



```
1 #Kevin Cabrera Luna
2 #1.1.1.11
3 #1
4 import mint
5 mint.make_money()
6 #####
7 #2
8 from mint import make_money
9 make_money()
10 #####
11 #3
12 from mint import make_money as mint_make_money
13 #####
14 #4
15 from mint import *
16 make_money()
17 #####
18
```



```
1 #Kevin Cabrera Luna
2 #1.2.1.17
3 #1
4 import math
5 result = math.e == math.exp(1)
6 True
7 #####
8 #2
9 #----los valores random seran los mismos
10 #####
11 #3
12 #----procesor()
13 #####
14 #4
15 import platform
16 print (len(platform.python_version_tuple()))
17
```



```
1 #Kevin Cabrera Luna
2 #1.3.1.11
3 #1
4 import sys
5 if __name__ == "__main__":
6     print "!Prohibido!"
7     sys.exit()
8 #####
9 #2
10 import sys
11 sys.path.append ("D:\\Python\\Project\\Modules")
12 #####
13 #3
14 abc
15     |__ def
16         |         |__ mymodule.py
17         |
18         |
19 import abc.def.mymodule
```



resumenes_seccion.py > ...

```
1  #Kevin Cabrera Luna
2  #2.2.1.15
3  #Ejercicio 1
4  #¿Cuál es la longitud de la siguiente cadena asumiendo que no hay espacios en blanco entre las comillas?
5  """
6  """
7  1
8  #Ejercicio 2
9  #¿Cuál es el resultado esperado del siguiente código?
10 s = 'yesteryears'
11 the_list = list(s)
12 print(the_list[3:6])
13 ['t', 'e', 'r']
14 #Ejercicio 3
15 #¿Cuál es el resultado esperado del siguiente código?
16 for ch in "abc":
17     print(chr(ord(ch) + 1), end='')
18 bcd
```

resumenes_seccion.py > ...

```
1  #Kevin Cabrera Luna
2  #2.3.1.17
3  #Ejercicio 1
4  #¿Cuál es el resultado esperado del siguiente código?
5  for ch in "abc123XYX":
6      if ch.isupper():
7          print(ch.lower(), end='')
8      elif ch.islower():
9          print(ch.upper(), end='')
10     else:
11         print(ch, end='')
12 #ABC123xyz
13 #Ejercicio 2
14 #¿Cuál es el resultado esperado del siguiente código?
15 s1 = '¿Dónde están las nevadas de antaño?'
16 s2 = s1.split()
17 print(s2[-2])
18 #de
19 #Ejercicio 3
20 #¿Cuál es el resultado esperado del siguiente código?
21 the_list = ['¿Dónde', 'están', 'las', 'nevadas?']
22 s = '*'.join(the_list)
23 print(s)
24 #¿Dónde*están*las*nevadas?
25 #Ejercicio 4
26 #¿Cuál es el resultado esperado del siguiente código?
27 s = 'Es fácil o imposible'
28 s = s.replace('fácil', 'difícil').replace('im', '')
29 print(s)
30 #Es difícil o posible
```

```
1 #Kevin Cabrera Luna
2 #2.4.1.5
3 #Ejercicio 1
4 #¿Cuál de las siguientes líneas describe una condición verdadera?
5 'smith' > 'Smith'
6 'Smiths' < 'Smith'
7 'Smith' > '1000'
8 '11' < '8'
9 #1, 3 y 4
10 #Ejercicio 2
11 #¿Cuál es el resultado esperado del siguiente código?
12 s1 = '¿Dónde están las nevadas de antaño?'
13 s2 = s1.split()
14 s3 = sorted(s2)
15 print(s3[1])
16 #de
17 #Ejercicio 3
18 #Cuál es el resultado esperado del siguiente código?
19 s1 = '12.8'
20 i = int(s1)
21 s2 = str(i)
22 f = float(s2)
23 print(s1 == s2)
24 #El código genera una excepción ValueError
```


 resumenes_seccion.py

```
1  #Kevin Cabrera Luna
2  #2.6.1.12
3  #Ejercicio 1
4  #¿Cuál es el resultado esperado del siguiente código?
5  try:
6      print("Tratemos de hacer esto")
7      print("#"[2])
8      print("¡Tuvimos éxito!")
9  except:
10     print("Hemos fallado")
11     print("Hemos terminado")
12     #Tratemos de hacer esto
13     #Hemos fallado
14     #Hemos terminado
15     #Ejercicio 2
16     #¿Cuál es el resultado esperado del siguiente código?
17     try:
18         print("alpha"[1/0])
19     except ZeroDivisionError:
20         print("cero")
21     except IndexingError:
22         print("índice")
23     except:
24         print("algo")
25     #cero
26
```

