uma variável em que nela está a função poly1d que está carregando um vetor, esse vetor são os coeficientes do polinômio.

Quando você faz a mesma coisa que eu fiz do lado, ou seja, usar a função poly1d e adicionar um vetor com coeficientes dentro dela é a mesma coisa que você escrever isso: $y = x^2 - 10x + 15$.

Figura II7: Exemplo louco primeiro passo

Fonte: Autoria própria

Segundo Passo: Criar um vetor de valores de x entre 0 e 10

Porque dar valores ao x? Bom, você está vendo que x é a variável aí do meio e ela consequentemente é a incógnita, o Python não trabalha com a variável sem nenhum tipo de valor, pelo menos uma variação ela deve ter, esse vetor serve dar um limite de valor ao x, isso por que se você não os coloca o x pode assumir valores infinitos e coitado do

processador do seu computador né? Podemos fazer isso usando o linspace que é só mostrar o intervalo que desejamos que o vetor será criado, assim:

```
8 from numpy import *
9 from matplotlib.pyplot import *
10
11
12
13 f = poly1d([1,-10,15])
14
15 x = linspace(0,10,10)
16
```

Nos tópicos anteriores eu lhe expliquei como usar o linspace né?
Então espero que ainda saiba usá-lo.

Mas eis a questão, por que não usamos o dtype=int? Por que se for assim o vetor irá reconhecer somente os números inteiros de 1 a 10 e não reconhecerá os números quebrados.

Figura 118: Exemplo louco segundo passo

Fonte: Autoria própria

Terceiro Passo: Calcular o valor do polinômio em cada ponto do x

Nesta parte vamos usar a função polyval, essa função deverá ter dentro dela a variável f (o polinômio) e a variável x (o intervalo), ela calculará o polinômio f para cada ponto do x. Se você não entendeu, essa função fará a mesma coisa que você pegar cada valor do x e substituir no polinômio e calcular.

```
8 from numpy import *
9 from matplotlib.pyplot import *
10
11
12
13 f = poly1d([1,-10,15])
14
15 x = linspace(0,10,10)
16
17 y = polyval(f,x)
```

Não esqueça que para colocar os dados dentro do polyval deve ter a ordem certa, primeiro se coloca o polinômio e depois a variável com seu intervalo.

Figura 119: Exemplo louco terceiro passo

Fonte: Autoria própria

Quarto Passo: Desenhar o gráfico

A partir de agora tudo que nós fizermos será para o gráfico, o polinômio já foi calculado.

Primeiro, devemos abrir uma figura, sim o Python produz uma figura só que se não a configurarmos ela sairá nua e crua, deve-se usar uma função que se chama "figure()" basta coloca-la em uma das linhas do script e pronto, precisa fazer mais nada.

Segundo, devemos "plotar" o polinômio em função de x, isso quer dizer, desenhar o polinômio em função de x, para isso usaremos a função chamada "plot()", dentro dela colocaremos primeiro o x e depois o y para que o polinômio seja plotado em x.

Até agora já fizemos isso:

```

8 from numpy import *
9 from matplotlib.pyplot import *
10
11
12
13 f = poly1d([1,-10,15])
14
15 x = linspace(0,10,10)
16
17 y = polyval(f,x)
18
19 figure()
20
21 plot(x,y)

```

Figura 120: Exemplo louco quarto passo a)

Fonte: Autoria própria

Terceiro, é hora de estabelecer os limites do gráfico, isto é, dizer até aonde a imagem irá mostrar a escala de x e y. Para isso basta usar as funções: `ylim()` e `xlim()`. Essa parte você pode escolher a vontade, mas lembre-se de estar em contexto com a função dada. Neste caso vamos fazer com que y vá de -10 a 15 e x de 0 a 10 (Se você inverter a ordem do y o gráfico ficará de cabeça para baixo, se você inverter a ordem do x o gráfico ficará espelhado). Se caso você se esquecer

```

8 from numpy import *
9 from matplotlib.pyplot import *
10
11
12 f = poly1d([1,-10,15])
13
14 x = linspace(0,10,10)
15
16 y = polyval(f,x)
17
18 figure()
19
20 plot(x,y)
21
22 ylim(-10,15)
23
24 xlim(0,10)

```

de colocar um dos dois limites, o Python irá reconhecer o limite como de 0 a 10.

Figura 121: Exemplo louco quarto passo b)

Fonte: Autoria própria

Quarto, isso que vou lhe falar é opcional, mas é importante, você pode colocar um nome para os eixos que apareceram na imagem, por exemplo: x, eixo x, eixo coordenadas, eixo y etc.... para cada eixo você pode dar um nome a eles usando a função `ylabel()` e `xlabel()`, basta colocar entre aspas o nome que você deseja (eu irei colocar x e y mesmo). Outra coisa legal que se pode fazer é ditar o tamanho da fonte usando um comando chamado “`fontsize`”, depois que você colocar o nome que querer entre aspas basta colocar uma vírgula e escrever o comando e colocar o sinal de igual e escrever o tamanho desejado em inglês que podem ser: `small` ou `large`, o tamanho deve ser escrito entre aspas simples e não duplas.

```

8 from numpy import *
9 from matplotlib.pyplot import *
10
11
12 f = poly1d([1,-10,15])
13
14 x = linspace(0,10,10)
15
16 y = polyval(f,x)
17
18 figure()
19
20 plot(x,y)
21
22 ylim(-10,15)
23
24 xlim(0,10)
25
26 xlabel("x",fontsize='large')
27
28 ylabel("y", fontsize='large')
29

```

Figura 122: Exemplo louco quarto passo c)

Fonte: Autoria própria

Por último temos duas opções que tornam o seu gráfico melhor. A primeira é a função "grid()", essa função tem por objetivo ativar ou não as linhas de grade que aparecem no fundo do gráfico, basta colocar dentro dela "True" ou "False" (primeira letra deve ser maiúscula), no meu caso vou optar por ativar as linhas de grade. A segunda opção é colocar um título para o gráfico usando a função "title()" é só escrever o seu título entre as aspas dentro da função e pronto, eu colocarei o nome de "Exemplo louco".

```

8 from numpy import *
9 from matplotlib.pyplot import *
10
11
12 f = poly1d([1,-10,15])
13
14 x = linspace(0,10,10)
15
16 y = polyval(f,x)
17
18 figure()
19
20 plot(x,y)
21
22 ylim(-10,15)
23
24 xlim(0,10)
25
26 xlabel("x",fontsize='large')
27
28 ylabel("y", fontsize='large')
29
30 grid(True)
31
32 title("Exemplo louco")
33

```

Figura 123: Exemplo louco quarto passo d)

Fonte: Autoria própria

E no resultado final veja sua obra prima, você pode usar esse seu conhecimento ao seu favor em muitas coisas importantes, não jogue este conhecimento no lixo, você vai precisar disso. Ao você ativar seu script o gráfico irá aparecer a você numa janelinha, nessa janelinha você pode optar por salvar a imagem em algum lugar do seu computador ou simplesmente só dar uma olhada no gráfico e fechar sem salvar. Veja o gráfico abaixo.

Exemplo louco

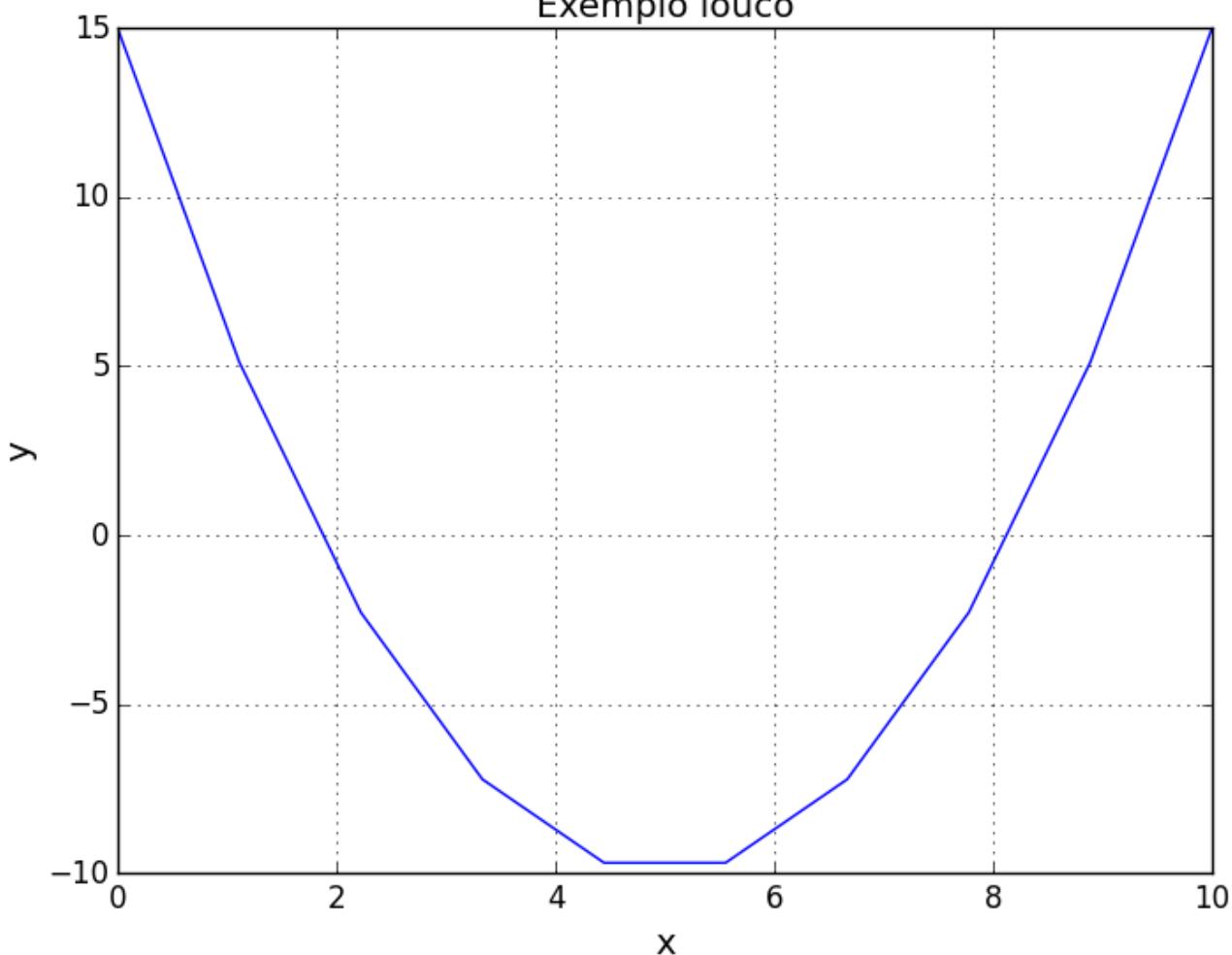


Figura I24: Gráfico do exemplo louco

Fonte: Autoria própria

5.4 Questões Resolvidas

Estas questões foram desenvolvidas pela ICOMP um instituto da UFAM e os algoritmos (scripts) apresentados aqui pertencem ao ICOMP.

Questão 1 – Vetor de zeros

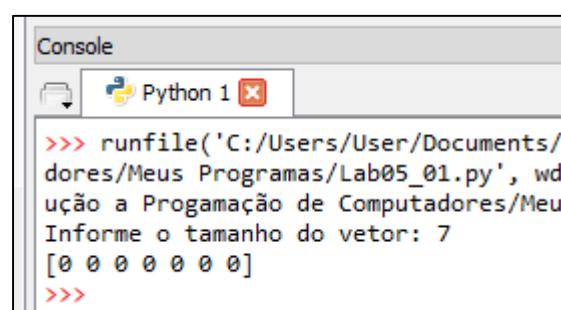
Escreva um script que crie e imprima um vetor inteiro de N posições, todas iguais a zero, sendo N um número natural informado pelo usuário via teclado. Como o vetor possui elementos do tipo inteiro, os zeros não podem estar seguidos do ponto indicador de separação de casas decimais.

Primeiro passo: Interpretação da questão

Depois de tanta explicação lá em cima essa está fácil demais, a questão pede simplesmente para que o usuário insira um número que será o tamanho de um vetor formado somente por zeros.

Acho que você já deve saber que o comando "zeros" faz justamente isso, você deve abrir o comando zeros e dentro dos parênteses coloca-se a quantidade de valores dentro do vetor e separado por uma vírgula deve-se colocar o comando `dtype=int` também.

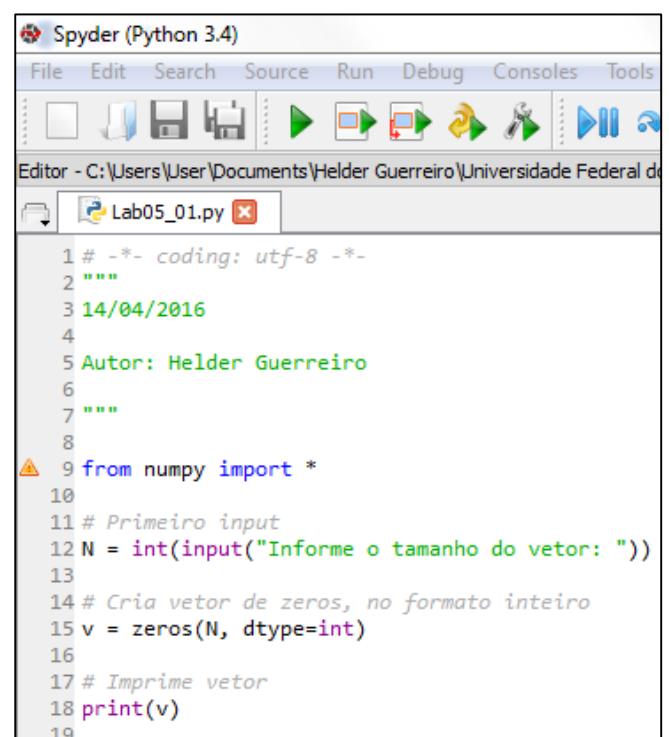
Acione a biblioteca numpy no início do script, faça uma variável (chamada `N`) que irá perguntar ao usuário qual será o tamanho deseja do vetor nulo. Logo abaixo crie o vetor nulo, crie a variável chamada "`v`" e acione o comando zeros com a variável `N` e o comando `dtype=int` dentro dele.



```
>>> runfile('C:/Users/User/Documents/dores/Meus Programas/Lab05_01.py', wdução a Progamação de Computadores/Meu Informe o tamanho do vetor: 7
[0 0 0 0 0 0 0]
>>>
```

Figura 126: Resultado questão 1E

Fonte: Autoria própria



```
# -*- coding: utf-8 -*-
"""
14/04/2016
Autor: Helder Guerreiro
"""

from numpy import *
# Primeiro input
N = int(input("Informe o tamanho do vetor: "))
# Cria vetor de zeros, no formato inteiro
v = zeros(N, dtype=int)
# Imprime vetor
print(v)
```

Figura 126: Questão 1E Primeiro Passo

Fonte: ICOMP, UFAM

Questão 2 – Vetor de uns

Escreva um script que crie e imprima um vetor inteiro de N posições, todas iguais a 1, sendo N um número natural informado pelo usuário via teclado. Como o vetor possui elementos do tipo inteiro, os 1s não deverão estar seguidos do ponto indicador de separação de casas decimais.

DICAS:

1. Modifique o script do problema anterior.
2. Para a entrada 5, a saída deverá ser [1 1 1 1], incluindo os colchetes.

Primeiro passo: Interpretação da questão

Esta questão é a mesma coisa da anterior, a única diferença aqui é que o comando não é mais o “zeros” e sim o comando “ones”. Acho quem não tenho muito mais o que falar né? É só você acionar o módulo numpy, construir uma variável N que venha requisitar do usuário o tamanho desejado do vetor, construir o vetor usando o comando ones e o comando dtype=int e no final usar o print para emitir o resultado.

The screenshot shows the Spyder Python 3.4 IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, and Tools. Below the menu is a toolbar with various icons. The main area is titled 'Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do' and shows the file 'Lab05_02.py'. The code is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 15/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Primeiro input
12 N = int(input("Informe o tamanho do vetor: "))
13
14 # Cria vetor de zeros, no formato inteiro
15 v = ones(N, dtype=int)
16
17 # Imprime vetor
18 print(v)
19
20
```

The screenshot shows the Python 1 console window. The command >>> runfile('C:/Users/User/Documents/dores/Meus Programas/Lab05_02.py', 1) is entered, followed by the user's input 'Informe o tamanho do vetor: 7'. The console then displays the output [1 1 1 1 1 1 1].

Figura I27: Questão 2E Primeiro Passo

Fonte: ICOMP, UFAM

Questão 3 - Geração de vetores linearmente espaçados

Escreva um programa que leia o número natural N inserido pelo usuário a partir do teclado. Em seguida, o programa deverá imprimir, na seguinte ordem:

1. Um vetor crescente de inteiros, de 1 até N (inclusive).
2. Um vetor crescente de inteiros, de 0 até N (inclusive).
3. Um vetor decrescente de inteiros, de N até 1 (inclusive).
4. Um vetor decrescente de inteiros, de N até 0 (inclusive).

DICAS:

1. Use a função `linspace()` da biblioteca `numpy`.
2. Para a entrada 5, a saída deverá ser a seguinte:

[12345]

[012345]

[54321]

[543210]

Primeiro passo: Interpretação da questão

Aqui não temos nenhum bicho de sete cabeças, vamos ver com calma como vamos resolver essa questão. Use o mesmo pensamento que você teve nas questões anteriores, você deverá criar um vetor que irá ter como tamanho a variável que irá perguntar do usuário qual o tamanho desejado. Você se lembra do `linspace()`? Então nós iremos usá-lo aqui por que ele é o responsável pela criação de vetores num determinado intervalo e os intervalos são aqueles apresentados de 1 a 4 no enunciado da questão.

Vou explicar novamente como funciona o `linspace()`: esse cara é uma função que irá produzir um vetor de números inteiros (com o `dtype=int`) ou não, as configurações do `linspace()` são separadas por vírgulas em que dentro dele deve ser colocar o intervalo desejado de onde até aonde, e, caso queira, tornar estes números inteiros (irá contar os números 1, 2, 3...) ou não (irá contar os números quebrados 1.1, 1.2, 1.3....). Dentro do `linspace()` o primeiro número a se colocar é o ponto de partida do intervalo (lembre-se: o intervalo do `linspace()` é um intervalo fechado, isso quer dizer que o número que você escolher como ponto de partida será contado também no intervalo) e o segundo número é o ponto final (que também será contado), o terceiro número é aquele que irá dizer quantos valores devem existir dentro do vetor independente do intervalo,

Por que tem um tal de inclusive ao lado dos intervalos dados pelo enunciado? Por que a questão está enfatizando que tanto o primeiro número como o último devem fazer parte da contagem do intervalo, eu sei que acabei de lhe falar que

o intervalo no `linspace()` é propriamente fechado, só que a casos em que uma das extremidades não entra na contagem e nós precisamos dar uma forcinha, vou dar um exemplo logo abaixo.

Temos a criação de um vetor que terá o intervalo de 1 a 5 com quantidade de valores 5, ou seja, nosso vetor será assim: $v = [1, 2, 3, 4, 5]$, mas se fizermos de 0 a 5 com quantidade de valores 5 teremos um problema, pois o zero será contado como um número e no final das contas teremos seis valores lá dentro enquanto o `linspace()` está ditando somente cinco valores, como consequência o 5 vai acabar fora do intervalo.

Então como fazemos para criar um vetor sem esse problema? Perceba que quando começamos do número 1 o mesmo número que fecha o intervalo é o mesmo número que dita a quantidade de valores dentro do vetor, ou seja, o mesmo N que irá determinar o final será o mesmo N que irá determinar o tamanho do vetor, então na hora de construir seu `linspace()` coloque o intervalo, por exemplo no primeiro intervalo, de 1 a N e o tamanho do vetor também será N, mas no caso como o segundo intervalo que é de 0 a N o tamanho do vetor será o próprio N mais uma unidade, por que se temos um intervalo de 0 a 5 a quantidade de valores é 6, se temos um intervalo de 0 a 7 a quantidade de valores é 8 e assim vai...

Nos casos em que os vetores estiverem em ordem decrescente basta inverter a ordem em que você coloca os intervalos e pronto, não esqueça de colocar o comando `dtype=int` em cada `linspace()` para que os valores sejam tidos como inteiros. Faça um vetor para cada intervalo requerido pela questão e um `print` para cada um, não precisa mudar as variáveis se quiser, coloque os vetores em ordem e um `print` debaixo de cada um, desse jeito não haverá confusão quanto às variáveis.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do
Lab05_03.py
1 # -*- coding: utf-8 -*-
2 """
3 15/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Primeiro input
12 N = int(input("Informe o tamanho do vetor: "))
13
14 # Cria vetor de 1 a N
15 v = linspace(1, N, N, dtype=int)
16 print(v)
17
18 # Cria vetor de 0 a N
19 v = linspace(0, N, N+1, dtype=int)
20 print(v)
21
22 # Cria vetor de N a 1
23 v = linspace(N, 1, N, dtype=int)
24 print(v)
25
26 # Cria vetor de N a 0
27 v = linspace(N, 0, N+1, dtype=int)
28 print(v)
29

```

Figura 129: Questão 3E Primeiro Passo

Fonte: ICOMP, UFAM

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/dores/Meus Programas/Lab05_03.py', wdução a Progamação de Computadores/Meu Informe o tamanho do vetor: 5
[1 2 3 4 5]
[0 1 2 3 4 5]
[5 4 3 2 1]
[5 4 3 2 1 0]
>>>

```

Figura 130: Resultado questão 3E

Fonte: Autoria própria

Questão 4 – Informações básicas de um vetor

Escreva um script em Python que leia um vetor digitado pelo usuário. Em seguida, o script deverá imprimir as seguintes informações sobre o vetor, necessariamente na ordem indicada:

- 1. Número de elementos do vetor*
- 2. Valor do primeiro elemento do vetor*
- 3. Valor do último elemento do vetor*
- 4. Maior elemento do vetor*
- 5. Menor elemento do vetor*
- 6. Soma dos elementos do vetor*
- 7. Média aritmética dos elementos do vetor (com duas casas decimais de precisão).*

DICAS:

- 1. Use as funções `size()`, `max()`, `min()` e `sum()`, da biblioteca `numpy`.*
- 2. Use a função `round(x, n)`, nativa da linguagem Python, para fazer arredondamento do número `x` com até `n` casas decimais de precisão.*
- 3. Para a entrada `[20, 10, 50, 40, 30]`, a saída deverá ser a seguinte: `5 20 30 50 10 150 30.0`, indicando que o vetor tem 5 elementos, sendo 20 o primeiro elemento e 30 o último; o maior elemento do vetor é 50 e o menor 10; a soma de todos os elementos do vetor é 150, e a média aritmética dos elementos do vetor vale 30.0.*

Primeiro passo: Interpretação da questão

Esta questão está aqui para exercitarmos o uso das funções que podem tirar informações dos vetores para o usuário, são funções simples assim como esta questão, a resolução dela é simplesmente a aplicação dessas funções.

A primeira informação é ter o número de elementos do vetor, isso quer dizer que devemos saber o tamanho desse vetor, isso pode ser facilmente feito usando a função `size()`, eu expliquei sobre no tópico de Funções e Operações de Vetores. Para fazer esta questão não precisa criar uma variável e igualá-la a função `size()` com o vetor dentro, você pode simplesmente colocar a função com o vetor dentro da própria função `print` sem nenhum problema (é mais rápido assim).

A segunda informação é o valor do primeiro elemento do vetor, também expliquei isso mais acima, sempre quando você quiser saber o primeiro elemento de qualquer vetor basta aplicar o comando "`vetor[0]`" (esse "vetor" é o nome do seu vetor, não precisa ser necessariamente a palavra vetor), então aplicando esse comando dentro da função `print` você já terá sua resposta pronta.

A terceira informação pede o último elemento que segue a mesma ideia do primeiro elemento, em vez de você usar o número zero use o número `-1` que lhe sairá a informação do valor do último elemento de qualquer vetor que seja. Lembre-se a contagem dos elementos do vetor é em ordem crescente a partir do zero, ou em ordem decrescente a partir do `-1` o

zero significa o primeiro elemento e o negativo um significa o último, se você for usar o comando `vetor[]` com qualquer outro número sem ser esses dois você estará selecionando qualquer outro elemento do vetor.

A quarta informação é o maior elemento do vetor, já expliquei como acha-lo, mas falarei de novo. Existe uma função específica para encontrar o maior elemento de qualquer vetor, a função `max()` basta colocá-la dentro da função `print` que o resultado será mostrado.

A quinta informação fala do menor elemento, ou seja, é a mesma questão da anterior a diferença que estamos procurando agora o menor elemento. A função `min()` é a oposta da função `max()` e faz o mesmo trabalho da `max()` só que oposto, então colocando essa função no `print` já resolve o que a quinta informação pede.

A sexta já pede a soma de todos os elementos, esta também é simples demais, pois quando se tem o conhecimento das funções de vetores se tem metade do caminho andado. A função `sum()` tem esse papel de pegar todos os elementos de qualquer vetor e soma-los.

A última já foi explicada antes de entrarmos no assunto de gráficos mais acima, mas dessa vez darei uma dica muito importante que vale a pena lembrar, em vez de você fazer o procedimento padrão e colocar soma pela quantidade, ou seja, `sum()/size()` tenho uma outra função a lhe apresentar que mudará isso para mais fácil, a função `mean()` faz esse papel sem precisar colocar divisão nem soma de nada basta usar a função e pronto! Não esqueça de usar a função `round` dentro do `print`, coloque a função `mean()` dentro da `round`.

```
>>> runfile('C:/Users/User/Documents/dores/Meus Programas/Lab05_04.py', wdd)
Leitura a Progamação de Computadores/Meu
Informe o vetor: 20,10,50,40,30
5
20
30
50
10
150
30.0
>>>
```

Figura 132: Resultado questão 3E

Fonte: Autoria própria

```
# -*- coding: utf-8 -*-
"""
18/04/2016
Autor: Helder Guerreiro
"""

from numpy import *
# Leitura do vetor
vetor = array(eval(input("Informe o vetor: ")))
# Tamanho de um vetor
print(size(vetor))
# Primeiro elemento de um vetor
print(vetor[0])
# Ultimo elemento de um vetor
print(vetor[-1])
# Maior valor de um vetor
print(max(vetor))
# Menor valor de um vetor
print(min(vetor))
# Soma dos elementos de um vetor
print(sum(vetor))
# Media dos elementos de um vetor
print(round(sum(vetor) / size(vetor), 2))
print(round(mean(vetor), 2))
```

Figura 131: Questão 4E Primeiro Passo

Fonte: ICOMP, UFAM

Questão 5 - Quantas ocorrências? (I)

Escreva um programa que leia um vetor digitado pelo usuário. Em seguida, o programa deve percorrer o vetor e imprimir cada posição cujo elemento for igual a 5. No final, o programa deve imprimir quantas ocorrências tiveram do número 5 procurado.

DICAS:

1. Utilize duas variáveis contadoras: uma para controlar a posição do vetor e outra para contar o número de ocorrências do valor procurado.
2. Para a sequência de entrada [5,11,5,22,5], a saída deverá ser 0 2 4 3, indicando que o valor foi encontrado nas posições 0, 2 e 4 do vetor, num total de 3 ocorrências.

Primeiro passo: Interpretação da questão

Nesta questão vamos trabalhar junto com o `while` e com o `if`, é simples não precisa se desesperar. O que devemos fazer primeiramente é fazer o `while` percorrer o vetor totalmente, mas como assim? Você deve saber que através de uma condição o `while` irá repetir seu `mine script`, então e se essa condição for que o `while` deva percorrer todo o vetor? Vamos pensar devagar, você sabe que o comando `vetor[i]` representar o valor de um elemento do vetor com o índice “*i*” né? Então, o índice começa do 0 e ao mudar esse índice crescentemente o comando `vetor[]` irá ficar pulando de elemento em elemento até chegar no último, mas o `while` não tem como saber quem é o último então por isso deve-se usar algo que sirva de limite para a repetição do `while`, ou seja, ENQUANTO o índice não chegar no limite o `while` continuará repetindo seu `mine script`, você se lembra de quem é o responsável em determinar quantos elementos existem dentro de um vetor? Sim estamos falando da função `size()` ela irá dizer qual o tamanho do vetor e claramente o índice não pode ser maior que o tamanho do seu vetor. Vale lembrar a você que o que entrará na condição será o índice e não o valor dele.

Você já deve ter percebido que para isso funcionar o índice “*i*” deve mudar de número toda vez que o `while` repetir sua ação, é claro que você já deve ter sacado (ou não, né) que para fazer isso basta fazermos que nem uma variável contadora, a cada final de repetição o índice “*i*” será somado pelo número um aumentando crescentemente o seu valor a cada repetição.

Lembre-se do que a questão está lhe pedindo, temos que saber o número de vezes que encontramos o número 5 dentro de um vetor. Como fazer isso? Tudo que eu expliquei acima foi como andar por todo o vetor usando o `while`, mas para identificar quem estamos procurando precisamos usar o `if`. Com o comando `vetor[i]` podemos identificar o valor de um certo elemento de índice “*i*”, ou seja, ao passo que o índice vai aumentando, o mesmo vai percorrendo por dentro do vetor e dessa forma usando o comando `vetor[i]` podemos saber o valor de cada elemento do vetor do primeiro até o último. Então SE o elemento de índice “*i*” for igual a 5 já teremos a presença do 5 uma vez, SE outro elemento também for igual a 5 já teremos duas contagens e assim se segue...

Para que a presença do 5 seja contada teremos que ter o apoio da variável contadora (o *i* não é uma variável contadora, ele é o corredor do vetor) e essa variável deve estar dentro do `mine script` da condicional `if`, toda vez que a condição do

elemento igual a 5 for respeitada o meu script será ativado e a variável contadora irá aumentar somando a si mesma a uma unidade.

Para fazer a condição do if use o comando `vetor[i]` que irá dizer o valor do elemento na posição "i" e use o operador condicional "`==`" para igualá-lo a 5 (você se lembra dos operadores condicionais?). Neste caso não há necessidade de usar o `else` isso por que não há necessidade de colocar uma exceção, SE a condição não for respeitada o Python irá pular para o meu script do while e rodar normalmente.

Como a questão pede para mostrar qual a posição em que o 5 se encontra então devemos colocar um `print` dentro do meu script do if em que toda vez que tal posição for satisfeita então ela será mostrada pelo print, e no final de tudo depois que o while repetir todas as vezes necessárias deveremos mostrar quantas vezes o 5 apareceu e essa informação quem carrega é a variável contadora, ou seja, devemos usar o `print`.

Agora irei falar de dúvidas simples que podem "cagar" o seu script. A condição do while é que o índice (`i`) não ultrapasse o tamanho do vetor analisado (`size()`), mas devemos usar o "menor que" ou "menor ou igual que"? Veja o seguinte, o índice deve ser contado a partir do número 0 por que é daí que começa a contagem dos elementos do vetor, porém a função `size()` não conta o tamanho do seu vetor a partir do 0, a função `size()` dizer QUANTOS elementos existem dentro do vetor sem considerar posição, então por isso a condição deve ser escrita assim: `i < size()`, por que assim estaremos igualando a diferença que existe entre os dois.

