Disciplina: Linguagem de Programação

Python

Profª. Drª. Giovana Angélica Ros Miola
giovana.miola@fatec.sp.gov.br

Fatec
Presidente Prudente

Centro Paula Souza

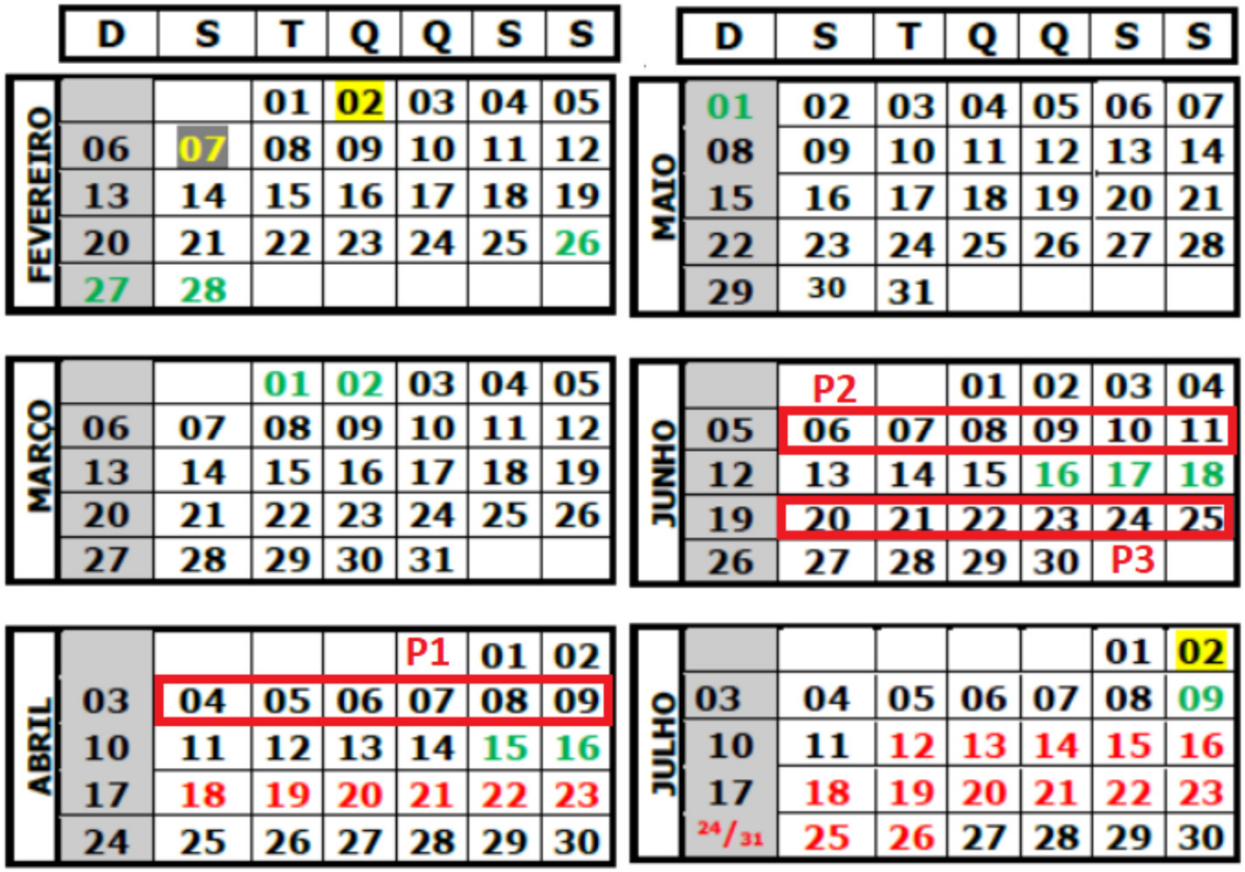
Conteúdo Programático

- Linguagens de Programação (Python)
  - Variáveis, constantes, operadores e expressões.
  - Comando de desvio (estrutura de decisão).
  - Controle de malhas (estruturas de repetição).
  - Vetores e ponteiros.
  - Funções de biblioteca.
  - Estruturas, uniões e tipos definidos pelo usuário.
  - Manipulação de arquivos.