Welcome

alo.py

al.py

gato.py

marvel.py

resumenes_seccion.py > ...

```
1 #Kevin Cabrera Luna
2 #2.7.1.8
3 #Ejercicio 1
4 #¿Cuál es la salida esperada del siguiente código?
5 try:
6     print(1/0)
7 except ZeroDivisionError:
8     print("cero")
9 except ArithmeticError:
10    print("arit")
11 except:
12    print("algo")
13 #cero
14 #Ejercicio 2
15 #¿Cuál es la salida esperada del siguiente código?
16 try:
17     print(1/0)
18 except ArithmeticError:
19     print("arit")
20 except ZeroDivisionError:
21     print("cero")
22 except:
23     print("algo")
24 #arit
25 #Ejercicio 3
26 #¿Cuál es la salida esperada del siguiente código?
27 def foo(x):
28     assert x
29     return 1/x
30 try:
31     print(foo(0))
32 except ZeroDivisionError:
33     print("cero")
34 except:
35     print("algo")
36 #algo
```



Welcome



alo.py



al.py



gato.py



marvel.py ●



resumenes_seccion.py ●



gif.py



resumenes_seccion.py > ...

```
1  #Kevin Cabrera Luna
2  #2.8.1.5
3  #Ejercicio 1
4  #¿Cuál de las excepciones se utilizará para proteger al código de ser interrumpido por el uso del teclado?
5  #KeyboardInterrupt
6  #Ejercicio 2
7  #¿Cuál es el nombre de la más general de todas las excepciones de Python?
8  #BaseException
9  #Ejercicio 3
10 #¿Cuál de las excepciones será generada a través de la siguiente evaluación fallida?
11 huge_value = 1E250 ** 2
12 #OverflowError
13
```

resúmenes_seccion.py

```
1 #Kevin Cabrera Luna
2 #3.1.1.8
3 #Ejercicio 1
4 #Si asumimos que pitones, víboras y cobras son subclases de la misma superclase, ¿cómo la llamarías?
5 #Serpiente, reptil, vertebrado, animal: todas estas respuestas son aceptables.
6 #Ejercicio 2
7 #Intenta nombrar algunas subclases de la clase Pitón.
8 #Pitón india, Pitón de Roca Sfricana, Pitón Bola, Pitón Birmana: la lista es larga.
9 #Ejercicio 3
10 #¿Puedes usar la palabra "class" para darle nombre a alguna de tus clases?
11 #¡No, no puedes, class es una palabra clave reservada!
```

```
1 #Kevin Cabrera Luna
2 #3.2.1.13
3 #Ejercicio 1
4 #Suponiendo que hay una clase llamada Snakes, escribe la primera línea de la declaración de clase Python
5 #expresando el hecho de que la nueva clase es en realidad una subclase de Snake.
6 class Python(Snakes):
7 #Ejercicio 2
8 #Algo falta en la siguiente declaración, ¿qué es?
9 class Snakes
10     def __init__():
11         self.sound = 'Sssssss'
12 #El constructor __init__() carece del parámetro obligatorio (deberíamos llamarlo self para cumplir con los estándares).
13 #Ejercicio 3
14 #Modifica el código para garantizar que la propiedad venomous sea privada.
15 class Snakes
16     def __init__(self):
17         self.venomous = True
18 #se veria
19 class Snakes
20     def __init__(self):
21         self.__venomous = True
```

resumenes_seccion.py > ...

```
1 #Kevin Cabrera Luna
2 #3.3.1.9
3 #Ejercicio 1
4 #¿Cuáles de las propiedades de la clase Python son variables de instancia
5 #y cuáles son variables de clase? ¿Cuáles de ellos son privados?
6 class Python:
7     population = 1
8     victims = 0
9     def __init__(self):
10         self.length_ft = 3
11         self.__venomous = False
12 #population y victims son variables de clase, mientras que length y __venomous
13 # son variables de instancia (esta última también es privada).
14 #Ejercicio 2
15 #Vas a negar la propiedad __venomous del objeto version_2, ignorando el hecho de que
16 #la propiedad es privada. ¿Cómo vas a hacer esto?
17 version_2 = Python()
18 version_2._Python__venomous = not version_2._Python__venomous
19 #Ejercicio 3
20 #Escribe una expresión que compruebe si el objeto version_2 contiene una propiedad de instancia
21 #denominada constrictor (¡sí, constrictor!).
22 hasattr(version_2, 'constrictor')
```


