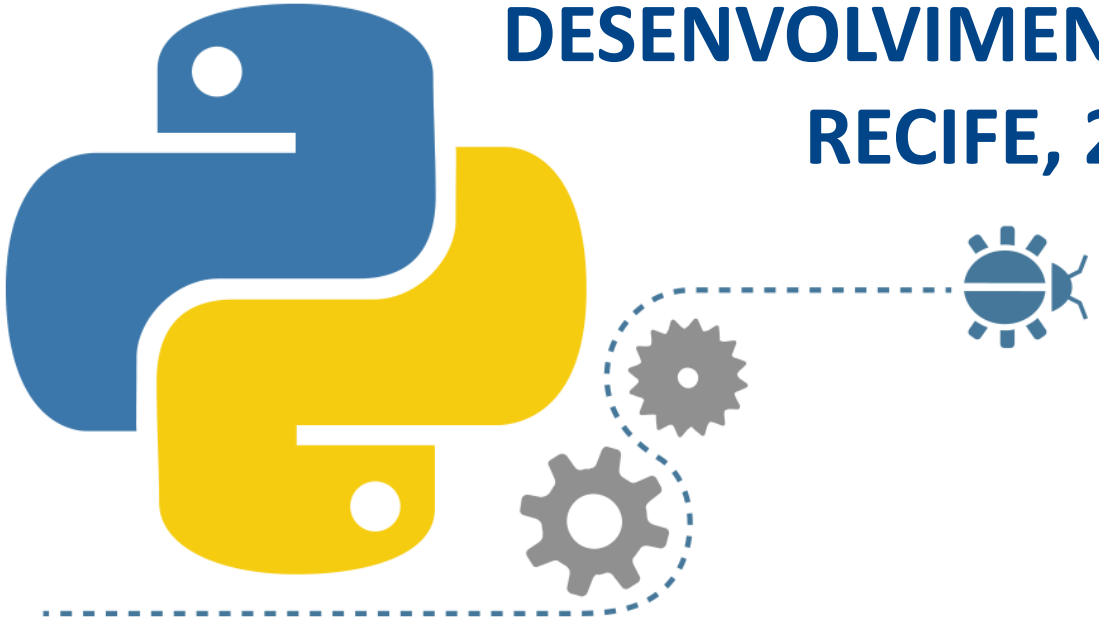
Unidade de Educação Profissional do Recife

DESENVOLVIMENTO DESKTOP
RECIFE, 2024



Tipos de Dados

Ponto Flutuante	float
Inteiro	int
Lógico	bool
Caractere	str

Mais conceitos ...

Variável é um espaço que reservamos na memória para armazenar um valor que pode mudar no decorrer do código, ou seja, receber e guardar valores diferentes.

Imagine as variáveis como um gaveteiro.

Podemos guardar um <u>único dado</u> em cada gaveta.

Tipos de Dados

As informações na memória RAM podem ser classificadas em: instruções e dados.

- As instruções comandam o funcionamento da máquina.
- Os dados são partes das informações que são processadas pelo computador.

Ex.: Proclamação da República (Informação), 15 de novembro (Dado)

A memória RAM (*Random Access Memory*) é um tipo de tecnologia que permite o acesso aos arquivos armazenados no computador e faz a leitura deles quando requeridos.

Tipos de Dados

- **Tipos numéricos em Python**
- Em Python, existem diferentes tipos de variáveis numéricas que podem ser usadas para representar números. Alguns dos tipos mais comuns são:
- **Inteiros (int)**
- Os inteiros em Python são números inteiros sem parte decimal. Eles podem ser positivos ou negativos. Por exemplo, 10, -5, 0 são exemplos de inteiros em Python.
- **Números de ponto flutuante (float)**
- Os números de ponto flutuante em Python são números com parte decimal. Eles podem representar números reais, como 3.14, -2.5, 0.0.
- **Números complexos (complex)**
- Os números complexos em Python são utilizados para representar números com uma parte real e uma parte imaginária. Eles são definidos na forma a + bi, onde a e b são números reais e i é a unidade imaginária (√-1). Por exemplo, 2 + 3j, -1 + 2j são números complexos em Python.

Tipos de Dados

Strings (str)

As strings em Python são utilizadas para representar sequências de caracteres. Elas são definidas entre aspas simples ou duplas. Por exemplo, “Olá, mundo!” é uma string em Python.

Listas (list)

As listas em Python são utilizadas para armazenar uma coleção ordenada de elementos. Elas podem conter diferentes tipos de dados e podem ser modificadas. Por exemplo, [1, 2, 3] é uma lista em Python.

Tuplas (tuple)

As tuplas em Python são semelhantes às listas, mas são imutáveis, ou seja, não podem ser modificadas após a sua criação. Elas também podem conter diferentes tipos de dados. Por exemplo, (1, 2, 3) é uma tupla em Python.

Dicionários (dict)

Os dicionários em Python são utilizados para armazenar dados em pares chave-valor. Cada elemento em um dicionário é acessado por sua chave única. Por exemplo, {“nome”: “João”, “idade”: 30} é um dicionário em Python.

