Introdução a python
Aula 02

Carlos Walter

Introdução a Python - Aula 02

list comprehension

Funções

Modularização

Exceções

Classes

Dica Do Dia

```
ord(character)
```

List comprehension

List comprehension

```
new_list = [expression for_loop_one_or_more condtions]
```

List comprehension

```
new_list = [expression for_loop_one_or_more condtions]

list = [1, 2, 3, 4]

squares = [x**2 for i in list]
```

List comprehension

```
new_list = [expression for_loop_one_or_more condtions]

list1 = [1, 2, 3, 4]

squares = [x**2 for i in list1]

list2 = [3, 4, 5, 6]

contains = [a for a in list for b in list2 if a == 2]
```

Funções

Funções

```
str(21)

float('15)

type(True)
```

Funções

```
str(21)

float(‘15)

type(True)

import math

math.sin(angulo)

math.log10(17)
```

Funções - Composição

```
import math

math.sin(math.log10(17))
```

Funções - definição

```
def nome_da_função(Parametros):

    Comandos
```

Funções - definição

```
def nome_da_função(Parametros):
    Comandos
def areaDoCirculo(raio):
    return math.pi*(raio**2)
```

Funções - Visibilidade

```
def sayMyName():
    name = ‘BOB’
    i = 2
    print(name*i)
name = “Martin”
i = 3
```

Funções - Visibilidade

```
def sayMyName():
    name = ‘BOB’
    #i = 2
    print(name*i)
name = “Martin”
i = 3
```

Funções - parâmetros

```
def sum(a, b, c):

    return a+b+c
```

Funções - parâmetros

```
def sum(a, b, c):
    return a+b+c

def sum(a, b=5, c):
    return a+b+c
```

Funções - parâmetros

```
def sum(a, b, c):

    return a+b+c

sum(1, 2, 3)
```

Funções - parâmetros

```
def sum(a, b, c):

    return a+b+c

sum(1, 2, 3)

sum(a=1, c=6, b=7)
```

Funções - parâmetros

```
def sum(a, b, c):

    return a+b+c

l1 = [1, 3, 4]

d1 = {‘a’: 1, ‘b’: 2, ‘c’: 3}
```

Funções - parâmetros

```
def sum(a, b, c):

    return a+b+c

l1 = [1, 3, 4]

d1 = {‘a’: 1, ‘b’: 2, ‘c’: 3}

sum(*l1)

sum(**d1)
```

Funções - apelidos

```
def sum(a, b, c):
    return a+b+c

i = sum
```

Funções - apelidos

```
def sum(a, b, c):
    return a+b+c

i = sum

i(1, 2, 3)
```

Funções - Recursividade

Funções - Recursividade

```
def fatorial(a):

    if a <= 1: return 1

    return a*fatorial(a-1)
```

Funções - Tipos

Python não dá a mínima para tipos de parâmetros e retornos

Funções - lambda 

Funções - lambda

```
lambda 1, … parametro_n: expressão
```

Modularização

Modularização

scripts

Modularização

scripts

import de códigos

Exceções

Exceções

```
while True:
    try:
        print(“Digite um número”)
        n = int(input())
    except:
        print(“Numero invalido, tente novamente”)
print(“:)”)
```

Classes

Classes

```
class NomeDaClasse:

    definição da classe

    atributos

    métodos
```

Classes

```
class Circulo:
```

Classes

```
class Circulo:

    x = 15

    y = 15

    raio = 5
```

Classes

```
class Circulo:

    x = 15

    y = 15

    raio = 5

    def area(self):

        return math.pi*self.raio
```

Classes

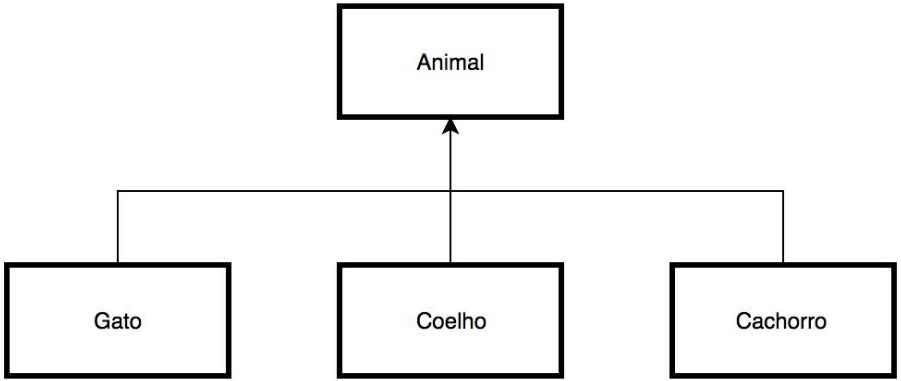
```
class Circulo:

    def __init__(self, x, y, raio):

        self.x = x

        self.y = y

        self.raio = raio

    def area(self):

        return math.pi*self.raio
```

Classes Documentação (__doc__)

```
class Circulo:

'''Representação de um círculo no plano cartesiano'''

    def __init__(self, x, y, raio):

        self.x = x

        self.y = y

        self.raio = raio

    def area(self):

        return math.pi*self.raio
```

Classes Herança

Classes Herança



Classes Herança

```
class FormaGeometrica:

    def __init__(self, x, y, nl, tl):

        self.x = x

        sel.y = y

        self.nl = nl

        self.tl = tl
```

Classes Herança

```
class Quadrado(FormaGeometrica):

    def __init__(self, x, y, nl, tl):

        super().__init__(x, y, nl, tl)

    def area(self):
```