resumenes_seccion.py > ...

```
1 #Kevin Cabrera Luna
2 #3.4.1.11
3 #Ejercicio 1
4 #La declaración de la clase Snake se muestra a continuación. Enriquece la clase con un método
5 #llamado increment(), el cual incrementa en 1 la propiedad victims.
6 class Snake:
7     def __init__(self):
8         self.victims = 0
9 class Snake:
10     def __init__(self):
11         self.victims = 0
12
13     def increment(self):
14         self.victims += 1
15 #Ejercicio 2
16 #Redefine el constructor de la clase Snake para que tenga un parámetro que inicialice el campo
17 #victims con un valor pasado al objeto durante la construcción.
18 class Snake:
19     def __init__(self, victims):
20         self.victims = victims
21 #Ejercicio 3
22 #¿Puedes predecir el resultado del siguiente código?
23 class Snake:
24     pass
25 class Python(Snake):
26     pass
27 print(Python.__name__, 'es una', Snake.__name__)
28 print(Python.__bases__[0].__name__, 'puede ser una', Python.__name__)
29 "Python es una Snake"
30 "Snake puede ser una Python"
```

resumenes_seccion.py > ...

```
1 #Kevin Cabrera Luna
2 #3.5.1.22/3
3 #Ejercicio 1
4 #¿Cuál es el resultado esperado del siguiente código?
5 print(rocky)
6 print(luna)
7 "Collie dice: ¡Guau! ¡No huyas, corderito!"
8 "Dobermann dice: ¡Guau! ¡Quédese donde está, intruso!"
9 #Ejercicio 2
10 #¿Cuál es el resultado esperado del siguiente código?
11 print(issubclass(SheepDog, Dog), issubclass(SheepDog, GuardDog))
12 print(isinstance(rocky, GuardDog), isinstance(luna, GuardDog))
13 "True False"
14 "False True"
15 #Ejercicio 3
16 #¿Cuál es el resultado esperado del siguiente código?
17 print(luna is luna, rocky is luna)
18 print(rocky.kennel)
19 "True False"
20 "2"
21 #Ejercicio 4
22 #Define una subclase de SheepDog llamada LowlandDog, y equipala con un método __str__() que anule
23 #un método heredado del mismo nombre. El nuevo método __str__() debe retornar la cadena "¡Guau! ¡No me gustan las montañas!".
24 class LowlandDog(SheepDog):
25     def __str__(self):
26         return Dog.__str__(self) + " ¡No me gustan las montañas!"
27
```



```
resumenes_seccion.py > ...
1 #Kevin Cabrera Luna
2 #3.6.1.9
3 #Ejercicio 1
4 #¿Cuál es el resultado esperado del siguiente código?
5 import math
6 try:
7     print(math.sqrt(9))
8 except ValueError:
9     print("inf")
10 else:
11     print("ok")
12 "3.0"
13 "ok"
14 #Ejercicio 2
15 #¿Cuál es el resultado esperado del siguiente código?
16 import math
17 try:
18     print(math.sqrt(-9))
19 except ValueError:
20     print("inf")
21 else:
22     print("ok")
23 finally:
24     print("fin")
25 "inf"
26 "fin"
27 #Ejercicio 3
28 #¿Cuál es el resultado esperado del siguiente código?
29 import math
30 class NewValueError(ValueError):
31     def __init__(self, name, color, state):
32         self.data = (name, color, state)
33 try:
34     raise NewValueError("Advertencia enemiga", "Alerta roja", "Alta disponibilidad")
35 except NewValueError as nve:
36     for arg in nve.args:
37         print(arg, end='! ')
38 #Advertencia enemiga! Alerta roja! Alta disponibilidad!
39
```

```

welcome al.py gato.py marvel.py resúmenes_seccion.py 4 gni
resúmenes_seccion.py > ...
1 #Kevin Cabrera Luna
2 #4.1.1.15
3 #Ejercicio 1
4 #¿Cuál es el resultado esperado del siguiente código?
5 class Vowels:
6     def __init__(self):
7         self.vow = "aeiouy " # Sí, sabemos que y no siempre se considera una vocal.
8         self.pos = 0
9     def __iter__(self):
10        return self
11    def __next__(self):
12        if self.pos == len(self.vow):
13            raise StopIteration
14        self.pos += 1
15        return self.vow[self.pos - 1]
16 vowels = Vowels()
17 for v in vowels:
18     print(v, end=' ')
19 "a e i o u y"
20 #Ejercicio 2
21 #Escribe una función lambda, estableciendo a 1 su argumento entero, y aplícalo a la función map()
22 #para producir la cadena 1 3 3 5 en la consola.
23 any_list = [1, 2, 3, 4]
24 even_list = # Completar las líneas aquí.
25 print(even_list)
26 list(map(lambda n: n | 1, any_list))
27 #Ejercicio 3
28 #¿Cuál es el resultado esperado del siguiente código?
29 def replace_spaces(replacement='*'):
30     def new_replacement(text):
31         return text.replace(' ', replacement)
32     return new_replacement
33 stars = replace_spaces()
34 print(stars("And Now for Something Completely Different"))
35 And*Now*for*Something*Completely*Different
36

