【模擬】Pythonカテスト

合計点 70/70 点

このテストは、Python3.9.13、BeautifulSoup4.11.1、Selenium4.6.0を想定している.

このフォームの送信時に回答者のメールアドレス(c0a2004727@edu.teu.ac.jp)が記録されました。

✓ 次のコードを実行した際の出力またはエラーとして適切なものを選びなさ 1/1 い.

```
1 x = tuple(range(10))
2 n = None
3 if n:
4    print(len(x))
5 else:
6    x[0] = 1
7    print(x[0])
```

- 10
- 0 1
- TypeError



✓ 次のコードを実行した際、7行目と8行目の出力として適切なものを選びな1/1 さい $1 \times = list(range(1, 11))$ 2 y = [str(i) for i in range(1, 11)]3 x.sort() 4 y.sort() 5 z = [int(i) for i in y]6 print(x, y) 7 print(x == z) #18 print(y == z) #2 ○ どちらもTrue どちらもFalse ○ 7行目だけTrue 8行目だけTrue ✓ 次のコードを実行した際の出力として適切なものを選びなさい。 1/1 $1 \times = list(range(10))$ 2 y = x.sort()3 print(type(y)) () list type sort NoneType

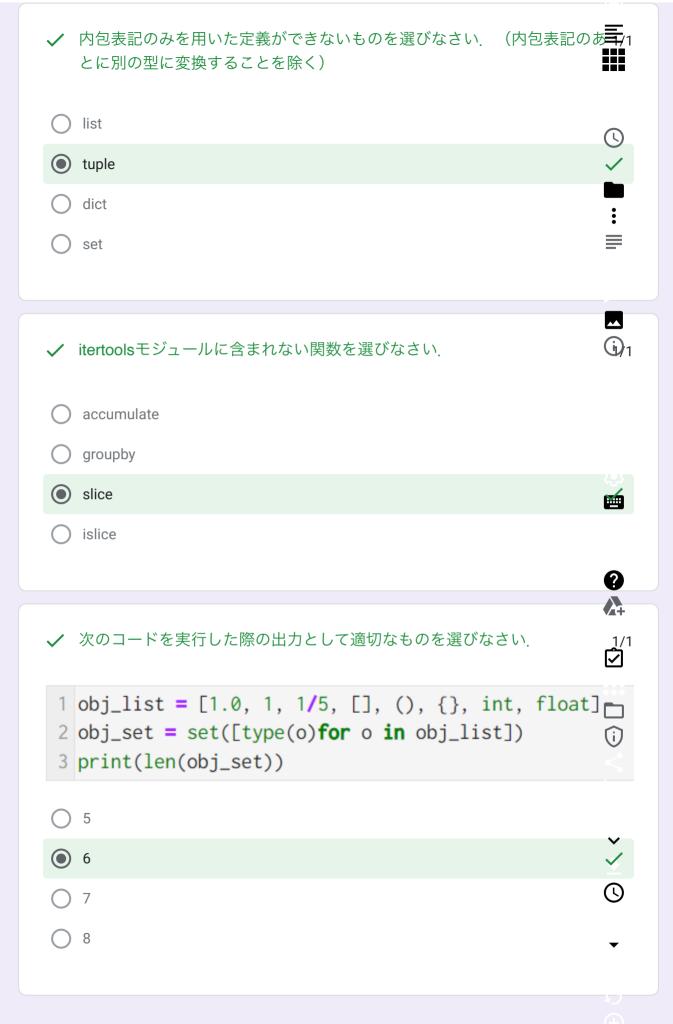
✓ 次のコードを実行した際に発生するエラーの原因として適切なものを選び 1/1 なさい。

```
1 class A:
 2
       def __init__(self, a, b, c=3):
 3
           self.a = a
           self.b = b
 4
           self.c = c
 5
 7 class A_d(A):
       def __init__(self, a, b=2):
           super.__init__(a, b)
 9
10
11
12 q = A_d(1, 5)
13 print(q.b)
```

- クラスA_dのインスタンスを生成する際、2つの引数を指定しているから
- クラスA_dは抽象クラスであるクラスAを継承しているから
- クラスAのイニシャライザのパラメータ数とクラスA_dのイニシャライザのパラメータ数が異なるから
- 9行目で親クラスのイニシャライザを呼び出す方法が間違っているから

```
✓ 次のコードを実行した際の出力として適切なものを選びなさい。
                                         1/1
 1 class A:
       def __init__(self):
           self.a = 0
 4 def __str__(self):
           return f"{self.a+1}"
 5
    def __repr__(self):
 6
           return f"{self.a+2}"
 7
 8
 9
10 a=A()
11 print(a)
( ) a
0
1
```

| ✓ 次のコードを実行した際に発生するエラーを選びなさい。エラない場合は「発生しない」を選びなさい。 | ーが発生し 1/1 |
|--|-----------|
| <pre>1 x_s = [tuple, set, str] 2 for x in x_s: 3</pre> | |
| ● 発生しない | ~ |
| ○ ValueError | |
| | |
| ☐ IndexError | |



