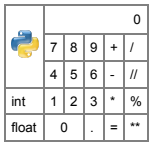
MC102 - Algoritmos e Programação de Computadores



Minicalculadora

Nesta tarefa, vamos reforçar os conceitos vistos na Tarefa de Laboratório 01 e ampliar o estudo de operações com objetos do tipo `int` e `float`.

Várias operações com `int` e `float`

Os tipos `int` e `float` permitem operações de soma, subtração, multiplicação, divisão, divisão inteira, cálculo do resto da divisão inteira e exponenciação. Veja alguns exemplos na Python Shell e a tabela que resume as possibilidades:

```
>>> 10 / 3              # divisão
3.3333333333333335
>>> 10 // 3             # divisão inteira
3
>>> 10 % 3              # resto da divisão de 10 por 3
1
>>> 10**3               # exponenciação com expoente inteiro
1000
>>> 10**2.9             # exponenciação com expoente real
794.3282347242813
```

Operando 1	Operador	Operando 2	Resultado
int	+ - * // % **	int	int
int	/	int	float
int	+ - * / // % **	float	float
float	+ - * / // % **	int	float
float	+ - * / // % **	float	float

Implementando a minicalculadora

Você deverá implementar um programa em Python que pode fazer várias operações com dois números. Um operando contendo o caractere `.` será considerado do tipo `float`, caso contrário será considerado do tipo `int`. O operador poderá ser um símbolo do conjunto `+, -, *, /, //, % ou **`, como esquematizado a seguir:

```
<operando_1>
<operador>
<operando_2>
```

Conforme o tipos dos operandos, o resultado será um `float` formatado com exatamente duas casas decimais ou um `int`.

Testes para o SuSy

No SuSy, para cada tarefa, criamos um conjunto de testes com arquivos de entrada `arq<i>.in` e para cada um deles temos uma saída esperada `arq<i>.res`. Para esta tarefa, os testes abertos estão listados na tabela abaixo. Note que se a entrada indicar uma divisão por zero, a operação não deve ser executada e a string `"Erro."` deve ser impressa.

Operador	Arquivos de entrada e de resultado				Operador	Arquivos de entrada e de resultado				Operador	Arquivos de entrada e de resu			
+	arq1.in	arq2.in	arq3.in	arq4.in	-	arq5.in	arq6.in	arq7.in	arq8.in	*	arq9.in	arq10.in	arq11.in	
	10 + 7	12 + 4.27	3.14 + 3	9.9873 + 3.3571		10 - 11	100 - 0.1	100.01 - 99	100.7569 - 100.0		3 * 7	5 * 2.1	45.2 * 0	0 * 0
	arq1.res	arq2.res	arq3.res	arq4.res		arq5.res	arq6.res	arq7.res	arq8.res		arq9.res	arq10.res	arq11.res	a
	17	16.27	6.14	13.34		-1	99.90	1.01	0.76		21	10.50	0.00	0
/	arq13.in	arq14.in	arq15.in	arq16.in	//	arq17.in	arq18.in	arq19.in	arq20.in	%	arq21.in	arq22.in	arq23.in	
	10 / 3	12.56 / 3	17 / 2.5	0.5 / 1.5		10 // 3	12.56 // 3	17 // 2.5	0.5 // 1.5		10 % 3	12.56 % 3	17 % 2.5	0 % 1
	arq13.res	arq14.res	arq15.res	arq16.res		arq17.res	arq18.res	arq19.res	arq20.res		arq21.res	arq22.res	arq23.res	a
	3.33	4.19	6.80	0.33		3	4.00	6.00	0.00		1	0.56	2.00	0
**	arq25.in	arq26.in	arq27.in	arq28.in	/	arq29.in	arq30.in	arq31.in	arq32.in	// ou %	arq33.in	arq34.in	arq35.in	

	3 ** 2	3 ** 2.1	3.1 ** 2	3.1 ** 2.1	zero	5 / 0	5 / 0.0	5.0 / 0	5.0 / 0.0	zero	5 // 0	5.0 % 0	5 // 0.0	6 % 0
	arq25.res	arq26.res	arq27.res	arq28.res		arq29.res	arq30.res	arq31.res	arq32.res		arq33.res	arq34.res	arq35.res	a
	9	10.05	9.61	10.76		Erro.	Erro.	Erro.	Erro.		Erro.	Erro.	Erro.	E

Dado um dos arquivos `arq<i>.in`, você pode utilizá-lo como entrada na linha de comando da seguinte forma:

```
$ python3 main.py < arq1.in
8
```

Você pode redirecionar a saída para um arquivo `arq<i>.out` da seguinte maneira:

```
$ python3 main.py < arq1.in > arq1.out
```

O SuSy, por padrão, compara a saída do seu programa com o arquivo `arq<i>.res` utilizando o comando `diff`:

```
$ diff arq1.res arq1.out
```

Uma saída vazia indicará que o programa executou corretamente no seu computador. Caso contrário, observe as diferenças apontadas e depure seu programa.

Para esta tarefa estão disponíveis dois scripts Python para ajudar a fazer todos os testes abertos, sendo um para ambiente GNU/Linux e o outro para ambiente Windows. Os scripts supõem que o arquivo `main.py`, os arquivos de entrada `arq<i>.in` e os arquivos com resultado `arq<i>.res` estão todos no mesmo diretório.

```
$ python3 executa_testes.py
```

```
> py executa-testes-windows.py
```

Esta tarefa tem mais quatro testes fechados, que são variações de alguns dos testes já apresentados.

Dicas de Python 3 para esta tarefa

- Reveja as dicas da Tarefa de Laboratório 01.
- Como não estamos fazendo tratamento de erros nos números da entrada, para ver se a entrada é um inteiro, podemos fazer o seguinte teste simplificado:

```
str1 = input()
if str1.isdigit() :
  operando = int(str1)
  tipo_operando = "int"
else:
  operando = float(str1)
  tipo_operando = "float"
```

Orientações para submissão

Veja aqui a página de submissão da tarefa. Lembre-se que o arquivo a ser submetido deve se chamar **`main.py`**. No link Arquivos auxiliares há um arquivo `arqs.zip` que contém todos os arquivos de testes abertos e seus respectivos resultados compactados. Os arquivos `executa-testes.py` e `executa-testes-windows.py` também estão neste pacote.

Todos os alunos matriculados até 16 de março estão inscritos. Se você entrou depois e/ou não estiver cadastrado corretamente, envie email para islene@ic.unicamp.br .

Observe o limite máximo de 20 submissões e que a nota final é proporcional ao número de testes que executaram corretamente.

O peso desta tarefa é 3.

O prazo final para submissão é 08/04/2018.
