Compte rendu TP1 de TLI :
Programme en ligne de commandes

Ligne de commande

1/ Les distances sémantiques à l'exécution et à l'interprétation sont respectivement l'effort a fournir pour comprendre comment lancer le programme et comment interpréter les résultats. Comme on ne sais ni comment se servir du programme (appels corrects ou incorrects) ni à quoi correspondent les résultats, on peut dire que la distance est infinie. Cependant lorsque l'on connaît la sortie du programme (dans le cas d'une erreur et d'une entrée correcte), le résultat est compréhensible sans être pour autant très explicite .

2 & 3/

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys
from math import *


def trace(function, xmin, xmax, nstep, output):
        output.write("x, %s\n" % function)
        function = eval("lambda x:" + function)

        step = 1.*(xmax-xmin)/nstep
        for i in range(nstep+1):
                x = xmin + i*step
                try:
```

```
                y = function(x)
            except:
                print("Veuillez entrer une fonction mathématique existante")
                break
            output.write("%s, %s\n" % (x, y))


def main(argv=None):
    if argv is None:
        argv = sys.argv

    import getopt
    try:
        options, argv = getopt.getopt(argv[1:], "o:", ["output="])
    except getopt.GetoptError as message:
        sys.stderr.write("%s\n" % message)
        sys.exit(1)

    if len(argv) == 1 :
        function = argv[0]
        output = sys.stdout
        xmin, xmax = 0., 1.
        nstep = 10
    elif len(argv) == 4 :
        try:
            function = argv[0]
            xmin = float(argv[1])
            xmax = float(argv[2])
            nstep = int(argv[3])
            output = sys.stdout
        except ValueError as message:
            sys.stderr.write("%s\n" % message)
            print("Rappel : \"fonction\" float float int")
            sys.exit(1)
    else :
        print("Erreur lors de l'utilisation de la fonction !!")
        print("Veuillez entrer un argument valide : une fonction mathématique valide entre guillemets.")
        print("Exemple d'utilisation : ./trace.py \"sin(x)\" ")
        print("Vous pouvez aussi entrer la valeur de départ, la valeur maximale à atteindre et le nombre d'etapes :")
        print("Exemple d'utilisation : ./trace.py \"sin(x)\" 0.0 1.0 10 ")
        print("il est possible de spécifier un fichier de sortie: ./trace.py [-o | --output, fichier] \"sin(x)\" [arguments] ")
        sys.exit(1)

    for option, value in options:
        if option in ["-o", "--output"]:
            print("Ecriture realisée dans le fichier : %s\n" % value)
            output = file(value, "w")
```

```
            else:
                assert False, "unhandled option"

    trace(function, xmin, xmax, nstep, output)


if __name__ == "__main__":
    sys.exit(main())
```

Langage graphique 2D

4/ Afin de réduire la distance d'évaluation, nous affichons la courbe obtenue en rouge et au format paysage.
Afin d'identifier la fonction, le nom de celle-ci est inscrit en dessous du graphique.
La courbe est, de plus, affichée en rouge afin de la différencier du repère et nous affichons l'échelle sur les axes afin de mieux comprendre la courbe obtenue.