次のコードを実行した際、3行目と4行目の出力として適切なものを選びな 1/1 さい.
 1 a = [i for i in range(0, 100, 5)]
 2 b = [i for i in range(100) if i%5==0]
 3 print(a == b) #1
 4 print(a is b) #2
 びちらもTrue
 どちらもFalse
 ③ 3行目だけTrue
 ✓
 4行目だけTrue

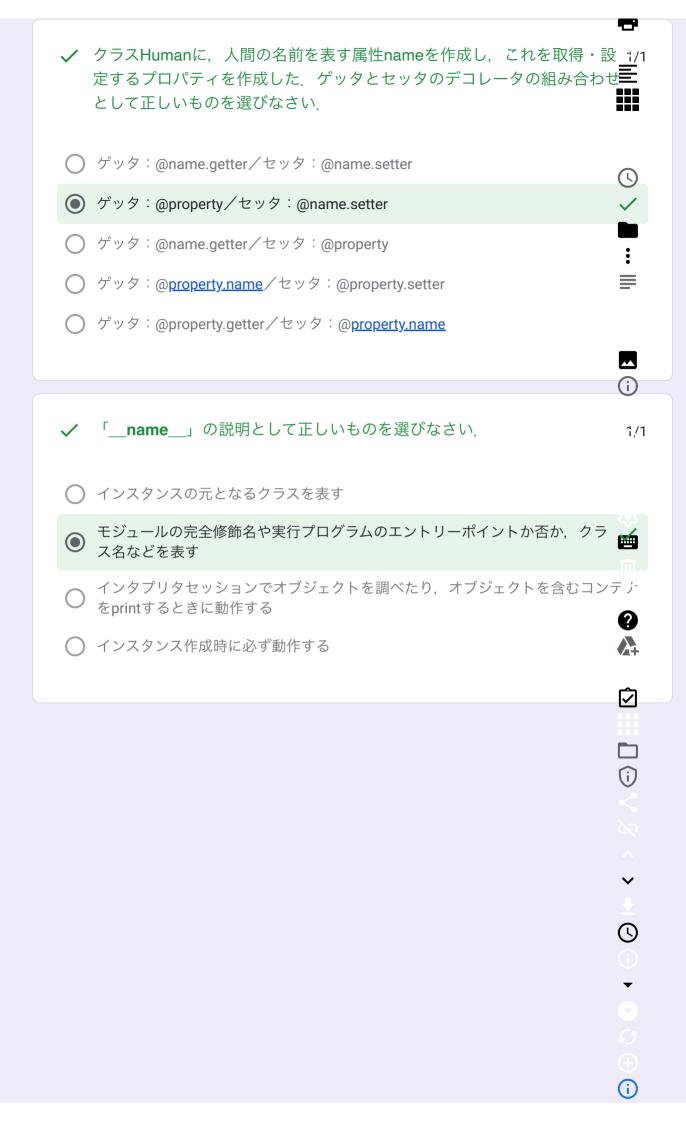
```
✓ 次のコードを実行した際の出力として適切なものを選びなさい。
                                              1/1
 1 from dataclasses import dataclass
 3 @dataclass
 4 class D:
 5 a :int
     b :float
 6
 7
     c :str
 8
 9
10 data = D(1, 2, 3)
11 print(type(data.c))
int
float
```

 \bigcirc D

✓ seleniumのwebdriverを用いて、次のQ.htmlから2つ目のハイパーリンクの 1/1

みを取得する際の適切なコードを選びなさい。 Q.html ▼ <!DOCTYPE html> 1 <html lang="ja"> <head> 4 <meta charset="UTF-8"> <title>HTML Sample</title> 5 6 </head> 7 <body> <h1>見出し</h1> 8 本文 9 1 10 link1 11 12 2 13 link2 14 15 </body> </html> 16 driver.find_element(By.TAG_NAME, "a") driver.find_element(By.NAME, "link2") driver.find_element(By.LINK_TEXT, "link2") driver.find_element(By.LINK_TEXT, "link1").next() ✓ 辞書から値だけを抽出したビューを返すメソッドを選びなさい。 1/1 value() item() key() values() items()) keys()

```
✓ 次のコードを実行した際、出力が「234」となった。空欄に当てはまるコ 1/1
   ードとして適切なものを選びなさい.
 1 lst = [2, 4, 3]
 2
 3 [空欄]
 4
 5 for num in 1st:
       print(num, end="")
 7 print()
Ist.sort()
 sort(lst)
 ) lst.sorted()
sorted(lst)
✓ 次のコードを実行した際に発生するエラーを選びなさい。
                                                     1/1
 1 def change_num(nums):
       nums[0] = nums[0] // nums[1] - nums[1]
 2
 3
 4
 5 \text{ numbers} = (0, 1)
 6 change_num(numbers)
 7 print(numbers[0]+numbers[1]+numbers[2])
 ) IndexError
TypeError
 ) ZeroDivisionError
  AttributeError
```