```

resumenes_seccion.py > ...

```
1 #Kevin Cabrera Luna
2 #4.2.1.12
3 #Ejercicio 1
4 #¿Cómo se codifica el valor del argumento modo de la función open() si se va a crear un nuevo archivo de texto?
5 "wt" o "w"
6 #Ejercicio 2
7 #¿Cuál es el significado del valor representado por errno.EACCESS?
8 "Permiso denegado: no se permite acceder al contenido del archivo."
9 #Ejercicio 3
10 #¿Cuál es la salida esperada del siguiente código, asumiendo que el archivo llamado file no existe?
11 import errno
12 try:
13     stream = open("file", "rb")
14     print("existe")
15     stream.close()
16 except IOError as error:
17     if error.errno == errno.ENOENT:
18         print("ausente")
19     else:
20         print("desconocido")
21 "ausente"
```

resumenes_seccion.py > ...

```
1  #Kevin Cabrera Luna
2  #4.3.1.18
3  #Ejercicio 1
4  #¿Qué se espera del método readlines() cuando el stream está asociado con un archivo vacío?
5  "Una lista vacía (una lista de longitud cero)."
6  #Ejercicio 2
7  #¿Qué se pretende hacer con el siguiente código?
8  for line in open("file", "rt"):
9      for char in line:
10         if char.lower() not in "aeiouy ":
11             print(char, end='')
12  "Copia el contenido del archivo file hacia la consola, ignorando las vocales."
13  #ejercicio 3
14  #Vas a procesar un mapa de bits almacenado en un archivo llamado image.png y quieres leer su
15  #contenido como un todo en una variable bytearray llamada image. Agrega una línea al siguiente código para lograr este objetivo.
16  try:
17      stream = open("image.png", "rb")
18      # Inserta una línea aquí.
19      stream.close()
20  except IOError:
21      print("fallido")
22  else:
23      print("exitoso")
24  image = bytearray(stream.read())
```



resumenes_seccion.py

```
1  #Kevin Cabrera Luna
2  #4.4.1.9
3  #Ejercicio 1
4  #¿Cuál es el resultado del siguiente fragmento si se ejecuta en Unix?
5  import os
6  print(os.name)
7  "posix"
8  #Ejercicio 2
9  #¿Cuál es el resultado del siguiente fragmento de código?
10 import os
11 os.mkdir("hello")
12 print(os.listdir())
13 ['hello']
14
```

resumenes_seccion.py > ...

```
1  #Kevin Cabrera Luna
2  #4.5.1.23
3  #Ejercicio 1
4  #¿Cuál es el resultado del siguiente fragmento de código?
5  from datetime import time
6  t = time(14, 39)
7  print(t.strftime("%H:%M:%S"))
8
9  "14:53:00"
10 #Ejercicio 1
11 #¿Cuál es el resultado del siguiente fragmento de código?
12 from datetime import datetime
13 dt1 = datetime(2020, 9, 29, 14, 41, 0)
14 dt2 = datetime(2020, 9, 28, 14, 41, 0)
15 print(dt1 - dt2)
16 "1 day, 0:00:00"
```

resumenes_seccion.py > ...

```
1  #Kevin Cabrera Luna
2  #4.6.1.14
3  #Ejercicio 1
4  #¿Cuál es el resultado del siguiente fragmento de código?
5  import calendar
6  print(calendar.weekheader(1))
7  "M T W T F S S"
8  #Ejercicio 2
9  #¿Cuál es el resultado del siguiente fragmento de código?
10 import calendar
11 c = calendar.Calendar()
12 for weekday in c.iterweekdays():
13     print(weekday, end=" ")
14 "0 1 2 3 4 5 6"
```