```
Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools
Editor - C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Rio Grande do Sul/Lab05_04.py
1 # -*- coding: utf-8 -*-
2 """
3 18/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Leitura do vetor
12 vet = array(eval(input("Informe o vetor: ")))
13
14 # Contadores
15 i = 0
16 cont = 0
17
18
19 while (i < size(vet)):
20     if (vet[i] == 5):
21         cont = cont + 1
22         print(i)
23     i = i + 1
24
25 print(cont)
26
```

Figura 133: Questão 5E Primeiro Passo

Fonte: ICOMP, UFAM

```
Console
Python 1
>>> runfile('C:/Users/User/Documentos/Meus Programas/Lab05_04.py')
Informe o vetor: 5,11,5,22,5
0
2
4
3
>>>
```

Figura 134: Resultado questão 5E

Fonte: Autoria própria

Questão 6 - Quantas ocorrências? (2)

*Escreva um programa que leia um vetor digitado pelo usuário. Em seguida, o programa deve percorrer o vetor e imprimir cada posição cujo elemento for **maior** que um número N informado via teclado. No final, o programa deve imprimir quantas ocorrências foram encontradas.*

DICAS:

1. Utilize duas variáveis contadoras: uma para controlar a posição do vetor e outra para contar o número de ocorrências do valor procurado.
2. Modifique o código da questão anterior.
3. Para a sequência de entrada [5,11,5,22,5] 5, a saída deverá ser 1 3 2, indicando que as posições 1 e 3 do vetor contêm valores maiores que 5, num total de 2 ocorrências.

Primeiro passo: Interpretação da questão

Esta questão é muito parecida com a anterior então pegue o script da questão anterior que nós vamos editá-la de acordo com o que se pede nesta questão 6. A única mudança aqui é a que o valor procurado deve ser maior que o número inserido pelo usuário, ou seja, se o usuário insere 3 então os números a partir de 4 para cima farão parte da nova condição da questão.

Então, o que vai mudar é quem será procurado e como procura-lo, neste caso teremos duas entradas que é o vetor e um "limiar" limiar é uma palavra da língua portuguesa se você não sabe que significa; aquilo que inicia, ou seja, esse número que dará a ordem do script reconhecer como satisfatório todos os números depois dele. A condição de procura do **if** também vai mudar claro, na questão anterior a condição era que o valor fosse igual a 5 e agora o valor deve ser MAIOR QUE o limiar... (acho que você já sacou como escrever essa condição).

The screenshot shows the Spyder Python 3.4 IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, and Tool. Below the menu is a toolbar with various icons. The main area is titled 'Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal' and contains the Python code for Question 6E. The code uses the numpy library to count elements in a vector greater than a given limit.

```
1 # -*- coding: utf-8 -*-
2 """
3 18/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Leitura do vetor
12 vet = array(eval(input("Informe o vetor: ")))
13 limiar = int(input("Digite o limiar: "))
14
15 # Contadores
16 i = 0
17 cont = 0
18
19 while (i < size(vet)):
20     if (vet[i] > limiar):
21         cont = cont + 1
22         print(i)
23     i = i + 1
24
25 print(cont)
26
```

Figura 135: Questão 6E Primeiro Passo

Fonte: ICOMP, UFAM

The screenshot shows the Python 1 console window. It displays the command `>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal/Meus Programas')`, followed by the user input `Informe o vetor: 5,11,5,22,5`. The program then prints the elements greater than 5, resulting in the output `1`, `3`, and `2`.

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal/Meus Programas')
Informe o vetor: 5,11,5,22,5
1
3
2
>>>
```

Figura 136: Resultado questão 6E

Fonte: Autoria própria

Questão 7 - Qual a porcentagem de acerto?

As respostas de um candidato a uma prova de múltipla escolha são guardadas em um vetor. O gabarito é guardado em outro vetor de mesmo tamanho. Escreva um programa que leia o vetor de respostas e o vetor do gabarito, nessa ordem, e imprima na saída a porcentagem de acerto do candidato, com até duas casas decimais de precisão. Considere que os dois vetores inseridos são sempre do mesmo tamanho e sempre têm um ou mais elementos.

DICAS:

1. Utilize duas variáveis contadoras: uma para controlar a posição do vetor e outra para contar o número de ocorrências em que um elemento da posição i de um vetor é igual ao elemento de mesma posição no outro vetor.
2. Porcentagem é um valor entre 0 e 100.
3. Use a função `round(x, n)`, nativa da linguagem Python, para fazer arredondamento do número x com até n casas decimais de precisão.
4. Eis alguns exemplos (não exaustivos) de entradas e saídas:
 - a. Para a entrada `[1, 2, 3, 4, 5] [1, 2, 3, 1, 1]`, a saída deverá ser `60.0`, indicando que o candidato acertou 3 entre cinco questões, ou seja `60,0%`.
 - b. Para a entrada `[1, 2, 3] [2, 3, 1]`, a saída deverá ser `0.0`, indicando que o candidato não acertou nenhuma das três cinco questões.
 - c. Para a entrada `[1, 2, 3, 4, 5, 2, 3, 4] [1, 2, 3, 4, 5, 2, 3, 4]`, a saída deverá ser `100.0`, indicando que o candidato acertou todas as oito questões

Primeiro passo: Interpretação da questão

Nesta questão vamos aprender a trabalhar com dois vetores ao mesmo tempo, um vetor será a resposta e o outro vetor será o gabarito dessa resposta. Já vou logo avisando que esta questão tem o mesmo objetivo que as duas anteriores, na quinta questão estávamos passeando pelo vetor a procura do número 5, dessa vez vamos procurar dentro de outro vetor.

Vou lhe dizer como vai funcionar, o vetor das respostas será o vetor a ser analisado, ou seja, vamos usar o `while` para passear dentro dele, e o outro vetor será o vetor que irá conter os valores que irão dizer se as respostas estão certas ou não, o `if` irá checar o vetor das respostas e usar o vetor do gabarito como base, sacou? É claro que para que as respostas estejam certas elas devem ser iguais ao gabarito.

Não se preocupe, o índice "i" continuará a ser usado normalmente, não precisamos de dois índices por que o índice é usado para passear pelo vetor e só vamos passear e um, a variável contadora continuará no mesmo script do if, isso porque a cada vez que a condição for respeitada, ou seja, uma resposta for acertada a variável contadora irá se encarregar de contabilizar o acerto toda vez que acontecer.

A condição do `while` vai seguir o mesmo esquema para passear no vetor das respostas, o índice não deve ser maior que o tamanho do seu vetor. E no `if` usaremos a condição que o elemento na posição "i" do vetor resposta deve ser

igual a outro elemento de mesma posição do vetor gabarito, dessa forma temos uma questão certa e o mine script do if irá rodar. Para fazer a condição do if use o comando “vetor[i]” para os dois vetores, lembre-se que o valor do elemento na posição “i” só será revelado se esse comando for usado.

Depois de tudo isso a questão nos pede que façamos a porcentagem dos acertos do vetor resposta, para fazer isso é a mesma coisa que você faria numa calculadora comum: você pega a quantidade de acertos e divide pelo total, ou seja, pegue a variável contadora e faça a divisão pelo tamanho do vetor resposta, não esqueça de multiplicar por 100 para dar a porcentagem e usar a função **round** com duas casas decimais.



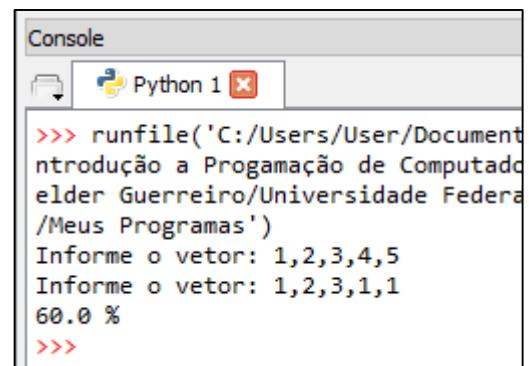
```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\Meus Programas\Lab05_07.py
1 # -*- coding: utf-8 -*-
2 """
3 18/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Leitura do vetor de respostas
12 resp = array(eval(input("Informe o vetor: ")))
13
14 # Leitura do vetor de gabarito
15 gabarito = array(eval(input("Informe o vetor: ")))
16
17 # Contadores
18 i = 0
19 cont = 0
20
21 while (i < size(resp)):
22     if (resp[i] == gabarito[i]):
23         cont = cont + 1
24     i = i + 1
25
26 acertos = cont/size(resp) * 100.
27
28 print(round(acertos, 2), "%")
29

```

Figura 137: Questão 7E Primeiro Passo

Fonte: ICOMP, UFAM



```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Introdução a Programação de Computadores/Helder Guerreiro/Universidade Federal do Amazonas/Meus Programas/Lab05_07.py')
Informe o vetor: 1,2,3,4,5
Informe o vetor: 1,2,3,1,1
60.0 %
>>>

```

Figura 138: Resultado questão 7E

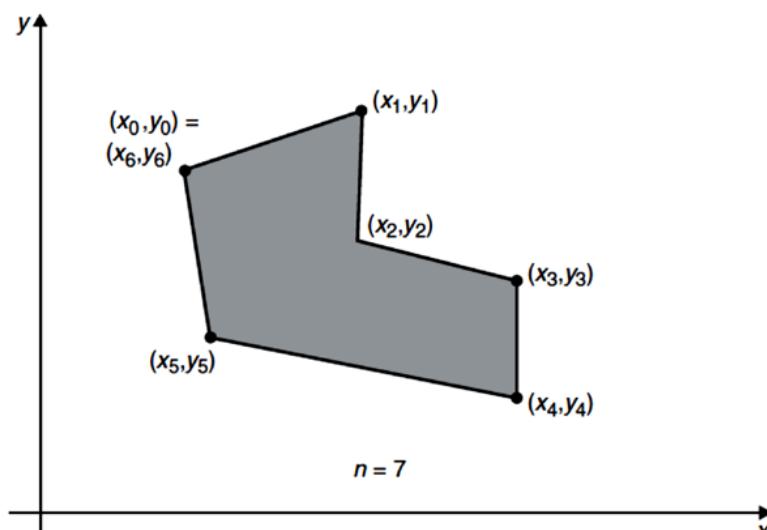
Fonte: Autoria própria

Questão 8 - Área de um polígono

Se "n" pontos estão ligados formando um polígono fechado, como mostrado abaixo, a área A do polígono pode ser determinada por:

$$A = \frac{1}{2} \left| \sum_{i=0}^{n-2} (x_{i+1} + x_i)(y_{i+1} - y_i) \right|$$

Observe que, embora o polígono ilustrado abaixo tenha apenas seis vértices distintos, temos que $n=7$ para este polígono, pois o algoritmo espera que o último ponto, (x_6, y_6) , seja uma repetição do ponto inicial, (x_0, y_0) .



Escreva um programa que receba como entrada as coordenadas (x, y) dos vértices do polígono como dois vetores de valores reais, um contendo os valores das abscissas (x) e o outro vetor contendo os valores das respectivas ordenadas (y). Na saída, deve-se informar a área do polígono, com até quatro casas decimais de precisão.

Para simplificar o código, considere que:

- Os vetores fornecidos são sempre do mesmo tamanho;
- Cada vetor possui pelo menos quatro elementos (o que garante pelo menos a existência de um triângulo); e
- O primeiro e o último elementos de cada vetor de entrada são sempre iguais.

DICAS

1. Utilize duas variáveis de laço: uma para controlar a posição dos vetores e outra para acumular o valor parcial da área.
2. Aplique a função módulo `abs()` apenas sobre o resultado do somatório, conforme a fórmula fornecida, ou seja, somente após o fim do laço `while`.
3. Atenção para a condição de parada do somatório: $n - 2$.
4. Por exemplo, para a entrada `[4,4,7,7,9,7,4] [0,7,5,7,5,3,0,0,0]`, a saída deve ser `25.5`.

Primeiro passo: Interpretação da questão

Acalma seu coração, tem muita gente que se desespera por causa do tamanho do enunciado, você leu a questão lá em cima ou pulou logo para a resolução? Se você não leu, como você quer aprender se você joga tudo nas minhas costas? Isso aqui não é uma apostila de leitura é uma apostila de aprendizagem, abra seu Python e comece a produzir seu script para começarmos a pensar na questão.

Esta questão é dada na sua cara só que você precisa de olhos de um programador para enxergar a resposta que está a sua frente. Perceba uma coisa, temos várias abscissas e ordenadas formando várias coordenadas e isso é sinal de que teremos dois vetores, um vetor para as abscissas e outro para as ordenadas.

Você percebe logo de cara que a parte mais importante dessa questão é essa fórmula do somatório que resulta na área. Acho que você já deve ter visto um somatório antes não? Mas vamos estuda-lo um pouco para ver como ele funciona?

O somatório é representado pela letra grega Σ (sigma), ele representa um sistema de uma soma organizada que funciona assim: o somatório carrega duas informações, uma em cima dele e outra em baixo, ao lado dele estará uma fórmula que será modificada de acordo com as ações do somatório.

$$A = \frac{1}{2} \left| \sum_{i=0}^{n-2} (x_{i+1} + x_i)(y_{i+1} - y_i) \right|$$

Vamos analisar o somatório que aparece no enunciado da questão: em baixo dele está a informação dizendo $i = 0$, esse é o ponto de partida do somatório, veja na fórmula ao lado que temos a aparição do "i" no meio dela, esse "i" será substituído pelo 0 até a última substituição que é a informação de cima "n - 2", ou seja, o "i" será substituído pelo 0 depois pelo 1, depois pelo 2, depois pelo 3... e no final por "n - 2". Ao longo que o "i" for substituído a fórmula será somada cada vez mais, isso quer dizer que quando o $i = 0$ a fórmula toda será substituída e somado todos os valores, depois quando o somatório seguir para $i = 1$ a fórmula será repetida e substituída novamente pelos valores atuais e depois somada, o resultado dessa soma irá somar com os outros resultados, assim:

$$(x_{0+1} + x_0)(y_{0+1} - y_0) + (x_{1+1} + x_1)(y_{1+1} - y_1) + \dots + (x_{(n-2)+1} + x_{n-2})(y_{(n-2)+1} - y_{n-2})$$

Para quem já tem uma certa prática no Python já deve ter sacado que esse somatório é praticamente meio caminho andado do script, o somatório age da mesma forma quando usamos o `while` para percorrer os vetores, quer ver como é verdade?

Usando o `while` nós podemos percorrer ambos os vetores, pois o `while` irá atualizar o índice "i" toda vez que seu script for repetido, e como o índice serve tanto para vetor das abscissas quanto para as ordenadas já que o que diferencia se o índice é de um vetor ou de outro é o comando "`vetor[i]`", enquanto esse comando não for usado o índice "i" é neutro e faz parte de qualquer vetor.

Veja que no exemplo quando a abscissa é 6 a ordenada também é 6, isso quer dizer que não importa qual o valor da abscissa a ordenada será o mesmo valor também e vice-versa, ou seja, o índice da abscissa será o mesmo da ordenada isso quer dizer que ao usar o `while` podemos trabalhar com o X e Y ao mesmo tempo. Não se preocupe, não vamos usar o `if` aqui, por que a única condição que temos aqui é a do `while`, não vejo outra condição para usar o `if`.

No seu script chame o vetor das abscissas de X e o vetor das ordenadas de Y, por que desse jeito ao construir aquela formula que fica ao lado do somatório a aparência será a mesma! Isso quer dizer que do jeito que está na formula será o jeito que você irá fazer a formula dentro do script.

TODA VEZ QUE VOCÊ SE DEPARAR COM UM SOMATÓRIO ele terá essas configurações: o somatório é como se fosse o while, o ponto de partida do somatório (a informação de baixo) junto com a última substituição (a informação de cima) forma a condição do while que ficará assim: Ponto de partida < Ponto de término. A formula que fica do lado somatório será o mine script do while, essa formula deverá se comportar como variável acumuladora, claramente por que essa formula será somada várias vezes então ela deve acumular o resultado de suas somas, no final do mine script não esqueça de colocar o índice "i" para atualizar. Neste caso a variável acumuladora é a área, por que o somatório dessa formula irá representar parte do resultado da área do polígono (não é o resultado completo, ainda tem que dividir por 2). TUDO que estiver do lado esquerdo do somatório não entra no mine script do while, essas partes devem ser resolvidas após o cálculo do somatório, assim como neste caso em que o cálculo do somatório deve ser dividido por 2, vamos dividir o resultado da variável acumuladora área fora dos domínios do while.

ESTA IMAGEM MOSTRA A VOCÊ QUE TODA VEZ QUE UM SOMATÓRIO FOR COLOCADO NO SCRIPT DEVE SER FEITO ASSIM:

$$A = \frac{1}{2} \left| \sum_{i=0}^{n-2} (x_{i+1} + x_i)(y_{i+1} - y_i) \right|$$


```
# Leitura do vetor de abscissas
x = array(eval(input("Vetor das abscissas: ")))

# Leitura do vetor de ordenadas
y = array(eval(input("Vetor das ordenadas: ")))

# Contadores
i    = 0
area = 0

while (i < size(x) - 2):
    area = (x[i+1] + x[i]) * (y[i+1] - y[i]) + area
    i = i + 1

area = abs(area) * 1/2.
```

Figura I39: Somatório no Python

Fonte: Autoria própria

Veja só como é parecido a formula que está ao lado do somatório com a formula que foi colocada dentro do while. Veja que o size(x) tem a mesma ação do "n", isso por que o "n" que foi falado na questão é a quantidade de vértices no polígono, ou seja, os pontos formados pelas abscissas e ordenadas, e o size() é a função que informa o tamanho exato do vetor sem incluir o número zero, pois na imagem apresentada no enunciado o vetor das abscissas e ordenadas são formados por 6 valores, mas a sua contagem começa do 0, o tamanho real dos vetores é 7. A área foi colocada na função **abs** para que o resultado seja mais organizado.

Com isso já fizemos todo o script, em cima coloca-se a biblioteca numpy e embaixo a função **print** com o **round** embutido com quatro casas decimais.

The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. Below the menu is a toolbar with various icons. The main area is titled 'Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas' and contains a file named 'Lab05_08.py'. The code in the file is:

```
1 # -*- coding: utf-8 -*-
2 """
3 19/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Leitura do vetor de abscissas
12 x = array(eval(input("Vetor das abscissas: ")))
13
14 # Leitura do vetor de ordenadas
15 y = array(eval(input("Vetor das ordenadas: ")))
16
17 # Contadores
18 i = 0
19 area = 0
20
21 while (i < size(x) - 2):
22     area = (x[i+1] + x[i]) * (y[i+1] - y[i]) + area
23     i = i + 1
24
25 area = abs(area) * 1/2.
26
27 print(round(area, 4))
28
```

Figura 140: Questão 8E Primeiro Passo

Fonte: ICOMP, UFAM

The screenshot shows the Python 1 console window. The command 'runfile' is used to execute the script 'Lab05_08.py'. The output shows the input vectors and the calculated area.

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/Lab05_08.py')
Vetor das abscissas: 4,4,7,7,9,7,4
Vetor das ordenadas: 0,7.5,7.5,3,0,0,0
25.5
>>>
```

Figura 141: Resultado questão 8E

Fonte: Autoria própria

Questão 9 – Raízes de um polinômio

Escreva um programa que leia um vetor correspondente aos coeficientes de um polinômio P de grau $n > 1$, bem como um valor Z . Na saída, o programa deve informar as raízes do polinômio e o valor de $P(Z)$.

DICAS:

1. No vetor de entrada, não omita os coeficientes $i < n$ que forem iguais a zero. Por exemplo, o polinômio $P(x) = x^3 - 15$ é representado pelo vetor $[1, 0, 0, -15]$.
2. Use a função `poly1d(vet)` para definir um polinômio $P(x)$ a partir de um vetor `vet` contendo seus coeficientes.
3. Use a função `roots(p)` para determinar as raízes de um polinômio p . Observe que a saída desta função também é um vetor.
4. Use a função `polyval(p, x)` para calcular o valor de um polinômio p no ponto x .
5. O Python utiliza a letra j para representar $\sqrt{-1}$.
6. Não é necessário usar nenhuma estrutura de decisão (`if-else`), nem de repetição (`laço while`) para resolver esta questão.
7. Alguns exemplos não exaustivos:
 - Para a entrada $[1, -2] 5$, a saída deve ser $[2, 3]$. Ou seja, para o polinômio de primeiro grau $P(x) = x - 2$ e $Z = 5$, então sua única raiz está guardada no vetor unitário $[2]$, e $P(Z=5) = 3$.
 - Para a entrada $[2, 0, -50] 10$, a saída deve ser $[5, -5, 150]$. Ou seja, o polinômio de segundo grau $P(x) = x^2 - 50$ possui duas raízes reais: 5 e -5 . Além disso, $P(10) = 150$.
 - Para a entrada $[1, 0, -4, 8, 35] 0$, a saída deve ser $[2. +1.7320508j, 2.-1.7320508j, -2.+1j, -2.-1j, 35]$. Ou seja, o polinômio de quarto grau $P(x) = x^4 - 4x^2 + 8x + 35$, possui quatro raízes complexas: $(2+ii\sqrt{3}), (2-ii\sqrt{3}), (-2+ii)$ e $(-2-ii)$. Além disso, $P(0)=35$.

Primeiro passo: Interpretação da questão

Assim de primeira a questão pode assustar, mas eu garanto que ela é bem simples, para resolver esta questão basta o conhecimento de como usar as funções de vetores do Python.

Vamos começando devagar, você sabe que numa questão dessas a primeira coisa a se fazer é importar a biblioteca numpy. Vendo o enunciado temos duas entradas, uma para os coeficientes do polinômio e outra para o chamado ponto Z , esse ponto é simplesmente um valor que será substituído no polinômio e depois o polinômio será calculado.

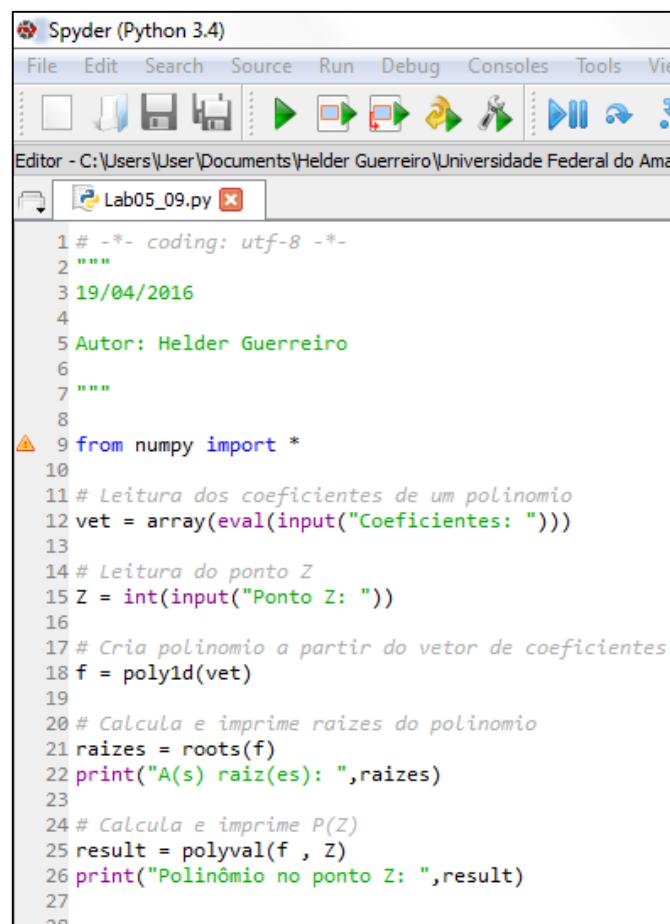
Veja com calma o que a questão está pedindo: é simplesmente inserir um polinômio e desse polinômio será calculado suas raízes e será calculado o polinômio no ponto Z . Vamos lembrar das funções?

Função `poly1d()`, essa é a função que cria um polinômio, dentro dela você deve colocar um vetor (crie o vetor por fora primeiro), ela funciona assim: temos o vetor $V = (1, 2, 0)$ ao colocar na função `poly1d(V)` temos a mesma coisa que escrevermos $X^2 + 2X$. Então usando essa função já temos o polinômio.

Função `root()`, essa função (como diz seu nome inglês) é a função que calcula as raízes dos polinômios é a mesma coisa que fazer Bhaskara e tudo mais, basta colocar a variável do polinômio dentro da função e pronto. Claro que se você quiser ver a resposta tem de usar a função `print`.

Função `polyval()`, seu nome já é bem sugestivo essa é a função que calcula o valor do polinômio num certo ponto (acho que você sabe qual é o ponto que estou falando), basta colocar a variável do polinômio e separar por vírgula o ponto escolhido para o cálculo. Use o `print` para lhe mostrar a resposta.

Viu como é fácil e você aí se assustando com o tamanho do enunciado. Leia primeiro, julgue depois.



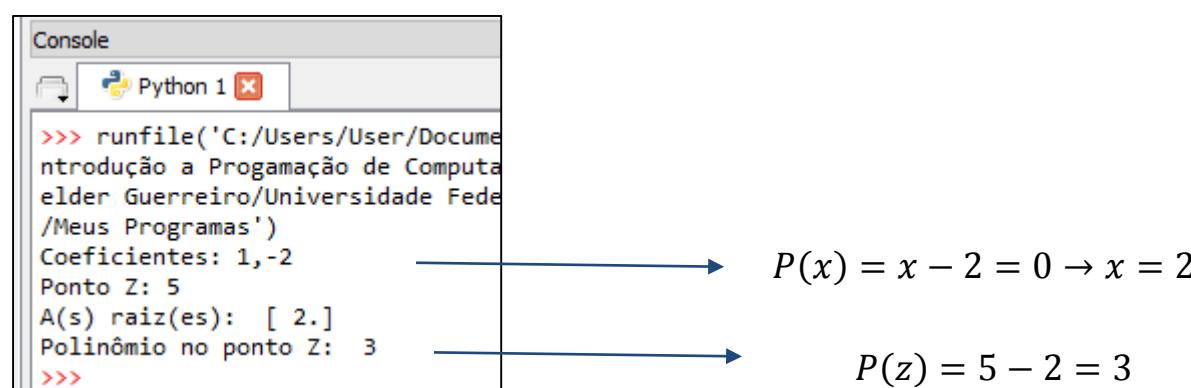
```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\Meus Programas\Lab05_09.py
1 # -*- coding: utf-8 -*-
2 """
3 19/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Leitura dos coeficientes de um polinomio
12 vet = array(eval(input("Coeficientes: ")))
13
14 # Leitura do ponto Z
15 Z = int(input("Ponto Z: "))
16
17 # Cria polinomio a partir do vetor de coeficientes
18 f = poly1d(vet)
19
20 # Calcula e imprime raizes do polinomio
21 raizes = roots(f)
22 print("A(s) raiz(es): ",raizes)
23
24 # Calcula e imprime P(Z)
25 result = polyval(f , Z)
26 print("Polinômio no ponto Z: ",result)
27
28

```

Figura I42: Questão 9E Primeiro Passo

Fonte: ICOMP, UFAM



```

Console
Python 1
>>> runfile('C:/Users/User/Documentos/Introdução a Programação de Computação/Helder Guerreiro/Universidade Federal do Amazonas/Meus Programas')
Coeficientes: 1,-2
Ponto Z: 5
A(s) raiz(es): [ 2.]
Polinômio no ponto Z: 3
>>>

```

$$P(x) = x - 2 = 0 \rightarrow x = 2$$

$$P(z) = 5 - 2 = 3$$

Figura I43: Resultado questão 9E

Fonte: Autoria própria

Questão 10 – Derivada de um polinômio

Suponha que você tenha uma placa quadrada de lado L e deseja construir um recipiente (sem tampa). Para tal, você corta um quadrado de lado x em cada vértice da placa, como mostra as áreas hachuradas na Figura A, e forma uma caixa como aparece na Figura B. Deseja-se saber qual o lado do quadrado (x) a ser cortado a fim de que a caixa tenha volume máximo.

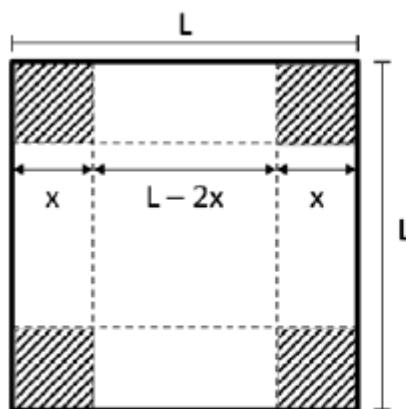


Figura A

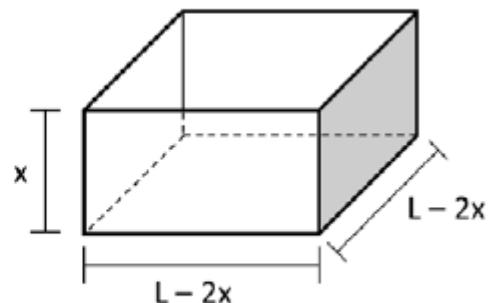


Figura B

Pela Figura B, vemos que o volume V da caixa é dado pelo seguinte polinômio:

$$V(x) = \text{base} \cdot \text{altura} = (L - 2x)^2 x = 4x^3 - 4Lx^2 + L^2 x \quad (\text{eq. 1})$$

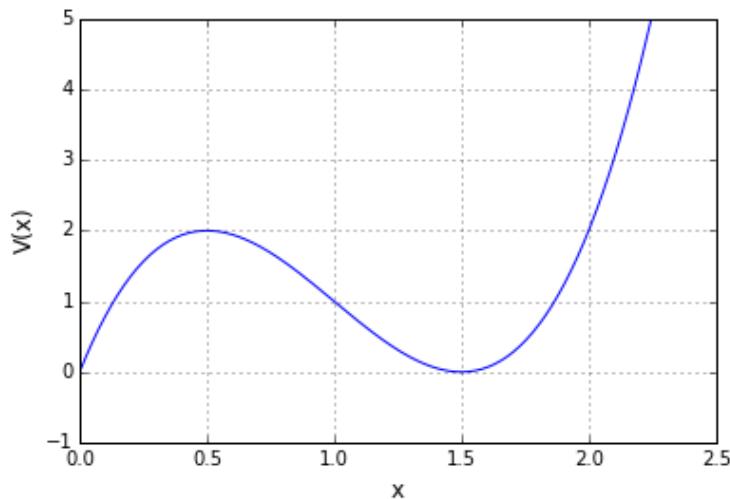
Lembre-se, a partir das aulas de Cálculo, que os pontos de máximo e de mínimo de uma função são aqueles onde sua derivada primeira é igual a zero.

OU seja, para encontrar qual o valor do lado do quadrado (x) a ser cortado a fim de que a caixa tenha volume máximo, devemos encontrar as raízes de $V'(x)$:

$$\frac{dV}{dx} = 12x^2 - 8Lx + L^2 = 0 \quad (\text{eq. 2})$$

EXEMPLO:

Para $L = 3$, o gráfico de $V(x)$ é dado abaixo, gerado pelas bibliotecas numpy e matplotlib:



Os dois valores que satisfazem a eq. 2 são $x = 0,5$ e $x = 1,5$. Quando $x = 1,5$, temos o valor mínimo de V , que é igual a zero. Já para $x = 0,5$, o volume é máximo: $V = 2$. Certamente, o gráfico mostra que $V(x)$ atinge valores maiores que 2, mas isso acontece em regiões onde o x é inválido para o problema, pois não faz sentido físico que ele seja maior que $L/2$.

Para saber se as raízes r_1 e r_2 da eq.2 são os pontos de máximo ou de mínimo sem auxílio do gráfico, é necessário determinar a segunda derivada de $V(x)$. Se $V''(r_1) > 0$, então r_1 é um ponto de máximo. Caso contrário, então r_2 é um ponto de máximo.

PROBLEMA:

Escreva um programa que leia o valor L correspondente ao comprimento da placa quadrada e informe o valor de x que maximiza o volume $V(x)$ da caixa, com duas casas decimais de precisão.

DICAS:

1. Use a função `poly1d()` para definir o polinômio $V(x)$.
2. Use a função `polyder()` para determinar a 1^a e 2^a derivadas de $V(x)$.
3. Use a função `roots()` para determinar as raízes de $V'(x)$, ou seja os pontos de inflexão de $V(x)$.
4. Use a função `polyval()` para calcular o valor de $V''(x)$ para cada raiz da equação $V'(x) = 0$.
5. Exemplos não exaustivos:
 - Para a entrada 3, a saída deve ser 1.5.
 - Para a entrada 4, a saída deve ser 2.0.
 - Para a entrada 5.678, a saída deve ser 2.84.

Primeiro passo: Interpretação da questão

Essa é uma questão muito importante para o seu aprendizado, preste muita atenção na resolução dela, você acha que está aprendendo cálculo I e II (ou sei lá quantos mais) para fazer continhos no papel? A aplicação do seu conhecimento em cálculo está aqui: no tão odiado Python.

Não se confunda, a questão não quer que você crie o gráfico da questão, ele quer que você encontre o ponto de máximo só isso.

De uma coisa nesse início você sabe, importar a biblioteca numpy e temos uma entrada que neste caso é o "L" que é o comprimento do lado da caixa. Como nós vamos trabalhar com um polinômio então temos que usar a função poly1d() para determinar o polinômio (o polinômio usado é o do volume), não se confunda e não vá usar o L como se fosse o X, isso quer dizer que o L será colocado dentro da função poly1d() também.

Vamos lembrar das aulas de cálculo, para sabermos quem é ponto máximo ou mínimo devemos derivar duas vezes o polinômio e substituir dentro dele os pontos críticos. Para derivar um polinômio usamos a função polyder(), basta colocar o polinômio dentro da função, e para encontrar a segunda derivada é só derivar novamente o polinômio já derivado. Use a função root() para calcular os pontos críticos (lembre-se que os pontos críticos são calculados na primeira derivada), por que a função root() faz a mesma coisa que igualar o polinômio a zero e encontrar as raízes.

Você sabe que as raízes de um polinômio podem ser uma, duas, três assim se segue, por isso a função root() funciona como vetor, no caso desta questão o polinômio a ser calculado a raiz será a derivada do volume, como o volume tem grau 3 a sua derivada terá grau 2, então com isso o polinômio derivado terá duas raízes, ou seja, a função irá representar a resposta como [x,y]. Já que nós precisamos desses pontos críticos para serem analisados dentro da segunda derivada do polinômio usaremos o comando vetor[i] para que possamos usar os valores encontrados pela função root, nesse caso teremos ou posição 0 ou posição 1 por que temos somente duas posições. Não se engane, não é para escrever root[0], root() é o nome da função o vetor é a resposta dela ou seja você deve colocar o nome da variável que está igualada a função root() e não a própria root().

Para calcularmos a segunda derivada nos pontos críticos precisamos usar a função polyval() que você já sabe como funciona né, lembre-se que você está correndo atrás do ponto de máximo e para que isso aconteça o polinômio aplicado no ponto crítico deve ser maior que zero, ou seja, SE o polinômio aplicado no primeiro ponto crítico for maior que zero o primeiro ponto será o ponto máximo OU SE não for maior que zero o outro ponto será o ponto de máximo. Perceba que iremos usar o **if** aqui, e a condição é justamente que o resultado do polyval() seja maior que zero, e para isso não é necessário calcular o polyval() para as duas posições, por que se um não for o outro será. No mine script como estamos usando o primeiro ponto crítico como referência, ou seja, estamos usando a posição 0, SE essa condição for satisfeita então o ponto de máximo é o da posição zero OU SE não for satisfeita o ponto máximo será o da posição um.

The screenshot shows the Spyder Python 3.4 IDE interface. The title bar says "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, and View. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Ama". A file named "Lab05_10.py" is open. The code is as follows:

```

1 # -*- coding: utf-8 -*-
2 """
3 19/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Leitura do Lado da caixa
12 L = int(input("Comprimento do Lado da caixa: "))
13
14 # Cria polinomio que representa o volume da caixa
15 v = poly1d([4, - 4*L, L**2, 0])
16
17 # 1a. derivada do volume: V'(x)
18 dv1 = polyder(v)
19
20 # 2a. derivada do volume: V''(x)
21 dv2 = polyder(dv1)
22
23 # Pontos críticos de V(x): V'(x) = 0
24 raizes = roots(dv1)
25
26 # Valor de V''(x) para o ponto de inflexão
27 f = polyval(dv2,raizes[0])
28
29 # Encontrar o valor maximo
30 if (f > 0):
31     print(round(raizes[0], 2))
32 else:
33     print(round(raizes[1], 2))
34

```

Figura 144: Questão 10E Primeiro Passo

Fonte: ICOMP, UFAM

The screenshot shows the Python 1 console window. The title bar says "Console" and "Python 1". The command prompt ">>> runfile('C:/Users/User/Documents/Introdução a Programação de Computador/Helder Guerreiro/Universidade Federal /Meus Programas')" is shown. The output is:

```

Comprimento do Lado da caixa: 3
1.5
>>>

```

Figura 145: Resultado questão 10E

Fonte: Autoria própria

5.5 Avaliações

Aqui teremos algumas avaliações para entender mais um pouco o assunto estudado neste primeiro capítulo. Aconselho você tentar fazer sozinho antes de ler a resolução.

Avaliação A - A5

Avaliação Parcial 05 (A5) a – Vetores

Informações

Questões

Notas

Questão 1

Escreva um script que crie um vetor x de 500 posições, contendo números reais igualmente espaçados no intervalo [-25; +25]. Em seguida, o script deverá determinar e imprimir um vetor y contendo os valores do polinômio $P(x)$ abaixo para cada valor do vetor x .

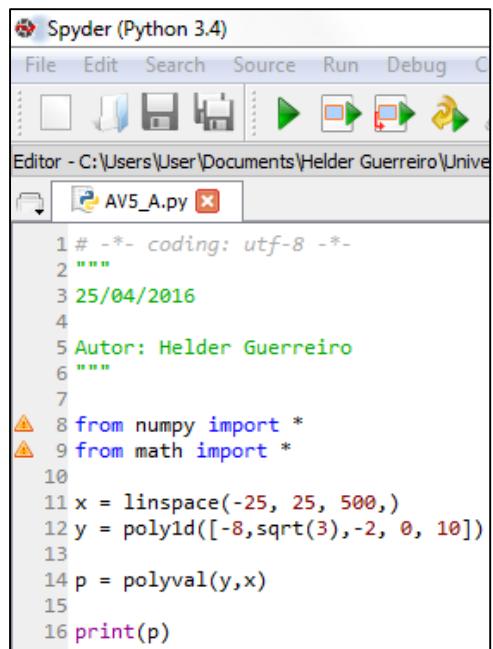
$$P(x) = -8x^4 + \sqrt{3}x^3 - 2x^2 + 10$$

Dicas:

1. Verifique se você está submetendo o código correto à questão correspondente desta Avaliação Parcial.
2. Use a função `linspace`, da biblioteca `numpy`, para criar o vetor x .
3. Use a função `poly1d(vet)`, da biblioteca `numpy`, para definir um polinômio $P(x)$ a partir de um vetor `vet` contendo seus coeficientes.
4. Use a função `polyval(p, x)`, da biblioteca `numpy`, para calcular o valor de um polinômio `p` no ponto `x`.

Questão simples, vamos só fazer com que este polinômio mostrado seja calculado usando as funções polinomiais. A função `linspace()` irá criar os limites de valores para a variável x , ao cria-la deve-se colocar os limites -25,25 e o número 500 para ditar quantos valores devem ter dentro do vetor. Mais uma vez eu explico que devemos fazer da variável x um vetor por que vamos substituir esse valor dentro do polinômio e calcular, se não dermos limites aos valores de x iremos trabalhar infinitamente.

Para escrever um polinômio devemos usar a função `poly1d` e perceba que dentro do polinômio temos uma raiz de 3, para isso deve-se usar a biblioteca `math` e usar a função `sqrt()`. Para substituição do valor de x temos de usar a função `polyval()` e mostrar a resposta final (ela será gigantesca pois são 500 valores).



The screenshot shows the Spyder Python 3.4 IDE interface. The title bar reads "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, and Help. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Unive". A tab labeled "AV5_A.py" is active. The code editor contains the following Python script:

```
1 # -*- coding: utf-8 -*-
2 """
3 25/04/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 from numpy import *
9 from math import *
10
11 x = linspace(-25, 25, 500)
12 y = poly1d([-8,sqrt(3),-2, 0, 10])
13
14 p = polyval(y,x)
15
16 print(p)
```

Figura 146a: Resposta Avaliação A 5A

Fonte: Autoria própria

Avaliação B - A5

Avaliação Parcial 05 (A5) b – Vetores

Informações

Questões

Notas

Questão 1

A regulamentação de trânsito estabelece que o nível de óxidos de nitrogênio (NOX) no escapamento de veículos não deve ser igual ou superior a 190 mg/km. Uma oficina armazena em um vetor de inteiros os níveis de NOX (medidos em mg/km) de vários veículos sob sua manutenção.

Escreva um programa que leia esse vetor, digitado a partir do teclado. Em seguida, o programa deve percorrer o vetor e imprimir o índice de cada posição correspondente a um veículo com emissão irregular de NOX. No final, o programa deve imprimir quantos veículos precisam de manutenção no escapamento.

Dicas:

1. Utilize duas variáveis contadoras: uma para controlar a **posição** do vetor e outra para contar o **número de ocorrências** do valor procurado.
2. Para a sequência de entrada **[216, 167, 161, 173, 190, 233]**, a saída deverá ser **0 4 5 3**, indicando que os veículos indicados pelos índices 0, 4 e 5 precisam de manutenção no escapamento, num total de três veículos.

Nesta questão vamos precisar passear pelo vetor. O vetor inserido é o vetor de dados das emissões de NOX dos veículos, usando o `while` vamos passear dentro do vetor e analisar cada índice dele usando o comando condicional `if`. Antes de iniciar o `while` devemos ter duas variáveis contadoras, uma para o `while` percorrer todo vetor e outra para o `if` contar quantos veículos precisarão de ajustes.

Para o `while` percorrer dentro do vetor faça da sua condição assim: que a variável contadora não ultrapasse o tamanho do vetor a ser percorrido. A cada execução do `while` será um índice diferente a ser analisado, cada índice deve ser analisado e quem faz a análise é o `if` que irá testar condições. A condição `if` é simplesmente que o valor do índice (use o comando `vetor[]`) seja maior que 190. No final do script do `if` devemos contar a ocorrência de uma condição satisfeita, para no final de tudo (fora do `while`) exibir o resultado de quantos veículos necessitam de manutenção, além de contar a ocorrência devemos exibir qual o índice que satisfez a condição do `if`.

The screenshot shows the Spyder Python IDE interface. The title bar reads "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, and Help. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Unive". A file tab labeled "AV5_B.py" is open. The code in the editor is:

```
1 # -*- coding: utf-8 -*-
2 """
3 25/04/2016
4
5 Autor: Helder Guerreiro
6 """
7
8 from numpy import *
9
10
11 V = array(eval(input("Vetor? ")))
12
13 i = 0
14
15 t = 0
16
17 while(i < size(V)):
18     if V[i] >= 190:
19         t = t + 1
20         print(i)
21     i = i + 1
22
23 print(t)
24
```

Figura 146b: Resposta Avaliação B 5A

Fonte: Autoria própria

6. Vetores no Modo Avançado - Estruturas de Repetição por Contagem

Chegou a hora de eu apresentar a você um novo comando, só que dessa vez não é um comando condicional como o `if` e o `while`, temos agora um comando cujo processo é por contagem. Apresento-lhe o “`for`”, esse é o comando que trabalha somente com vetores e trabalha em conjunto com o comando “`in`”. O comando `for` também trabalha com mine script, nesse comando o mine script será repetido para cada elemento do vetor escolhido. O comando `for` também trabalha com uma variável qualquer, essa variável será responsável por receber o valor do elemento do vetor na posição atual, isso quer dizer que se escolhermos a variável “`x`” e um vetor chamado `vet = [1,2,3]`, o `for` irá repetir seu mine script três vezes, na primeira repetição temos `x = 1`, na segunda repetição temos `x = 2` e na terceira e última repetição temos `x = 3`.

Para representar tudo isso que fizemos devemos escrever assim: `for x in vet`.

Se você for colocar no mine script do `for` para que ele reproduza o valor de `x` a saída será: 1,2 e 3.

Veja a diferença entre usar `while` e `for`:

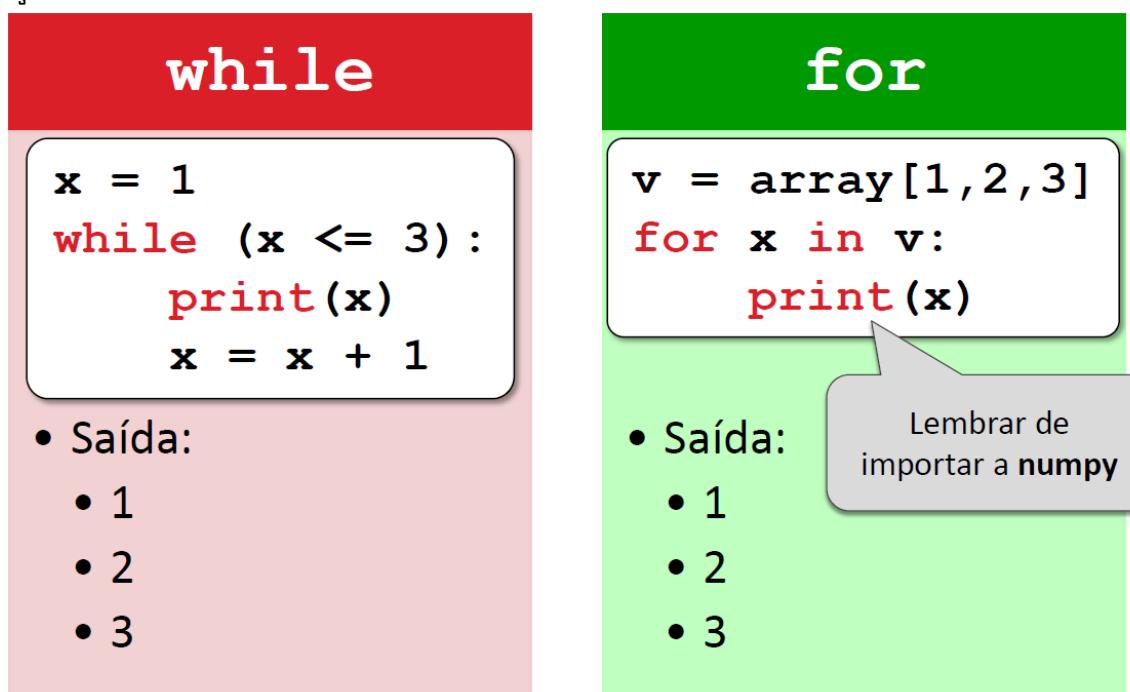


Figura 146: While e for

Fonte: ICOMP, UFAM

Como usar esse comando ao nosso favor? Para que ele serve? E como usá-lo para as atividades do dia a dia? Tudo isso você entenderá muito melhor quando formos fazer as questões resolvidas.

6.1 Uma Função Inovadora

Existe outra forma do `for` trabalhar, usando uma função chamada “`range`”, essa função permite que trabalhemos sem o uso de vetores, a própria função `range` serve para criarmos um intervalo e ela irá fazer desse intervalo um vetor. Para usá-la deve se escrever assim: `for x in range()`.

Podemos trabalhar com ela usando um, dois ou três argumentos, os argumentos são as informações que colocamos dentro da função. A imagem abaixo explica muito bem como funciona essa função:

```
for i in range(5):  
    print(i)
```

0, 1, 2, 3, 4

Com **01 argumento**, a sequência começa em **0**. O argumento é o primeiro valor não incluído na sequência.

Com **02 argumentos**, a sequência **começa** com o primeiro argumento, mas termina **antes** do segundo.

```
for i in range(1, 5):  
    print(i)
```

1, 2, 3, 4

```
for i in range(0, 5, 2):  
    print(i)
```

0, 2, 4

Com **03 argumentos**, o terceiro argumento é o **passo**.

Figura 147: Função range
Fonte: ICOMP, UFAM

Esse tal de "passo" que fala na imagem é como será a distribuição da contagem, no caso da imagem temos o passo como número 2, isso quer dizer que a distribuição dos números de 0 a 5 será de dois em dois como se pode perceber no exemplo acima.

Podemos fazer a contagem de forma decrescente também, se liga no exemplo abaixo:

```
for i in range(5, 0, -1):  
    print(i)
```

5, 4, 3, 2, 1

Figura 148: Função range decrescente
Fonte: ICOMP, UFAM

6.2 Questões Resolvidas

Estas questões foram desenvolvidas pela ICOMP um instituto da UFAM e certos algoritmos (scripts) apresentados aqui pertencem ao ICOMP.

Questão 1 – Quantos elementos são pares?

Escreva um script em Python que leia do teclado um vetor de tamanho qualquer. Em seguida, o script deve determinar quantos elementos são pares, quantos são ímpares e o número total de elementos. Seu script deve repetir esse procedimento indefinidamente para diversos vetores, de tamanhos variados, até que um vetor de apenas um elemento seja inserido, terminando a execução do script.

Primeiro passo: Interpretação da questão

Neste primeiro passo com o novo comando `for` perceba que já temos algumas condições aí, o script deve rodar até que o usuário determine seu fim (inserindo um vetor de um elemento), então você já deve ter percebido que ENQUANTO o usuário não inserir o vetor unitário o script continuará a rodar. Tudo bem já sabemos que vamos usar o `while`, mas vamos com calma.

No início de tudo o módulo `numpy` deve ser importado já que vamos trabalhar com vetores. O vetor que será pedido como entrada claramente deve estar antes de entrar no `mine script` do `while`,

Calma eu sei o que você está pensando, condição dentro de condição, `mine script` dentro de `mine script` isso tá virando uma bagunça... Se você aprendeu tudo que eu ensinei a você antes isso não será difícil de associar, o aprendizado desta matéria é estrutural e você precisa construir seu alicerce do conhecimento com calma.

Quem vem primeiro `while` ou `for`? Vamos pensar, o `for` irá repetir seu `mine script` de acordo com um vetor esse vetor é claramente o vetor inserido pelo usuário, pois esse vetor será analisado para que se saiba quem é par e quem é ímpar dentro dele. O `for` sendo o primeiro haverá uma repetição para cada elemento do vetor, só que quando acabar os elementos e o script tiver que pedir outro vetor do usuário como faremos? Ok podemos pedir outro vetor dentro do `mine script` do `for`, mas quando tivermos que parar a reprodução do script? Quem carregará a condição de parada será o `while`, certo, mas se o `while` estiver dentro do `for` toda vez que o `for` repetir seu `mine script` o `while` irá repetir o seu também, então para determinar a parada do script temos que executar o `for` primeiro para depois paramos o script ao chegar no `while`. Você percebeu que temos uma certa bagunça aí né, por isso temos que colocar o `while` na frente por que o `while` é que determina se o script para ou continua.

A condição do `while` é simplesmente que o vetor não seja unitário, ou seja, tenha somente um vetor, para fazer essa condição podemos usar a função que determina o tamanho dos vetores, estou falando do `size()`, se ele determina o tamanho dos vetores então o vetor inserido deve ter o seu tamanho maior que 1.

Aqui iremos usar dois contadores, um para o par e outro para o ímpar, isso por que a questão pede que encontramos a quantidade de números pares e ímpares dentro dos vetores. Como usaremos mais de um vetor, a contagem das variáveis deve ser zerada, então as variáveis devem estar dentro do mine script do while para que toda vez que for necessário a repetição do script as variáveis estejam prontas para contar novamente,

Chegou a hora de usarmos o for, use uma variável qualquer (eu vou chama-la de x), o vetor a ser analisado será o vetor inserido pelo usuário. Vou lhe dar uma dica: `for x in` vet é como se fosse “para a variável x em cada elemento de vet temos: ”. Isso que fazemos com o for é a mesma coisa que passemos dentro do vetor, só que dessa vez sem o while e sem o índice “i”. Então com o for já passemos por dentro do vetor e agora como faremos o seu mine script? Simples, agora que podemos andar por todo o vetor é hora de fazer com que o Python analise cada elemento para que ele seja par ou ímpar, em outras palavras: SE o elemento for par OU SE for ímpar.

Calma não vá se confundir, o while serve para repetir as ações do script, o for para passear dentro do vetor e o `if` para direcionar caminhos tomados através de uma certa condição. O `if` será usado para designar quem é par e ímpar aqui (você se lembra? Já fizemos uma questão envolvendo par e ímpar no segundo tópico desta apostila). Vamos ver se você se lembra, para sabermos SE um número é par ele dever ser divisível por 2, ou seja, seu resto deve ser zero OU SE não for divisível, ou seja, o resto não for zero então o número é par ...

Você já se tocou que estamos falando do velho conhecido resto da divisão? É meu caro leitor, nunca esqueça dele, até aqui ele se faz presente, ele é representado por “%” e será a condição do if. Use o `else` em caso do número não for divisível por 2. A condição sendo satisfeita ou não devemos entrar em ação com as variáveis contadoras, uma para contar a quantidade de ocorrências de números pares e outra para os ímpares.

Depois da execução do for tiver sido completa, é hora de mostrar os resultados para a quantidade de números pares, ímpares e o tamanho do vetor (isso não preciso explicar né?). Esses `print`'s devem estar dentro do while não fora, por que eles serão executados para cada vetor inserido. O pedido de um novo vetor é claramente dentro do mine script do while também, isso por que no final de tudo a pergunta será refeita e o vetor será mudado, como o while irá repetir novamente ele irá rever sua condição, ou seja, se o vetor inserido é unitário ou não.

The screenshot shows the Spyder Python 3.4 IDE interface. The title bar says "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, and Tools. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Acre\Lab06_01.py". The code in the editor is:

```
1 # -*- coding: utf-8 -*-
2 """
3 19/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import*
10 # Leitura do primeiro vetor
11 vet = array(eval(input("Primeiro vetor: ")))
12
13 # Verifica se o programa vai terminar
14 while (size(vet) > 1):
15     # Conta elementos pares
16     par = 0
17
18     # Conta elementos ímpares
19     impar = 0
20
21     for x in vet:
22         if (x % 2 == 0):
23             par = par + 1
24         else:
25             impar = impar + 1
26     print(par) # Elementos pares
27     print(impar) # Elementos ímpares
28     print(size(vet)) # Tamanho do vetor
29
30     # Próximo vetor
31     vet = array(eval(input("Próximo vetor: ")))
32
```

Figura 149: Questão IF Primeiro Passo

Fonte: ICOMP, UFAM

The screenshot shows the Spyder Python 1 console window. The title bar says "Console" and "Python 1". The command prompt shows ">>> runfile('C:/Users/User/Documents/introdução a Programação de Computador/holder Guerreiro/Universidade Federal /Meus Programas')". The output shows the execution of the code, printing the first vector (1,2,3,4,5), then the count of even (2) and odd (3) elements, and the size (5). It then prints the next vector (2,3,4,5,6), its even count (2), odd count (3), and size (5). Finally, it prints the next vector (1).

```
>>> runfile('C:/Users/User/Documents/introdução a Programação de Computador/holder Guerreiro/Universidade Federal /Meus Programas')
Primeiro vetor: 1,2,3,4,5
2
3
5
Próximo vetor: 2,3,4,5,6
3
2
5
Próximo vetor: 1
>>>
```

Figura 150: Resultado questão IF

Fonte: Autoria própria

Questão 2 – Aprovado, mas não pode ser monitor

Na UFAM, uma das condições para que um aluno seja monitor de uma disciplina é que ele tenha sido aprovado nela com nota mínima de 7. Outras condições que não serão consideradas neste problema é que ele tenha coeficiente de rendimento mínimo de 5 e que esteja pelo menos no 3º período letivo.

Escreva um script que leia um vetor contendo as médias finais dos alunos matriculados em uma disciplina. Em seguida, determine a quantidade de alunos aprovados na disciplina (nota maior ou igual a 5), mas não tem condições de ser monitor da mesma (nota menor que 7).

Seu script deve repetir esse procedimento indefinidamente para diversos vetores, de tamanhos variados, correspondentes a outras disciplinas, até que um vetor de apenas um elemento seja inserido, terminando a execução do script.

DICAS:

1. Modifique o script do problema anterior.

2. Para a entrada [0, 2, 4, 5, 6, 7, 8, 9] [10, 8, 9] [1, 2, 3] [], a saída deve ser 200.

Primeiro passo: Interpretação da questão

Esta questão é a cópia da anterior com algumas modificações. Você irá submeter um vetor com notas dos alunos, e esse vetor passará pela análise do `while` se está na condição ou não, estando na condição o `while` segue com seu próprio script onde estará a variável contadora zerada para que depois ela possa dizer quantos passaram, mas não podem ser monitores, logo abaixo o `for` irá passear pelo vetor e o `if` irá dizer quem é quem a cada elemento do vetor.

Isso descrito a cima é a mesma coisa que fizemos na primeira questão, a condição do `while` continua a mesma, um vetor unitário irá terminar a reprodução do script, e o `for` continuará passeando pelo mesmo vetor inserido pelo usuário, a variável acumuladora (dessa vez é só uma) será a responsável por contar quantos passaram e não podem ser monitores. Ao usar o `for` para passear dentro do vetor temos de usar o `if` para que ele possa analisar a condição que é a seguinte: SE o aluno tiver nota maior ou igual a 5 E tenha nota menor que 7, perceba que essa condição exige duas coisas, nota maior ou igual a 5 e nota menor que 7 para isso deve-se usar o operador condicional chamado `and` para que possamos colocar mais de um argumento na condição do `if`. Respeitada a condição do `if` é a vez da variável contadora ir guardado as ocorrências em que o `if` for satisfeito, você deve ter percebido que não precisamos do `else` aqui.

The screenshot shows the Spyder Python 3.4 IDE interface. The code editor window is titled 'Lab06_02.py' and contains the following Python script:

```
1 # -*- coding: utf-8 -*-
2 """
3 19/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10 # Leitura do primeiro vetor
11 vet = array(eval(input("Vetor com as notas: ")))
12
13 # Verifica se o programa vai terminar
14 while (size(vet) > 1):
15     # Variável contadora
16     Nmonitor = 0
17
18     for x in vet:
19         if (x >=5 and x < 7):
20             Nmonitor = Nmonitor + 1
21
22     #Quantidade de alunos aprovados, mas que não podem ser monitores :
23     print(Nmonitor)
24
25     # Leitura do proximo vetor
26     vet = array(eval(input("Proximo vetor: ")))
27
```

Figura 151: Questão 2F Primeiro Passo

Fonte: ICOMP, UFAM

The screenshot shows the Spyder Python 3.4 IDE's console window. It displays the execution of the script and its output:

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/1º Período/Introdução a Programação de Computadores/Meus Programas')
Vetor com as notas: 0,2,4,5,6,7,8,9
2
Proximo vetor: 10,8,9
0
Proximo vetor: 1,2,3
0
Proximo vetor: 1
>>>
```

Figura 152: Resultado questão 2F

Fonte: Autoria própria

Questão 3 - Contar ocorrências (I): Time de Futebol

Os resultados das partidas de um time de futebol ao longo de um campeonato são armazenados em dois vetores de mesmo tamanho. O primeiro vetor guarda o número de gols efetuados por esse time em cada partida. O segundo vetor guarda o número de gols efetuados pelo time adversário. Uma posição i de cada vetor indica a i -ésima partida realizada. Escreva um programa que leia esses dois vetores, na ordem em que foram explicados. Como saída, deve ser informado um vetor de três posições, contendo os seguintes dados, em ordem, em relação ao time:

- Número de vitórias
- Número de empates
- Número de derrotas

Considere que as entradas fornecidas são sempre válidas: tamanhos dos dois vetores de gols são iguais; cada elemento do vetor, que indica o número de gols, é um inteiro não-negativo.

DICAS:

1. Utilize um vetor para contar o número de ocorrências de cada situação elencada. Não se esqueça de zerar todos os elementos do vetor antes de começar a contar.
2. Exemplo (não exaustivo). Para a entrada $[4, 0, 2, 3, 1, 3] [0, 0, 2, 5, 1, 2]$, a saída deverá ser $[2\ 3\ 1]$. Isso significa que o time ganhou duas vezes (4×0 e 3×2 , primeiro e último jogo), empatou três vezes (0×0 , 2×2 e 1×1 , no segundo, terceiro e quinto jogo) e foi derrotado uma vez (3×5 , no quarto jogo).

Primeiro passo: Interpretação da questão

Aqui vamos trabalhar com dois vetores e com uma novidade: um vetor será usado como variável contadora. Vamos lembrar como funciona uma variável contadora? É uma variável que começa do zero e ao passar pela condição necessária para algo acontecer a variável irá adicionar a si mesma uma unidade. A contadora em forma de vetor funciona do mesmo jeito! O vetor deve ter um certo tamanho, o tamanho deve ter a quantidade de posições desejadas, neste caso precisamos de uma posição para informar o número de vitórias e outra para informar os empates e mais uma de derrotas, ou seja, teremos uma variável acumuladora de três posições. Por que estamos usando um vetor? Para se tornar mais prático, ou você prefere usar três variáveis contadoras de uma vez?

Antes de usar a variável contadora devemos zerá-la não? Então devemos zera o vetor também, para isso usamos a função zeros e criamos um vetor de 3 posições com valores iguais a zero.

Usaremos o `while` aqui? Não, o while seria usado se por acaso precisássemos pedir outro vetor para repetir as ações. Então aqui só vamos usar o `for`, só que dessa vez teremos a participação do `range`, mas por que? Vou explicar melhor o funcionamento dele: podemos usar um vetor como referência, só que ao usar o for e um vetor como referência estaremos passeando por dentro desse vetor, só que dessa vez não vamos somente um vetor, vamos passar por dois vetores e para isso temos de usar o range. Ao usar o range a variável escolhida não vai passar nem por um vetor e nem por outro, cabe a você dizer a quem pertence essa variável podendo usá-la para ambos os vetores.

Qual tamanho vamos usar para o range? O vetor inserido pelo usuário pode ter qualquer tamanho então não podemos usar um valor fixo para dizer o tamanho do vetor criado no range, para isso podemos usar o tamanho do próprio vetor inserido pelo usuário, ou seja, usar a função `size()`. Sabemos que a função `size()` conta o real tamanho do vetor começando a contagem a partir do 1 e não do 0, e o intervalo que devemos colocar dentro da função `range` é um intervalo que começa no ponto que você colocou, mas o intervalo termina uma unidade antes do ponto de término, mas ao colocar o intervalo de 0 à `size()` não iremos ter problemas, pois se o tamanho do vetor é 7 ao colocar dentro do intervalo de 0 à 7 na função `range` o vetor criado terá índices de 0 à 6 que é a mesma quantidade de índices do vetor de tamanho 7.

Usamos o `for` para passear e agora precisamos analisar os dois vetores ao mesmo tempo e para isso temos de usar o `if`, pois ele é o comando responsável por dizer se uma certa condição é satisfeita e neste caso temos três condições: vitórias, empates e derrotas. Então nossa análise dentro dos vetores deve ser: SE os números corresponderem a vitória? E SE os números corresponderem ao empate? OU SE os números corresponderem a derrota?

Já vai se tocando aí que nós vamos usar o `elif`, pois temos mais de duas condições então precisamos trabalhar em conjunto com os três comandos condicionais: `if`, `elif` e `else`.

Como vamos analisar dois vetores ao mesmo tempo? Usando o índice, o índice será a variável escolhida na formação do `for` com o `range`. Exemplo, escolhendo o "i" como variável temos: `for i in range`, a variável "i" se comportará como um índice e como nós estamos usando a função `range`, esse índice pode ser de qualquer vetor. Para designarmos de quem pertence o índice temos de usar o comando "`vetor[i]`", pois esse é o comando que através do vetor escolhido seleciona o valor pertencente ao índice "i".

