

1. Greek Mythology

Consider the following code for extracting the two names of a Mythological Figure:

```
def mythology_intro(name, realm="earth"):
    _____ (a)
    _____ (b)
    intro_print(name, greek_name, roman_name, realm)

def intro_print(name, get_greek_name_func, get_roman_name_func,
realm="earth"):
    print("Introducing the Great", name)
    print(" Greek name is", get_greek_name_func(name))
    print(" Roman name is", get_roman_name_func(name))
    print(" Rules over", realm)

mythology_intro('Athena Minerva', 'wisdom and war')
mythology_intro('Poseidon Neptune', "sea and waters")\
```

Output from the above code is:

```
Introducing the Great Athena Minerva
Greek name is Athena
Roman name is minerva
Rules over wisdom and war
Introducing the Great Poseidon Neptune
Greek name is Poseidon
Roman name is Neptune
Rules over sea and waters
```

(a) What lines of code could go in blank (a)? choose all options that could work

- `def greek_name(name):`
 `return name.split(' ')[0]`
- `def greek_name(name):`
 `return name.split()[1]`
- `greek_name = lambda full_name : full_name.split()[0]`

```
 greek_name = lambda full_name : full_name.split(" ")[1]
```

(b) What lines of code could go in blank (b)? — choose all options that could work

```
 def roman_name(name):  
    return name.split(' ')[1]
```

```
 def name(name):  
    return name.split(" ")[0]
```

```
 roman_name = lambda full_name : full_name.split(" ")[0]
```

```
 roman_name = lambda full_name : full_name.split(" ")[1]
```

2. Period of a Pendulum

The time for a pendulum to complete a cycle is $T = 2\pi\sqrt{I / mgL}$

I being the inertia of the center of mass

m being the mass

L being the distance between the center of mass and the pivot

g being the acceleration of gravity

```
import math
```

```
EARTH_GRAVITY = 9.81
```

```
MOON_GRAVITY = 1.62
```

```
def period (inertia, mass, length, gravity=____(a)):
```

```
    pi = ____ (b)
```

```
    def division(x, y):
```

```
        return ____ (c)
```

```
    return 2* pi * math.sqrt(division(inertia, (____(d))))
```

```
print(period(2(3**2), 2, 3))
```

(a) What line of code could go in blank (a) assuming we want the Earth's gravity to be the default?

```
 EARTH_GRAVITY
```

```
 MOON_GRAVITY
```

```
 3.141592
```

```
 math.sqrt
```

```
 x/y
```

```
 mass * length * gravity
```

```
 inertia / (mass * length * gravity)
```

(b) What line of code could go in blank (b) ?

- EARTH_GRAVITY
- MOON_GRAVITY
- 3.141592
- math.sqrt
- x/y
- mass * length * gravity
- inertia / (mass * length * gravity)

(c) What line of code could go in blank (c)

- EARTH_GRAVITY
- MOON_GRAVITY
- 3.141592
- math.sqrt
- x/y
- mass * length * gravity
- inertia / (mass * length * gravity)

(d) What line of code could go in blank (d)?

- EARTH_GRAVITY
- MOON_GRAVITY
- 3.141592
- math.sqrt
- x/y
- mass * length * gravity
- inertia / (mass * length * gravity)

3. Let's Draw

```
class Pencil:

    def __init__(self, color, weight):
        self.color = color
        self.weight = weight

    def __str__(self):
        return f'____(a) Pencil with the color {____(b)}
```

(a) What line of code could go in blank (a)?

- Pencil
- self.weight
- self.name
- nextPencil = nextPencil.next
- n += 1
- self.color
- currentPencil = currentPencil.rest
- n -= 1

(b) What line of code could go in blank (b)?