Provas
1º sem/2022

- P1:
  - 04/04/2022
- P2:
  - 06/06/2022
- P3:
  - 20/06/2022

Calendário Acadêmico



	D	S	T	Q	Q	S	S
FEVEREIRO			01	02	03	04	05
	06	07	08	09	10	11	12
	13	14	15	16	17	18	19
	20	21	22	23	24	25	26
	27	28					

	D	S	T	Q	Q	S	S
MAIO	01	02	03	04	05	06	07
	08	09	10	11	12	13	14
	15	16	17	18	19	20	21
	22	23	24	25	26	27	28
	29	30	31				

MARÇO			01	02	03	04	05
	06	07	08	09	10	11	12
	13	14	15	16	17	18	19
	20	21	22	23	24	25	26
	27	28	29	30	31		

JUNHO		P2		01	02	03	04
	05	06	07	08	09	10	11
	12	13	14	15	16	17	18
	19	20	21	22	23	24	25
	26	27	28	29	30	P3	

ABRIL					P1	01	02
	03	04	05	06	07	08	09
	10	11	12	13	14	15	16
	17	18	19	20	21	22	23
	24	25	26	27	28	29	30

JULHO						01	02
	03	04	05	06	07	08	09
	10	11	12	13	14	15	16
	17	18	19	20	21	22	23
	24/31	25	26	27	28	29	30

Linguagem de Programação

- Uma linguagem de programação é um vocabulário e um conjunto de regras gramaticais usadas para escrever programas de computador.

- Esses programas instruem o computador a realizar determinadas tarefas específicas.

- Cada linguagem possui um conjunto único de palavras-chave (palavras que ela reconhece) e uma sintaxe (regras) específica para organizar as instruções dos programas.

Linguagem de Programação

- Os computadores são usados em uma infinidade de diferentes áreas.

- Por causa dessa grande diversidade no seu espaço, linguagens de programação com metas muito diferentes têm sido desenvolvidas.

- Diversidades de aplicações:
  - Aplicações Científicas;
  - Aplicações Comerciais;
  - Inteligência Artificial;
  - Programação de Sistemas;
  - Linguagens de Scripting.

Aplicações Científicas

- Os primeiros computadores digitais, que surgiram na década de 40, eram usados para aplicações científicas.
- As aplicações científicas têm estruturas de dados simples, mas exigem um grande número de computações aritméticas.
- A eficiência era a primeira preocupação.
- A primeira linguagem para aplicações científicas foi o FORTRAN.

Aplicações Comerciais

- O uso de computadores para aplicações comerciais teve início na década de 50.

- Equipamentos especiais foram desenvolvidos para tal propósito, juntamente com linguagens especiais. A primeira linguagem bem sucedida foi o COBOL.

- As linguagens comerciais são caracterizadas por:
  - Facilidades para produzir relatórios elaborados;
  - Maneiras precisas de descrever e armazenar números decimais e textos;
  - Capacidade de especificar operações aritméticas decimais.

Inteligência Artificial

- A Inteligência Artificial (IA) é uma área abrangente das aplicações de computadores caracterizada pelo uso de computações simbólicas em vez de numéricas.
  - Símbolos (que consistem em nomes) são manipulados no lugar de números.

- A primeira linguagem de programação desenvolvida para aplicações de IA amplamente utilizada foi a funcional LISP, mais tarde surgiu a programação lógica com a linguagem Prolog.

Programação de Sistemas

- O Sistema Operacional e todas as ferramentas de suporte à programação de um computador são coletivamente conhecidos como seu software básico que é usado quase continuamente e, portanto, deve ter eficiência na execução.

- Uma linguagem para tal domínio deve oferecer uma execução rápida.

