django

Zadania rekrutacyjne - Python + Django

1.

Reverse the list without using builtin functions

```
odd_numbers = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

2.

What's the value of x?

```
x = map(lambda x: x * 2, range(5))
```

A. <map at 0x107c28a01>
B. [0, 2, 4, 6, 8]
C. (0, 2, 4, 6, 8)
D. TypeError: 'range' is not iterable

3.

```
users = [
    { "name": "Bill", "age": 37, "country": "USA" },
    { "name": "Susan", "age": 21, "country": "Canada" },
    { "name": "Glenda", "age": 61, "country": "Britain" },
    { "name": "Astro", "age": 12, "country": "USA" },
]
```

Get a list of users from the list for users in the USA.

DEVS-MENTORING

Additional:
Solve with a functional and imperative paradigm both.

4.

```
list_a = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23]
list_b = [4, 8, 15, 16, 23, 42]
```

Write a function that will return all numbers that exist in both lists.

Additional:
Maybe try to use sets?

5.

```
DEFAULT_SALUTATIONS = "Hello!"

def enhanced_greeting(name: str, salutation: str = DEFAULT_SALUTATIONS):
    salutation.replace("!", "")

    return f"{salutation} {name}!"

print(
    enhanced_greeting("Mr. Jones"),
    enhanced_greeting("Mrs. Jones", "Good morning!"),
    enhanced_greeting("Mrs. Smith", "Good afternoon")
)
```

When evaluating, what is printed?

A. Mr. Jones Mrs. Jones Mrs. Smith
B. Hello! Mr. Jones! Good morning! Mrs. Jones! Good afternoon Mrs. Smith!
C. Hello Mr. Jones! Good morning Mrs. Jones! Good afternoon Mrs. Smith!
D. Hello Mr. Jones Good morning Mrs. Jones Good afternoon Mrs. Smith

6.

```python
from random import shuffle

DECK_OF_CARDS = (
    (rank, suit)
    for rank in (*range(2, 11), "Jack", "Queen", "King", "Ace")
    for suit in ("Clubs", "Diamonds", "Hearts", "Spades")
)


def deal_hand_from_deck(deck: List[Tuple[Union[int, str], str]], *, hand_size: int = 5):
    """Returns the next hand of cards.

    Note: This fn mutates the deck.
    """

    num_of_cards = min(len(deck), hand_size)
    hand = deck[:num_of_cards]
    deck = deck[num_of_cards:]
    return hand

# game 1
deck = shuffle(list(DECK_OF_CARDS))
first_hand = deal_hand_from_deck(deck)

# game 2
deck2 = list(DECK_OF_CARDS)
shuffle(deck2)
first_hand2 = deal_hand_from_deck(deck2, 2)
```

Can you spot the errors in the code?

A. deck is None
B. deck2 is empty
C. deal_hand_from_deck(deck2, 2) is a TypeError
D. (*range(2, 11)) is a syntax error

DEVS-MENTORING

7.

```python
from functools import partial

def print_sum(num_list: Iterable[int], msg: Optional[str] = None):
  total = sum(num_list)
  print(f"{msg}: {total}" if msg else total)

calculate_total_cost = partial(print_sum, [11.22, 54.21, 100])

calculate_total_cost([50, 60])
```

What gets printed when evaluating this code?

A. 110
B. TypeError: print_sum takes 1 positional arguments, 2 were given
C. [50, 60]: 165.43
D. 165.43

8.

```python
# models.py
from django.db import models

class UserProfile(models.Model):
    birthdate = models.DateField(blank=True, null=True)
    github_name = models.CharField(max_length=255, blank=True, null=True)


class User(models.Model):
    first_name = models.TextField()
    last_name = models.TextField()
    profile = models.OneToOneField(UserProfile, on_delete=models.CASCADE)


# utils.py
from .models import User


def create_user(
    first_name: str,
    last_name: str,
    github_name: Optional[str] = None,
    birthdate: Optional[date],
):
    user = User.objects.create(first_name=first_name, last_name=last_name)

    if user.profile:
        user_profile = user.profile

        user_profile.github_name = github_name
        user_profile.birthdate = birthdate
        user_profile.save()

    return user
```

DEVS-MENTORING

Can you spot the errors in the code?

A. throws RelatedObjectDoesNotExist
B. models are missing save method
C. create_user function definition is invalid
D. User save method needs to be called to persist to DB

9.

```
def find_first_odd_number(numbers):
    if not numbers:
        return None

    return next((x for x in numbers if x % 2), None)
```

What's the most accurate typing for the function?

A. def find_first_odd_number(numbers: Maybe[List[int]]) -> Maybe[int]:
B. def find_first_odd_number(numbers: List[Any]) -> int:
C. def find_first_odd_number(numbers: Optional[List[int]]) -> Optional[int]:
D. def find_first_odd_number(numbers: int[]) -> int:

10.