A primeira condição fala que o time deve ser o time vitorioso e claramente para isso o número de gols do time deve ser maior que o número de gols do adversário. Nos dois vetores inseridos pelo usuário cada posição dos vetores significam um jogo ocorrido, a posição 0 do vetor dos gols do time é o mesmo jogo em que ocorreu os gols do time adversário na posição 0 do outro vetor, isso quer dizer que o índice "i" deve ser igual em todos os comandos. Para escrevermos a condição do `if` temos que colocar que o valor do vetor do time na posição "i" deve ser maior que o valor do vetor do adversário na mesma posição "i". O mine script do `if` é simplesmente colocar em ação a variável contadora que neste caso é um vetor, a primeira posição do vetor está destinada ao número de vitórias, ou seja, a posição 0 então vamos fazer contar somente a posição zero do vetor contador assim (Vou chamar o vetor contador de "vcont") :

$$vcont[0] = vcont[0] + 1$$

É o mesmo esquema que uma variável contadora comum, a diferença é que estamos lhe dando com um vetor então nós temos que modificar o valor do vetor numa posição específica. Essa continha aí cima significa que toda vez que o `if` for satisfeito a posição 0 do vetor contador será adicionado a uma unidade.

A próxima condição é a mesma estrutura que do `if`, só muda que agora estamos usando o `elif`. A condição agora é para caso o time empatasse, ou seja, o número de gols do time informado no vetor na posição "i" deve ser o mesmo número informado no vetor do adversário na posição "i". O seu mine script dessa vez vai contar o número de empates, ou seja, o número de vezes que a condição do `elif` foi satisfeita. Dessa vez o vetor contador irá contar na sua segunda posição que é a posição 1, faremos a mesma coisa que foi feita no `if`, iremos mexer somente com o valor do índice um do vetor contador,

Agora se não houve satisfação em nenhuma das condições acima, ou seja, se não houve nem vitória nem empate então só resta o fato de que o time perdeu e para isso basta usarmos o else e colocar no script dele a contagem na terceira posição do vetor contador que é a posição 2. No final de tudo fora do domínio do for basta exibir o resultado do vetor contador com a função `print`.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\Lab06_03.py
1 # -*- coding: utf-8 -*-
2 """
3 21/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Leitura
12 vet1 = array(eval(input("Gols do time: ")))
13 vet2 = array(eval(input("Gols do adversario: ")))
14
15 # Vetor de contagem
16 vcont = zeros(3, dtype=int)
17
18 for i in range(0, size(vet1)):
19     # Vitorias
20     if (vet1[i] > vet2[i]):
21         vcont[0] = vcont[0] + 1
22     # Empates
23     elif (vet1[i] == vet2[i]):
24         vcont[1] = vcont[1] + 1
25     # Derrotas
26     else:
27         vcont[2] = vcont[2] + 1
28
29 # Impressao dos resultados
30 print(vcont)
31
32

```

Figura 153: Questão 3F Primeiro Passo

Fonte: ICOMP, UFAM

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/Meus Programas/Lab06_03.py')
Introdução a Programação de Computadores
Helder Guerreiro/Universidade Federal do Amazonas
Gols do time: 4,0,2,3,1,3
Gols do adversario: 0,0,2,5,1,2
[2 3 1]
>>>

```

Figura 154: Resultado questão 3F

Fonte: Autoria própria

Questão 4 – Contar ocorrências (2): Aprovação em disciplina

Na Universidade Federal do Amazonas (UFAM), um aluno é aprovado em uma disciplina se atender a dois critérios: ter média igual ou superior a 5,0 e ter frequência igual ou superior a 75% da carga horária. Se sua frequência for menor que esse limiar, o aluno é reprovado por frequência, independentemente da sua nota. Por fim, o aluno é reprovado por nota se, tendo comparecido ao número mínimo de aulas, não atingiu a média exigida.

As notas dos alunos de uma classe são guardadas em um vetor de reais. A quantidade de presenças às aulas é armazenada em outro vetor de mesmo tamanho, mas contendo elementos inteiros.

Escreva um programa que leia o vetor de notas, o vetor de presença e a carga horária da disciplina, nessa ordem. Como saída, deve ser informado um vetor de três posições, contendo os seguintes dados, em ordem:

- Número de alunos aprovados.
- Número de alunos reprovados por nota.
- Número de alunos reprovados por frequência.

Considere que as entradas fornecidas são sempre válidas: notas entre 0 e 10, frequência entre 0 e a carga horária, tamanhos iguais dos vetores de notas e frequência.

DICAS:

1. Utilize um vetor para contar o número de ocorrências de cada situação elencada. Não se esqueça de zerar todos os elementos do vetor antes de começar a contar.
2. Exemplo (não exaustivo). Para a entrada [10, 10, 9] [44, 45, 46] 60, a saída deverá ser [2 0 1]. Isso significa que o primeiro aluno tirou dez, mas frequentou menos de 75% das 60h de aula previstas; logo foi reprovado por frequência. Os dois alunos seguintes atenderam aos dois critérios e por isso foram aprovados.

Primeiro passo: Interpretação da questão

Esta questão tem a mesma estrutura que a anterior, não é à toa que o nome da questão tem por título "Contar ocorrências (2)". Por isso fique de olho no script da questão anterior.

Aqui você já sabe que teremos três entradas: as notas, as frequências e a carga horária, são dois vetores e uma variável comum. Aqui teremos o uso do vetor contador de novo, pois devemos contar as ocorrências de alunos aprovados, reprovados por nota e reprovados por frequência. Como não há o requisito de um novo vetor então não precisamos usar o `while`, saiba que essa e a questão anterior pode ser usadas com o `while` sem problema, basta usar uma condição para o `while` e colocar todo o script dentro do `while` e pronto.

Agora o próximo passo é correr por dentro dos vetores para que eles sejam analisados, o `for` terá a mesma configuração que da questão anterior, como estamos trabalhando com mais de um vetor então temos de usar a função `range` e como o tamanho do vetor pode variar vamos usar o tamanho (a função `size`) desse vetor como base da criação do `range`.

Para analisar os vetores necessitamos dos comandos condicionais `if`, `elif` e `else`, precisamos do `elif` de novo por que estamos trabalhando com três condições diferentes. A primeira condição é a aprovação, mas a aprovação depende de duas situações: nota maior ou igual a 5 E a frequência deve ter pelo menos (maior ou igual) 75 % da carga horária, perceba que aqui vamos usar o comando `and` para junção de argumentos. Na organização dos vetores inseridos cada posição do vetor representa um aluno, então o aluno que tirou uma certa nota na posição "i" do vetor de notas é o mesmo aluno que teve uma certa frequência na mesma posição "i" do vetor de frequências, isso quer dizer que o mesmo índice "i" que estiver no vetor de notas também deve estar no vetor de frequências. Para selecionar o valor dos vetores na posição "i" não esqueça de usar o comando "`vetor[i]`", só assim iremos identificar o valor dentro dos vetores. No seu mine script vamos simplesmente contar a primeira posição do vetor contador, indicando um aprovado caso aja satisfação da condição do `if`.

A segunda condição presidida pelo `elif` diz que o aluno é reprovado por nota, mas não por frequência, isso quer dizer que o aluno deve tirar menos de 5 na sua nota E a sua frequência deve ter pelo menos (maior ou igual) 75 % da carga horária. Se você não se tocou conseguir 75 % da carga horária é a mesma coisa que multiplicar a variável da carga horária inserida pelo usuário por 0,75. No seu mine script vamos contar a segunda posição do vetor contador.

A última condição diz que o aluno será reprovado por frequência, mas não por nota. Para isso não precisamos do `elif`, o `else` já faz esse papel, pois se o aluno não atendeu a condição de tirar nota maior ou igual a 5 E pelo menos 75 % da carga horária e tirar nota menor que 5 E pelo menos 75 % da carga horária só resta o fato de que o aluno tem uma carga horária menor que 75 % independente da nota. Perceba que esse script não determina que o aluno reprovou pelas duas situações, para isso deveríamos adicionar mais um `elif`, porém não há necessidade, pois, as frequências na UFAM são prioridade independente da nota do aluno. O mine script do `else` irá contar a terceira posição do vetor contador e no final de tudo basta apresentar a resposta que é o vetor contador.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas
Lab06_04.py

1 # -*- coding: utf-8 -*-
2 """
3 21/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Leitura
12 notas = array(eval(input("Vetor de notas: ")))
13 freq = array(eval(input("Vetor de frequencia: ")))
14 carga = int(input("Carga horaria: "))
15
16 # Vetor de contagem
17 vcont = zeros(3, dtype=int)
18
19 for i in range(0, size(notas)):
20     # No. de aprovados
21     if (notas[i] >= 5) and (freq[i] >= 0.75*carga):
22         vcont[0] = vcont[0] + 1
23     # No. de reprovados por nota
24     elif (notas[i] < 5) and (freq[i] >= 0.75*carga):
25         vcont[1] = vcont[1] + 1
26     # No. de reprovados por frequencia
27     else:
28         vcont[2] = vcont[2] + 1
29
30 # Impressao dos resultados
31 print(vcont)
32

```

Figura 155: Questão 4F Primeiro Passo

Fonte: ICOMP, UFAM

```

Console
Python 1
>>> runfile('C:/Users/User/Documentos/Introdução a Programação de Computadores/Helder Guerreiro/Universidade Federal do Amazonas/Meus Programas')
Vetor de notas: 10,10,9
Vetor de frequencia: 44,45,46
Carga horaria: 60
[2 0 1]
>>>

```

Figura 156: Resultado questão 4F

Fonte: Autoria própria

Questão 5 - Contar ocorrências (3): Faltas ao trabalho

O setor de Recursos Humanos (RH) de uma empresa está preocupado com as faltas de seus empregados ao trabalho. Como estudo inicial, o RH deseja determinar quantas faltas acontecem em cada dia de semana de trabalho (segunda a sábado).

Um vetor armazena o histórico de faltas registradas na empresa. Cada posição armazena um número que representa o dia da semana em que algum empregado faltou ao trabalho. O número 2 indica que houve uma falta na segunda-feira, 3 indica uma falta em dia de terça-feira, e assim por diante, até o número 7, que indica uma falta em dia de sábado.

Escreva um programa que leia esse vetor, de tamanho qualquer. Na saída, o programa deve devolver um outro vetor de seis posições contendo o percentual de faltas que aconteceram na segunda, terça, ..., até sábado, nesta ordem, com até uma casa decimal de precisão.

Considere que as entradas fornecidas são sempre válidas: elementos do vetor é sempre um inteiro entre 2 e 7.

DICAS:

1. Utilize um vetor para contar o número de ocorrências de cada situação elencada.
2. Não confunda o número usado para representar o dia da semana com o índice do vetor de contagem.
3. Não se esqueça de zerar todos os elementos do vetor de contagem antes de começar a contar.
4. Use a função `round(x, n)`, nativa da linguagem Python, para fazer arredondamento do número x com até n casas decimais de precisão.
5. Serão necessários dois laços de repetição. O primeiro deverá percorrer o vetor de histórico de faltas. O outro deverá percorrer o vetor de contagem, a fim arredondar o percentual de faltas para a precisão desejada.

6. Exemplos (não exaustivos):

- a. Para a entrada [2,2,3,3,7,7,7,7], a saída deverá ser [30. 20. 0. 0. 0. 50]. Isso significa que 30,0% das faltas aconteceram na segunda-feira, 20,0% na terça e 50,0% no sábado.
- b. Para a entrada [2,3,4,5,6,7], a saída deverá ser [16.7 16.7 16.7 16.7 16.7 16.7]. Isso significa que cada dia de trabalho (segunda a sábado) contribui com 16,7% das faltas.
- c. Para a entrada [4,4,4], a saída deverá ser [0. 0. 100. 0. 0. 0.]. Isso significa que todas as faltas se concentraram na quarta-feira.

Primeiro passo: Interpretação da questão

Esta questão é um pouco diferente, mas é importante você aprender as várias formas de se trabalhar com os vetores e suas funções. Esta é uma questão interessante que vale a pena saber programar.

Agora a entrada é diferente, continua sendo um vetor, mas esse vetor deve ter a numeração somente de 2 a 7, pois cada unidade representa um dia da semana e dia de domingo que seria o número 1 não há trabalho. Neste caso o enunciado não pede restrição caso o usuário digite um número inválido como 0, 1 ou acima de 7, mas eu mostrarei lá para o final que podemos dar algumas acrescentadas bem maneiras nesse script.

O nosso vetor contador dessa vez deve ter 6 posições, ou seja, seis números zeros, esses vetores contadores também pode ser criados criando um vetor com um bocado de zeros, mas não é mais fácil usar a função zeros()?

Ao usarmos o `for` temos aquela pequena dúvida, vamos usar o `range`? O `range` é usado quando estamos trabalhando com mais de um vetor cujo índice deve pertencer a todos ao mesmo tempo, nesta questão não temos a entrada de dois vetores, mas temos um vetor contador também, nas outras questões não precisávamos usar o índice do `for` para trabalhar com o vetor contador, e agora será que precisamos do `range`? Vamos estudar isso agora.

A questão fala que o usuário irá inserir um vetor com números de 2 a 7, cada número inserido representa uma falta no dia especificado pela numeração, ou seja, se o 2 for inserido três vezes no vetor teremos três faltas no dia de segunda feira. Ou seja, a função do `for` é correr pelo vetor da história de faltas e fazer com que cada dia inserido seja contado no vetor contador, o vetor contador tem seis zeros cada posição representa um dia da semana começando pela segunda e terminando no sábado. O uso do `range` irá depender de como esse script será produzido, isso mesmo temos duas opções de construção desse script, a mais fácil e a mais rápida.

O JEITO MAIS FÁCIL deve ser aquilo que você pensou enquanto lia esta interpretação: "há vamos usar o `if`, `elif` e o `else`, de novo que nem as outras questões", posso lhe dizer que você não pensou errado, é simples fazer tudo isso com os comandos condicionais, mas é a forma mais demorada de ser fazer esta questão. Veja como é simples: SE o índice "i" do vetor das faltas for igual a 2 então contamos uma unidade na primeira posição do vetor contador, E SE o índice "i" do vetor das faltas for igual a 3 então contamos uma unidade na segunda posição do vetor contador OU SE o índice "i" do vetor das faltas não for igual a nenhum número de 1 a 6 então temos a última opção (sábado), com isso contamos uma unidade na última posição do vetor contador. Você percebe que estamos trabalhando somente com o vetor das faltas e não precisamos usar o índice "i" no vetor contador, pois ele usa suas próprias posições, nesse caso não precisamos usar o `range`, usamos somente o tamanho do vetor inserido que é o vetor das faltas.

```
# Contagem das faltas
for i in faltas:

    if faltas[i] == 2:
        vcont[0] = vcont[0] + 1

    elif faltas[i] == 3:
        vcont[1] = vcont[1] + 1

    elif faltas[i] == 4:
        vcont[2] = vcont[2] + 1

    elif faltas[i] == 5:
        vcont[3] = vcont[3] + 1

    elif faltas[i] == 6:
        vcont[4] = vcont[4] + 1

    else:
        vcont[5] = vcont[5] + 1
```

Veja ao lado a construção simples, o `for` percorre todo o vetor das faltas e os comandos condicionais analisam cada situação e caso certa situação for satisfeita por uma das condições o vetor contador entra em ação e começa a contar a ocorrência.

É bem fácil, porém veja o tamanho disso e o tempo para se construir um script tão simples chega a demorar e ser entediante, existe uma maneira mais rápida de se resolver essa questão, eu vou lhe ensinar, é melhor aprender os atalhos para que lá na frente você possa saber se safar de situações complicadas.

Figura 157: Questão 5F Primeiro Passo

Fonte: Autoria própria

O JEITO MAIS RÁPIDO é feito usando dois vetores, o segundo vetor é o próprio vetor contador, ou seja, o vetor contador fará parte do processo de análise sem precisar dos comandos if, elif e else. Para usarmos o range você sabe que o tamanho do vetor criado pelo range deve ser do tamanho do vetor inserido pelo usuário, ou seja, devemos usar a função size() para criarmos um intervalo a partir do zero que possa englobar perfeitamente qualquer vetor inserido pelo usuário. LEMBRE-SE DESSA DICA, QUANDO VOCÊ FOR USAR O RANGE A CONFIGURAÇÃO QUE ENGLOBA QUALQUER VETOR É ESSA:

for i in range(0, size(vetor)):

Se nós não vamos usar as condicionais então vamos pular diretamente para os vetores. Quando usamos o for estamos passeando pelos vetores, podemos passear pelo vetor contador também, toda vez que o índice correto chegar ao vetor contador o vetor irá adicionar uma unidade à posição indicada pelo índice, mas como fazer com que os índices de cada situação encaixem no vetor contador? Vamos pensar um pouco...

O vetor das faltas é um vetor que começa do 2 e vai até o 7, a distribuição das posições desse vetor fica: posição 0 é o número 2 (segunda); posição 1, número 3 (terça); posição 2, número 4 (quarta); posição 3, número 5 (quinta); posição 4, número 6 (sexta); posição 5, número 7 (sábado). Já o vetor contador tem a sua organização do mesmo jeito, só que para acessarmos o índice que corresponde a segunda (número 2) temos que digitar zero, para acessar o índice que corresponde aos valores de terça (número 3) temos que digitar um e assim se segue até que para acessar o índice que corresponde aos valores de sábado (número 7) temos que digitar 5. Você percebeu a diferença entre o índice e o valor real? Se temos o número 2 o índice é 0, se temos o número 7 o índice é 5; temos aqui a diferença de duas unidades, ou seja, caso estejamos mexendo com os valores de segunda feira devemos subtrair o 2 pelo número 2 que chegamos ao zero que é o índice do número 2, a mesma coisa se repete com todos os outros números, com isso chegamos à conclusão de que para alcançar a real posição a qual está o dia da semana devemos subtrair o valor por 2.

Então para que possamos identificar qual é o dia que está no certo índice do vetor temos que subtrair o valor que está nesse índice por 2, por exemplo o for começa a percorrer o vetor das faltas e na primeira posição, usando o comando vetor[i], chegamos ao valor que está nessa primeira posição, caso esse valor seja o número 3 (terça feira) ao subtrairmos esse número por 2 chegamos ao número 1, esse número 1 será usado como índice para o vetor contador que irá somar uma unidade na posição 1 do vetor. A mesma coisa se repetirá a cada vez que o for passear pelo vetor das faltas. Veja a construção dessa ideia:

```
# Contagem das faltas
for i in range(0, size(faltas)):
    dia = faltas[i] - 2
    vcont[dia] = vcont[dia] + 1
```

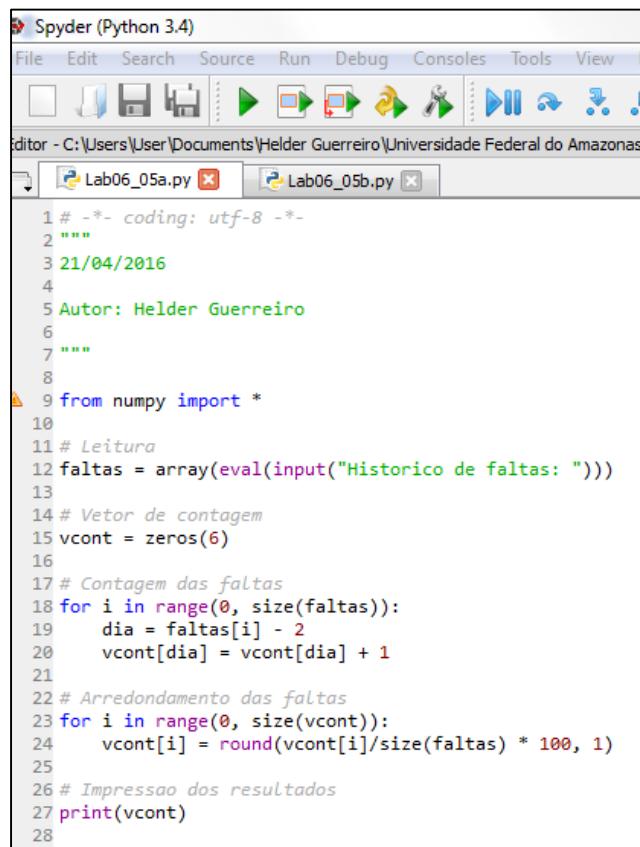
Veja a grandiosa diferença entre resolver esta questão com as condicionais if, elif e else e resolver esta questão usando somente a relação entre os índices dos vetores.

Figura 158: Questão 5F Primeiro Passo

Fonte: Autoria própria

Calma a questão não acabou, ainda temos que resolver um assunto: temos que apresentar o resultado em percentual, ou seja, se datamos cinco faltas e dessas cinco temos três faltas dos dias de segunda então o dia de segunda feira tem um percentual de $2/5 = 40\%$ das faltas cometidas. Como estamos trabalhando com vetores não podemos simplesmente dividir o vetor pelo total, nós temos que mexer nos valores do vetor e para isso existe o for, que irá percorrer o vetor indo de índice em índice e assim possamos modificar os valores dentro de cada índice,

Esse for será usado depois do outro ter terminado seu trabalho, assim o vetor contador já estará terminado e pronto, para fazermos o percentual de cada valor é só pegar o valor de cada índice do vetor contador e dividir pelo total, o total é quantidade de faltas inseridas pelo usuário, ou seja, o tamanho do vetor inserido, o for será configurado com o range da mesma forma que foi feito no for anterior. Se você não entendeu com será feito o mine script do segundo for vou explicar melhor: o for irá percorrer o vetor contador, use o comando vetor[i] para selecionar o valor da atual posição, esse valor será igual a sua modificação, isso quer dizer que o valor selecionado será modificado e a modificação é justamente o percentual, dividir o próprio valor inserido pelo total de faltas (use o size) e multiplicar por 100, sem se esquecer do **round** arredondando para uma casa decimal. No final de tudo basta apresentar o valor final do vetor contador.



```

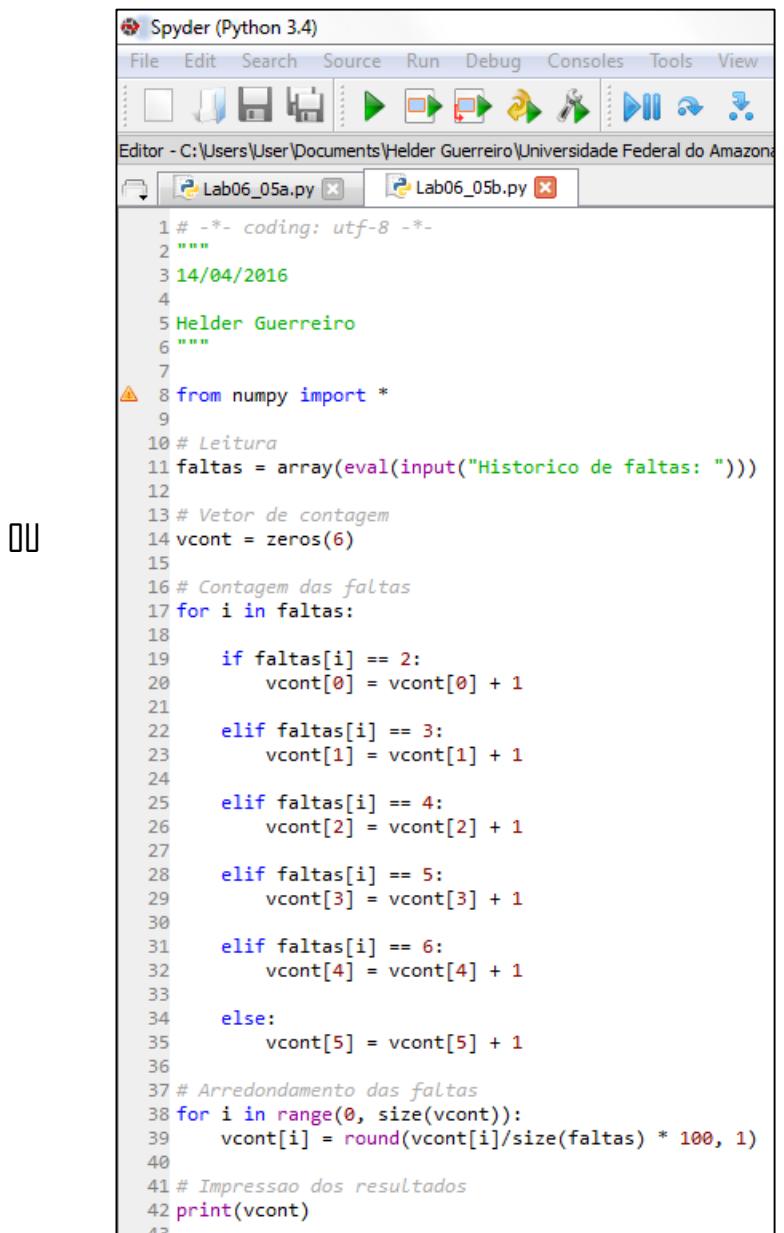
Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas
Lab06_05a.py Lab06_05b.py

1 # -*- coding: utf-8 -*-
2 """
3 21/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Leitura
12 faltas = array(eval(input("Historico de faltas: ")))
13
14 # Vetor de contagem
15 vcont = zeros(6)
16
17 # Contagem das faltas
18 for i in range(0, size(faltas)):
19     dia = faltas[i] - 2
20     vcont[dia] = vcont[dia] + 1
21
22 # Arredondamento das faltas
23 for i in range(0, size(vcont)):
24     vcont[i] = round(vcont[i]/size(faltas) * 100, 1)
25
26 # Impressao dos resultados
27 print(vcont)
28

```

Figura 159: Questão 5F Primeiro Passo

Fonte: ICOMP, UFAM



```

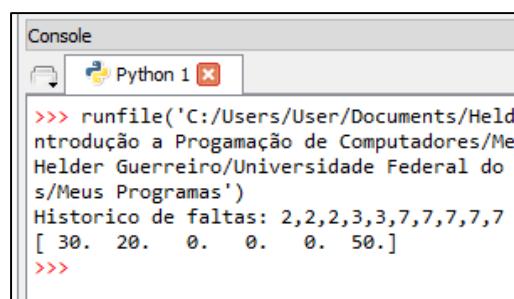
Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas
Lab06_05a.py Lab06_05b.py

1 # -*- coding: utf-8 -*-
2 """
3 14/04/2016
4
5 Helder Guerreiro
6 """
7
8 from numpy import *
9
10 # Leitura
11 faltas = array(eval(input("Historico de faltas: ")))
12
13 # Vetor de contagem
14 vcont = zeros(6)
15
16 # Contagem das faltas
17 for i in faltas:
18
19     if faltas[i] == 2:
20         vcont[0] = vcont[0] + 1
21
22     elif faltas[i] == 3:
23         vcont[1] = vcont[1] + 1
24
25     elif faltas[i] == 4:
26         vcont[2] = vcont[2] + 1
27
28     elif faltas[i] == 5:
29         vcont[3] = vcont[3] + 1
30
31     elif faltas[i] == 6:
32         vcont[4] = vcont[4] + 1
33
34     else:
35         vcont[5] = vcont[5] + 1
36
37 # Arredondamento das faltas
38 for i in range(0, size(vcont)):
39     vcont[i] = round(vcont[i]/size(faltas) * 100, 1)
40
41 # Impressao dos resultados
42 print(vcont)
43

```

Figura 160: Questão 5F Primeiro Passo

Fonte: Autoria própria



```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Held
ntrodução a Progamação de Computadores/M
Helder Guerreiro/Universidade Federal do
s/Meus Programas')
Historico de faltas: 2,2,2,3,3,7,7,7,7,7
[ 30.  20.   0.   0.  50.]
>>>

```

Figura 161: Resultado questão 5F

Fonte: Autoria própria

Questão 6 - Fração contínua (I): Raiz quadrada de dois

Escreva um programa que leia um número N do teclado e determine o valor de $\sqrt{2}$ através da sua expansão em fração contínua com N termos. Na saída, devem ser mostrados os resultados intermediários, com seis casas decimais de precisão. Utilize uma semente igual a 1.

$$\sqrt{2} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{\dots}}}}$$

DICAS:

1. Na expressão da fração contínua, identifique qual a parte constante e qual a parte que se repete. Esta última deverá ser controlada por uma variável acumuladora dentro de um laço de repetição.
2. Utilize uma variável acumuladora para guardar o valor parcial da parte cíclica, e uma variável contadora para controlar o número de iterações. Note que elas são independentes uma da outra.
3. Use a função `round(x, n)`, nativa da linguagem Python, para fazer arredondamento do número x com até n casas decimais de precisão.
4. Eis alguns exemplos (não exaustivos) de entradas e saídas:
 - a. Para a entrada 1, a saída deverá ser 1.333333.
 - b. Para a entrada 2, a saída deverá ser 1.333333 1.428571.
 - c. Para a entrada 5, a saída deverá ser 1.333333 1.428571 1.411765 1.414634 1.414141.

Primeiro passo: Interpretação da questão

Sobre esse tipo de questão vou lhe dar uma dica: SEMPRE DEVE-SE REPRESENTAR A FÓRMULA MOSTRADA NO ENUNCIADO DENTRO DO PYTHON, DA MESMA FORMA, isso quer dizer que nosso objetivo é fazer com que o mesmo resultado que a formula apresentada no enunciado consigue também devemos conseguir através dos nossos recursos do Python, sempre teremos que representar séries e formulas no script em questões como essas.

Essa questão é tão simples que no final você nem vai acreditar pelo tamanho do script. Primeiramente sabemos que precisamos somente de uma entrada, essa entrada é o número de iterações, ou seja, o número de vezes que aquelas frações da formula irão se repetir. Aqui não precisamos usar a biblioteca aqui, por que a entrada não é um vetor e o `for` não depende dessa biblioteca.

Como não temos vetor algum temos que trabalhar com o `range` usando o intervalo estabelecido de 0 a N , esse N é o número de interações, é para isso que ele serve: para definir o intervalo do range. E agora como representar aquela formula estranha do $\sqrt{2}$?

Vamos analisar com calma.

$$\sqrt{2} = 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \frac{1}{\dots}}}}$$

Olhando com calma a formula temos que analisar as partes que sempre se repetem, as partes repetitivas são o coração das formulas e séries e são elas que entrarão no script.

- O número 1 no numerador se repete sempre;
- O número 2 somando a fração se repete sempre;
- Sempre haverá uma fração.

Vamos usar essas repetições como padrão para a criação dessa formula no Python.

Se juntarmos todas as informações repetitivas que pegamos temos o seguinte:

$$\frac{1}{2}$$

Só isso? Mas é um grande passo!

Veja esse é o número que sempre se repete, a diferença que temos aqui é que o 2 soma com outra fração, ou seja, o 2 é o último número que não repete e depois dele já teremos outra iteração fazendo a mesma coisa.

Mas como dar continuidade as frações depois do 2?

Lembre-se que estamos usando o comando for e temos o N que fará parte do intervalo do range, o N é o número de iterações, ou seja, a quantidade de vezes que essas frações se repetirão. Se a base de toda essa formula é o número $\frac{1}{2}$ então ele é que será repetido! A base da formula é a fonte de repetição.

Se fizermos $x = \frac{1}{2}$ teremos uma variável x correspondente a uma iteração, isso quer dizer que uma iteração é igual a $\frac{1}{2}$!

Lembre-se, depois do 2 não temos a continuidade das iterações? Então quer dizer que o 2 é o limite da base e depois dele temos o começo de uma nova iteração, ou seja, um novo x!

Colocando nossa ideia em prática temos:

$$x = \frac{1}{2 + x}$$

Mas e aquele número 1 somando ali no início da formula?

Veja bem o que esse número 1 está somando... Ele está somando todas as iterações ao mesmo tempo! Isso mesmo, preste atenção que todos os números 2 que estão na formula estão no denominador, ou seja, eles são dependentes das frações e não poder ser somados depois da conta feita, mas o número 1 está sozinho e somente depois de todas as iterações calculadas é que o 1 será somado (lembre-se daquela história de primeiro a divisão depois multiplicação e depois soma e subtração). Quero dizer com tudo isso que podemos colocar esse número 1 somando tudo no final da conta sem problema. As iterações não são representadas por x? Então basta somar o x depois da conta das iterações.

O for entra em ação para fazer a variável x acumular todo o seu valor, ou seja, a cada contada do for o x irá acumular mais uma iteração e assim se segue.

Antes de cortermos felizes para o abraço temos que tomar cuidado com uma coisa, usamos o x para que seja a variável responsável por carregar todas as iterações, mas não podemos criar uma variável dentro da outra assim sem

que ela tenha um valor antes, não tem problema criar variáveis só que essa variável está igualando a ela mesma sem ter valor algum. Não entendeu?

Se criamos uma variável devemos dar um valor a ela e aí sim usar ela dentro dela mesma, mas do jeito que fizemos o x ainda não tem valor, o seu valor seria $\frac{1}{2}$, mas nós colocamos o x ali de novo, o x não pode estar ali por que ele ainda não tem valor, o script ainda não foi lido totalmente, então se colocarmos a formula do jeito que está aí o Python irá ignorar esse x no denominador por que ele ainda não tem valor (como se ele fosse 0) e desse jeito a conta iria ter um resultado diferente.

Depois dessa explicação já posso falar que devemos dar um valor ao x antes da conta começar, ou seja, antes de chegar no for. Mas qual valor colocar? Olha essa dica: SEMPRE A QUESTÃO DEVERÁ DAR A VOCÊ, é o que chamamos de "semente" no enunciado da questão fala que a semente deve ser igual a 1. Essa semente é uma referência a qual será crucial para o cálculo das frações.

Depois de tanto trabalho o processo das iterações já está pronto, agora não devemos nos esquecer daquele número 1 que soma todo mundo, basta criar uma variável (vamos chama-la de "raiz2") e fazer $x + 1$, onde x é o conjunto acumulado de todas as iterações. Depois dessa guerra hora de mostrar o resultado com a função `print` e `round` de 6 casas decimais, só que devemos analisar, como a questão está pedindo para demostrar o resultado para cada iteração então as funções devem estar dentro do mine script do for, por que toda vez que o for repetir seu mine script as funções print e round serão repetidas também.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade F
Lab06_06.py

1 # -*- coding: utf-8 -*-
2 """
3 22/04/2016
4
5 Autor: Helder Guerreiro
6 """
7 """
8
9
10 # Leitura
11 N = int(input("Numero de iteracoes: "))
12
13 # Semente
14 x = 1
15
16 for i in range(0, N):
17     x = 1 / (2 + x)
18
19     raiz2 = x + 1
20
21     print(round(raiz2, 6))
22

```

Figura 162: Questão 6F Primeiro Passo

Fonte: ICOMP, UFAM

```

Console
Python 1
>>> runfile('C:/Users/User/Introdução a Programação deelder Guerreiro/Universidade/Meus Programas')
Número de iteracoes: 2
1.333333
1.428571
>>>

```

Figura 163: Resultado questão 6F

Fonte: Autoria própria

Questão 7 – Arte ASCII (I)

Escreva um programa que imprima uma figura semelhante à representada abaixo, a partir da leitura do número de asteriscos presentes na base. Por exemplo, para uma entrada igual a 5, o resultado produzido será igual ao da figura abaixo.

```
*****  
****  
***  
**  
*  
*  
**  
***  
****
```

DICAS:

1. Utilize duas variáveis contadoras de laço: uma para controlar o número de asteriscos impressos em uma linha e outra para controlar a quantidade de linhas a serem impressas.
2. Divida para conquistar: veja que a figura é composta por dois triângulos. Use um laço para desenhar o primeiro e outro laço para desenhar o segundo triângulo.