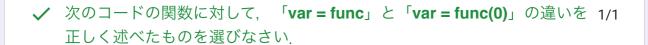
✓ 次のコードを実行した際、7行出力された。6行目に出力されるものを選 1/1 びなさい。

```
1 import copy
3 class Robot:
      latest_version = 0
5
     def __init__(self, type_):
6
7
         self._type = type_
          Robot.latest_version += 1
8
          self.version = Robot.latest_version
10
11
     def version_up(self):
          Robot.latest_version += 1
12
          self.version = Robot.latest version
13
15 robots = [Robot(["家事用"])]
16 robots.append(robots[0])
17 robots.append(copy.copy(robots[0]))
18 robots.append(copy.deepcopy(robots[0]))
19
20 robots[0].version_up()
21 robots[0]._type.append("娯楽用")
22
23 robots.append(robots[2])
24 robots.append(copy.copy(robots[2]))
25 robots.append(copy.deepcopy(robots[2]))
27 robots[2].version_up()
28 robots[2]._type.append("工場用")
30 for rbt in robots: print(rbt.version, rbt._type)
```

- 1 ['家事用']
- () 2['家事用']
- 3 ['家事用']
- 1 ['家事用', '娯楽用']
- 2 ['家事用', '娯楽用']
- 3 ['家事用', '娯楽用']
- 1 ['家事用', '娯楽用', '工場用']
- 2 ['家事用', '娯楽用', '工場用']

| ○ 3 ['家事用', '娯楽用', '工場用'] | <u>^</u> |
|--|------------------------------|
| ✓ with構文を使わないでファイルを読み込む際の注意点を選びなさい | \. 1 |
| ○ ファイルのディスクリプタを直接扱うことになるので、環境依存のコー | ドになる |
| ● ファイルが自動的に閉じられないため、忘れると他のプログラムで不具き起こす危険性がある。 | 合を引 ✓ |
| ○ ファイルの文字コードを自動的に判別してくれないため、手動で設定しらない. | なければな |
| ○ ファイルの読み込み・書き込みのようなモードを自動的に判別しないの 設定しなければならない. |)で, 手動で |
| | (i) |
| | |
| ✓ 次のような内容のタブ区切りのテキストファイルを開き、ファイル変数senに読み込む、このとき、2行目の文を抽出するコードとしまのを選びなさい、なお、改行文字は「\n」、タブ文字は「\t」で 1 ID 投稿時刻 共有数 内容 2 0 41 50 C++23には添え字演算子の提案を採用してほしいた。 3 1 42 1000 Unicodeの対応をもっと改善しておくれ・・ 4 2 93 150 それよりも、コルーチンや契約の機能の方が先だ ○ sen.split('\n')[-1] | て適切な ある. = ・ |
| sen.split('\n')[-3] | |
| sen.split('\n')[0] | Ø |
| sen.split('\t')[-3] | <u> </u> |
| 3cm.spmt(\tau)[0] | 0 |
| | * |





- 1 **def** func(a):
- 2 return a**2
- O funcに括弧をつけない場合,引数に乱数が暗黙的に代入されるので,括弧が無い方のvarには何かの値を2乗したものが代入される.
- O funcに括弧をつけない場合、文法的に誤っているので、括弧が無い方の記述はエラーとなり実行できない。
- funcに括弧をつけない場合, varにはfunc本体が代入されるので, 「var(0)」の ✓ ような記述ができる.
- O funcに括弧をつけない場合、未定義の動作が発生するので何が起きるか分からない。
- ✓ 型アノテーションについて正しく述べたものを選びなさい。

1/1

- 変数の初期化をその型の値で行う.
- 変数に入る値の種類や関数が返す値の種類を示すヒントである。
- 変数の宣言を静的に行うための機能であり、その型以外の値を代入できないよう にする.
- 型アノテーションを使う場合は、必ず代入を行わなければならない。

```
1 def divide(func):
       def _func(a):
           return func(a) // 3
       return _func
 4
 5
 6 def multiply(func):
       def _func(a):
 7
           @divide
 8
           def _calculate(a):
 9
                return a * 10
10
           return _calculate(func(a))
11
       return _func
12
13
14 @multiply
15 def caluclate(a):
       return a ** 2
16
17
18 print(caluclate(9))
81
54
270
○ 構文エラーにより、計算できない
```





✓ 次のコードのクラスPersonの属性ageとnameに対する処理のうち、エラー1/1 が発生するものを選びなさい.

```
1 class PersonDescriptor:
        def __get__(self, obj, ojtype=None):
 2
            return self.value
 3
        def __set__(self, obj, value):
 4
            if type(value) is int:
 5
                 if value < 0:</pre>
 6
                     raise ValueError('')
 7
                 self.value = value
 8
            elif type(value) is str:
 9
                 self.value = value
10
            else:
11
                 raise TypeError('')
12
13
14 class Person:
       age = PersonDescriptor()
15
        name = PersonDescriptor()
16
17
18 person = Person()
 ) person.age = "I don't know"
  person.name = 0
  person.age = 150000
person.age = 5.0
```