Tipos Primitivos/ Saída e Entrada

Int – float-boolen-string

```
var=True
print(type(var))
```

```
n1=int(input("digite um valor"))
print(type(n1))
```

```
n1=int(input("digite um numero"))
n2=int(input("digite outro numero"))
soma=n1+n2
print("o valor total é", soma)
```

Utilizando o format

```
n1=int(input("digite um numero"))
n2=int(input("digite outro numero"))
soma=n1+n2
print("o valor total entre {0} e {1} é {2}".format(n1,n2,soma))
```

Conversão de Tipos

```
idade='45'
print(idade,type(idade))

idade_inteira=int(idade)
print(idade_inteira,type(idade_inteira))neste exemplo convertemos qualquer tipo

altura=input("digite sua altura: ")
print(altura,type(altura))

altura=float(input("digite sua altura: "))
print(altura,type(altura))

nome=input('qual o seu nome? ')
print(f'o usuario digitou {nome} e o tipo da variavel é' f' {type(nome)}')# permite digita o texto e informar o tipo de dados
```

```
#inteiro
idade = 20
idade = 17
#ponto flutuante
estatura = 1.82
#string (caractere)
saudacao = "olá"
#lógico (booleano)
aprovado = True

print(type(idade))
print(type(estatura))
print(type(saudacao))
print(type(aprovado))
```

```
qual o seu nome:luciana oliveira
qual sua idade? 40

 luciana oliveira tem 40 anos. luciana oliveira nasceu em 1984.
nome=input("qual o seu nome:" )
idade=input("qual sua idade? ")
anonascimento=2024-int(idade)

print(f' {nome} tem {idade} anos.'
     f' {nome} nasceu em {anonascimento}.')
```

As f-strings vão servir para que você consiga colocar uma variável dentro de um texto, e isso é feito utilizando a letra “f” antes do texto e colocando a sua variável dentro de {} chaves

Obs. O python ele recebe a string e converte o tipo

Exercício

Modifique o Programa 2.2, de forma que ele calcule um aumento de 15% para um salário de R$ 750. salário = 750 aumento = 15 print(salário + (salário * aumento / 100))

```
salário = 750
aumento = 15
print(salário + (salário * aumento / 100))
```

```
salario=750
aumento=15
print("o valor do salario com desconto é",salario+(salario*aumento/100))
```

Podemos fazer recebendo o valor do salário e aumento

```
salario=float(input("digite o valor do salario: "))
aumento=float(input("digite o valor do aumento:" ))
print("o valor do salario com desconto é",salario+(salario*aumento/100))
```

Tipos Primitivos

- valor=input('digite alguma coisa: ')
- print('o tipo primitivo é ?', type(valor))
- print('tem espaço:?', valor.isspace()) #verificar se tem espaço
- print('é um numero:?', valor.isnumeric()) # verificar se é numerico
- print('é alfabetico:?', valor.isalpha()) # verificar se é alfabetico
- print('é alfanumero:?', valor.isalnum()) # verificar se é alfanumerico
- print('está em maiucula:?', valor.isupper()) # verificar se é maiuscula
- print('está em minuscula:?', valor.islower()) # verificar se é numerico
- print('Aprimeira letra é Maiúscula:?', valor.istitle()) # verificar se é primeira maiuscula
- # type: ignore

Strings no Python

Posição de uma string

l	u	@	g	m	a	i	l	.	c	o	m
0	1	2	3	4	5	6	7	8	9	10	11

```
print(len('lu@gmail.com'))
12
texto='lu@gmail.com'
print(len(texto))#utilizando variável, contar caracteres
12
texto='lu@gmail.com‘ #a partir da 2ª posiçao
print(texto[2])
@
texto='lu@gmail.com‘ #comando find,indica posiçao
print(texto.find('@'))
```

Strings no Python

```
texto='lu@gmail.com‘ #apartir do 3º caracteres exibir todos
texto='lu@gmail.com'
print(texto[3:])#também podemos extrair tudo apartir do 3º
                [3:9]#apartir do 3º 9 caracteres
                [:4]#tudo até o 4º


texto='lucianaoliveria@gmail.com'
posicao=texto.find('@')#desta forma ele posso inserir um email e ele pega apartir do arroba
print(texto[posicao+1:])
```

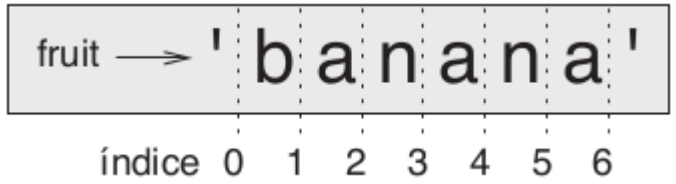
Fatiamento de Strings Python

- Python é uma linguagem de programação poderosa que oferece uma variedade de maneiras de manipular estruturas de dados. Um desses recursos é o fatiamento (também conhecido pelo seu termo em inglês, *slicing*), que permite extrair partes de uma sequência como uma lista, tupla ou string. Entender como o fatiamento funciona pode aumentar significativamente sua eficiência na codificação. Vamos entender a mecânica da notação de fatiamento em Python.