- A linguagem C tem características que a torna boa para a programação de sistemas. Ela é relativamente fácil de portar ou mover, para máquinas diferentes e sua execução é eficiente.

Linguagens de Scripting

- As linguagens de scripting desenvolveram-se lentamente ao longo dos últimos 25 anos.

- São usadas colocando-se uma lista de comandos, chamados de script, em arquivos para serem executados.

- A JavaScript é uma linguagem de scripting desenvolvida pela Netscape para uso tanto em servidores Web como para navegadores. Ela tem crescido, tanto como linguagem, como em popularidade nos últimos anos.

- Outros exemplos: VBScript (sistemas Windows), BashScript (sistemas Linux) etc.

Categorias

- As linguagens de programação podem ser classificadas, em uma escala relativa à sua semelhança com a linguagem humana, em:
    - Linguagem de Máquina;
    - Linguagem Assembly;
    - Linguagem de Alto Nível;
    - Linguagem de 4ª Geração;
    - Linguagem de 5ª Geração.

Linguagem de Máquina

- É a linguagem de mais baixo nível de entendimento pelo ser humano e a única, na verdade, entendida pelo processador (CPU).
- É constituída inteiramente de números, o que torna praticamente impossível entendê-la diretamente.
- Cada CPU tem seu conjunto único de linguagem de máquina, definido pelo fabricante do chip.
- Uma instrução típica em linguagem de máquina seria algo como:

0100 1111 1010 0110

- As instruções presentes na linguagem de máquina são as mesmas da linguagem do nível mais acima (linguagem assembly).
- Os programas escritos nas linguagens de mais alto nível são convertidos (compilados ou montados) para a linguagem de máquina específica, para que possam ser executados pelo computador.
- Essa linguagem é também classificada como linguagem de primeira geração.

Linguagem Assembly

- É a linguagem de nível imediatamente acima da linguagem de máquina.
- Possui a mesma estrutura e conjunto de instruções que a linguagem de máquina, porém permite que o programador utilize nomes (chamados mnemônicos) e símbolos em lugar dos números.
- A linguagem assembly é também única para cada tipo de CPU, de maneira que um programa escrito em linguagem assembly para uma CPU poderá não ser executado em outra CPU de uma família diferente.
- Nos primórdios da programação todos os programas eram escritos em Assembly.
- Hoje, a linguagem Assembly, é utilizada quando a velocidade de execução ou o tamanho do programa executável gerado são essenciais.

Linguagem Assembly

- Atualmente a maioria dos programas é escrita em linguagens de alto nível.
- Todos os programas escritos nessas linguagens são convertidos para a linguagem de máquina para serem executados pelo processador.
- A conversão da linguagem Assembly para a linguagem de máquina se chama montagem, e é feita por um programa chamado montador (ou assembler).
- Exemplo de uma típica instrução em assembly :

  ORG 100
  END

- Essa linguagem é também classificada como linguagem de segunda geração, e, assim como a linguagem de máquina, é considerada uma linguagem de baixo nível.

Baixo nível x Alto nível

```
%ifdef  OpenBSD
section .note.openbsd.ident
        align   2
        dd      8,4,1
        db      "OpenBSD",0
        dd      0
        align   2
%endif
section .text
%ifidn __OUTPUT_FORMAT__, macho64       ; MacOS X
        %define SYS_exit        0x2000001
        %define SYS_write       0x2000004
        global  start
        start:
%elifidn __OUTPUT_FORMAT__, elf64
        %ifdef  UNIX            ;
Solaris/OI/FreeBSD/NetBSD/OpenBSD/DragonFly
                %define SYS_exit        1
                %define SYS_write       4
        %else                   ; Linux
                %define SYS_exit        60
                %define SYS_write       1
        %endif
        global  _start
        _start:
%else
        %error  "Unsupported platform"
%endif
        mov     rax,SYS_write
        mov     rdi,1           ; stdout
        mov     rsi,msg
        mov     rdx,len
        syscall
        mov     rax,SYS_exit
        xor     rdi,rdi         ; exit code 0
        syscall
section .data
msg     db      "Hello, world!",10
len     equ     $-msg
```

Assembly

```
print('Hello word!');
```

Python

Linguagens de Alto Nível

- São as linguagens de programação que possuem uma estrutura e palavras-chave que são mais próximas da linguagem humana, tornando os programas mais fáceis de serem lidos e escritos.
  - Esta é a sua principal vantagem sobre as linguagens de nível mais baixo.