| ✓ dataclassについて正しく述べたものを選びなさい. | ∰ 3,/1 ≣ |
|---|------------------------------|
| 比較演算に関する特殊メソッドが自動的に定義されないので、必要な場合は 定義しなければならない。 | |
| ○ インスタンス変数のアドレスが共有されるので、低レベルな操作に向いてい | ³ . ن |
| ● 算術演算に関する特殊メソッドが自動的に定義されないので、必要な場合は 分で定義しなければならない。 | |
| ○ メソッドを定義することができない. | : ≡ |
| | |
| ✓ 次のうち、正しい記述を選びなさい。ただし、リスト、集合、辞書、 ルは組み込み型を用いる。 | タプ ム /1 ① |
| ○ 空のリスト(list)10個からなる集合(set)を作ることができる. | |
| ● タプル(tuple)を辞書(dict)のキーとして使うことができる。 | ✓ |
| ○ intやfloatなどの数値をfor-in文のinの後ろに置くことができる. | |
| ● 集合(set)を別の集合(set)の要素とすることができる。 | |
| | 0 |
| | A + |
| ✓ 伏見が嫌いな(=パイソニックでない)書き方をすべて選びなさい。 | 1/1 🗘 |
| for i in range(len(monsters)): print(monsters[i]) | ✓ |
| for i in monsters: print(i) | 8 |
| for mon in monsters: print(mon) | |
| print(monsters[0], monsters[1], monsters[2], monsters[3], monsters[4], monsters[5]) | ✓ |
| | |
| | U |
| | 0 |

() (+)



```
1 @f1(arg)
2 @f2
3 def func():
4    pass
```

```
1 def func(arg):
2  pass
3
4 func = f1(f2(func))
```

選択肢 2

```
1 def func():
2 pass
3
4 func = f2()(f1(func))
```

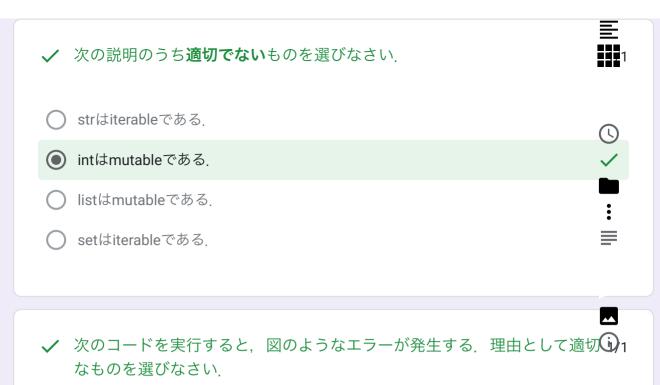
選択肢3

```
1 def func():
2 pass
3 func = f1(arg)(f2(func))
```

```
1 def func():
2  pass
3
4 func = f1(arg)(f2(func()))
```

選択肢4

✓ 次のコード (poke name.py) をエラーなく実行するためのコマンドを選 1/1 びなさい import sys 2 3 def read_names(fp): 省略 4 return names # リスト 5 6 7 file_path = sys.argv[1] $8 \text{ row_num} = int(sys.argv[2])-1$ 9 names = read_names(file_path) 10 print(names[row_num]) python poke_name.py 10 data/poke_names.txt python poke_name.py data/poke_names.txt 10) python poke_name.py python poke_name.py sys.argv[1] sys.argv[2] ✓ 次の説明のうち適切でないものを選びなさい。 1/1)iterableは,for-in文で繰り返し可能なオブジェクトである. 🔘 sizedは, len()関数で要素数を返すオブジェクトである. immutableは、作成後に値を変更できるオブジェクトである。) callableは,() を付して呼び出せるオブジェクトである.



```
class Monster:
        def __init__(self, name, level):
                                                        self.__name = name
              self.__level = level
 4
                                                        0
 5
                                                        6 m = Monster("ピカチュウ", 25)
 7 print(m.__level)
                                                        N
AttributeError
                           Traceback (most recent call last)
Input In [2], in <cell line: 7>()
   4 self.__level = level
   6 m = Monster("ピカチュウ", 25)
----> 7 print(m.__level)
AttributeError: 'Monster' object has no attribute '__level'

○ インスタンス変数

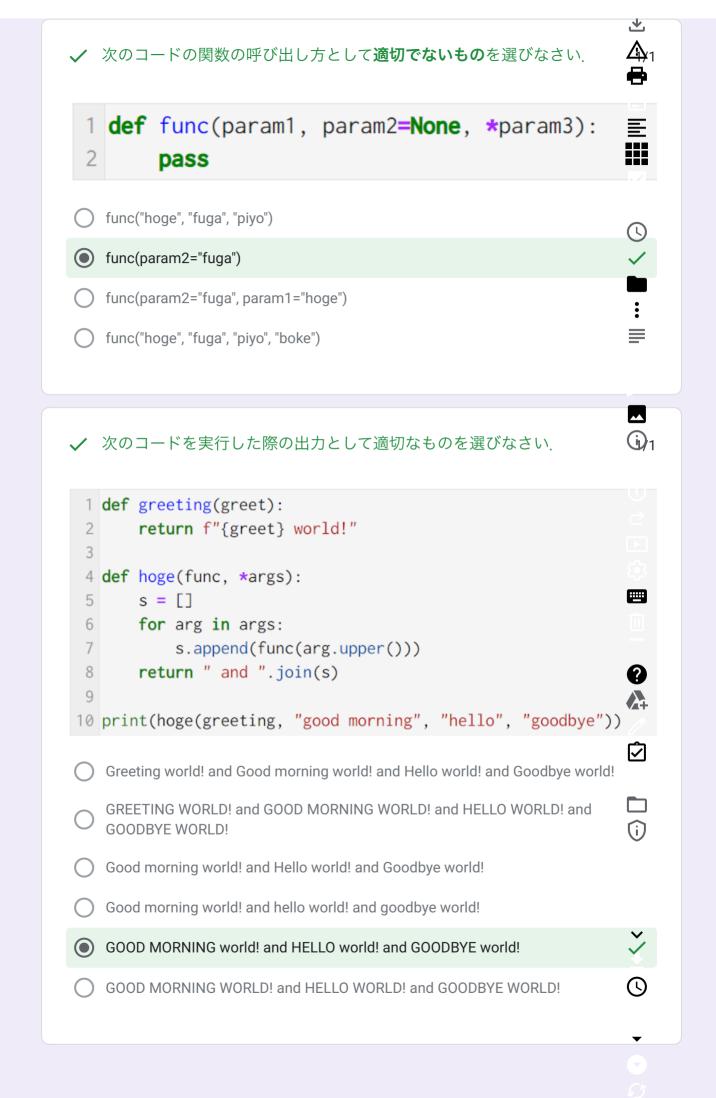
Llevelのアンダースコアが多いため

 ) 名前マングリングにより「__level」は「_level」という名前に変わっているから(()
```