Primeiro passo: Interpretação da questão

Para que uma questão dessa? Bom eu não sei exatamente.... Mas o ASCII é a lista de códigos que estão presentes no seu teclado, ou seja, ponto, ponto e vírgula, asterisco, barra etc. Tudo isso faz parte do ASCII.

O que temos de fazer é o seguinte: o usuário irá inserir um número, esse número é o tamanho da base da imagem que nós vamos fazer, ou seja, se a pessoa colocou o número 5 então o início e o fim da figura terá três asteriscos. Para fazermos essa coisa toda temos que trabalhar com o **for** e como não temos vetores o **range** também entra no meio, não tem nada de bico papão aqui, nós só vamos trabalhar com **for** e **print** aqui.

Como usar o **for** para fazer isso? Primeiro temos que fazer com que a cada linha a quantidade de asteriscos caia uma unidade até chegar a um e amentar de novo, para isso temos que usar dois **for**! Isso mesmo, nós não podemos simplesmente multiplicar o asterisco por 5 e pronto, precisamos fazer com que a quantidade de asteriscos mude a cada linha e só podemos fazer isso mudando o tamanho do vetor do **range**, e o tamanho do vetor **range** não muda assim sozinho não, precisamos de outro **for** para que esse **for** venha mudar o outro a cada linha.

A entrada é o tamanho da base da arte e através dela os dois **for** irão trabalhar. O primeiro **for** irá comandar o segundo, ele serve para mudar o tamanho do **range** a cada repetição, você já sabe que pode usar qualquer variável (chame-a de "j" se quiser) e o **range** terá seus limites estabelecidos de 0 ao número de entrada (chame-o de N), estamos usando o limite de 0 a N por que só temos eles como base para o nosso intervalo não?

O primeiro for só serve para isso mesmo, ele irá modificar o tamanho do range do segundo for e irá pular uma linha para a continuidade da exibição de asteriscos.

O segundo for será responsável pela exibição de asteriscos numa mesma linha somente, só quando o primeiro for repetir o segundo será acionado novamente e dessa vez numa outra linha com uma quantidade diferente de asteriscos. Use uma outra variável no segundo for (chame-o de "i" se quiser) e o range deve ter uma configuração que possa fazer seu vetor diminuir a cada repetição, é só pensar assim: quais as referências que temos? Temos o tamanho da base N, a variável do primeiro for j, e claro que temos que começar o intervalo do 0. Já sabemos que o intervalo do segundo range começa do 0, já o final do intervalo tem que ter uma relação com o primeiro for. Pensa assim comigo, na primeira repetição qual será o valor do "j"? Se o range do primeiro for é de 0 a N então o j vale 0 na primeira repetição do for, logo depois que sai do primeiro for o script já vai logo pulando para o segundo, esse for será repetido até que acabe seu intervalo e assim possa seguir a repetição do primeiro for.

Vamos supor que o usuário insere uma base de 5:

O segundo for deve primeiro executar o print cinco vezes;

Quando ele tiver de executar de novo, o print deve ser executado quatro vezes;

Na terceira vez, o print deve ser executado três vezes;

Na quarta vez, o print deve ser executado duas vezes;

Na quinta e última vez, o print deve ser executado somente uma vez.

É assim que o segundo for deve ser executado, de forma decrescente e linear, vamos analisar o primeiro for agora?

O primeiro for terá sua variável igual a 0 na primeira execução;

Na segunda execução, a variável é igual a 1;

Na terceira execução, a variável é igual a 2;

Na quarta execução, a variável é igual a 3;

Na quinta e última execução (o final do intervalo é 5, mas o range conta com o zero), a variável é igual a 4.

Percebeu a relação?

- 1) Primeiro for: j = 0, segundo for: print 5 vezes;
- 2) Primeiro for: j = 1, segundo for: print 4 vezes;
- 3) Primeiro for: j = 2, segundo for: print 3 vezes;
- 4) Primeiro for: j = 3, segundo for: print 2 vezes;
- 5) Primeiro for: j = 4, segundo for: print 1 vez.

Com isso eu posso dizer a você que a chave desse enigma é uma subtração da base pelo j (variável do primeiro for):

$$5 - 0 = 5$$

$$5 - 1 = 4$$

$$5 - 2 = 3$$

$$5 - 3 = 2$$

$$5 - 4 = 1$$

Ou seja, o intervalo do range do segundo for é simplesmente de 0 a base menos j.

No meu script do segundo estará o print para que aja a representação dos asteriscos a cada execução do for, só que não somente fazer um print simples com um asterisco não... para isso vou ensinar a você comandos que vão lhe ajudar a trabalhar melhor com o print.

Dois novos comandos vou ensinar a você aqui: Comando "end=" e comando "\n".

O comando "end=" é um comando que permite com que a próxima execução do print seja ao lado da última impressão e não embaixo como de costume, ele deve ser colocado dentro da função print separado com vírgula, o comando deve ser precedido de duas aspas, assim: print("", end="")

Se você colocar alguma coisa dentro das aspas do end o que você colocou aparecerá ao lado da mensagem que o print executou, não é inteligente usar isso para resolver essa questão.

O comando "\n" serve para se pular um espaço, ou seja, se terminamos os asteriscos numa linha usamos esse comando para ir para outra linha. Por que temos de pular para outra linha se o print normal já executa na outra linha? Por que ao usar o comando "end=" a próxima impressão será exatamente ao lado da última, ao usar o comando "\n" nós cancelamos o comando do "end=".

O comando "\n" deve ser colocado dentro das aspas do print como se fosse uma mensagem normal, mas devemos colocar o comando "end=" também aqui por que ao colocar somente o comando de pular a linha o espaço será grande demais, então os dois comandos juntos formam tipo a mesma coisa ao apertar a tecla ENTER no Word.

O primeiro for irá englobar o segundo, ele serve somente para fazer variar o range do segundo for e o segundo for serve para imprimir os asteriscos numa fila, então é função do primeiro for pular uma linha. No segundo for o print dos asteriscos deve estar acompanhado do comando "end=" para que eles sejam imprimidos em fileira.

Se você for executar esse script você perceberá que estaremos amostrando somente uma parte da arte, só a parte de cima! Isso quer dizer que temos que fazer a parte de baixo também. Se você olhar a imagem do enunciado os asteriscos (considerando base 5) começa com 5 e ao descer vai diminuindo até o 1, depois vai crescendo de novo até o 5.

Para fazermos a parte de baixo é a MESMA coisa que a de cima, basta repetir tudo que fizermos até agora (se quiser pode copiar mesmo), a ÚNICA coisa que vai mudar é o intervalo do range do segundo for, por que o intervalo do primeiro for não tem como mudar e ele ainda continua pulando as linhas e o segundo for continua imprimindo os asteriscos em fila. Temos que fazer um intervalo para o range de tal forma que ele faça com que os asteriscos cresçam de 1 até o tamanho da base.

Vamos voltar a análise do primeiro e do segundo for? Nesta parte de baixo eles deverão se comportar assim:

- 1) Primeiro for: j = 0; segundo for: print 1 vez;
- 2) Primeiro for: j = 1; segundo for: print 2 vezes;
- 3) Primeiro for: j = 2; segundo for: print 3 vezes;
- 4) Primeiro for: j = 3; segundo for: print 4 vezes;
- 5) Primeiro for: j = 4; segundo for: print 5 vezes.

Analice a relação entre os dois, vou lhe dar uma dica: AS RELAÇÕES SEMPRE DEVEM SER FEITAS MATEMÁTICAMENTE, então a relação que temos aqui é que a quantidade de print é uma unidade a mais do j, é dessa for que temos de colocar no intervalo do range: j + 1.

The screenshot shows the Spyder IDE interface with the title bar "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, and Tools. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do A..." and contains a file named "Lab06_09.py". The code in the file is as follows:

```

1 # -*- coding: utf-8 -*-
2 """
3 25/04/2016
4
5 Autor: Helder Guerreiro
6
7
8 *****
9 ****
10 ***
11 **
12 *
13 *
14 **
15 ***
16 ****
17 *****
18
19 """
20
21 # Leitura
22 base = int(input("Tamanho da base: "))
23
24 # Triangulo superior
25 for j in range(0, base):
26     for i in range(0, base - j):
27         print("*", end="")
28     print("\n", end="")      #Termina a Linha
29
30 # Triangulo inferior
31 for j in range(0, base):
32     for i in range(0, j+1):
33         print("*", end="")
34     print("\n", end="")      #Termina a Linha
35

```

Figura 164: Questão 7F Primeiro Passo

Fonte: ICOMP, UFAM

The screenshot shows the Spyder IDE console window titled "Console" with a tab "Python 1". The command "runfile('C:/User/introdução a Programação/Helder Guerreiro/Univ/Meus Programas')" is entered, followed by the output of the program which prints a triangular pattern of asterisks.

```

>>> runfile('C:/User/introdução a Programação/Helder Guerreiro/Univ/Meus Programas')
Tamanho da base: 5
*****
*****
***
**
*
*
**
***
*****
*****

```

Figura 165: Resultado questão 7F

Fonte: Autoria própria

Isso poderia ser feito também com um monte print's num script só, mas veja quanto é mais prático fazer do nosso jeito: ($N = 3$)

```

20
21 # Arte de tamanho de base 3
22
23 print("*",end="")
24 print("*",end="")
25 print("*",end="")
26
27 print("\n",end="")
28
29 print("*",end="")
30 print("*",end="")
31
32 print("\n",end="")
33
34 print("*",end="")
35
36 print("\n",end="")
37
38 print("*",end="")
39
40 print("\n",end="")
41
42 print("*",end="")
43 print("*",end="")
44
45 print("\n",end="")
46
47 print("*",end="")
48 print("*",end="")
49 print("*",end="")
50
51 print("\n",end="")
52
53

```

Figura 164: Questão 7F Primeiro Passo

Fonte: Autoria própria

Quanto maior a base, mais print's.

6.3 Avaliações

Aqui teremos algumas avaliações para entender mais um pouco o assunto estudado neste primeiro capítulo. Aconselho você tentar fazer sozinho antes de ler a resolução.

Avaliação A - A6

Avaliação Parcial 06 - Repetição por contagem - AV6 - a

Informações

Questões

Notas

Questão 1

Uma empresa fez um levantamento dos anos de nascimento dos seus funcionários. Apurou-se que os funcionários mais antigos nasceram em 1950 e os funcionários mais novos nasceram em 1995. Os anos de nascimento de cada funcionário foram armazenados em um vetor.

Escreva um programa que leia esse vetor a partir do teclado e conte o número de funcionários que nasceram em cada ano de 1950 a 1995. A saída do programa deverá ser armazenada em outro vetor, cujo primeiro elemento indica o número de funcionários que nasceram em 1950, e assim sucessivamente até o último elemento, que indica o número de funcionários que nasceram em 1995.

Dicas:

1. Verifique se você está submetendo o código correto à questão correspondente desta Avaliação Parcial.
 2. Exemplo (não exaustivo): para a entrada [1994, 1995, 1994, 1994, 1950, 1995], a saída deverá ser [1 0 3 2], indicando que um funcionário nasceu em 1950, três nasceram em 1994 e dois nasceram em 1995.

Vou lhe ensinar um jeito fácil de fazer esta questão, preste atenção. Veja que o script deve ditar quem são os funcionários nascidos de 1950 até 1995, a diferença entre os dois é 46, tá certo que se subtraímos 1995 por 1950 o resultado será 45 e não 46, mas lembre-se que estamos trabalhando com anos e não simples números, a diferença numérica entre os dois pode até ser 45, mas para que um ano seja contado por completo ele deve ir até ao final, pois se chegarmos até a data de primeiro de janeiro de 2016 não se pode contar o ano de 2016 numa conta por que ele não está completo, mas o de 2015 sim.

O vetor que irá dar o resultado é um vetor contador, ou seja, o cara tem que ter 46 zeros, você não vai digitar tudo isso né, use a função zeros() e não esqueça do comando dtype=int. A entrada será um vetor contendo os anos de nascimentos dos funcionários da empresa.

Já fizemos uma questão em que usamos um vetor contador para contar as ocorrências de acordo com uma condição, mas eu acho que ninguém quer fazer 46 condições diferentes em `if`, `elif` e `else` né? Na questão 5 mostrei que existe uma alternativa para fugir dessa grande quantidade de condições e terminar tudo em duas linhas, vamos usar essa

saída. Lembre-se quando se for tentar resolver questões nesse estilo tenha em mente que a resolução é sempre variável de acordo com a questão, no caso das condições é só montar os comandos, mas neste caso não, deve-se usar a cabeça.

Você se lembra de como eu fiz para achar a diferença de 46 entre o primeiro e o último ano? Essa é a nossa chave, toda vez que você estiver correndo atrás para resolver um problema, pense matematicamente! Aqui iremos usar o `for` com certeza, para que possamos passear dentro do vetor inserido a partir daí possamos analisar cada índice desse vetor.

Para que possamos contar as ocorrências dentro do vetor contador precisamos encontrar um jeito de fazer com que cada valor inserido venha entrar no seu respectivo índice do vetor contador, por exemplo, se o ano de 1950 é inserido ele pertence ao índice 0 do vetor contador que será somado a 1 contando a ocorrência. Lembre-se, com a diferença de anos cheguei à conclusão que temos 46 anos de diferença entre o primeiro e o último, 46 também é o último índice do vetor, ou seja, ao fazer $1995 - 1950 = 45$ estaremos selecionando o índice 45 do vetor contador (lembre-se os vetores contam o zero como número, então o índice 45 representa o 46º zero do vetor). Perceba uma coisa, estamos usando o 1950 como referência, veja que se fizermos $1950 - 1950 = 0$ estaremos selecionando o primeiro índice do vetor e realmente esse é o lugar dele.

Com isso devemos usar o comando “vetor[i]” para selecionar o valor do vetor inserido, pois se o valor selecionado for 1950 ao subtrair por 1950 resultamos 0 que será usado como índice no vetor contador, se o valor selecionado for 1985 ao subtrair por 1950 resultamos 35 que será usado como índice no vetor contador, ou seja, primeiro vamos calcular a qual índice o ano inserido pertence e depois esse resultado será usado como índice no vetor contador.

Chamando a conta que calcula o índice de x, ao usar dentro do vetor contador, usando o comando “vetor[i]”, selecionamos o índice representante do ano inserido, e agora temos que fazer com que essa ocorrência seja contada.

Quando se iguala uma variável a ela mesma com algumas modificações você está trocando um valor por outro, ou seja, antes o Python entendia tal variável como x e agora ele troca a leitura por y. Por isso ao igualar o vetor contador, com o índice x selecionado, a ele mesmo estamos dizendo que agora o valor do vetor contador no índice x será modificado, como todos os valores do vetor contador são iguais a zero então estamos dizendo que: “vetor[x]” = 0, mas como queremos modificar o valor nós somamos o próprio comando a 1, para que toda vez que o for se repetir o índice x seja calculado e o valor em que esse índice x estiver seja somado a 1.

The screenshot shows the Spyder Python IDE interface. The title bar reads "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Pe" and contains the file "AV_A.py". The code in the editor is:

```
1 """
2 26/04/2016
3
4 Autor: Helder Guerreiro
5
6 """
7
8
9
10 from numpy import *
11
12 z = zeros(46, dtype = int)
13
14 vet = array(eval(input("Digite os anos de nascimento: ")))
15
16 for i in range(size(vet)):
17     x = vet[i] - 1950
18     z[x] = z[x] +1
19 print(z)
20
```

Figura 167: Resposta Avaliação A SA

Fonte: Autoria própria

Avaliação B - A5

Avaliação Parcial 06 - Repetição por contagem - AV6 - b

Informações

Questões

Notas

Questão 1

Escreva um programa que leia um número N do teclado e determine o valor da raiz quadrada de 26 através da sua expansão em fração contínua com N termos. Na saída, devem ser mostrados os resultados intermediários, com oito casas decimais de precisão. Utilize uma semente igual a 14.

$$\sqrt{26} = 5 + \cfrac{1}{10 + \cfrac{1}{10 + \cfrac{1}{10 + \cfrac{1}{10 + \dots}}}}$$

Dicas:

1. Na expressão da fração contínua, identifique qual a parte constante e qual a parte que se repete. Esta última deverá ser controlada por uma variável acumuladora dentro de um laço de repetição.
2. Utilize uma variável **acumuladora** para guardar o valor parcial da parte cíclica, e uma variável **contadora** para controlar o número de iterações. Note que elas são independentes uma da outra.
3. Use a função **round(x, n)**, nativa da linguagem Python, para fazer arredondamento do número x com até n casas decimais de precisão.
4. Eis alguns exemplos (não exaustivos) de entradas e saídas:
 1. Para a entrada 1, a saída deverá ser **5.04166667** ($= 5 + 1/(10 + 14)$).
 2. Para a entrada 2, a saída deverá ser **5.04166667 5.09958506** ($= 5 + 1/(10 + 1/(10 + 14))$).
 3. Para a entrada 5, a saída deverá ser **5.04166667 5.09958506 5.09901397 5.09901957 5.09901951**.

Fizemos uma questão parecida com essa, a resolução é a mesma. A entrada será o número de iterações desejadas pelo usuário, ou seja, quão mais próximo do real valor o usuário deseja, sendo que nunca se chegará ao resultado final por que essas frações são contínuas, isto é, infinitas.

Ao criar o for não temos um vetor para percorrer então devemos usar o **range**, o intervalo do range é claramente o número N inserido pelo usuário, o número de iterações, então seu intervalo será de 0 a N. Dentro do for lembre-se que primeiramente deve se identificar qual é a base da fração contínua, ou seja, qual a numeração que sempre se repete. Olhe com cuidado as frações do enunciado...

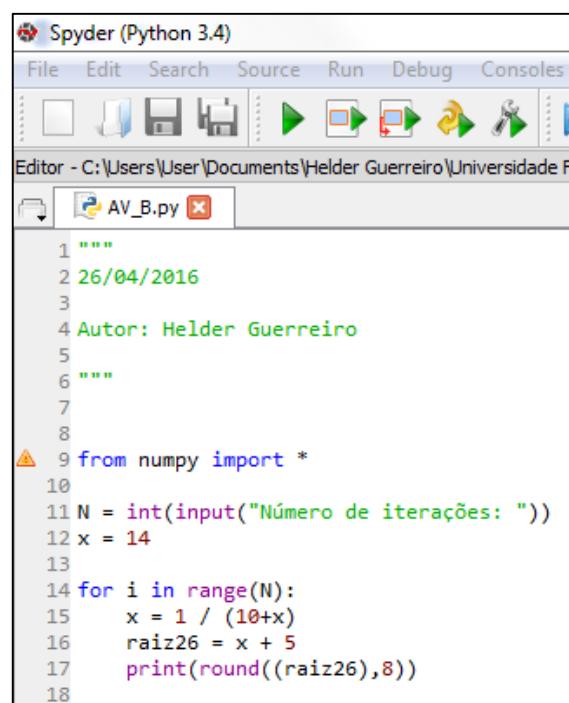
Temos então que a numeração que sempre se repete é:

$$\cfrac{1}{10 + x}$$

I/10 sempre se repete, a variável x representa uma outra iteração. Como o x representa uma iteração e a pequena equação acima é uma iteração também então o x é igual a equação acima, ele irá servir como se fosse uma variável

acumuladora que irá ajudar cada iteração feita dentro do `for`. Lembre-se que antes do `for` trabalhar temos que colocar a semente que neste caso é 14, então $x = 14$ antes dos cálculos começarem.

Depois da base e semente prontas, hora de pegar todas as iterações e somar pelo número individual, que é aquele número que fica somando todas as iterações e não depende de ninguém, neste caso é o número 5. Então crie uma variável chamada `raiz26` e iguale a $x + 5$, onde x é o acumulador de iterações. No final de tudo não esqueça de arredondar o resultado com oito casas decimais.



```
Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade F
AV_B.py

1 """
2 26/04/2016
3
4 Autor: Helder Guerreiro
5
6 """
7
8
9 from numpy import *
10
11 N = int(input("Número de iterações: "))
12 x = 14
13
14 for i in range(N):
15     x = 1 / (10+x)
16     raiz26 = x + 5
17     print(round((raiz26),8))
18
```

Figura 168: Resposta Avaliação B 5A

Fonte: Autoria própria

7. No Coração da Álgebra Linear – As Matrizes

Depois de encarar os vetores hora da evolução, dos vetores sairão as matrizes e elas irão nos mostrar o quanto elas são úteis no nosso dia a dia, e pare de reclamar que você vai estudar matrizes, você estuda isso desde o primeiro ano do ensino médio.

Nesta parte você verá que as mesmas contas que você faz no papel com matrizes você também fará no Python. Veja a construção de uma matriz na velha e conhecida álgebra linear:

- Em notação matemática, em uma matriz A qualquer, cada elemento é indicado por a_{ij} .
 - O índice i indica a **linha**.
 - O índice j indica a **coluna**.

$$A_{m \times n} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Figura 169: A Matemática de uma Matriz.

Fonte: ICOMP, UFAM

7.1 A Construção do Império Matricial

Para construção de matrizes precisamos usar também o módulo dos vetores, a biblioteca numpy. Agora iremos utilizar outra biblioteca para o manuseio de matrizes usamos a biblioteca numpy.linalg.

```
from numpy import *
from numpy.linalg import *
```

Figura 170: As Bibliotecas

Fonte: ICOMP, UFAM

A criação de uma matriz é bem parecida com a dos vetores, ainda continuaremos usando o array para criar uma matriz e as funções `eval` e `input` também estão na hora da entrada de matrizes pelo usuário. Na construção de um vetor separávamos os valores por vírgula, ainda faremos isso, só que dessa vez ao construir uma matriz iremos construí-la linha por linha, a divisão será feita entre colchetes, ou seja, a cada par de colchetes temos uma linha da matriz e os colchetes também serão separados por vírgulas.

```
A = array([[3, -4, 0], [5, 2, -1]])
```

Lembrar de importar a `numpy`

$$A = \begin{bmatrix} 3 & -4 & 0 \\ 5 & 2 & -1 \end{bmatrix}$$

Figura 171: A Construção

Fonte: ICOMP, UFAM

Perceba que além dos colchetes que dividem as linhas temos os colchetes que englobam todos os outros, se esquecer de qualquer um deles dará erro no seu script ao executar.

7.2 Funções e Operações de Matrizes

Os vetores têm seus comandos e suas artimanhas além de suas variadas funções e a interação com a matemática dos polinômios, pois é, as matrizes também têm suas próprias funções e comandos que nos ajudarão a fazer contas simples que podem formar de pequenos scripts grandes ideias.

7.2.1 As Dimensões

Esses comandos nos dão várias informações que podem nos ser bem úteis dependendo da situação:

- Número de `elementos`:

```
TAM = size(quadro)
```

- Número de `linhas` e de `colunas`:

```
FORMA = shape(quadro)
```

- Número de `linhas`:

```
LIN = quadro.shape[0]
```

- Número de `colunas`:

```
COL = quadro.shape[1]
```

Figura 172: As Dimensões

Fonte: ICOMP, UFAM

Essa palavra “quadro” é o nome da matriz analisada.

7.2.2 As Funções

Essas funções nos auxiliaram na geração de matrizes de acordo com o que queremos, dentre elas estão funções até mesmo usadas entre os vetores, mas que servem de ajuda aqui também:

<code>zeros((n,n))</code>	• Gera uma matriz n x n de zeros
<code>zeros((m,n))</code>	• Gera uma matriz m x n de zeros
<code>ones((n,n))</code>	• Gera uma matriz n x n de uns
<code>ones((m,n))</code>	• Gera uma matriz m x n de uns
<code>eye(n)</code>	• Gera uma matriz identidade n x n
<code>eye(m,n)</code>	• Gera uma matriz identidade m x n

Figura 173: As Funções

Fonte: ICOMP, UFAM

7.2.3 Seleção de Valores

Quando queremos selecionar um valor de um índice do vetor usamos o comando “vetor[i]”, aqui não muda muita coisa a única diferença é que agora temos dois índices e podemos fazer seleções de várias formas. Para selecionar simplesmente um elemento da matriz basta usa o comando “matriz[i, j]” onde o índice “i” é a linha e o índice “j” é a coluna, para outras formas de seleções temos:

- Seleção de todos os elementos da linha **i**:

`x = quadro [i, :]`

- Seleção dos **n primeiros** elementos da linha **i**:

`x = quadro [i, :n]`

- Seleção dos **n últimos** elementos da linha **i**:

`x = quadro [i, -n :]`

Figura 174: Seleção de Valores na Linha

Fonte: ICOMP, UFAM

□ Seleção de todos os elementos da coluna j :

$x = \text{quadro}[:, j]$

□ Seleção dos n primeiros elementos da coluna j :

$x = \text{quadro}[:n, j]$

□ Seleção dos n últimos elementos da coluna j :

$x = \text{quadro}[-n:, j]$

Figura 175: Seleção de Valores na Coluna

Fonte: ICOMP, UFAM

Devo lhe avisar que as matrizes também contam o zero tanto na linha quanto na coluna, ou seja, quando a contagem dos índices começa do 0 em diante.

Veja um exemplo dessas funções em ação:

Determinar:

1. $M[4, 1]$
2. $M[5, 3]$
3. $M[2, 3]$
4. $M[5, M[1, 3]]$
5. $M[M[4, 2], 2]$
6. $M[5, [M[2, 3] + M[4, 1]]]$
7. $M[3, :]$
8. $M[:, 3]$

1.	0	2.	-2
3.	1	4.	-1
5.	4	6.	-1
7.	-3, 2, 0, 0		
8.	4, 0, 1, 0, 1, -2		

	0	1	2	3
0	1	2	3	4
1	5	-5	4	0
2	1	1	1	1
3	-3	2	0	0
4	0	0	1	1
5	-1	-1	-2	-2

M

Figura 176: Exemplo do Uso de Seleções

Fonte: ICOMP, UFAM

Neste exemplo veja que do 1 ao 3 houve somente a seleção de um único valor da matriz nas coordenadas indicadas; na 4 foi selecionado a linha 5 e o índice da coluna é o mesmo valor do elemento nas coordenadas $M[1, 3]$ que é 0 (veja na tabela), ou seja, na 4 temos linha 5 e coluna índice 0 que vale -1; na 5 é a mesma coisa e a 6 só houve uma soma dos valores dos índices indicados e o resultado usado como índice da coluna; na 7 houve a seleção de toda a linha 3 e na 8 a seleção de toda a coluna 3.

7.2.4 Operações Entre Matrizes

Aqui chegamos numa parte onde iremos pôr em prática tudo aquilo que aprendemos a fazer no papel, só que dessa vez o Python irá fazer os cálculos para você.

Operação	Em Python
Soma estrutural	<code>a + b</code>
Subtração estrutural	<code>a - b</code>
Multiplicação estrutural	<code>a * b</code>
Multiplicação matricial	<code>dot(a,b)</code>

Figura 177: Operações Entre Matrizes a

Fonte: ICOMP, UFAM

Operação	Em Python
Elementos da diagonal principal	<code>diagonal(m)</code>
Transposta	<code>m.T</code>
Determinante	<code>det(m)</code>
Inversa	<code>inv(m)</code>

Figura 178: Operações Entre Matrizes b

Fonte: ICOMP, UFAM

Operações estruturais são operações que devem ser feitas com matrizes idênticas, as operações matriciais fazem os cálculos usando a velha e conhecida álgebra linear.

Apresento a você a uma função que ajuda bastante evitando cálculos desnecessários, a função `solve()` veja como ela funciona com esse exemplo simples da imagem:

$$x = A^{-1}B$$

x = inv(A) * B

ou

x = solve(A, B)

Figura 179: Função Solve

Fonte: ICOMP, UFAM

7.3 Questões Resolvidas

Estas questões foram desenvolvidas pela ICOMP um instituto da UFAM e certos algoritmos (scripts) apresentados aqui pertencem ao ICOMP.

Questão 1 – SEL(1): Cesta de frutas

Alice comprou três abacates, uma dúzia de bananas e um caqui por R\$23,60. Beto comprou uma dúzia de abacates e dois caquis por R\$ 52,60. Carol comprou duas bananas e três caquis por R\$ 27,70. Escreva um programa que determine o preço unitário de cada fruta.