```
def doubler():
    x = 5

    def inner_doubler():
        global x
        return x * 2

    return inner_doubler()

x = 2

y = doubler()
```

What's the value of y?

DEVS-MENTORING

A. 10
B. NameError: name ‘x’ is not defined
C. TypeError: name ‘x’ can not be redefined
D. 4

11.

Write a function to group items into a dict by language.

```python
class Library(TypedDict):
    language: str
    name: str


libraries: List[Library] = [
    { "language": "java", "name": "Spring" },
    { "language": "javascript", "name": "React" },
    { "language": "javascript", "name": "Svelte" },
    { "language": "python", "name": "Django" },
    { "language": "python", "name": "Flask" },
    { "language": "ruby", "name": "Ruby on Rails" },
]


def group_by_lang(xs: List[Library]) -> Dict[str, List[Library]]:
    """Returns a dictionary of lists, keyed by language."""
    # Your code here
```

12.

```python
DEFAULT_FAVORITE_NUMBER = 1

def not_a_great_function(name, age, *, favorite_color, favorite_numbers=[]):
    # If the user hasn't selected any numbers, we'll use some defaults
    if not favorite_numbers:
        favorite_numbers.append(DEFAULT_FAVORITE_NUMBER)

    return User(
        name=name,
        age=age,
        favorite_numbers=favorite_numbers,
        favorite_color=favorite_color,
    )
```

Can you spot the errors in the code?

A. * is a syntax error
B. ‘favorite color’ must have a default value
C. ‘favorite numbers’ should not default to a list
D. Cannot evaluate ‘favorite_numbers’ list as boolean

13.

```
x = (
     "Lorem ipsum dolor sit amet, "
     "consetetur sadipscing elitr, sed diam nonumy eirmod "
     "tempor invidunt ut labore et dolore magna aliquyam "
     "erat, sed diam voluptua. "
)
```

What is the type of x?

A. tuple
B. str
C. Syntax error
D. List

14.

```
from random import randint


def main():
    """Returns a list of integers, sometimes 1-5, otherwise 1-10."""
    x = (1, 2, 3, 4, 5)

    if randint(1, 10) >= 7:
        x.extend(6, 7, 8, 9, 10)

    return x
```

Can you identify the issues in the code?

A. ‘extend’ takes an iterable, not variadic args
B. ‘randint’ does not accept parameters

C. ‘extend’ is not a method, should use ‘push’ instead
D. Should use list instead of tuple

15.

```python
# models.py
class State(models.Model):
    name = models.TextField()
    abbreviation = models.CharField(max_length=3)
    country_abbreviation = models.CharField(max_length=3)


class Location(models.Model):
    street_address = models.TextField(null=True)
    postal_code = models.CharField(max_length=32, null=True)
    city = models.TextField(null=True)
    state = models.ForeignKey(State, on_delete=models.CASCADE)


class Event(models.model):
    title = models.TextField()
    location = models.ForeignKey(Location, on_delete=models.CASCADE)
    created_by = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.SET_NULL, null=True)

# views.py
class EventListView(ListView):
    model = Event
    paginate_by = 25

    def get_queryset(self, **kwargs):
        return YOUR_QUERYSET_SELECTION


# myapp/event_list.html
{% block content %}
<h3>Upcoming Events</h3>
<ul>
{% for event in object_list %}
  <li>{{event.title}} ({{event.location.city}}, {{event.location.state.abbreviation}})</li>
{% empty %}
  <li>No Events</li>
{% endfor %}
</ul>
{% endblock content %}
```

Which Queryset would be the most efficient?
A. return super().get_queryset()

DEVS-MENTORING

B. models.Event.objects.select_related('location', 'state')
C. super().get_queryset().select_related('location', 'location__state')
D. super().get_queryset().prefetch_related('location')

16.

Craft model class for Employees.

```
# Requirements
# This is an Employee record which will be linked to a User
# We want to track information such as,
# - Company
# - User who hired this employee
# - Hire date
# - Start date
# - End date
# - Status
# - Employment type (part time, full time, etc)
# - Anything else you think might be important!
```

17.

```
@require_http_methods(["POST"])
def edit_profile_view(request):
    if not request.user or not request.user.is_authenticated:
        return HttpResponseForbidden()

    user = request.user

    user.first_name = request.POST.get("first_name", user.first_name)
    user.last_name = request.POST.get("last_name", user.last_name)
    user.email = request.POST.get("email", user.email)
    user.updated_at = datetime.now()
    user.save()

    return render(
        request,
        "templates/edit_profile.html",
        context=dict(),
    )
```

Which improvements could be made?

A. Use django timezone util

B. Add @login_required decorator
C. Specify update_fields on save
D. