Question 4

Having a second chain link for the static grandparent would increase subprogram linkage time, but it’ll allow the subprograms to reach grandparent references/variables faster.

Question 6

•Are local variables statically or dynamically allocated?

Python local subprogram variables are stack dynamically allocated

a = “hello”

b = 3

both variables are stack dynamic

•Can subprogram definitions appear in other subprogram definitions?

Yes, Python supports subprogram nesting

def a():

def b():

•What parameter-passing method or methods are used?

Python uses pass-by-assignment where all data values are objects, where the actual is assigned to the formal. If you pass a simple object, like an int, it does it as pass by value, but something like a list will be pass by reference.

>>> def a(x):

... x = x+1

>>> a(3)

>>> test = 3

>>> a(3)

>>> test

3

>>> def a(x):

... x[0]=x[0]+1

>>> test = [3]

>>> a(test)

>>> test

[4]

>>>

•Are the types of the actual parameters checked against the types of the formal parameters?

Python in general does type checking at execution time. However for parameters, you can add type hints and return types. However, these types are not enforced, a static type checker is required for that.

def headline(text: str, align: bool = True) -> str:

This states that the type of text is str, aligh is a boolean with default value of True, and return type is str

•If subprograms can be passed as parameters and subprograms can be nested, what is the referencing environment of a passed subprogram?

In python, if a subprogram is passed as a parameter, then its referencing environment is within the subprogram that it was passed to.

Def outter(x):

print(x)

def inner():

def test(x):

return x

outter(test(x))

•Can subprograms be overloaded?

Python does not directly support overloading subprograms, the most recent instance will be the one called. However there’s a pseudo implementation where you can give a subprogram default parameters, and only use specific parameters, but the limitation is where you have to do it left to right.

def temp(input, append = “test”):

…

ob = temp()

ob(“foo”)

ob(“foo”, “bar”)

•Can subprograms be generic?

Python does not explicitly support generics, however parameters without typing are “generic” to the overarching object type. You can pass any object and it’ll run until it reaches an error.

>>> def p(x):

... print(x)

...

>>> p(3)

3

>>> p("foo")

foo

>>> test = (3,4,5)

>>> p(test)

(3, 4, 5)

>>>

•If the language allows nested subprograms, are closures supported

Yes, python support closures.

Def outer(x):

x = x

def inner():

print(x)

return inner

test = outer(“hello”)

test()