- Pencil
- self.weight
- self.name
- nextPencil = nextPencil.next
- n += 1
- self.color
- currentPencil = currentPencil.rest
- n -= 1

```
class PencilBox:

    def __init__(self, currPencil, nextPencil):
        assert isinstance(type(currPencil), ____ (c)) or type(currPencil) ==
(c)
        assert isinstance(type(nextPencil), ____ (c)) or type(nextPencil) ==
(c)
        self.current = currPencil
```

```

self.ext = nexPencil

def __len__(self):
    n = 1
    nextPencil = self.next
    while isinstance(nextPencil, Pencil):
        __ (d) __
        __ (d) __
    If nextPencil is not None:
        raise TypeError("Length attempted on mixed pencil box")
    return n

```

(c) What line of code could go in both blanks (c) ?

- Pencil
- self.weight
- self.name
- nextPencil = nextPencil.next
- n += 1
- self.color
- currentPencil = currentPencil.rest
- n -= 1

(d) What **TWO** lines of code could go in both blanks (d) ?

- Pencil
- self.weight
- self.name
- nextPencil = nextPencil.next
- n += 1
- self.color
- currentPencil = currentPencil.rest
- n -= 1

(e) Suppose you want to use inheritance to create classes of specific types of pencil. Would the following be a valid Python classes that inherit from the Pencil class?

- ```

class Crayola(Pencil):
 color = 'red'
 weight = 5
 def __init__(self):
 super().__init__(Crayola.color, Crayola.weight)

```

#### 4. Scheme List

Complete the function `interleave`, which takes a two lists `s1` and `s2` as arguments.

`interleave` should return a new list that interleaves the elements of the two lists. (In other words, the resulting list should contain elements alternating between `s1` and `s2`.)

Additionally complete `my-filter`, which takes a predicate `func` and a list `lst`, and returns a new list containing only elements of the list that satisfy the predicate.

```
(define (my-filter func lst)
 (if (_____ (a) _____ lst)
 nil
 (if (func (car lst))
 (cons (_____ (b) _____ lst) (my-filter func (_____ (c) _____
lst)))
 (my-filter func (cdr lst)))
)
)
)
```

```
(define (no-repeats lst)
 (if (null? lst)
 lst
 (cons
 (car lst)
 (no-repeats (_____ (d) _____ (lambda (x) (not (= x (car
lst)))) (cdr lst)))
)
)
)
```

(a) Which of the following would go in blank (a) above?

- `null?`
- `list?`
- `number?`
- `< (car lst) cdr`

(b) Which of the following would go in blank **(b)** above?

- car
- cdr
- null?
- list?

(c) Which of the following would go in blank **(c)** above?

- car
- cdr
- null?
- list?

(d) Which of the following would go in blank **(d)** above?

- null?
- lambda
- my-filter
- def x

## 5. Decode the message

```
"""The list is each letter attached to the corresponding index, a = 0, z = 25"""
```

```
superSecretList = [a, b,... , z]
```

```
def DecodeLetter(n):
 return superSecretList[n]
```

```
def DecodeNumber(n):
 return (n + 3)
```

```
def decodeMessage(message):
 decodedMessage = []
```

```

for i in (a) :
 if i (b) :
 decodedMessage += DecodeLetter[i]
 elif i (c) :
 decodedMessage += DecodeNumber[i]
 else:
 print("Message could not be decoded")
 return
return (d)

```

(a) Which of the following would go in blank **(a)** above?

- message
- decodedMessage
- DecodeLetter(message)
- decodeMessage(myStr)
- superSecretList

(b) Which of the following would go in blank **(b)** above?

- isalpha()
- isdigit()
- Is true
- I is not num
- is\_prime

(c) Which of the following would go in blank **(c)** above?

- isalpha()
- isdigit()
- Is true
- I is not num
- is\_prime

(d) Which of the following would go in blank **(d)** above?

