Lógica de programação – 1 semestre

11/02

Por: Gabi Oliveira

* *Lista*
* *Funções*
* *Trabalho com strings*
* *Operadores*
* *Entrada e saída de dados*
* *Estrutura de seleção*

Python:

***Legível*** – tenho identação obrigatória

***Multiplataforma***: Windows

***Tipagem forte de dinâmica:*** é possível atribuir qualquer valor a uma variável e em qualquer momento do código fonte

***Multiparadigma:*** estruturado, funcional e orientado a objetos

***Usada em várias aplicações***: desktop, web, mobbie

Diferencia maiúsculas e minúsculas

*(variável PODE ficar grande, não usar acentos etc)*

Variáveis:

- Iniciar obrigatoriamente com uma letra

- Pode conter números e caracteres sublinhado

- Permite a utilização de acentos (conjunto de caracteres chamado UTF-8) a partir da versão 3

- ***Exemplo***: nome, n1, média\_notas, \_a

- Evite acentos, python aceita, no entanto, não é recomendado

Tipos de dados:

* numéricos
* Inteiros (1,2,3...)
* Ponto flutuante ( 1,33,3,3,1415,7,0)

***Logica ou booleano:***

* True para verdadeiro
* False para falso

***String:***

* S = “abc” ou S= ‘abc’
* Listas
* Tuplas
* Dicionários

Operadores:

Adição +

Subtração -

Multiplicação\*

Divisão /

Divisão de inteiros//

Potenciação\*\*

Modulo (resto da divisão inteira) %

Op. Relacionais:

Igualdade ==

Maior que >

Menor que <

Diferente !=

Maior ou igual >=

Menor ou igual<=

Atribuição =

# Phyton – Aula 13/02

Operadores lógicos:

>>>a =1

>>>b = 5

>>>c =2

>>>d = 5

Saída de dados:

* Função print
* Exibe uma mensagem na tela do computador

Entrada de dados:

* Imput
* Espera o usuário digitar um texto no teclado e executa o comando

Aula 18/02

*If <condicao>:*

*Bloco verdadeiro*

* (:) - informa que existe um bloco de linhas a seguir.
* Bloco
* Representado com um deslocamento do início da

linha para a direita (indentação).

* Continua até a primeira linha com deslocamento diferente.

Estruturas de Repetição –Aula 25/02

Permitem executar a mesma parte de um programa várias vezes, sob determinadas condições.

* Imprimindo de 1 a 3 - forma básica:

Exemplo:

# -\*- coding: UTF-8 -\*-

print(1)

print(2)

print(3)

* Imprimindo de 1 a 3, com variável.

Exemplo:

# -\*- coding: UTF-8 -\*-

x = 1

print(x)

x = 2

print(x)

x = 3

print(x)

* Imprimindo de 1 a 3, com incremento.

Exemplo:

# -\*- coding: UTF-8 -\*-

x = 1

print(x)

x = x + 1

print(x)

x = x + 1

print(x)

While -

while <condição>:

bloco de código

Onde:

condição: condição a ser avaliada.

Contadores :

* Modifique o programa anterior para imprimir de 1 até o número digitado pelo usuário, mas, dessa vez, apenas os números ímpares.
* Reescreva o programa anterior para escrever os 10 primeiros múltiplos de 3.

Interrompendo a repetição

* A estrutura while só verifica a sua condição de parada no início de cada repetição.
* Instrução break
* Interrompe a execução de while, independentemente do valor atual de sua condição.

For in range - 27/02

* v: um nome de variável.
* Range(argumento(s)): função para gerar valores com até

três inteiros passados a ela.

* Uso da função range(início, fim, incremento), com

saltos. Ex:

*# -\*- coding: UTF-8 -\*-*

*# Programa para imprimir os números 0, 2, 4, 6 e 8.*

*for v in range(0, 10, 2):*

*print(v)*

* Usando for para iterar sobre uma string. Ex:

*# -\*- coding: UTF-8 -\*-*

*# Programa para imprimir os caracteres 1 a 1 da string texto.*

*texto = "Olá Mundo!"*

*for letra in texto:*

*print(letra)*

Funções e variável global - 11.03

* Uma função é um bloco de código que tem nome,

pode receber parâmetros e gerar um resultado.

* São interessantes para isolar uma tarefa específica em

um trecho de programa (técnica de modularização).

* Aumentam a possibilidade de reuso e reduzem a

complexidade de manutenção.

* Já usamos várias funções de Python, como por

exemplo: int, float, print e input.

* def deve ser utilizada para definir uma função e return

para devolver algum valor.

variável locais e globais

Local

* Existe apenas dentro da função.
* Não pode ser acessada fora da função em que foi

criada.

Global

* Definida fora de uma função.
* Pode ser vista e alterada por qualquer função do

programa.

* Deve ser utilizada com cuidado, como por

exemplo em constantes.

Prova 20.03

* Caderno/estojo
* Sem consulta
* Temas: print, input,if,else,elif,for,while,funções,listas,dicionários
* Individual,no papel
* Peso: 0-100

Git-Hub 27.03

* Finalidade:Compartilhar códigos
* repositório: onde guardamos os códigos (pasta ou diretório)
* git config--global user.name seu nome(digitar comando assim)
* git config -- global user.email seu email (digitar também)
* criação de repositório git local:
* mkdir meu\_projeto, cd meu\_projeto,git init
* entra na pasta do VS, escreve o comando teste.py
* git add
* git commit -m "Mensagem descrevendo as alterações feitas"
* ex: git commit -m “estamos adicionando a idade de fulano"
* git remote add origin URL\_do\_seu\_repositório\_no\_GitHub(depois de criar git hub online)
* no git hub entrar no seu perfil, criar um novo repositório, dê nome ao repositório (nome:git hub)
* o repositório deve ser público
* add a README file(marcar)
* add gitignore(none)
* license(MTI license)
* agora o repositório está sendo criado
* Branch(versões do repositório) permitem trabalhar em várias funcionalidades ou correções simultaneamente sem interromper o desenvolvimento principal. A branch principal é normalmente chamada de main ou master, e outras branches são criadas para diferentes tarefas (como adicionar um novo recurso ou corrigir um bug).
* Merge(mesclar)
* Agora no VS digite git remote add origin URL\_do\_seu\_repositório\_no\_GitHub (ir em code no git hub online e colocar no lugar da URL)
* git push -u origin master(para enviar mudanças para o github)
* git checkout master
* git merge nome\_do\_branch
* “U” arquivo novo criado no repositório
* Git desketop, no commit USE titulo e descrição

Passo a Passo:

* alterar: git add.
* Git commit -m “...”
* Git push (-u origin)
* Git pull