- Os programas escritos nessas linguagens são convertidos para a linguagem assembly por meio de um programa compilador ou são interpertados por meio de um interpretador.

- Exemplos:
  - FORTRAN, Prolog, COBOL, PHP, C, Java, C#, Pascal, C++, Python, entre outras

Linguagens de 4ª Geração

- São linguagens de programação com estrutura mais próxima da linguagem humana do que as linguagens de programação de alto nível.

- A maioria delas é usada para acessar bancos de dados, a SQL (Structured Query Language) por exemplo.

- Também se encontram nesta geração as linguagens orientadas a objeto.

Linguagens de 5ª Geração

- Aqui se encontram as linguagens orientadas à inteligência artificial.
- Tais linguagens ainda não estão suficientemente desenvolvidas.
- Exemplo de linguagens específicas: LISP e Prolog.
- Linguagens como Python, entre outras, possuem módulos que atuam em IA.
- As bibliotecas de matemática e estatísticas disponíveis no Python são praticamente únicas. O NumPy se tornou tão onipresente que é quase uma API padrão para operações de tensor, e o Pandas traz os poderosos quadros de dados do R ao Python. Para o processamento de linguagem natural, você tem o venerável NLTK e o incrivelmente rápido SPACY. Para **machine learning**, há o Scikit-learn. E quando se trata de **deep learning**, todas as bibliotecas atuais (TensorFlow , PyTorch , chainer , Apache MXNet Theano , etc) são projetos pioneiros do Python.

Tradutores: Interpretador e Compilador

- O problema básico das linguagens de alto nível é como ser implementada em um computador cuja linguagem de máquina é bastante diferente, e de nível muito mais baixo.

- A solução encontrada foi traduzir essas linguagens usando a própria linguagem de máquina da CPU em questão. Este é o papel de um tradutor.

- Os principais tipos de tradutores são o Interpretador e o Compilador.

Interpretador

- Nesta solução, as ações indicadas pelos comandos da linguagem não são diretamente executadas.
- Em geral existe, para executar cada ação possível, um subprograma (escrito na linguagem de máquina do computador hospedeiro).
- Assim a interpretação de um programa é feita pela chamada àquele subprograma em uma sequência apropriada.
- Mais precisamente, um interpretador é um programa que executa repetidamente a seguinte sequência:
  - Obter o próximo comando a ser executado.
  - Determinar que ações devem ser executadas.
  - Executar essas ações
- É uma situação análoga a um diálogo com uma pessoa que fala um idioma diferente do nosso, com o auxílio de um intérprete.

Compilador

- Nesta solução, programas escritos em linguagem de alto nível são traduzidos em versões de linguagem de máquina, antes de serem executados.
- Essa tradução é feita em vários passos. Por exemplo:
  - Subprogramas podem ser inicialmente traduzidos para código de montagem (assembly);
  - Esse código pode depois ser traduzido para código relocável em linguagem de máquina (código objeto);
  - Em seguida, unidades de código relocável podem ser ligadas em uma única unidade relocável (linkagem), gerando o código executável;
  - E finalmente, o programa é carregado na memória principal, como código executável de máquina.
- Os tradutores usados em cada um destes passos tem nomes especiais: compilador, montador, ligador e carregador, respectivamente.