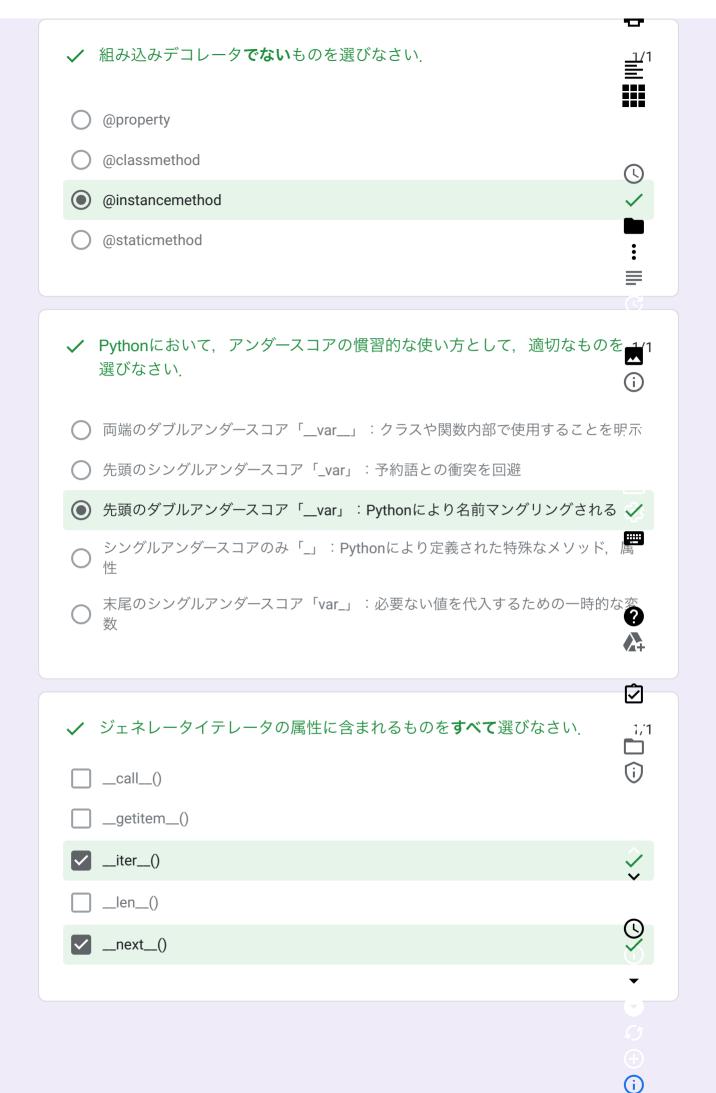
- 名前マングリングにより「_level」はプライベート変数になりクラス外部からアク セスできなくなっているから
- 名前マングリングにより「__level」は「_Monster__level」という名前に変わっ 🗸 ているから







