Leon Zhang  
llz210000  
CS 4337.501

Homework 1

1. Python objects and classes using functions.

def function\_class(starting\_var: int):

var = starting\_var

print(var)

def add(increment):

nonlocal var

var += increment

print(var)

double()

return var

def double():

nonlocal var

var \*= 2

print(var)

double()

return add

obj = function\_class(3)

obj(9)

obj(3)

obj(-5)

2. Implementing strings using dictionaries and integers

def int\_to\_char(ascii\_val):

ASCII\_table = {

0: ' ', 1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f', 7: 'g', 8: 'h', 9: 'i', 10: 'j', 11: 'k', 12: 'l', 13: 'm',

14: 'n', 15: 'o', 16: 'p', 17: 'q', 18: 'r', 19: 's', 20: 't', 21: 'u', 22: 'v', 23: 'w', 24: 'x', 25: 'y', 26: 'z'

}

return ASCII\_table[ascii\_val]

def string\_builder():

string = []

string.append(int\_to\_char(8))

string.append(int\_to\_char(5))

string.append(int\_to\_char(12))

string.append(int\_to\_char(12))

string.append(int\_to\_char(15))

string.append(int\_to\_char(0))

string.append(int\_to\_char(23))

string.append(int\_to\_char(15))

string.append(int\_to\_char(18))

string.append(int\_to\_char(12))

string.append(int\_to\_char(4))

string\_printer(string)

def string\_printer(string):

for x in string:

print(x, end = '')

string\_builder()

3. Object-Orientated Design – each user’s account will be an object of a class called “Users”. This is so that data such as their id, account balance, transaction history, personal info, etc. can be easily stored and accessed when needed. Each user will have the same type of data stored, so an individual user will just be a user object. Withdrawing and depositing will also be done within the class as it easy to modify data of an object’s attributes. Simple modifier functions such as:

void deposit(double amount){

balance += amount;

}

Are easy and intuitive to implement using OOP design.

Procedural – procedural design will be used for app functionality and logic – user interactions. For users to have a good experience with the app, things must happen in the right order and when the user wants it to happen. For example, login shouldn’t happen unless username and password match correctly, withdraw/deposit shouldn’t happen until users press a button/confirm, etc.

4. One of the ways implementing a programming language is done using a compiler. This takes the source code and runs it through a series of linear steps until it can be executed. The first steps are through the lexical analyzer, which tokenizes all the words, numbers, and symbols and sends it to the syntax analyzer, which makes sure things are in the right order. Then, everything is converted to intermediate then machine code, an object file is made, all the files are linked, and an executable file is created. Essentially, the compiler breaks down the code into code that a machine can understand and execute.

Another implementation is a pure interpreter, which executes code line by line from the source code. While it still uses a lexical and syntax analyzer, these work line by line instead of doing the whole file before moving on like in the compiler. Unlike compilers which have to work before a file can be run, interpreters are dynamic and work during runtime.

The last type of implementation is a hybrid between the two, where source code goes through the usual, linear path from lexical analyzer, syntax analyzer, and lastly intermediate language code, where it is then interpreted as its intermediate code. Hybrid implementation in general is faster than a pure interpreter because interpreting the intermediate code is simpler.

Compilers are generally used when maintaining performance is a key aspect of your language. They are usually the fastest languages because the compiler can make a lot of optimizations because it’s not restrained by having to do everything at runtime. Additionally, due to their more static nature, compiled languages are great at optimization for specific tasks and platforms. Interpreters are used when performance is not that big a factor, but other things like dynamic behavior, prototyping, or scripting are highlighted features. Hybrid is used when you need better performance than a pure interpreter and also more platform mobility than compilers.

5. A paper with writing on it

Description automatically generated

6.

A piece of paper with writing on it

Description automatically generated

7. if ≡ λx.λy.λz.xyz

This statement is an if statement where x is the conditional and if it is true, then following the logic that true represents taking the first option and false represents taking the second option, and it will evaluate to y. Therefore, if the condition x is false, then it will choose the else path and evaluate to z.

Given that x = t, (λx.λy.λz.xyz) T y z

= (λy.λz.Tyz) y z < perform beta reduction

= (λz.Tyz) z <perform beta reduction

= Tyz

= y