Histórico de Python

- Python foi criado no final dos anos oitenta(1989) por Guido van Rossum no Centro de Matemática e Tecnologia da Informação (CWI, Centrum Wiskunde e Informatica), na Holanda, como sucessor da linguagem de programação ABC, capaz de lidar com exceções e interagir com o sistema operacional Amoeba.
- O nome da linguagem vem do gosto de seu criador pelos humoristas britânicos Monty Python. Van Rossum é o principal autor de Python, e seu papel central contínuo na decisão da direção de Python é reconhecido, referindo-se a ele como Ditador de Vida Benevolente (em inglês: Benevolent Dictator for Life, BDFL).
- Atualmente, é gerenciada pela Python Software Foundation. Possui uma licença de código aberto, chamada Python Software Foundation License, que é compatível com a GNU General Public License a partir da versão 2.1.1 e incompatível em certas versões anteriores.

Versões da Criação de Python

Version	Latest micro version	Release date	End of full support	End of security fixes
0.9	0.9.9[2]	1991-02-20[2]	1993-07-29[a][2]	
1.0	1.0.4[2]	1994-01-26[2]	1994-02-15[a][2]	
1.1	1.1.1[2]	1994-10-11[2]	1994-11-10[a][2]	
1.2		1995-04-13[2]	Yes	
1.3		1995-10-13[2]	Yes	
1.4		1996-10-25[2]	Yes	
1.5	1.5.2[34]	1998-01-03[2]	1999-04-13[a][2]	
1.6	1.6.1[34]	2000-09-05[35]	2000-09[a][34]	
2.0	2.0.1[36]	2000-10-16[37]	2001-06-22[a][36]	
2.1	2.1.3[36]	2001-04-15[38]	2002-04-09[a][36]	
2.2	2.2.3[36]	2001-12-21[39]	2003-05-30[a][36]	
2.3	2.3.7[36]	2003-06-29[40]	2008-03-11[a][36]	
2.4	2.4.6[36]	2004-11-30[41]	2008-12-19[a][36]	
2.5	2.5.6[36]	2006-09-19[42]	2011-05-26[a][36]	
2.6	2.6.9[23]	2008-10-01[23]	2010-08-24[b][23]	2013-10-29[23]
2.7	*2.7.17*[43]	2010-07-03[43]	2020-01-01[c][43]	
3.0	3.0.1[36]	2008-12-03[23]	2009-02-13[44]	
3.1	3.1.5[45]	2009-06-27[45]	2011-06-12[46]	2012-06[45]
3.2	3.2.6[47]	2011-02-20[47]	2013-05-13[b][47]	2016-02-20[47]
3.3	3.3.7[48]	2012-09-29[48]	2014-03-08[b][48]	2017-09-29[48]
3.4	3.4.10[49]	2014-03-16[49]	2017-08-09[50]	2019-03-18[a][49]
3.5	*3.5.9*[51]	2015-09-13[51]	2017-08-08[52]	2020-09-13[53]
3.6	*3.6.10*[54]	2016-12-23[54]	2018-12-24[b][54]	2021-12[54]
3.7	*3.7.6*[55]	2018-06-27[55]	2020-06-27[b][55]	2023-06[55]
3.8	*3.8.1*[56]	2019-10-14[56]	**2021-04**[56]	**2024-10**[56]
3.9	*3.9.0 alpha 1*[57]	2020-10-05[57]	2022-05[58]	2025-10[57][58]

Legend: Old version | Older version, still maintained | **Latest version** | Latest preview version | Future release

Italic is the latest micro version of currently supported versions as of 2019-11-29.

Table notes:

a. ^ a b c d e f g h i j k l Date of last micro release.

b. ^ a b c d e Date of last non security only release.

c. ^ Official support ended on 2020-01-01, but the final release 2.7.18 is slated to occur around 2020-04.[43]

Características da linguagem Python

- Python é uma linguagem interpretada de alto nível e que suporta múltiplos paradigmas de programação: imperativo, orientado a objetos e funcional.
- É uma linguagem com tipagem dinâmica e forte, escopo léxico e gerenciamento automático de memória.
- Possui algumas estruturas de dados embutidas na sintaxe – como tuplas, listas e dicionários – que aumentam muito a expressividade do código.
- Python possui uma vasta biblioteca padrão com diversos utilitários poderosos.