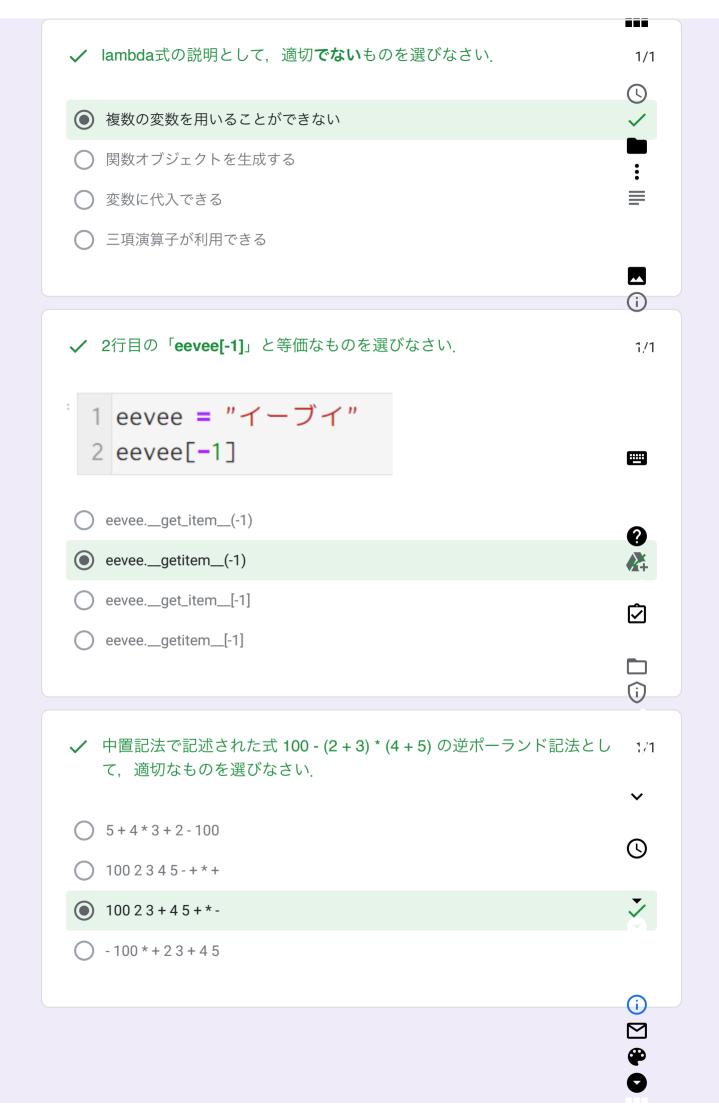
```
def make_greeting(time):
        def inner1(text):
 2
             return f"Good morning {text}!"
 4
        def inner2(text):
 5
             return f"Hello {text}!"
 6
 7
        def inner3(text):
 8
             return f"Good evening {text}!"
 9
10
        if 5 < time < 10:
11
             return inner1
12
        if 10 < time < 17:
13
             return inner2
14
15
        else:
             return inner3
16
17
18 print( make_greeting(6) )
  <function make_greeting at 0x7fdfd035b9d0>
<function make_greeting.<locals>.inner1 at 0x7fdfd0310c10>
  Good morning world!
  Good morning world! Hello world! Good evening world!
```

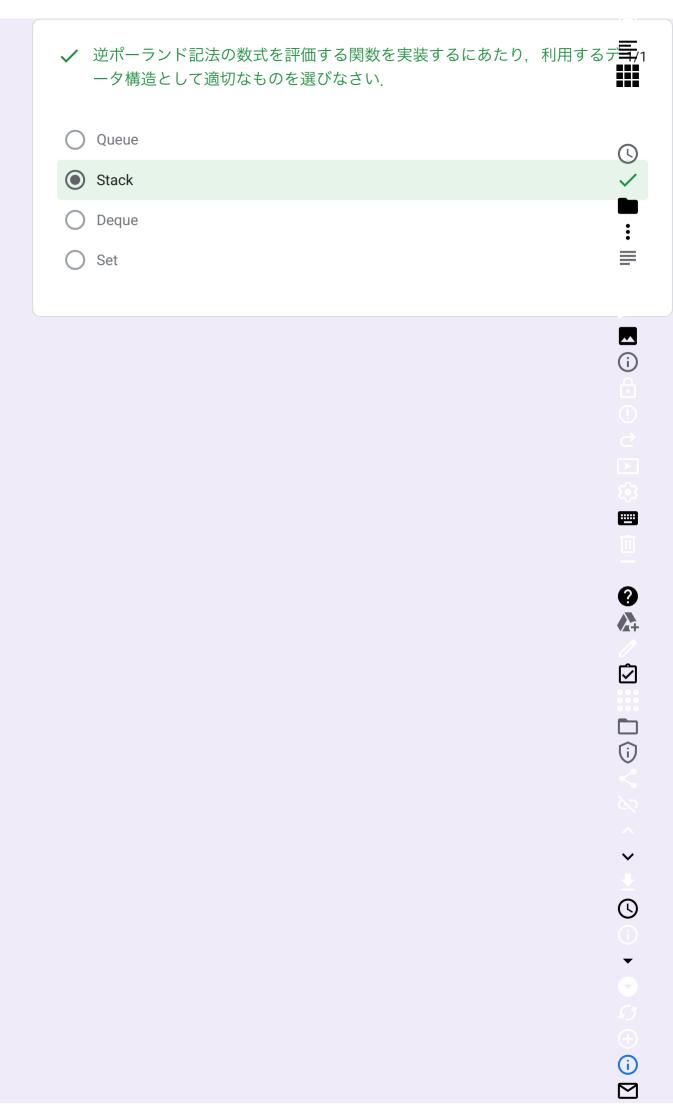






(i)





