Python 3 Cheat Sheet

Tipos Básicos

Tipo	Descrição	Exemplo
boolean	Valor Lógico	*True*, *False*
integer	Inteiro	1, 20, -45
float	Real	0.5, 1.5, -7.667
string	Texto	"*Python é Legal*"
list	Lista	[1, 2, "oi", -12]
dictionary	dicionário	{"um": 1, "dois": 2}
tuple	tupla	(1, 2, "a", "teste")
set	conjunto	{1, 2, 9, 1 "a", 0}

tuple: lista de elementos *imutáveis*

set: lista de elementos *únicos*

Operações Aritméticas

Operação	Descrição	Exemplo
+	Adição	5 + 3 => 8
-	Subtração	5 - 3 => 2
*****	Multiplicação	5 * 3 => 15
/	Divisão	5 / 3 => 1.667
******	Exponenciação	5 ** 3 => 125
%	Módulo/Resto	5 % 3 => 2
//	Divisão Inteira	5 // 2 => 1

Operadores Lógicos

and	*True* **and** *False* => *False*
or	*True* **or** *False* => *True*
not	**not** *True* => *False*

Caracteres especiais

comentário no código

\n nova linha

\t tabulação

""" *comentário em multiplas linhas* **"""**

Operadores de Comparação

Op.	Descrição	Exemplo
==	igualdade	2 **==** 3 => *False*
!=	diferença	2 **!=** 3 => *True*
>	maior que	2 **>** 3 => *False*
<	menor que	2 **<** 3 => *True*
>=	maior igual que	2 **>=** 3 => *False*
<=	menor igual que	2 **<=** 3 => *True*

Funções Embutidas (built-in functions)

print(s, sep='y') => imprime **s**
input(s) => retorna o valor digitado
len(x) => retorna o tamanho de **x**
str(x) => converte **x** pro tipo **str**
list(x) => convert **x** pro tipo **list**
int(x) => converte **x** pro tipo **int**
float(x) => converte **x** pro tipo **float**
type(x) => retorna o tipo de **x**
min(L) => menor valor em **x**
max(L) => maior valor em **x**
range(n1, n2, n) => retorna uma sequência de números de **n1** até **n2** em **n** passos.

x, y para qualquer tipo de dado, **s** para *string*, **n** para número, L para ***lista*** onde **i**,**j** são os índices da lista

Estrutura de Decisão/Condicional

```
if <condicao> :
    <codigo>
else if <condicao> :
    <codigo>
...
else:
    <codigo>

if <valor> in <lista>:
    <codigo>
```

```
x = int(input('numero: '))
if x == 0:
    print("Zero")
elif x >= 1:
    print("Positivo")
else:
    print("Negativo")

if 2 in [1, 2, 3, 4, 5]:
    print("2 está na lista")
```

Palavras Reservadas

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Tratamento de Exceções

```
try:
    <codigo>
except<erro>:
    <codigo>
else:
    <codigo>
```

Módulos

```
import modulo
modulo.funcao()

from modulo import *
funcao()
```

Funções

```
def <funcao>(<parametro>):
    <codigo>
    return <dado>
```

```
def somar(n1, n2):
    n3 = n1 + n2
    return n3
print(somar(2, 3)) # 5
```

Estrutura de Repetição / Laço (loop)

```
while <condicao>:
    <codigo>

for <variavel> in <lista>:
    <codigo>
```

```
while True:
    print("loop infinito")

lista = [1,2,3,4]
for numero in lista:
    print(numero)
```

```
for <variavel> in range(inicio, final, passo):
    <codigo>
```

```
for n in range(0, 10, 2):
    print(n)
```

```
for chave, valor in dict.items():
    <codigo>
```

```
dicionario = {'um': 1, 'dois': 2}
for chave, valor in dicionario.items():
    print(chave, " => ", valor)
```

Comandos de controle do loop

break finaliza o loop/laço
continue pula para próxima iteração
pass não faz nada

Índices e *slice notation*

índice negativo =	*-5*	*-4*	*-3*	*-2*	*-1*
índice positivo =	*0*	*1*	*2*	*3*	*4*

lista = [10, 39, "*Brasil*", 7.47, 5]

```
lista[0] -> 10
lista[-5] -> 10
lista[-1] -> 5
lista[:] -> [10, 39, "Brasil", 7.47, 5]
lista[::2] -> [10, "Brasil", 5]
lista[1:-1] -> [39, "Brasil", 7.47]
lista[:-1] -> [10, 39, "Brasil", 7,47]
lista[1:3] -> [39, "Brasil"]
lista[-3:-1] -> ["Brasil", 7,47]
lista[:3] -> [10, 39, "Brasil"]
lista[4:] -> [10, 39, "Brasil", 7.47]
lista[::-1] -> [5, 7.47, "Brasil", 39, 10]
```

para list, str, tuple...

Range e *generator*

padrão 0 — Não incluso

range([inicio,]final [,passo])

range(5) -> [0, 1, 2, 3, 4]
range(3, 8) -> [3, 4, 5, 6, 7]
range(2, 12, 3) -> [2, 5, 8, 11]

range() retorna um objeto ***generator*** para converter para uma lista faça **list**(**range**(n))

```
print(list(range(4)))
> [0, 1, 2, 3]
```