Um segmento de uma string é chamado de fatia. Selecionar uma fatia é como selecionar um caractere:

```
>>> s = 'Monty Python'
>>> s[0:5]
'Monty'
>>> s[6:12]
'Python'
```

O operador `[n:m]` retorna a parte da string do "enésimo" caractere ao "emésimo" caractere, incluindo o primeiro, mas excluindo o último. Este comportamento é contraintuitivo, porém pode ajudar a imaginar os índices que indicam a parte entre os caracteres, como na Figura



Se você omitir o primeiro índice (antes dos dois pontos), a fatia começa no início da string. Se omitir o segundo índice, a fatia vai ao fim da string:

```
>>> fruit = 'banana'
>>> fruit[:3]
'ban'
>>> fruit[3:]
'ana'
```

Se o primeiro índice for maior ou igual ao segundo, o resultado é uma string vazia, representada por duas aspas:

```
>>> fruit = 'banana'
>>> fruit[3:3]
''
```

Uma string vazia não contém nenhum caractere e tem o comprimento 0, fora isso, é igual a qualquer outra string

Estruturas Condicionais

Quando programamos, muitas vezes precisamos que determinado bloco de código seja executado apenas se uma determinada condição for verdadeira. Em casos assim, devemos fazer uso de uma estrutura de condição. É ela que faz com que seu Algoritmo decida qual caminho seguir.

Estruturas Condicionais

- Linguagem Python;
  - Introdução;
  - Estruturas Condicionais
    - if
    - if-else
    - If-elif-else

Estruturas Condicionais

IF

O **if** é uma estrutura de condição que permite avaliar uma expressão e, de acordo com seu resultado, executar uma determinada ação.

```
1 idade = 18
2 if idade < 20:
3    print('Você é jovem!')
```

Um pouco mais da linguagem

Vamos construir um programa que recebe do teclado o nome de um dos animes da lista a baixo e mostre quantas temporadas existem:

#If Composto

Naruto → 9 temporadas

The Chosen→ 4 temporadas

O Agente Noturno → 7 temporadas

Resolução

```
1  anime = input("Digite um dos animes: Naruto, One Piece ou Inuyasha ")
2
3  if anime=="Naruto":
4      print(anime,"possui 9 temporadas")
5  if anime=="One Piece":
6      print(anime,"possui 20 temporadas")
7  if anime=="Inuyasha":
8      print(anime,"possui 7 temporadas")
```

Python - **Estruturas Condicionais**

IF-ELSE

O **if-else** é uma estrutura de condição que permite 2 caminhos. 1º caminho é caso o resultado seja **verdadeiro**, e o 2º caminho é caso o resultado seja **falso**.

```
1 idade = 18
2 if idade >= 18:
3       print('maior de idade')
4 else:
5       print('menor de idade')
```

Dessa vez, caso a condição avaliada na linha 3 não seja atendida, definimos o fluxo alternativo que o código deve seguir. Ou seja, se a idade não for maior ou igual a 18, o bloco abaixo da palavra reservada **else** deverá ser executado. Nesse caso, temos apenas uma instrução de impressão (linha 5).

Python - **Estruturas Condicionais**

IF-ELIF-ELSE

Adicionalmente, se existir mais de uma condição alternativa que precisa ser verificada, devemos utilizar o **elif** para avaliar as expressões intermediárias antes de usar o **else**, da seguinte forma:

```
1 idade = 18
2 if idade < 12:
3                 print('criança')
4 elif idade < 18:
5                 print('adolescente')
6 elif idade < 60:
7                 print(adulto')
8 else:
9                 print('idoso')
```

Na linha 2 definimos a primeira condição (idade < 12). Caso essa não seja atendida, o programa seguirá para a linha 4 e avaliará a próxima condição (elif), que se for verdadeira fará com que o bloco logo abaixo (a linha 5, nesse caso) seja executado.

Caso essa condição ainda não seja atendida (elif), há uma outra alternativa na linha 6 que será avaliada e que fará com que o bloco logo abaixo seja executado se ela for atendida. Por fim, se nenhuma das condições for satisfeita, o programa seguirá para a linha 8, executando o que é definido pelo else.

Um pouco mais da linguagem

Vamos construir um Programa em Python que o usuário possa entrar pelo teclado com o saldo de uma aplicação bancária e escolher a forma de aplicação:

- **Poupança:** imprime (escreva no console) o novo saldo em 1 mês, considerando o reajuste de 0.5 % ao mês.
- **CDB:** imprime (escreva no console) o novo saldo em 1 mês, considerando o reajuste de 1 % ao mês.

Obs: desconsiderar os tributos pegos pelo cdb.

Resolução

```
1  valor = float(input("Digite o valor investido: "))
2  formaA = input("Digite a forma de aplicação(Poupança ou CDB): ")
3
4  if formaA=="Poupança":
5      print("O novo saldo após 1 mês de aplicação é R$",(valor*1.005))
6  elif formaA=="CDB":
7      print("O novo saldo após 1 mês de aplicação é R$",(valor*1.01))
8  else:
9      print("Você digitou uma aplicação inválida")
```