✓ 次の図のように、read_main.pyでaisatsu.pyをimportしhello関数を呼び出 1/1 した際、「AttributeError: module 'aisatsu' has no attribute 'hello'」というエラーが発生した。エラーの原因として適切なものを選びなさい。

```
# aisatsu.py
2 if __name__ == "__main__":
       def hello():
3
            print("Hello World")
4
       def goodbye():
5
            print("GoodBye World")
6
1 # read_main
2 import aisatsu
4 aisatsu.hello()
 read_main.pyで、「import hello」としていないから
 read_main.pyで、「import aisatsu.hello」としていないから
 read_main.pyで、「hello()」としていないから
 read_main.pyで, 「if __name__ == "__main__"」の中で「hello()」としていないか
 aisatsu.pyで、hello関数が定義されているのが「if __name__ == "__main__"」の 🗸
  中だから
```

i

0

Ŷ

□

(1)

```
1 def make_total():
     total = 0
       def _total():
          nonlocal total
 4
           total += 1
 5
          return total
 6
 7
       return _total
 8
 9
10 funcs = []
11 for i in range(5):
       funcs.append(make_total())
12
13
14 for fun in funcs:
print(fun(), end=" ")
00000
11111
01234
12345
44444
```

✓ 次のコードを実行した際、「インスタンス」と出力された。空欄のコード 1/1 として適切なものを選びなさい。

```
1 class niClass:
2 __cl= "クラス"
3     def __init__(self):
4         self.__in = "インスタンス"
5     def __method(self):
7         return "メソッド"
8     9 t = niClass()
```

- print(<u>t.in</u>)
- print(t._in)
- print(t.__in)
- print(t._niClass__in)

```
1 class Pokemon:
       def __init__(self, name, types):
 2
            self.__name = name
 3
            self.__types = types
 4
 5
       def set_name(self, name):
 6
            self.\__name = name
 7
 8
 9
       def set_types(self, types):
            self.__types = types
10
11
12
       def get_name(self):
            if self.__name == '':
13
                raise ValueError
14
            return self.__name
15
16
17
       def get_types(self):
            if len(self.\_types) \Leftarrow 0:
18
19
                raise ValueError
            elif len(self.__types) >= 3:
20
21
                raise ValueError
            return self.__types
22
23
24 pokemon = Pokemon('フシギダネ', ('くさ', 'どく'))
   print(pokemon.get_name())
print(pokemon.name)
  print(pokemon.get_types())
print(pokemon.__name)
   print(pokemon._Pokemon__name)
```

✓ 次のコードを実行した際の出力として、適切なものを選びなさい. 1/1

```
1 class a:
2 a = 100
3 b = 200
4 def __init__(self):
     self.a = a
5
6
7 a1 = a()
8 a1.a = 50
9 print(a.a)
```

None

- 100
- 50

✓ 3行目の空欄に入る戻り値の型アノテーションとして、適切なものを選び 1/1 なさい.

1 from typing import Iterable, Iterator

2 def f(_x: Iterable[int]) → [空欄]:
 x = list(_x)
 left, right = iter(x), iter(x)
 next(right)
 return [1 / r for l, r in zip(left, right)]

○ list[int]

○ list[float]
 lterable[int]
 lterator[float] | ZeroDivisionError

✓ 「Hi! I am Koukaton. I am majoring in computer science.」と1行で書 1/1 かれたファイルsentence.txtを読み込み、文章から単語を抽出するプログラムを作成したい.空欄A~Cに当てはまるものとして適切なものを選びなさい.

```
1 rfo = open('sentence.txt', 'r')
 2 row = 「空欄AT
 3 keywords = row.rstrip().split(' ')
 4 \text{ results} = []
 5 for keyword in keywords:
       if keyword[-1] == '!':
 6
           results.append([空欄B])
 7
       elif keyword[-1] == '.':
 8
           results.append([空欄C])
 9
       else:
10
           results.append(keyword)
11
12
13 print(results)
```

['Hi', 'I', 'am', 'Koukaton', 'I', 'am', 'majoring', 'in', 'computer', 'science']

- 空欄A:rfo.read(3)/空欄B:keyword.rstrip('!')/空欄C:keyword.rstrip('.')
- 空欄A:rfo.readlines()/空欄B:keyword.rstrip(-1)/空欄C:keyword.rstrip(-1)
- 空欄A:rfo.readline()/空欄B:keyword.rstrip('!')/空欄C:keyword.rstrip('.')
- ② 空欄A:rfo.readlines()/空欄B:keyword.split('!')[0]/空欄C:keyword.split(")[0]



