TOPÍCOS QUE JÁ FORAM EXPLORADOS:

DEFINIÇÃO DE SEMÂNTICA

Processos de significação, atribuição de signos significados;

Uma estrutura que explique como esse digno deve ser interpretado

Aplicação de teoria matemáticas e lógicas para estudar esses processos de significação, e essas entidades de significados. No Campo da computação a semântica formal possui duas áreas principais:

1 voltada ao estudo de como dotar os sistemas computacionais designificado, isto é, como especificar que uma determinada estrutura, o que essa estrutura significa, como o computador pode entender o conhecimento que está codificado

# Semântica formal e suas aplicações para resoluções de problemas na computação

Semântica formal utiliza a lógica para determinar signos e significados; em programas críticos por exemplo semântica formal vai auxiliar de modo a fazer com que não tenhas acidentes de avião devido a bug no software desse sistema.

Compiladores e Interpretadores: Na implementação de compiladores e interpretadores, a semântica formal é usada para traduzir o código-fonte de uma linguagem de programação para código executável. A semântica formal fornece regras claras para a geração de código a partir das construções da linguagem, garantindo que o comportamento do programa seja preservado durante a compilação ou interpretação

Verificação de Tipos: A semântica formal é usada nos sistemas de tipos das linguagens de programação para verificar a corretude dos tipos em um programa. Através de regras de inferência e restrições semânticas, a semântica formal permite detectar erros de tipos, como tentativas de atribuir valores incompatíveis ou usar operações inválidas.

1. Python 3.0: Essa versão trouxe uma série de mudanças significativas em relação à versão anterior, Python 2.x. Algumas atualizações e definições de semântica formal incluem:
   * Mudança na sintaxe do print: No Python 3.0, a função **print** foi modificada para se tornar uma função real, em vez de um statement especial. Isso implicou em uma mudança semântica na forma de uso do **print**, exigindo a utilização de parênteses para envolver os argumentos.
   * Divisão de inteiros: Em Python 3.0, a operação de divisão de inteiros usando o operador **/** sempre retorna um valor de ponto flutuante. Essa mudança foi feita para evitar confusões e inconsistências, tornando o comportamento mais previsível.
2. Python 3.6: Nessa versão, algumas atualizações relevantes e definições de semântica formal incluem:
   * Formatted string literals: Python 3.6 introduziu as chamadas "f-strings", que são literais de string formatados. Essa atualização permitiu a interpolação de expressões e variáveis diretamente em strings usando a sintaxe **{expressão}**. A semântica formal define como essas expressões são avaliadas e inseridas nas strings formatadas.
   * Order preservation in dictionaries: Antes do Python 3.6, a ordem dos elementos em um dicionário não era garantida. A partir do Python 3.6, a ordem de inserção de elementos em um dicionário foi preservada, o que implica em mudanças na semântica formal relacionada à iteração e comparação de dicionários.
3. Python 3.8: Nessa versão, uma atualização relevante e sua semântica formal associada incluem:
   * Assignment Expressions (Walrus Operator): O Python 3.8 introduziu o operador "walrus" (**:=**), que permite a atribuição dentro de expressões. Essa atualização forneceu uma sintaxe mais concisa para definir variáveis em contextos onde antes seria necessário repetir a expressão. A semântica formal descreve como a atribuição ocorre dentro das expressões e como isso afeta o escopo e o valor das variáveis.

Esses são apenas alguns exemplos de atualizações relevantes na linguagem Python e suas definições de semântica formal associadas. Cada atualização traz mudanças no comportamento da linguagem, que são especificadas formalmente para garantir consistência, corretude e previsibilidade no uso da linguagem.