Micro langage

Micro language en python avec ply

GitHub repo

Fonctionnalités

Calcul de base, print

```
cmd > print((2+6)*3);
24
```

Print d'une chaine de caractère

```
cmd > print("ma jolie string");
ma jolie string
```

Print concaténation

```
cmd > a = 10;
cmd > print("a=",a);
a= 10

cmd > print("1+2=",1+2);
1+2= 3
```

Opérations booléennes

```
cmd > print(3+6 > 9);
False

cmd > print( (3+6>9) | (2-2==0));
True
```

Affectation, print

```
cmd > x = 4; x=x+3; print(x);
7
```

Affectation élargie

```
cmd > x = 4; x++; print(x);
5

cmd > x = 66; x--; print(x);
65

cmd > x = 100; x+=3; x-=5; print(x);
98
```

Condition

```
a = 10;
if a > 9 then {
print(a);
}
if  a > 10 then {
print("never");
}

10
```

While, For

```
cmd > x=4; while x > 0 { print(x); x--;}
4
3
2
1

cmd > for (i=0; i < 10; i+=2;) { print(i); }
```

```
0
2
4
6
8
```

Fonction void avec et sans paramètres

```
function fibo() {
    n=10;
    first=0;
    second=1;
    while n > 0 {
        tmp=first+second;
        first=second;
        second=tmp;
        print(first);
        n--;
    }
}
fibo();

1
1
2
3
5
8
13
21
34
55

function fibo(n) {
    first=0;
    second=1;
    while n > 0 {
        tmp=first+second;
        first=second;
```

```
        second=tmp;
        print(first);
        n--;
    }
}
a=5;
fibo(a);

1
1
2
3
5
```

Fonction récursive et scope des variables

```
function rec(n) {
    if n > 0 then {
        n--;
        rec(n);
        print(n);
    }
}
a=5;
rec(a);
print("----");
print(a);

0
1
2
3
4
----
5
```

Retour de fonction

```
function factorial(a) {
    if a > 1 then {
        return a * factorial(a-1);
    }
    return a;
}
print(factorial(10));

3628800
```

Tableaux

Affectation d'un tableau

Un tableau peut être affecté à une variable et peut contenir plusieurs expressions différentes :

```
cmd > array = ["coucou", "toi", 1, 2, 3+3];
cmd > print(array);

coucou toi 1 2 6
```

Taille d'un tableau

On peut récupérer la taille d'un tableau

```
cmd > print(len([1,2,3,4]));

4

cmd > array = [1,2];
cmd > print(len(array));

2
```

Accès à une case du tableau

On peut accéder à n'importe quel case du tableau pour en récupérer l'expression. Si l'index est hors de portée, une exception est levée.

```
cmd > array = ["coucou", "toi", 1, 2, 3+3];
cmd > print(array[1]);
```

```
toi
```

Concaténation dans le tableau

Il est possible de rajouter de l'élément dans un tableau existant de la manière suivante :

```
cmd > array = [ 1, 2, 3 ];
cmd > array <- 4;
cmd > array <- 4 + 1;
cmd > print(array);

1 2 3 4 5
```

Chargement de fichiers de code au démarrage

Possibilité de charger des fichiers au lancement pouvant contenir des déclarations de fonctions ou même des instructions.

Fichier *fibo* :

```
function fibo(n) {
    first=0;
    second=1;
    while n > 0 {
        tmp=first+second;
        first=second;
        second=tmp;
        print(first);
        n--;
    }
}
```

Fichier *factorial* :

```
function factorial(a) {
    if a > 1 then {
        return a * factorial(a-1);
    }
    return a;
}
```

Exécution du programme

```
python main.py fibo factorial


cmd > fibo(10);


1
1
2
3
5
8
13
21
34
55
```

Chargement de fichiers de code pendant l'exécution

Possibilité de charger des fichiers pendant l'exécution pouvant contenir des déclarations de fonctions ou même des instructions.

Fichier *fibo* :

```
function fibo(n) {
    first=0;
    second=1;
    while n > 0 {
        tmp=first+second;
        first=second;
        second=tmp;
        print(first);
        n--;
    }
}
```

Exécution du programme

```
python main.py
```

```
cmd > load("fibo");
cmd > fibo(10);

1
1
2
3
5
8
13
21
34
55
```

Revision #1
Created 9 May 2022 21:50:15 by Noé Larrieu-Lacoste
Updated 9 May 2022 21:50:43 by Noé Larrieu-Lacoste