● b = copy.deepcopy(a)としてこれらのリストは作成された

○ b = a によってこれらのリストは作成された

✓ 次のコードを実行した際,下段の図の6行目のprint文で最初に出力される 1/1 ものを選びなさい.

```
1 class Zukan:
      def __init__(self):
 2
          self.titles = ["フシギダネ", "フシギソウ", "フシギバナ",
 3
                      "ヒトカゲ","リザード","リザードン",]
          self.idx = 0
 5
 6
      def __iter__(self):
 7
          return self
 8
 9
    def __next__(self):
10
          if self.idx == len(self.titles):
11
             raise StopIteration()
12
         title = self.titles[self.idx]
13
14
         self.idx += 1
15
         return title
1 zukan = Zukan()
2 print(next(zukan))
3 print(next(zukan))
4
5 for i, z in enumerate(zukan, 1):
     print(f"{i:03d}\t{z}")
001 フシギダネ
003 フシギダネ
001 フシギバナ
003 フシギバナ
StopIteration
```

| ✓ | 「gen = (i*i for i in range(20) if i%2 == 0)」に対して、エラーが出ない print文を すべて 選びなさい. | ∑ ₁ |
|----------|---|----------------------------------|
| | print(gen[4]) | |
| | print(len(gen)) | |
| | print(next(gen)) | |
| | print(16 in gen) | Y |
| | print(gen()) | = : = |
| / | 「 rng = range(20) 」に対して、エラーが 出ない print文を すべて 選びなさい. | |
| | print(rng[4]) | |
| | print(len(rng)) | |
| | print(next(rng)) | |
| | print(16 in rng) | |
| | print(rng()) | 2 |
| | | |
| ✓ | 特殊メソッド「 get 」と「 set 」を持つオブジェクトをなんと呼ぶか、適切なものを選びなさい. | |
| 0 | イテレータ | (i) |
| 0 | ジェネレータ | ~ |
| • | ディスクリプタ | |
| 0 | クロージャ | |
| 0 | コンテキストマネージャ | () |
| | | _ |
| | | |

✓ 図の下段 (special.py) を「**python special.py**」のようにコマンドライン 1/1 で実行した際に、図の上段の9行目のprint文について正しい記述を選びなさい.

```
1 # pokemon.py
2 print(f"{__file__}の{__name__}(ifの外)")
4 class Monster:
     def __init__(self):
6
         print(f"{__class__.__name__}クラスの{__class__.__init__.__name__}関数")
8 if __name__ == "__main__":
    print(f"{__file__}の{__name__}(ifの中)")
10 m = Monster()
1 # special.py
2 from pokemon import Monster
4 if __name__ == "__main__":
    print(f''\{\_file\_\} \mathcal{O}\{\_name\_\} (if \mathcal{O} 中)'')
   m = Monster()
    「pokemon.pyの_main_(ifの中)」と出力される
() 「special.pyの__main__(ifの中)」と出力される
NoneのNone(ifの中)」と出力される
実行されない
```

- ✓ 特殊属性「__dict__」について正しい記述を選びなさい. 1/1
- すべてのオブジェクトが持っている
- インスタンスは持っているが、クラスは持っていない
- listオブジェクトは__dict__を持っていない
- intオブジェクトの__dict__の値は書き換え可能である

✓ 次のコードにおいて、上段3行目の「self.attrname」に代入される値を選 1/1 びなさい.

```
1 class PokeDesc:
      def __set_name__(self, owner, attrname):
2
           self.attrname = attrname
 3
4
      def __get__(self, obj, objtype=None):
5
           return obj.__dict__[self.attrname]
6
7
      def __set__(self, obj, value):
8
           obj.__dict__[self.attrname] = value
9
1 class Monster:
      title = PokeDesc()
2
3
      def __init__(self, title):
4
           self.title = title
 1 m = Monster("ピカチュウ")
Monster
PokeDesc
( "title"
self.title"
() "ピカチュウ"
```

```
1 from collections import defaultdict
 2
 3 class Monster:
       def __init__(self):
 4
 5
            pass
 6
 7 class Pokemon:
       def __init__(self, title):
 8
            self.title = title
 9
10
   def func():
       return
12
13
14 gen = (i for i in range(10))
15
16 def_dct1 = defaultdict(list)
17 def_dct2 = defaultdict(Monster)
18 def_dct3 = defaultdict(Pokemon)
19 def_dct4 = defaultdict(func)
print(def_dct1[3])
print(def_dct2[3])
print(def_dct3[3])
print(def_dct4[3])
```

✓ 次のコードの5行目の「super()」を「self」とした場合のエラーとして適 1/1 切なものを選びなさい。

```
1 class EvenList(list):
        def append(self, value):
 2
 3
             if value%2 != 0:
 4
                 raise ValueError(f"奇数は設定できません:{value}")
 5
             super().append(value)
 6
 7 eve_lst = EvenList()
 8 for i in range(10):
 9
        try:
10
             eve_lst.append(i)
        except Exception as e:
11
12
            print(e)
13 print(len(eve_lst), eve_lst)
奇数は設定できません: 1
奇数は設定できません:3
奇数は設定できません:5
奇数は設定できません:7
奇数は設定できません:9
5 [0, 2, 4, 6, 8]
   RecursionError: maximum recursion depth exceeded in comparison
    AttributeError: 'EvenList' object has no attribute 'append'
    ValueError: 奇数は設定できません:1
```

このフォームは 東京工科大学 内部で作成されました。

TypeError: not all arguments converted during string formatting

Google フォーム