DICA:

Utilize a biblioteca numpy.linalg

Primeiro passo: Interpretação da questão

Logo na primeira questão já temos uma novidade, as equações lineares. Você deve se lembrar das suas aulas de álgebra linear que podemos solucionar um sistema transformando-o em uma matriz e utilizando métodos de soluções para encontrar as respostas né, então como estamos trabalhando com matrizes estamos também trabalhando com sistemas, vou lhe mostrar como.

Primeiro temos que formar as matrizes, na primeira linha temos Alice que comprou 3 abacates, 12 bananas e 1 caqui, na segunda linha temos Beto que comprou 12 abacates e 2 caquis e na terceira está a Carol com 2 bananas e 3 caquis. Por que não colocamos os preços no meio também? Por que o Python não trabalha diretamente com sistemas, somente com matrizes, se criar uma matriz com todos os dados não teremos como resolver os sistemas, temos que ter uma matriz com os dados de quantidade e outra matriz com os dados de preços.

Formando a matriz acima temos:

$$qtd = \begin{pmatrix} 3 & 12 & 1 \\ 12 & 0 & 2 \\ 0 & 2 & 3 \end{pmatrix}$$

Veja que temos que colocar zeros aonde não temos valores, isso não é nenhuma novidade para você.

A segunda matriz é a matriz dos preços, veja que quando criamos as matrizes, nós escrevemos seus dados linha por linha e não coluna por coluna, desse jeito ao criarmos a matriz com os preços estaremos criando uma matriz com somente uma linha e três colunas, por isso devemos fazer a trasposta dessa matriz que irá fazer a matriz linha virá matriz coluna.

$$preço = (23,6 \quad 52,6 \quad 27,7) \rightarrow preço^T = \begin{pmatrix} 23,6 \\ 52,6 \\ 27,7 \end{pmatrix}$$

Estamos fazendo a trasposta dela justamente por que os preços devem ficar ao lado das quantidades né? Pelo menos eu aprendi sistemas assim. Para fazer a transposta use o comando “matriz.T”, depois de fazer a matriz igualar a variável da matriz a esse comando e pronto.

Então nosso objetivo é fazer isso:

$$\begin{cases} 3x & 12y & 1z = 23,6 \\ 12x & 0y & 2z = 52,6 \\ 0x & 2y & 3z = 27,7 \end{cases}$$

A resolução é mais simples do que se pensa graças a função `solve()` podemos resolver essa equação em dois temos, basta as matrizes estarem compatíveis e pronto, a função `solve` junta as matrizes e calcula o sistema para você da mesma forma que fazemos no papel, mas é claro que no computador é rápido demais. Para usar a função `solve()` deve-se colocar as variáveis das duas matrizes dentro dela separadas por vírgula, primeiro vem a matriz das quantidades e depois a dos preços. Pronto a questão acabou!

Se fossemos resolver esta questão no papel iríamos precisar usar os métodos que temos disponíveis para resolver essas contas, são eles: regra de Cramer, método de Gauss e método de Gauss-Jordan, esses são os mais famosos.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools
Editor - C:\Users\Helder Guerreiro\Universidade Federal do A...
Lab07_01.py
1 # -*- coding: utf-8 -*-
2 """
3 27/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10 from numpy.linalg import *
11
12 # Matriz do sistema linear
13 qtd = array([[3, 12, 1], [12, 0, 2], [0, 2, 3]])
14
15 # Matriz linha
16 preço = array([23.6, 52.6, 27.7])
17 # Matriz coluna
18 preço = preço.T
19
20 # Resolução do sistema de equações lineares
21 solução = solve(qtd, preço)
22
23 print(solução)

```

Figura 180: Questão 1G Primeiro Passo

Fonte: Autoria própria

```

Console
Python 1
>>> runfile('C:/Users/User/.../Meus Programas')
[ 2.9  0.5  8.9]
>>>

```

Figura 181: Resultado questão 1G

Fonte: Autoria própria

Questão 2 – SEL(2): Alimentando Bactérias

Um biólogo colocou três cepas de bactérias (indicadas por I, II e III) num tubo de ensaio, onde elas se alimentam de três fontes diferentes de alimentos (A, B e C). A cada dia são colocadas no tubo de ensaio 2300 unidades do alimento A, 800 unidades de B, e 1500 unidades de C. Cada bactéria consome um certo número de unidades de cada alimento por dia, como mostra a tabela abaixo.

	Cepa I	Cepa II	Cepa III
Alimento A	2	2	4
Alimento B	1	2	0
Alimento C	1	3	1

Escreva um programa que determine quantas bactérias de cada cepa pode coexistir no tubo de ensaio e consumir toda a comida?

DICAS:

1. Utilize a biblioteca `numpy.linalg`, além da própria `numpy`.
2. Não há entradas de teste. Os parâmetros do problema devem ser informados como constantes no corpo do script.
3. A saída deverá ser um vetor de três elementos reais, semelhante a `[10. 20. 30.]`, indicando que o biólogo deve colocar dez bactérias da cepa I, vinte da cepa II e trinta da cepa III no tubo de ensaio, a fim de consumir todo o alimento.

Primeiro passo: Interpretação da questão

A resolução desta questão é exatamente igual a anterior, aqui o biólogo quer otimizar sua cultura de bactérias fazendo com toda a quantidade de alimento seja consumida pelas bactérias. A ideia é a mesma, a tabela acima já nos mostra a primeira matriz, é a matriz das quantidades, neste caso é as quantidades de alimentos consumidos.

A outra matriz será a matriz dos valores a serem consumidos, é aquele mesmo esquema: faz a matriz linha e torne-a matriz coluna usando a transposta a fim de fazermos um sistema.

$$\begin{cases} 2x & 2y & 4z = 2300 \\ 1x & 2y & 0z = 800 \\ 1x & 3y & 1z = 1500 \end{cases}$$

Agora não tem mais aquela tribulação que é encher o papel de várias matrizes e contas e no final errar toda a conta, agora é só usar a função `solve()` e tchau e benção.

The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, and Tools. Below the menu is a toolbar with various icons. The main area is titled 'Editor - C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Rio Grande do Norte' and contains the following Python code:

```
1 # -*- coding: utf-8 -*-
2 """
3 27/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10 from numpy.linalg import *
11
12 # Matriz do sistema linear
13 qtd = array([[2, 2, 4], [1, 2, 0], [1, 3, 1]])
14
15 # Vetor de constantes
16 consumo = array([2300, 800, 1500])
17 consumo = consumo.T
18
19 # Resolucao do sistema de equacoes lineares
20 solucao = solve(qtd, consumo)
21
22 print(solucao)
23
```

Figura 182: Questão 2G Primeiro Passo

Fonte: Autoria própria

The screenshot shows the Spyder Console window with a tab labeled 'Python 1'. The console output is:

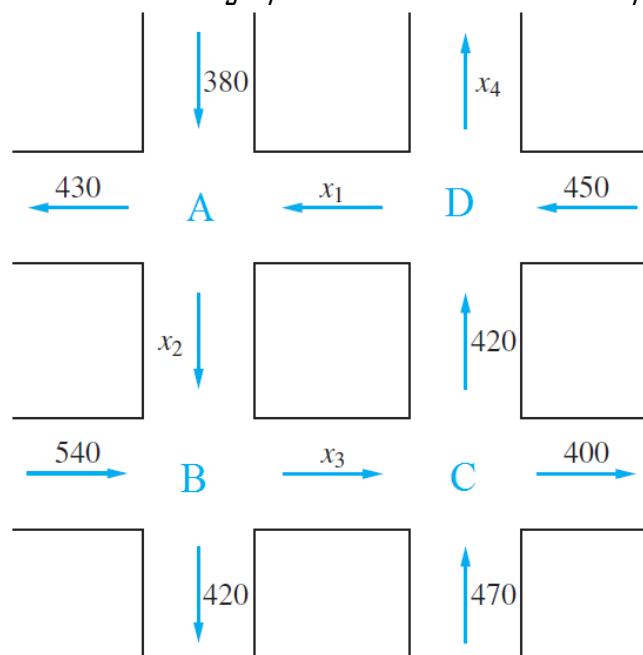
```
>>> runfile('C:/Users/User/introducao_a_Programacao_de_C/elder_Guerreiro/Universidad/Meus_Programas')
[ 100.  350.  350.]
>>>
```

Figura 181: Resultado questão 1G

Fonte: Autoria própria

Questão 3 - SEL(3): Fluxo de Tráfego

No centro de certa cidade, dois conjuntos de ruas de sentido único cruzam como se mostra na figura abaixo. O volume médio de tráfego por hora que entra e deixa essa parte da cidade durante a hora do rush é dada no diagrama. Escreva um programa que determine o fluxo de tráfego que circula entre cada um dos quatro cruzamentos A, B, C e D.



Observe que, em cada interseção A, B, C e D, o número de veículos que entra deve ser o mesmo que o número que sai. Por exemplo, na interseção A, o número de veículos que entra é $x_1 + 380$ e o número de veículos que saem é $x_2 + 430$. Assim:

$$x_1 + 380 = x_2 + 430 \text{ (interseção A)}$$

Similarmente:

$$x_2 + 540 = x_3 + 420 \text{ (interseção B)}$$

$$x_3 + 470 = 420 + 400 \text{ (interseção C)}$$

$$x_4 + x_1 = 420 + 450 \text{ (interseção D)}$$

Portanto, a solução do presente problema requer a solução do seguinte sistema de equações lineares:

$$x_1 - x_2 = 50$$

$$x_2 - x_3 = -120$$

$$x_3 = 350$$

$$x_1 + x_4 = 870$$

DICAS:

1. Utilize a biblioteca `numpy.linalg`, além da própria `numpy`.
2. Não há entradas de teste. Os parâmetros do problema devem ser informados como constantes no corpo do script.
3. A saída deverá ser um vetor de quatro elementos reais, semelhante a [10, 20, 30, 40], indicando que dez carros por hora passam no trecho do mapa indicado por x_1 , vinte carros por hora no trecho indicado por x_2 , trinta carros por hora no trecho x_3 , e quarenta carros por hora no trecho x_4 .

Primeiro passo: Interpretação da questão

Se você é um mal estudante você se desesperou com a questão, por que você viu o tamanho dela e pulou para a interpretação, mas se você é um bom estudante e leu o enunciado você viu que o enunciado da questão te deu a faca e o queijo na mão. Se você fez o que um mal estudante faria que tal repensar no seu modo de estudar? Você está aprendendo algo assim?

O enunciado da questão dá toda a interpretação da situação e te dá o sistema prontinho para você passar para o Python, veja o sistema dado no enunciado da questão:

$$\begin{cases} x_1 - x_2 = 50 \\ x_2 - x_3 = -120 \\ x_3 = 350 \\ x_1 + x_4 = 870 \end{cases}$$

Só temos que tomar cuidado com uma coisa, sempre que uma linha da matriz ou sistema não tiver todas as quantidades de variável existentes na matriz ou sistema devemos completar esses buracos, isso quer dizer que o sistema escrito como acima está errado o jeito certo é este:

$$\begin{cases} x_1 & -x_2 & 0 & 0 & = 50 \\ 0 & x_2 & -x_3 & 0 & = -120 \\ 0 & 0 & x_3 & 0 & = 350 \\ x_1 & 0 & 0 & x_4 & = 870 \end{cases}$$

Aqui também temos que ter duas matrizes claro, uma para os coeficientes dessas variáveis e outra para os valores correspondentes a eles, não esqueça que a matriz dos valores deve ser uma matriz coluna, isso quer dizer: use a matriz transposta. Acho que você já sabe o final né, use a função `solve()` e tudo estará tranquilo.

The screenshot shows the Spyder Python IDE interface. The title bar reads "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, and Run. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Introdução a Programação" and contains the following Python code:

```
1 # -*- coding: utf-8 -*-
2 """
3 27/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10 from numpy.linalg import *
11
12 # Matriz do sistema linear
13 coeficientes = array([[1, -1, 0, 0], [0, 1, -1, 0], [0, 0, 1, 0], [1, 0, 0, 1]])
14
15 # Matriz linha
16 valores = array([50, -120, 350, 870])
17 # Matriz coluna
18 valores = valores.T
19
20 # Resolução do sistema de equações lineares
21 solução = solve(coeficientes, valores)
22
23 print(solução)
```

Figura 184: Questão 3G Primeiro Passo

Fonte: Autoria própria

The screenshot shows the Spyder Python 1 console window. The title bar says "Console" and "Python 1". The console output shows the execution of a Python script and its results:

```
>>> runfile('C:/Users/User/D
ntrodução a Programação de Co
elder Guerreiro/Universidade
/Meus Programas')
[ 280. 230. 350. 590.]
```

Figura 185: Resultado questão 3G

Fonte: Autoria própria

Questão 4 - Impressão de padrão

Escreva um programa que crie matrizes quadradas $N \times N$, sendo N lido a partir do teclado, de modo a formar padrão semelhante ao seguinte:

1	1	1	1	1
1	2	2	2	2
1	2	3	3	3
1	2	3	4	4
1	2	3	4	5

DICAS:

1. Desenhe uma matriz 5×5 e explice numericamente cada um dos índices i e j dos elementos. Procure um padrão.
2. Exemplo (não exaustivo). Para a entrada 5, a saída deverá ser $[[1\ 1\ 1\ 1\ 1]\ [1\ 2\ 2\ 2\ 2]\ [1\ 2\ 3\ 3\ 3]\ [1\ 2\ 3\ 4\ 4]\ [1\ 2\ 3\ 4\ 5]]$.

Primeiro passo: Interpretação da questão

Agora vamos sair da rotina e dificultar um pouco as coisas, a criação de uma matriz nesse estilo não é difícil, mas exige de você as coisas que aprendeu lá atrás.

A entrada aqui é somente uma mesma, o tamanho da matriz (chame de N se quiser). Você se lembra que podemos usar variáveis contadoras? E vetores contadores também né? Pois é, matrizes também entram na jogada, podemos usar matrizes como matrizes contadoras para nossos scripts, assim como a variável contadora e o vetor contador a matriz deve ser zerada para começar a contar, use o comando `dtype=int` aqui também para que todos os números da matriz sejam inteiros. Você se lembra que podemos passear dentro de vetores? Então se podemos passear dentro dos vetores também podemos passear dentro das matrizes, usando o `for` passemos tranquilamente dentro dos vetores, já para as matrizes nós necessitamos de dois `for's` para passear dentro de uma matriz.

Por que precisamos de dois `for's`? Por que precisamos passear pelas linhas e pelas colunas, cada `for` fica responsável por uma, o primeiro `for` fica com a variável “ i ” representando seu passeio pela linha e o segundo `for` fica dentro do primeiro usando a variável “ j ” representando seu passeio pela coluna, os dois `for's` em conjunto um dentro do outro fazem com que possamos passear dentro de toda a matriz, para passear dentro da matriz claramente devemos usar a função `range` já que não temos nenhum vetor aqui. Você se lembra para que serve a `range`? Ela impõe limite à execução do `for`, então neste caso o `for` deverá executar até que a quantidade N seja atingida, em ambos os `for's`.

A execução do meu script que você deseja atuar na matriz deve ficar no segundo `for`, por que o primeiro tem o dever somente de acionar a linha desejada e o segundo irá acionar a coluna desejada, ou seja, só depois do segundo `for` é que a coordenada estará completa.

Para fazer a matriz desejada precisamos selecionar o valor nos determinados índices toda vez que os `for's` forem repetidos, ou seja, temos que usar o comando “`matriz[i,j]`” para selecionar o valor. E agora? O que queremos na verdade?

Devemos criar uma matriz cuja numeração deve ser de acordo com sua linha e coluna. Para isso vou lhe ensinar uma técnica e guarde-a bem para um futuro próximo:

Quando selecionamos qualquer valor desde o primeiro até o último, sempre o valor dos índices selecionados será o número zero por que a matriz contadora é uma matriz nula. Se somarmos mais 1 a cada valor da matriz vamos ter uma matriz só de números 1, e se colocarmos qualquer outro valor para somar também nós teremos um valor constante na matriz e não conseguiremos fazer a numeração variar de acordo com sua linha e coluna.

Se igualarmos o valor da matriz com o valor da sua linha ou coluna também estará errado, pois as numerações dos valores da matriz dependem tanto da linha quanto da coluna, ou seja, temos que dá um jeito de relacionar os dois índices.

Vamos analisar: Não importa se o outro valor tiver um número maior, o menor sempre prevalece, veja...

Se tivermos linha 1 e coluna 2, o valor deve ser 1;

Se tivermos linha 5 e coluna 2, o valor deve ser 2.

Isso por que o valor deve depender da sua linha e coluna, veja de novo a tabela:

1	1	1	1	1
1	2	2	2	2
1	2	3	3	3
1	2	3	4	4
1	2	3	4	5

Veja como seria a visualização dos seus índices:

[1,1]	[1,2]	[1,3]	[1,4]	[1,5]
[2,1]	[2,2]	[2,3]	[2,4]	[2,5]
[3,1]	[3,2]	[3,3]	[3,4]	[3,5]
[4,1]	[4,2]	[4,3]	[4,4]	[4,5]
[5,1]	[5,2]	[5,3]	[5,4]	[5,5]

Analisando a tabela o menor valor de cada coordenado é o valor que fica na matriz, ou seja, basta pegar as coordenadas e tirar o menor valor dela. Só que como fazemos isso?

É isso é uma pegadinha para o pessoal que só vive se esquecendo do que estuda, você se lembra da função `min`?

É meu caro leitor, faz tempo que não ouvimos falar dessa função e agora temos que usá-la, pois ela é a chave do nosso problema. Relembrando, a função `min()` permite que se coloque números ou variáveis dentro dela separados por vírgulas, quantos você quiser, e de todos esses valores a função `min()` irá ter o valor do menor deles. Seu irmão é função `max`.

Mas tome muito cuidado com uma coisa, você se lembra que as matrizes do Python também contam o zero como se fosse um número? Isso quer dizer que aqueles índices que aparecem na tabela acima são os índices de uma matriz como a conhecemos, mas os índices começam a contagem a partir do zero isso quer dizer que se usarmos somente a função `min` os valores da matriz irão começar do número zero, mas como devemos começar a partir do número 1 basta somarmos a função `min()` com 1 e acabou-se. No final de tudo é só imprimir a matriz contadora.

The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, and Tools. Below the menu is a toolbar with various icons. The editor tab is titled 'Lab07_04.py' and contains the following Python code:

```
1 # -*- coding: utf-8 -*-
2 """
3 27/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Entradas
12 N = int(input("Dimensao da matriz quadrada: "))
13
14 # Cria vetor de zeros, no formato inteiro
15 mat = zeros((N,N), dtype=int)
16
17 for i in range(0,N):
18     for j in range(0,N):
19         mat[i,j] = min(i,j)+1
20
21 # Imprime vetor
22 print(mat)
```

Figura 186: Questão 4G Primeiro Passo

Fonte: ICOMP, UFAM

The screenshot shows the Python 1 console window. The input 'Dimensao da matriz quadrada: 20' is followed by the output of a 20x20 matrix where each element is the minimum of its row and column indices plus one. The matrix is printed as a list of lists.

```
Dimensao da matriz quadrada: 20
[[ 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1]
 [ 1  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2]
 [ 1  2  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3]
 [ 1  2  3  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4]
 [ 1  2  3  4  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5]
 [ 1  2  3  4  5  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6]
 [ 1  2  3  4  5  6  7  7  7  7  7  7  7  7  7  7  7  7  7  7]
 [ 1  2  3  4  5  6  7  8  8  8  8  8  8  8  8  8  8  8  8  8]
 [ 1  2  3  4  5  6  7  8  9  9  9  9  9  9  9  9  9  9  9  9]
 [ 1  2  3  4  5  6  7  8  9  10 10 10 10 10 10 10 10 10 10 10]
 [ 1  2  3  4  5  6  7  8  9  10 11 11 11 11 11 11 11 11 11 11]
 [ 1  2  3  4  5  6  7  8  9  10 11 12 12 12 12 12 12 12 12 12]
 [ 1  2  3  4  5  6  7  8  9  10 11 12 13 13 13 13 13 13 13 13]
 [ 1  2  3  4  5  6  7  8  9  10 11 12 13 14 14 14 14 14 14 14]
 [ 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 15 15 15 15 15]
 [ 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 16 16 16 16]
 [ 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 17 17 17]
 [ 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 18 18]
 [ 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 19]
 [ 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20]]
```

Figura 187: Resultado questão 4G

Fonte: Autoria própria

Questão 5 – Horas de trabalho em uma empresa (I)

Suponha que as horas de trabalho semanais dos funcionários de uma empresa são armazenadas em uma tabela semelhante à do exemplo abaixo. Cada linha registra o número de horas trabalhadas por um funcionário em sete colunas, uma para cada dia da semana, de domingo a sábado. Por exemplo, a tabela a seguir armazena as horas de trabalho para quatro funcionários.

	DOM	SEG	TER	QUA	QUI	SEX	SAB
Funcionário 0	2	4	3	4	5	8	8
Funcionário 1	7	3	4	3	3	4	4
Funcionário 2	3	3	4	3	3	2	2
Funcionário 3	9	3	4	7	3	4	1

Escreva um script Python que leia uma matriz $N \times 7$, sendo $N > 1$, e determine quantas horas cada um dos funcionários trabalhou durante a semana. A resposta deve ser dada em um vetor de N elementos.

DICAS:

1. O atributo `shape` da biblioteca numpy, quando aplicado a uma matriz, retorna um vetor contendo o número de linhas na primeira posição e o número de colunas na segunda.
2. O operador `:` (dois pontos) toma todos os elementos de uma linha ou coluna.
3. Para a tabela do exemplo acima, temos os seguintes exemplos de entrada e saída:
 - a. Entrada: `[[2, 4, 3, 4, 5, 8, 8], [7, 3, 4, 3, 3, 4, 4], [3, 3, 4, 3, 3, 2, 2], [9, 3, 4, 7, 3, 4, 1]]`
 - b. Saída: `[34 28 20 31]`

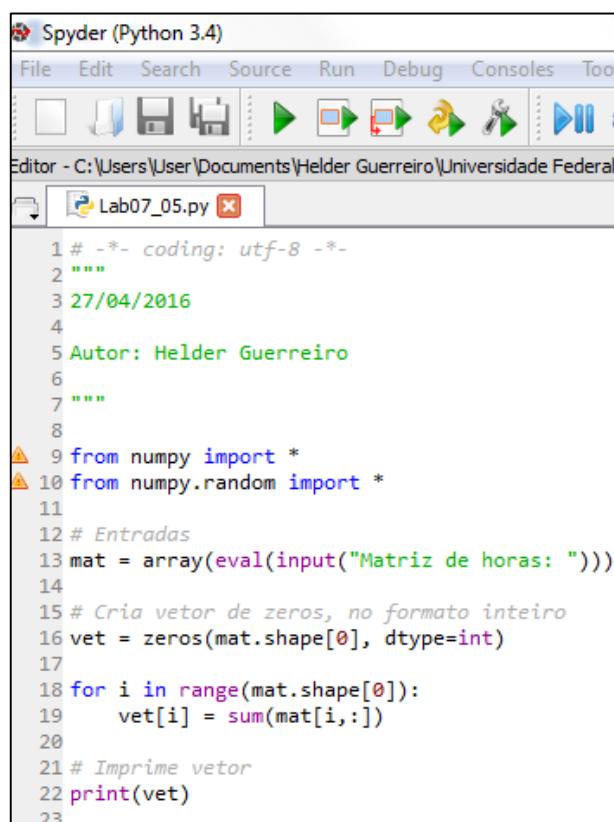
Primeiro passo: Interpretação da questão

Essa questão é mais fácil que a anterior, perceba que só temos que fazer a soma de todas as horas de cada funcionário na semana, para isso não precisamos correr as colunas, pois todas as horas de cada funcionário podem ser pegas nas linhas.

Dessa vez teremos que dar entrada numa matriz, não esqueça de usar as funções `array()`, `eval` e `input`. O resultado deverá sair num vetor, esse vetor é um vetor acumulador que irá guardar a soma das horas de cada funcionário, mas esse vetor deve ter o mesmo tamanho que as linhas da matriz inserida, para isso vamos usar o comando `"matriz.shape[0]"`, usamos o número zero para representar linha e um para representar coluna, ao usar a função `zeros()` para criar um vetor nulo use o comando da dimensão da linha da matriz selecionada para informar quantos zeros serão criados, não esqueça de usar a função `dtype=int`.

Para passear dentro de uma matriz precisamos de dois for's certo? Então, neste caso precisaremos somente de um for, por que vamos passear somente pelas linhas e não pelas colunas. Para usar o for precisamos determinar o intervalo do range, claramente devemos fazer o for percorrer até o fim das linhas da matriz, ou seja, o range deve ter o mesmo intervalo que o tamanho da linha da matriz, por isso usaremos o comando `"matriz.shape[0]"` de novo para estabelecer a quantidade de vezes necessária para o for ser executado.

No nosso script do for temos que fazer exatamente o que a questão está nos pedindo, para isso devemos selecionar os valores do vetor nulo (usando o comando "vetor[i]"), por que iremos trocar esses zeros pela soma de horas. Devemos igualar cada valor do vetor nulo a soma de todas as horas numa determinada linha. Você se lembra como fazermos para somar vários valores? Outra função antiga que não usamos faz um tempo é a função `sum`, essa função soma todos os valores que estiverem dentro dela, mas dentro o que colocaremos? Claro devemos selecionar toda a linha, por que é na linha que estão as horas trabalhadas na semana, para selecionar toda a linha basta usar o que aprendemos: "matriz[i, :]". com esse comando nós selecionamos toda a linha a partir do índice "i", esse índice com certeza será o primeiro índice da linha por que não estamos percorrendo colunas, somente linhas. Com isso basta imprimir o vetor acumulador e tchau e benção,



```

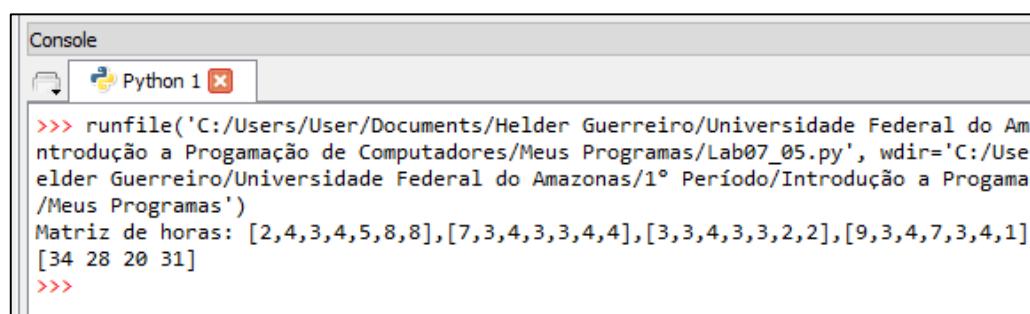
Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal
Lab07_05.py

1 # -*- coding: utf-8 -*-
2 """
3 27/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10 from numpy.random import *
11
12 # Entradas
13 mat = array(eval(input("Matriz de horas: ")))
14
15 # Cria vetor de zeros, no formato inteiro
16 vet = zeros(mat.shape[0], dtype=int)
17
18 for i in range(mat.shape[0]):
19     vet[i] = sum(mat[i,:])
20
21 # Imprime vetor
22 print(vet)
23

```

Figura 188: Questão 5G Primeiro Passo

Fonte: ICOMP, UFAM



```

Console
Python 1

>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/1º Período/Introdução a Programação de Computadores/Meus Programas/Lab07_05.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/1º Período/Introdução a Programação de Computadores/Meus Programas')
Matriz de horas: [2,4,3,4,5,8,8],[7,3,4,3,3,4,4],[3,3,4,3,3,2,2],[9,3,4,7,3,4,1]
[34 28 20 31]
>>>

```

Figura 189: Resultado questão 5G

Fonte: Autoria própria

Questão 6 – Horas de trabalho em uma empresa (2)

Suponha que as horas de trabalho semanais dos funcionários de uma empresa são armazenadas em uma tabela semelhante à do exemplo abaixo. Cada linha registra o número de horas trabalhadas por um funcionário em sete colunas, uma para cada dia da semana, de domingo a sábado. Por exemplo, a tabela a seguir armazena as horas de trabalho para quatro funcionários.

	DOM	SEG	TER	QUA	QUI	SEX	SAB
Funcionário 0	2	4	3	4	5	8	8
Funcionário 1	7	3	4	3	3	4	4
Funcionário 2	3	3	4	3	3	2	2
Funcionário 3	9	3	4	7	3	4	1

Escreva um script Python que leia uma matriz Nx7, sendo $N > 1$, e determine qual o dia da semana em que os funcionários mais trabalham. A saída do programa deverá ser um número inteiro de 1 a 7, indicando respectivamente domingo, segunda, ..., sábado.

DICAS:

1. Se houver coincidência de dois ou mais dias serem igualmente as demais horas trabalhadas, os números correspondentes a todos eles devem ser impressos.
2. Para a tabela do exemplo acima, temos os seguintes exemplos de entrada e saída:
 - a. Entrada: `[[2, 4, 3, 4, 5, 8, 8], [7, 3, 4, 3, 3, 4, 4], [3, 3, 4, 3, 3, 2, 2], [9, 3, 4, 7, 3, 4, 1]]`
 - b. Saída: 1, indicando que o domingo é o dia em que mais se trabalha nessa empresa.
 - c. Entrada: `[[2, 2, 3, 4, 5, 8, 8], [7, 7, 4, 3, 3, 4, 4], [3, 3, 4, 3, 3, 2, 2], [9, 9, 4, 7, 3, 4, 1]]`
 - d. Saída: 1 2, indicando que o domingo e a segunda são os dias em que mais se trabalha nessa empresa.

Primeiro passo: Interpretação da questão

Nada de especial, aqui temos uma questão idêntica a anterior com algumas modificações. Além de soma das horas precisamos determinar qual dia da semana é o mais trabalhado pelos funcionários. Tudo aquilo que fizemos antes ainda permanece. Por que ainda precisamos da soma das horas se não vamos imprimir? Precisamos sim por que é pelo acúmulo das horas que vemos qual foi o dia mais trabalhado, né?

Só que dessa vez devemos mudar os limites do vetor acumulador e do `for`, pois dá outra vez os limites eram de acordo com o tamanho da linha da matriz inserida, só que dessa vez estamos usando a semana como referência e dessa forma o tamanho do vetor e da análise do `for` deve ser obrigatoriamente 7, o vetor terá 7 índices cada um indicando um dia da semana. Outra coisa é muito importante analisar, na questão anterior selecionamos todas as linhas uma por uma, para que através da seleção aja a soma das horas para cada funcionário, se as linhas representam cada uma um funcionário então as colunas representam cada uma um dia da semana, por isso temos que mudar o comando “`matriz[i, j]`” que está dentro da função `soma`, pois na questão anterior selecionamos a variável “`i`” no lado esquerdo do comando, esse é o lado

das linhas, o que queremos na verdade é selecionar toda a coluna e através dessa seleção aja a soma de todas as horas trabalhadas no determinado dia, então o comando fica assim: "matriz[:, i]"

Como estamos com metade do caminho em nossas mãos, já temos o acúmulo da soma das horas de cada dia da semana. Ao final de tudo o vetor estará pronto com 7 posições, cada posição representando um dia da semana. Se temos que indicar qual o dia mais trabalhado da semana então temos que analisar o vetor acumulador para encontrar o maior valor somado.

Para analisar um vetor é necessário primeiro percorre-lo, ou seja, vamos usar o for de novo aqui para passear dentro do vetor contador. Não esqueça que estamos trabalhando com um vetor de 7 índices, o `range` deve ter o tamanho de 7. Agora o que queremos é encontrar o maior valor dentro do vetor, claro que você deve se lembrar da função `max`, mas como fazer? Bom a solução pode ser simplesmente pegar uma variável e igualá-la ao valor máximo do vetor e pronto imprimir, mas e se acontecer de dois valores máximos iguais? A questão pede que se houver números iguais ambos devem ser impressos, e agora? Para isso devemos usar condições: SE o valor analisado for igual ao valor máximo do vetor então devemos imprimir o dia da semana, usando o if nós testamos a condição em que o valor num certo índice "i" seja igual ou não ao valor máximo do vetor, caso for respeitado devemos imprimir o dia da semana que corresponde ao índice do vetor, MAS lembre-se que o vetor conta o zero, então esse índice deve ser somado a 1 para representar o dia da semana.