Características

Virtual Machine - Máquina virtual. Um computador definido inteiramente em software. A máquina virtual de Python executa o bytecode emitido pelo compilador de bytecode.

- A comunidade Python preza muito pela legibilidade do código e possui dois elementos que reforçam ainda mais essa questão: PEP-8 e The Zen of Python.
- O PEP-8 (VAN ROSSUM, 2001) é um guia de estilos de código Python que é amplamente empregado e existem diversas ferramentas para checá-lo automaticamente.
- O The Zen of Python (PETERS, 2004) é um pequeno texto que fala muito sobre o estilo de programação em Python.
- Apesar de ser uma linguagem interpretada, existe um processo de compilação transparente que transforma o código texto em bytecode, que, por sua vez, é interpretado por uma Virtual Machine (VM).
- A implementação padrão da linguagem Python é chamada de CPython e, apesar de existirem outras implementações da especificação, é nesta que trabalharemos.

Características

- PEP – tirado da documentação Python
- Proposta de melhoria do Python. Uma PEP é um documento de design que fornece informação para a comunidade Python, ou descreve uma nova funcionalidade para o Python ou seus predecessores ou ambientes. PEPs devem prover uma especificação técnica concisa e um racional para funcionalidades propostas.

- PEPs tem a intenção de ser os mecanismos primários para propor novas funcionalidades significativas, para coletar opiniões da comunidade sobre um problema, e para documentar as decisões de design que foram adicionadas ao Python. O autor da PEP é responsável por construir um consenso dentro da comunidade e documentar opiniões dissidentes.

PEP – 8

- É importante ler a PEP pelo menos uma vez, pois há vários detalhes interessantes. Certos pontos são considerados indispensáveis por quase toda comunidade.
- Eis alguns tópicos importantes que vale destacar:
  - Use 4 espaços para indentação;
  - Nunca misture Tabs e espaços;
  - Tamanho máximo de linha é 79 caracteres;
  - lower_case_with_underscore para nomes de variáveis;
  - CamelCase para classes.
- O guia tem mais itens sobre indentação, espaçamento, sugestões de como usar imports e muito mais. A grande maioria dos programadores Python acaba adotando grande parte da PEP-8 (VAN ROSSUM, 2001) no seu dia a dia. Por isso, a sua leitura é recomendada. www.python.org/dev/peps/pep-0008

The Zen of Python (Filosofia Python)
Lista de princípios de projeto e filosofias do Python que são úteis para a compreensão e uso da linguagem.

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one – and preferably only one – obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than **right** now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea – let's do more of those!

Tim Peppers

Bonito é melhor que feio.
Explícito é melhor que implícito.
Simples é melhor que complexo.
Complexo é melhor que complicado.
Linear é melhor que aninhado.
Esparso é melhor que denso.
Legibilidade conta.
Casos especiais não são especiais o suficiente para quebrar regras.
Embora praticidade prevaleça sobre pureza.
Erros nunca devem ser silenciados.
A não ser explicitamente.
Diante de uma ambiguidade, não caia na armadilha do chute.
Deve existir um – e preferencialmente um – jeito óbvio de se fazer algo.
Embora possa não parecer óbvio a não ser que você seja holandês.
Agora é melhor que nunca.
Embora nunca normalmente seja melhor que exatamente agora.
Se a implementação é difícil de explicar, ela é uma má ideia.
Se a implementação é fácil de explicar, talvez seja uma boa ideia.
Namespaces são uma grande ideia – vamos usá-los mais!

IDEs - Ambiente de Desenvolvimento Python

IDLE

WING PYTHON IDE

ATOM IDE

PyDev

Python IDE for Eclipse

Sublime Text

PyScripter

Spyder IDE

The Scientific Python Development Environment