- superSecretList
- True
- decodeMessage(decodedMessage)
- decodedMessage
- i

## 6. Scheme

Implement a procedure `pow` for raising the number `base` to the power of a nonnegative integer `exp` for which the number of operations grows logarithmically, rather than linearly (the number



of recursive calls should be much smaller than the input `exp`). For example, for `(pow 2 32)` should take 5 recursive calls rather than 32 recursive calls. Similarly, `(pow 2 64)` should take 6 recursive calls.

```
((a) (pow base exp)
 (if ((b))
 (c)
 (if (even? (d))
 (pow (square base) (/ exp 2))
 ((e) (pow (square base) (/ (f) 2))))))
```

(a) Which of the following would go in blank **(a)** above?

- define
- lambda
- if
- cond
- Unquote

(b) Which of the following would go in blank **(b)** above?

- (= exp 0)
- (exp = 0)
- (base = 0)
- (= base 0)

(c) Which of the following would go in blank **(c)** above?

- 1
- 0
- pow 0 1
- base
- exp

(d) Which of the following would go in blank **(d)** above?

- 1
- 0
- pow 0 1
- base
- exp

(e) Which of the following would go in blank **(e)** above?

- nothing
- \* base
- + base
- + exp
- \* exp

(f) Which of the following would go in blank **(f)** above?

- (- exp 1)
- exp
- (/ exp 2)
- (- exp base)
- base

## 7. TREES

```
class FamilyTree:

 def __init__(____(a)____):
 self.name = name
 self.mother = None
 self.father = None

 def set_mother(self, branch):
 self.mother = branch

 def set_father(self, branch):
 self.father = branch

def find_names(tree):
 """Lists the names of all the family members """
 if ____ (b) ____:
 return None
 else:
 names = ()
 names += ____ (c) ____
 names += ____ (d) ____

 return ____ (e) ____ + names
```

(a) What expression should go in blank (a) above?

- `self, name=None`
- `self, name, mother=None, father=None`
- `self, name`

(b) What expression should go in blank (b) above? — **choose all options that could work**

- `not tree`
- `tree == None`
- `isinstance(tree, Tree)`

(c) What TWO expressions should go in blank (c) and (d) above?

- `tree.name`
- `find_names(tree.father)`
- `find_names(tree.mother)`
- `tree.father`
- `tree.mother`

(d) What expression should go in blank (e) above?

- `tree.mother`
- `tree.father`
- `tree.name`
- `self.name`

## 8. Regular Expressions

(a) Given the pattern “[code]+23M?[1-5]”, which of the following would be matched?

- `e234`
- `c23M12345`
- `cod23M1`
- `231`

(b) Which of the following patterns would match `!CS111!`, `!*CS111*!`, but not `CS111*?`

- `+\\**?(CS111)\\**?!+`
- `!+[(CS111)*]+!+`
- `![!(CS111)*]+`
- `!?![(CS111)*]+!?`

## 9. BNF

Recall Extended Backus-Naur Form (EBNF) which allows us to create a language. Assume we want a language that describes all possible student ratings.

```
?start: _____ (a)
NUM_RATING: "5" | "4" | "3" | "2" | "1"
TAKE_AGAIN: "Yes" | "No"
feed_back: LETTER | NUMBER | SPACE
LETTER: "a".. "z"
NUMBER: "0".. "9"
SPACE: " "
```

(a) What lines of EBNF code could go in blank (a) ? — **choose all options that could work.**

- NUM\_RATING " Would you take again?" TAKE\_AGAIN " Feedback " feed\_back\*
- NUM\_RATING " Would you take again?" TAKE\_AGAIN " Feedback " feed\_back
- NUM\_RATING " Feedback " feed\_back\*

## 10. Tail-Recursion

**Tail recursion** is defined as a recursive function in which the recursive call is the last statement that is executed by the function. So basically nothing is left to execute after the recursion call.

(a) Which of the following scheme procedures is **tail-recursive**?

- ```
def fact(n):
    if (n == 0):
        return 1
    return n * fact(n-1)
```
- ```
def prints(n):
 if (n < 0):
 return
 print(n-1)
```