```

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools
Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Introdução a Programação de Computadores\Meus Programas\Lab07_06.py
Lab07_06.py
1 # -*- coding: utf-8 -*-
2 """
3 27/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10 from numpy.random import *
11
12 # Entradas
13 mat = array(eval(input("Matriz de horas: ")))
14
15 # Cria vetor de zeros, no formato inteiro
16 vet = zeros(7, dtype=int)
17
18 for i in range(7):
19     vet[i] = sum(mat[:,i])
20
21 # Imprime dia da semana em que mais se trabalha
22 for i in range(7):
23     if vet[i] == max(vet):
24         print(i + 1)
25

```

Figura 190: Questão 6G Primeiro Passo

Fonte: ICOMP, UFAM

```

Console
Python 1
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/1º Período/Introdução a Programação de Computadores/Meus Programas/Lab07_06.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/1º Período/Introdução a Programação de Computadores/Meus Programas')
Matriz de horas: [2, 2, 3, 4, 5, 8, 8], [7, 7, 4, 3, 3, 4, 4], [3, 3, 4, 3, 3, 2, 2], [9, 9, 4, 7, 3, 4, 1]
1
2
>>>

```

Figura 191: Resultado questão 6G

Fonte: Autoria própria

Se você for uma pessoa caprichosa (sem piada de boi), gostará dessa outra forma de resolver a questão, só que o enunciado da questão não pede isso.

The screenshot shows the Spyder Python 3.4 IDE interface. The title bar says "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools. Below the menu is a toolbar with various icons. The main area is titled "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Introdução a Progamação de Computadores\Meus Programas\Lab07_06.py". The code editor contains the following Python script:

```
1 # -*- coding: utf-8 -*-
2 """
3 27/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10 from numpy.random import *
11
12 # Entradas
13 mat = array(eval(input("Matriz de horas: ")))
14
15 # Cria vetor de zeros, no formato inteiro
16 vet = zeros(7, dtype=int)
17
18 for i in range(7):
19     vet[i] = sum(mat[:,i])
20
21 # Imprime dia da semana em que mais se trabalha
22 for i in range(7):
23     if vet[i] == max(vet):
24         if (i + 1) == 1:
25             print("Domingo")
26         elif (i + 1) == 2:
27             print("Segunda")
28         elif (i + 1) == 3:
29             print("Terça")
30         elif (i + 1) == 4:
31             print("Quarta")
32         elif (i + 1) == 5:
33             print("Quinta")
34         elif (i + 1) == 6:
35             print("Sexta")
36         else:
37             print("Sábado")
```

Figura 192: Questão 66b Primeiro Passo

Fonte: Autoria própria

The screenshot shows the Python 1 console window. The title bar says "Console" and "Python 1". The command prompt shows the execution of the script and its output:

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/1º Período/Introdução a Progamação de Computadores/Meus Programas/Lab07_06.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Universidade Federal do Amazonas/1º Período/Introdução a Progamação de Computadores/Meus Programas')
Matriz de horas: [2, 2, 3, 4, 5, 8, 8], [7, 7, 4, 3, 3, 4, 4], [3, 3, 4, 3, 3, 2, 2], [9, 9, 4, 7, 3, 4, 1]
Domingo
Segunda
>>>
```

Figura 193: Resultado questão 66b

Fonte: Autoria própria

Questão 7 – Distância entre duas cidades

O tempo que um determinado avião dispensa para percorrer o trecho entre duas cidades distintas está disponível através da seguinte tabela:

	111	222	333	444	555	666	777
111	0	2	11	6	15	11	1
222	2	0	7	12	4	2	15
333	11	7	0	11	8	3	13
444	6	12	11	0	10	2	1
555	15	4	8	10	0	5	13
666	11	2	3	2	5	0	14
777	1	15	13	1	13	14	0

Por exemplo, o tempo de viagem entre a cidade 222 e a cidade 444 é de 12 horas. Escreva um programa que, tendo a tabela acima armazenada como uma matriz, informe ao usuário o tempo necessário para percorrer duas cidades por ele informadas.

DICAS:

1. Considere que o usuário sempre insere valores válidos.
2. Converta o número da cidade em um índice válido da matriz antes de consultar o valor de seus elementos.
3. Eis alguns exemplos (não exaustivos) de entradas e saídas:
 - a. Para as entradas 222 e 444, a saída deverá ser 12.
 - b. Para as entradas 555 e 555, a saída deverá ser 0.

Primeiro passo: Interpretação da questão

Nesta questão vamos aprender a selecionar os valores da matriz de acordo com as coordenadas. Mas já não sabemos fazer isso? Sim, só que dessa vez é diferente, temos umas modificações a mais.

Neste caso o usuário irá inserir as cidades desejadas através das numerações mostradas nas identificações das linhas e colunas acima, ou seja, a primeira cidade é 111, a segunda 222 e assim por diante até a sétima com 777.

O que tem de se fazer nessa questão é que quando o usuário inserir as cidades desejadas o Python responda com suas respectivas coordenadas, por exemplo, escolhemos as cidades 555 e 777, olhando na tabela acima o resultado a ser mostrado pelo Python deve ser o número 13. Com isso você já deve ter percebido que precisamos copiar toda essa tabela em forma de matriz para dentro do Python, só que sem essa numeração de identificação das linhas e colunas né.

Ao inserir o número da cidade precisamos converter esse número na respectiva coordenada da cidade, mas como fazer isso? Vamos analisar com calma.

A primeira cidade é 111, mas qual o seu índice? Ela pertence a primeira linha e primeira coluna, seu índice é [0,0]. Sabemos que não podemos zerar logo de uma vez o número 111, pois se multiplicar ou dividir os números por 0 vai dar tudo zero, não?

Dividindo o 111 por ele mesmo temos $111/111 = 1$, já é um grande passo para chegar no zero é só subtrair por 1. E com os outros, será que funciona?

Com 222, se dividirmos por ele mesmo temos o número 1, ta certo que esse é o índice dele mesmo, mas se estamos tentando criar uma forma geral para qualquer número ao subtrair por 1 chegamos no zero, então dividir por ele mesmo não funciona. Mas podemos usar o 111 como referência, pois ele é o primeiro e o primeiro deve-se sempre ser usado como referência, ao dividir $222/111 = 2$ e ao subtrair por 1 temos o número 1 como resultado, agora sim chegamos aonde queremos.

Esses números de três dígitos ao serem divididos por outro de três dígitos se tornam a mesma coisa como se fossem um simples número, por que $1/1 = 1$ e $2/1 = 1$.

Bom já encontramos a solução, faça uma pequena formula que divida o número inserido por 111 e depois subtraia por 1, isso funcionará com qualquer número de referência das linhas e colunas, pois $555/111 = 5$ e $5 - 1 = 4$, que realmente é o índice dele. Essa formula vai calcular somente o índice, use o comando "matriz[i, j]" e insira esses índices dentro dele em seus respectivos lugares (i = linha, j = coluna).

Perceba que essa questão é mais uma "frescurite" do que realmente aprendizado, se tivéssemos usado como referência das linhas e colunas os números de 0 a 6, não precisaríamos usar formula nenhuma somente jogar dentro do comando "matriz[i, j]" e pronto.

The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, and View. Below the menu is a toolbar with various icons. The main area shows a code editor with the file 'Lab07_07.py'. The code is as follows:

```
1 # -*- coding: utf-8 -*-
2 """
3 27/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 tab = array([
12 [0, 2, 11, 6, 15, 11, 1],
13 [2, 0, 7, 12, 4, 2, 15],
14 [11, 7, 0, 11, 8, 3, 13],
15 [6, 12, 11, 0, 10, 2, 1],
16 [15, 4, 8, 10, 0, 5, 13],
17 [11, 2, 3, 2, 5, 0, 14],
18 [1, 15, 13, 1, 13, 14, 0]
19 ])
20
21 # Número das cidades
22 C1 = int(input("Digite o numero da 1a. cidade: "))
23 C2 = int(input("Digite o numero da 2a. cidade: "))
24
25 i = C1/111 - 1
26 j = C2/111 - 1
27
28 x = tab[i,j]
29
30 print(x)
31
```

Figura 194: Questão 76 Primeiro Passo

Fonte: Autoria própria

The screenshot shows the Python 1 console window. The command `>>> runfile('C:/Users/User/Documents/introdução a Programação de Computador/elder Guerreiro/Universidade Federal/Meus Programas')` was run. The console then prompts for two city numbers: "Digite o numero da 1a. cidade: 555" and "Digite o numero da 2a. cidade: 777". The output shows the result of the calculation: "13".

```
>>> runfile('C:/Users/User/Documents/introdução a Programação de Computador/elder Guerreiro/Universidade Federal/Meus Programas')
Digite o numero da 1a. cidade: 555
Digite o numero da 2a. cidade: 777
13
>>>
```

Figura 195: Resultado questão 76

Fonte: Autoria própria

8. Fechando Com Chave de Ouro - Strings

Estamos chegando na hora final, esse último assunto pode não parecer difícil e mamão com açúcar, bom em si esse assunto é simples, mas ele exige de você tudo que foi aprendido nos tópicos anteriores, então se você não aprendeu os assuntos que eu lhe ensinei, você só vai bater cabeça aqui.

String quer dizer corda em inglês, esse é o assunto que envolve o trabalho com caracteres, ou seja, iremos trabalhar com textos, o Python pode fazer vários trabalhos envolvendo caracteres e muito mais, inclusive criptografia.

Para identificar uma string dentro do script do Python é simplesmente usar aspas (simples ou dupla), quando você pega uma variável e iguala ela a um texto com aspas está criando uma string.

```
texto = "Odeio o Python "
```

```
texto = ' Odeio o Python '
```

Figura 196: Exemplo de String

Fonte: ICOMP, UFAM. Editado pelo Autor

Podemos colocar as próprias aspas dentro do texto usando um simples comando: \"

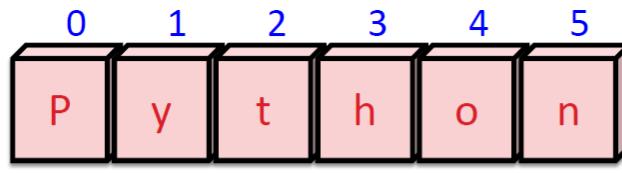
```
>> texto2 = "Odeio o \" Python \" . "
>> texto2
Odeio o "Python" .
```

Figura 197: Aspas no Texto

Fonte: ICOMP, UFAM. Editado pelo Autor

8.1 Um Comportamento Conhecido

O comportamento das strings é semelhante ao comportamento de vetores, com o nome da variável da string e usando índices podemos selecionar os caracteres da string, veja abaixo:

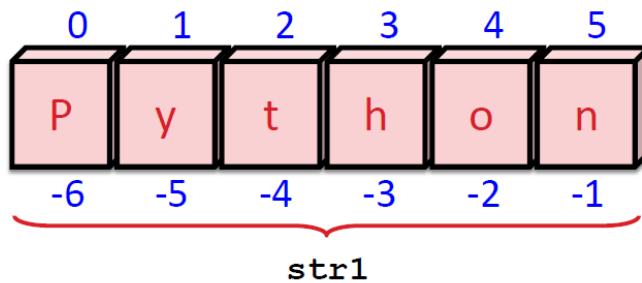


Não usar `str` como identificador, pois é uma classe nativa

```
str1 = 'Python'
str1[0]    # primeiro caractere: 'P'
str1[5]    # ultimo caractere: 'n'
str1[6]    # ERRO
```

Figura 198: Comportamento Vetorial a

Fonte: ICOMP, UFAM.

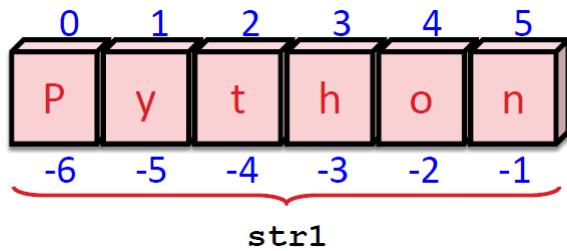


```
str1[-1]      # ultimo caractere: 'n'
str1[-2]      # penultimo caractere: 'o'
str1[-6]      # primeiro caractere : 'P'
str1[-7]      # ERRO
```

Figura 199: Comportamento Vetorial b

Fonte: ICOMP, UFAM.

Assim como fazemos nas matrizes podemos usar dois índices nas strings, dessa vez o primeiro índice significa que o caractere no índice informado está selecionado, o segundo índice significa que o texto será selecionado do primeiro índice até o caractere do segundo índice. E os comando que usamos para selecionar toda uma linha ou toda uma coluna ainda entra aqui.



```
# carac. da posição 0 a 1: 'Py'
str1[0:2]
# carac. da posição inicial até a 1:'Py'
str1[:2]
# carac. da posição 2 a 5: 'thon'
str1[2:6]
# carac. da posição 2 até o final: 'thon'
str1[2:]
```

Figura 200: Seleção de Texto

Fonte: ICOMP, UFAM.

8.2 Trabalhando com Strings

Vamos conhecer algumas funções e comandos para nos ajudar a trabalhar com strings:

- Quantidade de caracteres da string

```
>> len(string)
```

- Primeiro caractere da string

```
>> string[0]
```

- Último caractere da string

```
>> string[-1]
```

Figura 201: Tamanho, primeira e último,

Fonte: ICOMP, UFAM

Duas funções que são interessantes, "string.upper()" faz seu texto ficar todo em maiúsculo, caso queira deixar somente uma parte maiúscula use o comando "string[i].upper()", outra função que é oposta dessa é a "string.lower()" deixa seu texto todo minúsculo.

```
str1 = 'Isso é um teste!'  
  
str1.upper()      # ISSO É UM TESTE!  
str1.lower()      # isso é um teste!
```

Figura 202: Maiúsculo e Minúsculo.

Fonte: ICOMP, UFAM

Podemos fazer algumas operações com as strings como se fossem números, as operações são de soma e multiplicação.

```
primeira = 'Auto'  
ultima = 'escola'  
  
primeira + ultima          # 'Autoescola'  
primeira + ' ' + ultima    # 'Auto escola'
```

Figura 203: Somando Strings.

Fonte: ICOMP, UFAM

```
str1 = 'UFAM'  
  
str1 * 3                  # 'UFAMUFAMUFAM'  
(str1 + ' ') * 3         # 'UFAM UFAM UFAM '  
  
3 * str1                  # 'UFAMUFAMUFAM'  
3 * (str1 + ' ')          # 'UFAM UFAM UFAM '
```

Figura 204: Multiplicando Strings.

Fonte: ICOMP, UFAM

Uma outra função que devo apresentar a você é a função "string.split()" essa função só tem o deve de separar uma string em pedaços de acordo com a configuração da função.

```
str1 = 'Fulano de Tal'  
str1.split()           # ['Fulano', 'de', 'Tal']  
str1.split(' ')        # ['Fulano', 'de', 'Tal']  
str1.split(',')         # ['Fulano de Tal']  
'1,2,3'.split(',')    # ['1', '2', '3']
```

Figura 205: Função Split.

Fonte: ICOMP, UFAM

Podemos usar uma função que pode buscar uma string dentro de outra string, é como se estivéssemos buscando uma letra dentro de um nome, essa é a função "string.find(string2)":

```
str1 = 'Fulano de Tal'  
str2 = 'de'  
str1.find(str2)          # 7
```

Figura 206: Função find.

Fonte: ICOMP, UFAM

Uma função interessante é a função "string.replace(string2, string3)", essa função troca pedaços do texto da string por outros pedaços, como representado acima a função replace irá trocar a string2 pela string3 dentro da string principal, veja o exemplo abaixo:

```
str1 = 'Na minha família há vários  
médicos. Meu avô, meu pai, minha mãe e eu  
somos médicos.'  
str2 = 'médicos'  
str3 = 'engenheiros'  
str1.replace(str2,str3)
```

Figura 207: Função replace

Fonte: ICOMP, UFAM

8.3 Testes Lógicos

Todas as funções ou comandos que emitem como resultado True (verdadeiro) ou False (falso), são funções ou comandos de teste lógico, elas servem para verificar se uma certa situação está acontecendo ou não, veja quem são e como funcionam:

Assim como procuramos pedaços de string dentro de uma string principal, podemos afirmar que um pedaço de string está em uma string principal, cabe ao comando `in` dizer se é verdadeira ou falsa essa afirmação:

```
'de' in 'Fulano de Tal'    # True
```

Figura 208: Comando `in`

Fonte: ICOMP, UFAM

Um aviso sobre esse pessoal, não vá pensando que é só colocar a função `print` e lá no console vai sair o resultado "True" ou "False", esses comandos ou funções emitem esses resultados mas não com o objetivo de imprimi-los, mas sim com o objetivo de condicional, é como se fosse o `if`, o if testa se a condição seja verdadeira, não para imprimir o resultado "Sim" ou "Não", mas para se seguir com seu mine script, só que essas funções não têm mine script.

8.3.1 Funções "is"

Elas recebem esse nome por que todas elas começam com "is" e elas são todas de teste lógico, veja quais são e para que servem:

- Função `isalpha()`
 - Verifica se todos os caracteres na string são letras do alfabeto.
- Função `isnumeric()`
 - Verifica se todos os caracteres na string são numéricos.
- Função `isalnum()`
 - Verifica se todos os caracteres na string são alfanuméricos.

Figura 209: Funções "is" a

Fonte: ICOMP, UFAM

Pode parecer meio estanho o que essas funções fazem, mas você entenderá melhor quando ver esses exemplos abaixo:

```
str1 = 'Fulano'  
str2 = '123'  
str3 = str1 + str2  
  
str1.isalpha()      # True  
str3.isalpha()      # False  
  
str2.isnumeric()    # True  
str3.isnumeric()    # False  
  
str1.isalnum()      # True  
str2.isalnum()      # True  
str3.isalnum()      # True
```

Figura 210: Exemplo de funções "is" a
Fonte: ICOMP, UFAM

Tem mais duas funções que quero apresentar a você, uma testa as strings minúsculas e outra testa as strings maiúsculas:

- Função **islower()**
 - Verifica se todos os caracteres na string são minúsculos.
- Função **isupper()**
 - Verifica se todos os caracteres na string são maiúsculos.

Figura 211: Funções "is" b
Fonte: ICOMP, UFAM

Agora veja como elas são aplicadas:

```

str1 = 'fulano'
str2 = 'FULANO'
str3 = 'Fulano'

str1.islower()      # True
str3.islower()      # False
str2.isupper()      # True
str3.isupper()      # False

```

Figura 212: Exemplo de funções "is" a
Fonte: ICOMP, UFAM

8.4 De Strings Para Números Ou Vice-Versa

Duas funções você está careca de saber são as funções `in` e `float`, as duas pegam a string e transformam em números, uma em números inteiros e outra em números reais, ao inserir números como entrada no Python estamos na verdade inserindo caracteres, é necessário usar essas duas funções para que o Python as reconheça como números e aí sim possamos fazer operações e tudo mais com esses números. A nova função que devo apresentar a vocês é a função `str`, essa função tem como objetivo pegar números e transformá-lo em strings, é o oposto das duas outras funções.

str()

- Converte números em strings

int()

- Converte strings em números inteiros

float()

- Converte strings em números reais

Figura 213: De Strings Para Números Ou Vice-Versa

Fonte: ICOMP, UFAM

Veja uns exemplos marotos logo abaixo:

```
str1 = '123'  
num1 = 123  
  
str(num1)                  # '123'  
int(str1)                  # 123  
float(str1)                # 123.0  
  
int(str1) + num1           # 246  
float(str1) + num1         # 246.0  
str(num1) + str1           # '123123'  
  
str1 + num1                # ERRO
```

Figura 214: Exemplo Das Funções De Strings Para Números Ou Vice-Versa

Fonte: ICOMP, UFAM

Perceba que lá no final deu um erro, é deve ser meio difícil somar um número com uma letra né? Não importa se a string é "1, 2, 3" para o Python são a mesma coisa que "a, b, c".

8.5 Questões Resolvidas

Estas questões foram desenvolvidas pela ICOMP um instituto da UFAM e os algoritmos (scripts) apresentados aqui pertencem ao ICOMP.

Questão 1 – Data por extenso

Escreva um script em Python que leia do teclado uma data no formato “ddmmaaaa” e imprima essa data por extenso (suponha valores sempre válidos).

DICA:

Exemplo (não exaustivo) com a data de fundação da UFAM:

Entrada: 17011909

Saída: 17 de janeiro de 1909

Primeiro passo: Interpretação da questão

Bom é a nossa primeira questão envolvendo strings, lembra-se que as strings se comportam como vetores? Para isso devemos trabalhar com a biblioteca numpy. Já sabemos que nossa entrada será a data, e ela será nossa string a ser trabalhada, essa entrada não terá a função `int` e nem `float`, por que se não a entrada vai deixar de ser string.

Vamos analisar com calma, dos números que colocaremos de entrada, qual será transformado e qual não será mudado? Os dias com certeza serão representados por números e o ano também, então pelo visto a mudança aqui acontecerá ao inserirmos o mês, pois o número se tornará o nome do respectivo mês.

O número do mês inserido deverá ser um indicador que corresponderá ao mês de janeiro, com isso a única forma de representar uma palavra por um número é utilizando um vetor, um vetor com o nome de todos os meses e cada índice corresponderá a um mês, os índices serão de 0 a 11.

Usando o comando “vetor[i]” podemos selecionar o índice correspondente e o comando irá retornar o nome do mês selecionado pelo índice, mas como tirar esse índice da data inserida? Lembra-se de que as strings se comportam como vetores, então a string data também terá seus índices. Então vamos analisa-los:

17011909

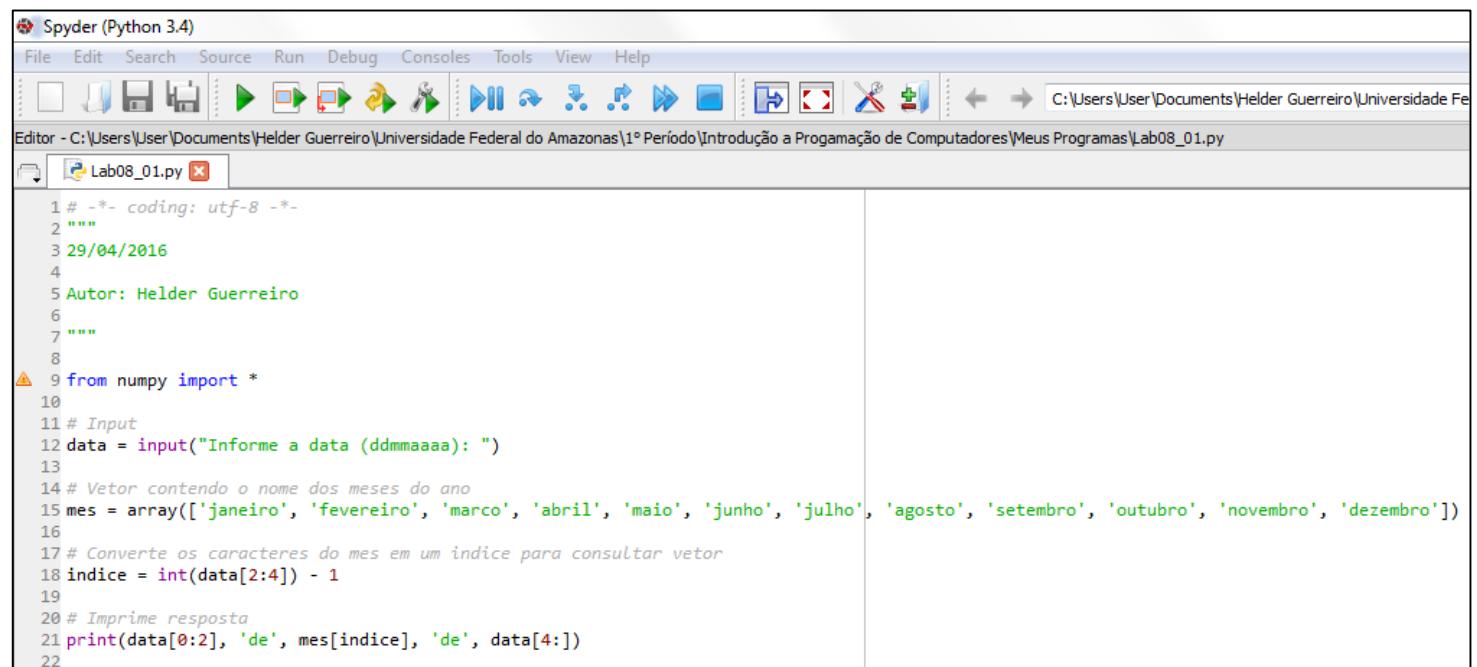
01234567

Veja, os índices 0 e 1 é o dia, 2 e 3 é o mês e do 4 em diante é o ano. Com isso usando o comando “string[i]” podemos selecionar as partes da data, a parte que iremos utilizar como índice é o mês que está nos índices 2 e 3 da string. Usando o comando “strig[i]” vamos selecionar o mês para usar esse número como índice, o mês está nos índices 2 e 3, mas lembre-se que o intervalo começa aberto e termina fechado, isso quer dizer que ao colocarmos o primeiro número do intervalo esse número estará dentro do intervalo, mas ao colocar o último número esse último não estará mais no intervalo somente o anterior. Então o intervalo que pega a numeração do mês é [2:4], lembre-se que o intervalo das strings é feito com dois pontos e não com vírgula como os vetores.

O que faremos depois de conseguir essa numeração? Bom, saiba que quando nós pedimos aquele `input`, não colocamos nenhum `int` e nem `float`, mas agora temos que tornar essa string em número. Por que?

Por que temos de subtrair esse valor por 1, veja que a data dita como janeira tem sua numeração "01", só que a primeira posição do vetor dos meses é "0" e não 1, então para fazermos operações precisamos transformar essa string em números, basta usar a função `int`.

Depois de conseguir o índice basta imprimir o resultado, já? Isso mesmo terminamos aqui, mas o `print` deve ter algumas coisinhas a mais. A primeira coisa a se colocar dentro do `print` é o dia, para imprimir o resultado vamos usar o comando "`string[i]`" para selecionar os caracteres para a função `print`, o intervalo que seleciona o dia é de `[0:2]`, por que no zero está o primeiro dígito do dia e como o 2 não entra no intervalo o 1 é o último dígito do dia (por acaso existe dia de três dígitos?). Então (sem as aspas por favor) coloque o comando "`string[i]`" com o intervalo `[0:2]` dentro da função `print`, separando por vírgula (cada termo diferente deve ser separado por vírgula no `print`) podemos colocar entre aspas a preposição "de" (ó ele sabe português), mas por que? Acho que na frase dizendo "17 de janeiro de 1909" esse "de" não apareceu num passe de mágica né? Depois do dia e da primeira preposição "de" hora de colocar o mês (que na verdade foi o único problema dessa questão), o índice do mês já sabemos qual é, coloque esse índice no comando "`vetor[i]`" do vetor dos meses e pronto vírgula de novo. E no final colocamos mais um "de" e depois o ano que será colocado da mesma forma que o dia (seu intervalo começa do índice 4 até o final `[4:]`).



The screenshot shows the Spyder Python 3.4 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. The toolbar has various icons for file operations like Open, Save, and Run. The status bar at the bottom shows the path: C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Introdução a Programação de Computadores\Meus Programas\Lab08_01.py. The code editor window contains the following Python script:

```
1 # -*- coding: utf-8 -*-
2 """
3 29/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Input
12 data = input("Informe a data (ddmmaaaa): ")
13
14 # Vetor contendo o nome dos meses do ano
15 mes = array(['janeiro', 'fevereiro', 'março', 'abril', 'maio', 'junho', 'julho', 'agosto', 'setembro', 'outubro', 'novembro', 'dezembro'])
16
17 # Converte os caracteres do mes em um indice para consultar vetor
18 indice = int(data[2:4]) - 1
19
20 # Imprime resposta
21 print(data[0:2], 'de', mes[indice], 'de', data[4:])
22
```

Figura 215: Questão IH Primeiro Passo

Fonte: ICOMP, UFAM

The screenshot shows a Python console window titled "Python 1". The code run is:

```
>>> runfile('C:/Users/User/Documents/Introdução a Programação de Computadores/elder Guerreiro/Universidade Federal /Meus Programas')
Informe a data (ddmmaaaa): 17011909
17 de janeiro de 1909
>>>
```

Figura 216: Resultado questão 1H

Fonte: Autoria própria

Questão 2 - Formato de datas

Escreva um script em Python que leia do teclado uma data por extenso ('dia'+'mês'+'ano') e imprima essa data no formato "ddmmaaaa" (suponha valores sempre válidos).

DICAS:

1. Modifique o script do problema anterior.
2. Use o método `.split(separador)`, que quebra uma string em substrings, usando o parâmetro 'separador' como critério de separação, descartando-o no final.
3. Utilize um laço de repetição para descobrir qual índice do vetor contendo os nomes dos meses corresponde ao mês inserido na entrada. Qual o laço mais apropriado - `while` ou `for` -, uma vez que você não precisa percorrer todo o vetor para encontrar o nome do mês inserido?
4. Use a estrutura `if-else` para inserir o caractere '0' em dias e meses menores que dez.

5. Exemplo (não exaustivo) com a data de fundação da UFAM:

Entrada: 17 janeiro 1909

Saída: 1701909

Primeiro passo: Interpretação da questão

Essa questão fico me perguntando por que criaram, quem é que vai querer fazer isso com a data? É mais feio e mais difícil, mas ela é boa para você ver a relação entre os assuntos que estudamos.

Vamos com calma, primeiro saiba que temos que usar a biblioteca numpy por que as strings são trabalhadas igualmente como os vetores. A nossa entrada é colocar a data sem as preposições "de", isso por que temos que usar a função `"string.split()"`, essa função como foi explicado tem por objetivo dividir a string em pedaços, e o nosso objetivo aqui é dividir esse string em três: dia, mês e ano. Por que isso? Ao usar essa função para separar a string iremos criar uma string de três posições, só que dessa vez em das posições serem por letra agora essa posição será por palavra e o limite de cada posição é dada por você ao colocar o limitador dentro da função `split()`, no nosso caso o que delimita uma palavra da outra é o espaço, então devemos colocar `"string.split(' ')"` para que onde houver espaço aja uma delimitação dizendo que para um lado é uma posição e para o outro é outra posição, assim:

```
datax = "17 janeiro 1909"
data = datax.split(' ') # ['17', 'janeiro', '1909']
```

Então quer dizer que se usarmos `data[0]` teremos: 17;

Ao usar `data[1]` teremos: janeiro;

Ao usar `data[2]` teremos: 1909.

Precisamos criar o mesmo vetor dos meses de novo (chame-o de `vet-mes`), por que ele vai servir para que o Python reconheça que o mês "tal" é a mesma coisa que o número "tal", ou seja, janeiro é um mês inserido pelo usuário, o Python deve reconhecer esse janeiro como o número 1, e a única forma de relacionarmos os nomes dos meses com números é usando índice de vetores.

Agora com a entrada, strings e o vetor dos meses prontos hora de fazer com que os nomes sejam convertidos em suas respectivas representações numéricas, mas como fazer isso? Temos que pegar o vetor do mês, e analisar índice por índice até que possamos identificar que o índice analisado tem a mesma string que a da data inserida, quando se fala em analisar você pode até pensar em condicional, ou seja, `if`, mas você acha que realmente deveríamos fazer doze condições diferentes para analisar cada mês? Por isso vou ensinar a você fazer isso com `while`, para isso precisamos de uma célula contadora, por que temos que passear por dentro do vetor dos meses, mas não podemos fazer isso com o `for`? Sim, mas senão teríamos que usar as condicionais, e usando o `while` não precisaremos dela. A condição do `while` é: ENQUANTO a string do vetor dos meses for diferente da string da data inserida, continuaremos mudando o índice até que cheguemos a igualdade, isso quer dizer que iremos usar o comando “`vetor[i]`” para selecionar o mês do vetor dos meses no índice indicado e usaremos o comando “`string[i]`” para selecionar o mês da data inserida (lembre-se que o índice `i` na string é o índice do mês), então ENQUANTO o mês no vetor dos meses for DIFERENTE do mês da data inserida, o índice continuará contando, se você não entendeu a dica você deve usar o operador condicional “`!=`” que significa “diferente de”, o mine script do `while` será contar o índice do “`i`”. O nosso objetivo aqui é somente chegar ao índice que pertence o mês inserido, pois lá na frente iremos usá-lo, pois ele vai funcionar assim:

Inserimos o mês de dezembro;

O `while` tem como condição que `vet_mes[i]` sendo diferente de `data[i]` o mine script será executado;

Começa com `i = 0`, `vet_mes[0]` é o mês de janeiro, `data[0]` é o mês de dezembro, eles são diferentes, `while` continua;

Com `i = 1`, `vet_mes[1]` é o mês de fevereiro, `data[1]` é o mês de dezembro, eles são diferentes, `while` continua;

:

Com `i = 11`, `vet_mes[11]` é o mês de dezembro, `data[11]` é o mês de dezembro, eles são iguais, `while` para.

O nosso objetivo com o `while` aqui é somente encontrar o índice correspondente ao mês inserido pelo usuário, nós iremos usar somente o índice.

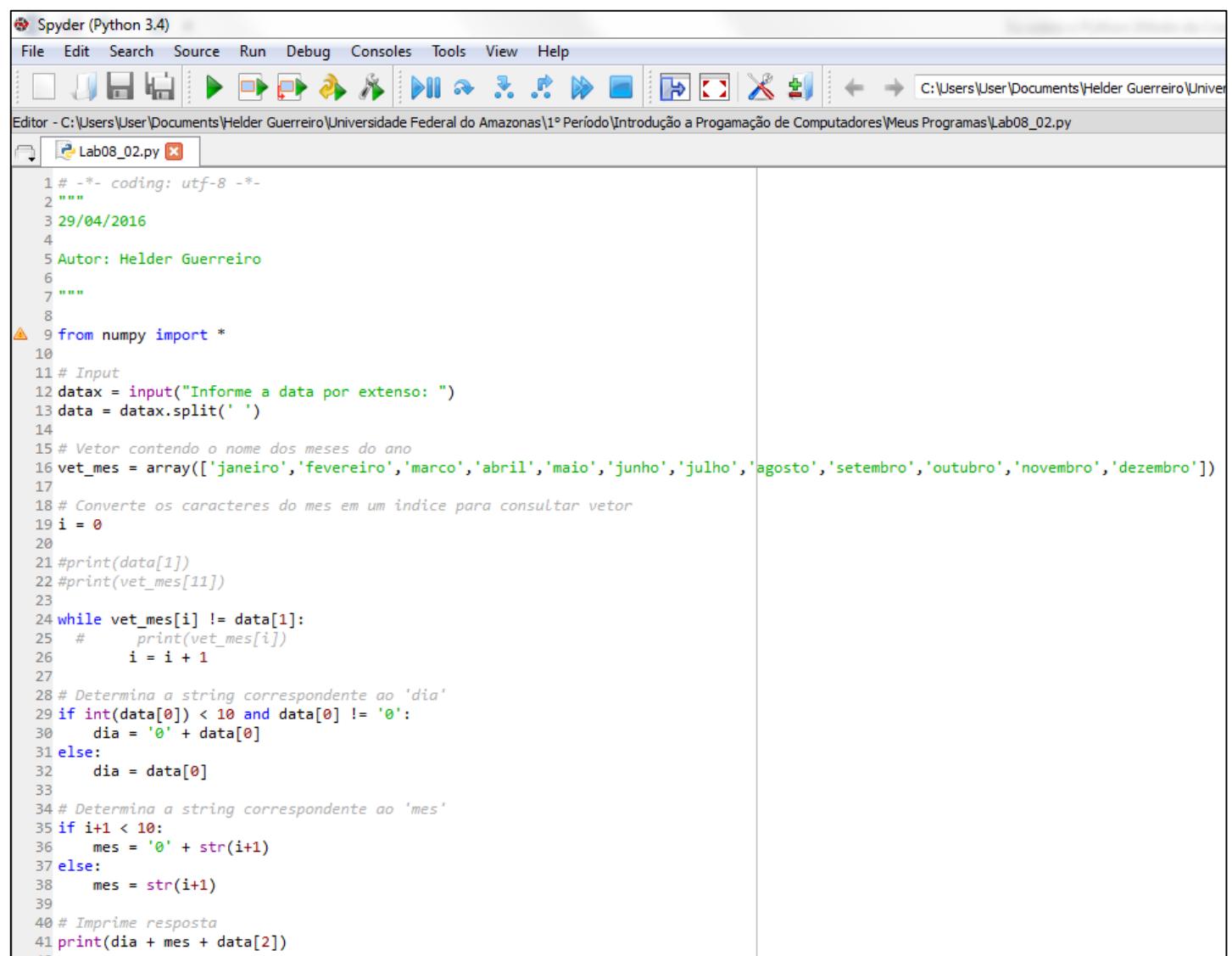
Com esse índice pronto já temos a numeração do mês pronta, pois a numeração do dia já foi inserida pelo usuário em forma de número e o ano também, somente o mês é por extenso, você já deve ter percebido que para chegar ao valor exato do ano devemos somar mais um no índice né? Por que no exemplo acima chegamos em índice 11 para dezembro enquanto ele é o mês 12.

Antes de corrermos para o abraço temos que prestar atenção em uma coisa, se colocarmos a data “1 de janeiro de 1909” o dia deverá ter um zero na frente e o mês também, por que? Pura frescura da questão. Para colocarmos um zero na frente de cada numeração precisamos de uma condicional `if`. A `if` tem por condição o seguinte: se o dia for menor que 10, então o mine script será acionado para colocar um zero na frente.

O primeiro `if` que é o do dia terá por condição, que o dia inserido que está na string `data` seja menor que 10, lembre-se que usamos a função `split()` na data e assim a string foi dividida em três, 0 para o dia, 1 para o mês e 2 para o ano, então para selecionar o dia inserido use “`data[0]`” que deve ser transformado em número, por que o dia inserido será lido como string e para usar o operador condicional “menor que” devemos usá-lo somente com números, então use a função `int` para converter a string em número. Outra condição também é necessária, que o dia seja diferente de zero, por que se o usuário (retardado) inserir um “dia zero” para que colocar outro zero na frente?

O **mine** script será uma soma de strings, por que quando somamos strings estamos juntando letras e palavras, então basta somar zero (entre aspas para ser string) com o dia inserido, esse é o valor do dia. Caso não aja uma satisfação da condição do if, o dia será simplesmente o próprio dia inserido.

O segundo if que é o do mês terá por condição que o índice somado a 1 seja menor que 10, o **mine** script é o mesmo esquema do dia, some o zero mais o índice somado a 1, só que esse índice somado a 1 deve ser transformado em string com a função str, pois string só se pode somar com string. Caso seja maior ou igual a 10 o mês será somente o índice somado a 1 (em string).



The screenshot shows the Spyder Python 3.4 IDE interface. The title bar says "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. The toolbar has various icons for file operations like Open, Save, Run, and Stop. The status bar at the bottom shows the path "C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas\1º Período\Introdução a Programação de Computadores\Meus Programas\Lab08_02.py". The main code editor window contains the following Python script:

```
1 # -*- coding: utf-8 -*-
2 """
3 29/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 from numpy import *
10
11 # Input
12 datax = input("Informe a data por extenso: ")
13 data = datax.split(' ')
14
15 # Vetor contendo o nome dos meses do ano
16 vet_mes = array(['janeiro','fevereiro','marco','abril','maio','junho','julho','agosto','setembro','outubro','novembro','dezembro'])
17
18 # Converte os caracteres do mes em um indice para consultar vetor
19 i = 0
20
21 #print(data[1])
22 #print(vet_mes[11])
23
24 while vet_mes[i] != data[1]:
25     #    print(vet_mes[i])
26     i = i + 1
27
28 # Determina a string correspondente ao 'dia'
29 if int(data[0]) < 10 and data[0] != '0':
30     dia = '0' + data[0]
31 else:
32     dia = data[0]
33
34 # Determina a string correspondente ao 'mes'
35 if i+1 < 10:
36     mes = '0' + str(i+1)
37 else:
38     mes = str(i+1)
39
40 # Imprime resposta
41 print(dia + mes + data[2])
42
```

Figura 217: Questão 2H Primeiro Passo

Fonte: ICOMP, UFAM

The screenshot shows a Windows-style application window titled "Console". Inside, there's a tab bar with "Python 1" selected. The main area contains the following text:

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Introdução a Programação de Computadores/Meus Programas/Meus Programas')  
Introdução a Programação de Computadores/Meus Programas  
Helder Guerreiro/Universidade Federal do Amazonas  
/Meus Programas'  
Informe a data por extenso: 17 janeiro 1909  
17011909  
>>>
```

Figura 218: Resultado questão 2H

Fonte: Autoria própria

Questão 3 - Operações básicas envolvendo strings

Escreva um programa que leia uma string digitada pelo usuário. Em seguida, imprima as seguintes informações sobre a string, necessariamente na ordem indicada:

- 1. Quantidade de caracteres da string*
- 2. Primeiro caractere da string*
- 3. Último caractere da string*
- 4. Conversão da string em caracteres maiúsculos*
- 5. Conversão da string em caracteres minúsculos*
- 6. Impressão de 789 cópias da string*

DICAS:

- 1. Utilize as funções e métodos vistos em sala de aula: `len()`, `upper()` e `lower()`.*
- 2. Exemplo (não exaustivo). Para a entrada `Guido van Rossum`, a saída deverá ser:*

*16
G
m
GUIDO VAN ROSSUM
guido van rossum
Guido van Rossum....Guido van Rossum*

A última linha acima foi truncada para apenas duas cópias da string, para caber no espaço. Porém, seu programa deverá imprimir as 789 cópias solicitadas pela especificação.

Primeiro passo: Interpretação da questão

Esta questão é simples, nós só vamos pôr em prática as funções de strings que aprendemos no início do assunto.

Coloque de entrada uma string, lembre-se que para darmos entrada a uma string devemos usar somente a função `input`, não importa se a entrada for de números ou não.

A primeira coisa a se fazer é ver a quantidade de caracteres dentro de uma string, para isso temos a função `len`, basta colocarmos a string dentro dessa função e imprimir o resultado final.

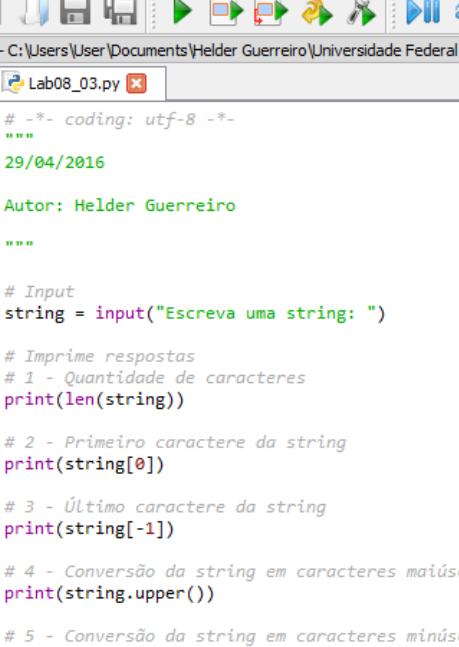
O primeiro caractere de uma string é sempre representado pelo índice 0, ou seja, basta imprimir o comando `"string[0]"` e o primeiro caractere será apresentado.

O último caractere de uma string é sempre representado pelo índice -1, use o comando `"string[-1]"` e coloque-o dentro da função `print`.

Para converter strings para todas as letras maiúsculas precisamos da ajuda da função `string.upper()`. Com isso basta imprimir a função e pronto.

A próxima exigência da questão trabalha com o oposto da função anterior, a função `string.lower()` faz com que todas as letras da string fiquem minúsculas.

E por último basta fazermos uma multiplicação da string por 789, lembre-se multiplicar string com número é uma cópia da string.



The screenshot shows the Spyder Python 3.4 IDE interface. The title bar reads "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, and View. Below the menu is a toolbar with icons for file operations like Open, Save, and Run. The main area is the "Editor - C:\Users\User\Documents\Helder Guerreiro\Universidade Federal do Amazonas" tab, which contains the following Python code:

```
1 # -*- coding: utf-8 -*-
2 """
3 29/04/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 # Input
10 string = input("Escreva uma string: ")
11
12 # Imprime respostas
13 # 1 - Quantidade de caracteres
14 print(len(string))
15
16 # 2 - Primeiro caractere da string
17 print(string[0])
18
19 # 3 - Último caractere da string
20 print(string[-1])
21
22 # 4 - Conversão da string em caracteres maiúsculos
23 print(string.upper())
24
25 # 5 - Conversão da string em caracteres minúsculos
26 print(string.lower())
27
28 # 6 - Impressão de 789 cópias da string
29 print(string*789)
30
```

Figura 219: Questão 2H Primeiro Passo

Fonte: ICOMP, UFAM

Figura 220: Resultado questão 2H

Fonte: Autoria própria

9. Arquivos

A partir de agora não vou mais submete-lo a questões, vamos estudar algumas coisas que seriam bem interessantes de se ter habilidade. A primeira delas é o assunto arquivos, como assim? Isso mesmo, o Python também pode dar uma cutucada em arquivos e deixar algumas coisas mais organizadas para você, vamos lá?

9.1 Pra que isso?

Para facilitar sua vida, veja que quando fazemos qualquer script ao reproduzi-lo os dados aparecem, mas após você fechar o script ou o programa os dados se perdem e já era. Além de podermos criar arquivos o Python também ler os arquivos e os transforma em dados. Para preparamos tudo, tenho que lhe avisar que o script do Python cria e ler arquivos que estão na sua mesma pasta raiz.

9.2 Abrindo e fechando

Podemos abrir um arquivo para lê-lo ou para escrever alguma coisa, para ler um arquivo ele deve estar na pasta raiz do script e para criar um arquivo não há necessidade em fazer nada, ele simplesmente será criado pelo nome que você escolher. As duas e primeiras funções a aprendermos aqui são `open` e `close`, uma para abrir e outra para fechar o arquivo, sua formulação deve ser assim:

```
<variável> = open("arquivo.ext", <modo>)
```

Figura 221: Função open

Fonte: ICOMP, UFAM

```
arq.close()
```

Figura 222: Função close

Fonte: ICOMP, UFAM

Por que isso? Temos que abrir um arquivo para podermos escrever algo ou lê-lo.

9.3 Modos

Modos são na verdade comandos que ao coloca-los na função `open` você estará dizendo à função `open` que você está abrindo o arquivo para um certo objetivo, esses objetivos são:

Modo	Significado
r	Leitura apenas
w	Escrita, apagando o conteúdo existente
a	Escrita, acrescentando após conteúdo existente
rb	Leitura de arquivos binários
wb	Escrita de arquivos binários, apagando o conteúdo existente
ab	Escrita de arquivos binários, acrescentando após conteúdo existente

Figura 223: Modos

Fonte: ICOMP, UFAM

Para um arquivo novo use "w" mesmo, mas se quiser modificar um arquivo qualquer então use "a" para que os dados que estavam no arquivo não se percam e sim permaneçam.

9.4 Escrevendo

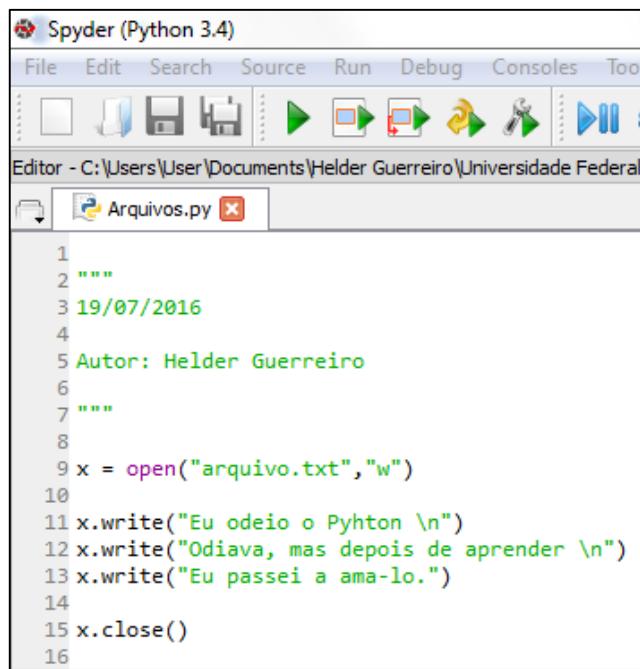
Ao identificar a abertura de um arquivo com a finalidade de escrevê-lo, usamos um comando chamado write para que você possa escrever o que quiser nesse arquivo, esse comando deve ser feito assim:

```
<arquivo>.write ("Eu Odeio o Python. \n")
```

Figura 224: Comando write

Fonte: ICOMP, UFAM. Editado pelo autor

Quando você faz isso você está abrindo o arquivo e escrevendo exatamente o que está dentro das aspas, não esqueça que aquele "\n" quer dizer que o texto pulou uma linha. Veja um exemplo abaixo:



Screenshot of the Spyder Python 3.4 IDE. The window title is "Spyder (Python 3.4)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, and Tools. Below the menu is a toolbar with various icons. The main area shows the code for "Arquivos.py":

```
1
2 """
3 19/07/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 x = open("arquivo.txt","w")
10
11 x.write("Eu odeio o Pyhton \n")
12 x.write("Odiava, mas depois de aprender \n")
13 x.write("Eu passei a ama-lo.\n")
14
15 x.close()
16
```

Figura 225: Escrevendo
Fonte: Autoria própria

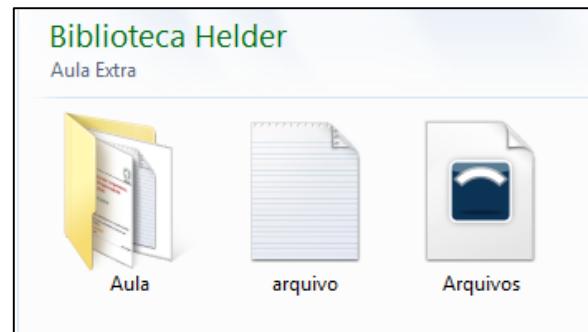


Figura 226: Pasta raiz
Fonte: Autoria própria



Figura 227: Arquivo
Fonte: Autoria própria

9.5 Lendo

Há três formas de se ler um arquivo, e quando o Python lê o arquivo você pode usá-lo para alguma ação que queira fazer com esses dados. As três formas de se ler um arquivo é usando os comandos: `read()`, `readline()` e `readlines()`.

□ Método `readline`

- Lê uma única linha do arquivo aberto e guarda-a em uma variável do tipo `string`.

□ Método `read`

- Lê todas as linhas do arquivo aberto e guarda-as em uma variável do tipo `string`.

□ Método `readlines`

- Lê todas as linhas do arquivo aberto e guarda-as em uma lista de strings.

Figura 228: Lendo

Fonte: ICOMP, UFAM

Para usar esses comandos devemos usar o comando `for`, pois eles trabalham e funcionam em conjunto. Veja como elas devem ser representadas.

```
# Abre arquivo
arq = open("filosofos.txt", "r")

for linha in arq.readlines():
    print(linha)

# Fecha arquivo
arq.close()
```

```
# Abre arquivo
arq = open("filosofos.txt", "r")

for linha in arq.readline():
    print(linha)

# Fecha arquivo
arq.close()
```

```
# Abre arquivo
arq = open("filosofos.txt", "r")

for linha in arq.read():
    print(linha)

# Fecha arquivo
arq.close()
```

Figura 229: comandos de leitura

Fonte: ICOMP, UFAM

A diferença de `readlines()` para `read()` é que `read()` vai pegar caractere por caractere do texto como se fossem uma linha e `readlines()` analisa linha por linha, vamos ver alguns exemplos:

The screenshot shows the Spyder Python 3.4 IDE interface. On the left, the code editor displays a script named 'Arquivos.py' with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 19/07/2016
4 
5 Autor: Helder Guerreiro
6 """
7 
8 
9 x = open("arquivo.txt","r")
10
11 for y in x.readlines():
12     print(y,end="")
13
14 x.close()
15
```

The screenshot shows the Python 1 console window. The command `>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Arquivos.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Arquivos/Aula Extra')` was run, followed by the output:

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Arquivos.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Arquivos/Aula Extra')
Eu odeio o Pyhton
Odiava, mas depois de aprender
Eu passei a àma-lo.>>>
```

Figura 231: Readlines resultado

Fonte: ICOMP, UFAM

Figura 230: Readlines

Fonte: ICOMP, UFAM

The screenshot shows the Spyder Python 3.4 IDE interface. On the left, the code editor displays a script named 'Arquivos.py' with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 19/07/2016
4 
5 Autor: Helder Guerreiro
6 """
7 
8 
9 x = open("arquivo.txt","r")
10
11 for y in x.readline():
12     print(y,end="")
13
14 x.close()
15
```

The screenshot shows the Python 1 console window. The command `>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Arquivos.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Arquivos/Aula Extra')` was run, followed by the output:

```
>>> runfile('C:/Users/User/Documents/Helder Guerreiro/Arquivos.py', wdir='C:/Users/User/Documents/Helder Guerreiro/Arquivos/Aula Extra')
Eu odeio o Pyhton
>>>
```

Figura 233: Readline resultado

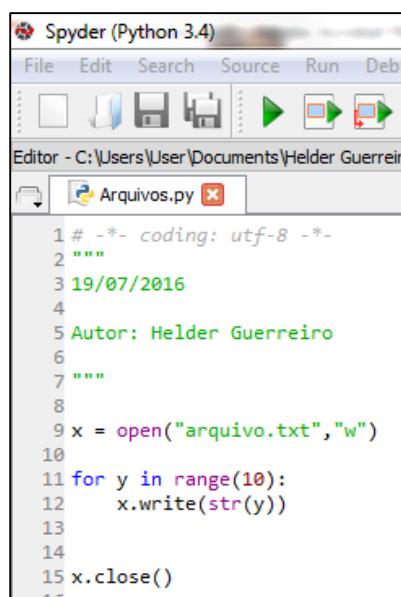
Fonte: ICOMP, UFAM

Figura 232: Readline

Fonte: ICOMP, UFAM

9.6 Dados numéricos

Quando for colocar números dentro de um arquivo, primeiro transforme-o em string usando a função `str`, e aí sim o número está pronto para ser escrito num arquivo, isso por que o número normal serve para ser operado, o número em forma de string serve para ser representado.



```
# -*- coding: utf-8 -*-
"""
19/07/2016
Autor: Helder Guerreiro
"""

x = open("arquivo.txt","w")
for y in range(10):
    x.write(str(y))
x.close()
```

Figura 234: Dados numéricos

Fonte: Autoria própria

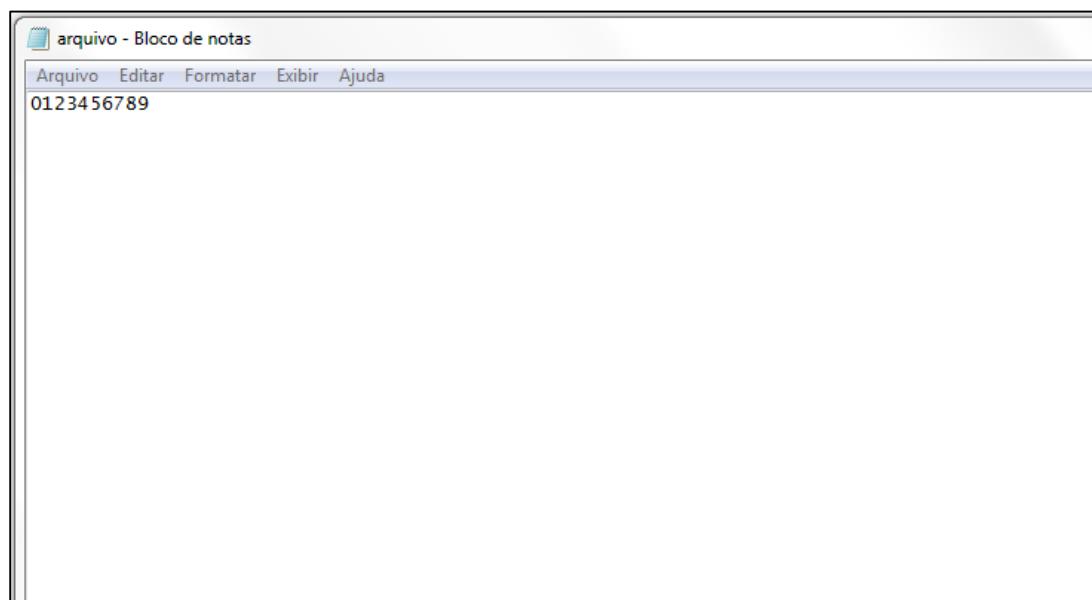


Figura 235: Dados numéricos

Fonte: Autoria própria

Veja que usei o comando `range` para criar um intervalo de 0 a 10 para `y`, e depois o comando `write` copiava cada número para dentro do arquivo. Se quisesse colocar um espaço entre os números bastava colocar + " " dentro do comando `write`.

Agora para pegarmos os números de um arquivo de texto e usa-los no script precisamos do comando `read`, mas dessa vez fora do `for`. O que vai acontecer é que o arquivo será aberto com a finalidade de ser lido e o comando `read` irá armazenar todos os dados em forma de strings, para transformarmos esses valores em números inteiros precisamos correr dentro da string usando o `for` assim aplicar a função `int` em cada valor da string, mas preste atenção, se os números forem pegos tudo de uma vez teremos um erro, por isso usamos o comando `split()` para aplicar uma separação entre cada número do arquivo.

```

Spyder (Python 3.4)
File Edit Search Source Run Deb
Editor - C:\Users\User\Documents\Helder Guerreiro
Arquivos.py
1 # -*- coding: utf-8 -*-
2 """
3 19/07/2016
4
5 Autor: Helder Guerreiro
6
7 """
8
9 x = open("arquivo.txt", "r")
10 z = x.read()
11 for y in z.split():
12     print(int(y))
13 x.close()
14
15
16
17
18

```

```

Console
Python 1
>>> runfile('C:/Users/User/.../Arquivos.py', wdir='C:/.../Aula Extra')
0
1
2
3
4
5
6
7
8
9
>>>

```

Figura 237: Resultado de números
Fonte: Autoria própria

Figura 236: Números de arquivos
Fonte: Autoria própria

Assim como o exemplo acima foi feito com o `int`, você pode também fazer com o `float`. Caso você queira especificar que a numeração não é somente 1 2 3, mas sim 12 e 3, então vá ao seu arquivo e coloque um ponto ao lado do número desejado e no script use o `float`, dessa forma o resultado mostrado terá a numeração 12 e 3 em vez de 1 2 e 3.

Um Futuro Não Tão Distante

Aqui chegamos ao fim da nossa aventura, você agora já sabe muitas coisas sobre o Python e já tem condições de fazer seus próprios scripts sozinho. Você deve estar se perguntando acerca de alguns assuntos:

Por que o último assunto foi tão curto?

O assunto String é muito complicado, tanto que para sua total compreensão é necessária uma apostila somente para ele, o - que você aprendeu aqui foi somente uma introdução para não deixar você aéreo no assunto.

Por que não teve avaliações resolvidas nos assuntos 7 e 8?

- No assunto 7 (matrizes) houve uma falha na apresentação completa da avaliação no site oficial de trabalhos e avaliações de Python da UFAM o CodeBench, sem uma apresentação legível da avaliação não foi possível criar a resolução da mesma.
- No assunto 8 (strings) não houve avaliações na época dos meus estudos, então como não existiu avaliações não tem como existir resolução.

O assunto de gráfico foi superficial?

- Sim, esse assunto requer uma apostila somente para esse assunto.

Houve questões que não foram resolvidas aqui nesta apostila, por que?

- Todas as questões que pude resolver e mostrar aqui eu fiz, mas houve casos como nos assuntos 4 (repetição por condição) e 6 (repetição por contagem) que questões apresentaram problemas, no assunto 4 foi tudo normal, mas a partir da questão 7 em diante houve uma repetição do mesmo procedimento tomado na 6, no assunto 6 houve questões que não corresponderam ao resultado esperado, depois de muita análise chegou-se ao resultado que as questões de frações contínuas eram todas problemáticas (exceto a que eu fiz). No assunto 7 (matrizes) houve questões que forma omitidas por causa da sua grande complexidade envolvendo gráficos e scripts gigantes, então como estamos somente numa apostila de introdução é desnecessário apresentar esse assunto agora. No assunto 8 (strings) as questões foram até a 3 por que as demais exigiam muito do programador até de assuntos não estudados, é melhor deixar isso para outra hora.

Em um futuro não tão distante, novas apostilas virão, é muito importante nos focarmos no aprendizado do Python por que algum dia iremos precisar e muito, mesmo que você não trabalhe com o Python, mas você terá que programar e programar é uma técnica universal independente do programa ou da intenção. Para quem estuda Engenharia e Tecnologias e aconselho a se focar bem nos assuntos envolvidos em programação para que lá na frente você possa desfrutar de um bom aprendizado. Eu tenho certeza que agora você não tem mais o sentimento que tinha quando começou a estudar este assunto bem complexo, então acho que já podemos dizer que: Eu amo o Python.

O objetivo final

Chegamos até aqui, tudo que fiz aqui nesta apostila tem somente um único objetivo: que você aprenda a programar de forma independente, meu objetivo é que qualquer leigo consiga ler e entender e assim possa passar o conhecimento para a tela do computador. Esta apostila nunca será vendida, nem servirá de uso científico, eu sou só um estudante de graduação, não tenho as habilidades de um mestre e um doutor, reconheço o meu lugar, tudo que fiz aqui foi pela minha boa vontade de querer ajudar qualquer um que precisasse de ajuda neste assunto.

Criei esta apostila sozinho, mas não adquiri conhecimento sozinho, meus agradecimentos aos professores do ICOMP por terem permitido a circulação desta apostila nesta nova edição, os algoritmos finais das respostas das questões são de responsabilidade e direitos autorais do próprio ICOMP com exceção de alguns algoritmos produzidos por mim mesmo. Meu objetivo não é passar para você a "cola" de todas as questões do laboratório para assim você passar na matéria e tirar nota, se você fizer estará sendo um sem disciplina por que aqui te dou todas as habilidades possíveis para criar seu próprio script e aí sim depois de toda a lógica apresento o algoritmo para você comparar suas respostas e encontrar os seus erros.

Aqui termino a segunda edição da apostila "Eu odeio o Python", a propósito esse nome me veio à cabeça depois de perceber que os meus professores usavam muito o exemplo "Eu amo o Python" para explicar certos assuntos, então eu olhava para minha turma e via muita gente sem entender o assunto e com muita dificuldade em criar seus próprios algoritmos, então o que vem a cabeça não é "Eu amo o Python" e sim "Eu odeio o Python", bom mas depois que você aprende e realmente sabe mexer nesse programa você começa a vê-lo como algo interessante e bom e de usar, então passamos de odiá-lo no início dessa apostila para amá-lo agora no final, lembre-se nunca julgue um livro pela capa.

Helder Guerreiro, 20 de julho de 2016 – Manaus, AM.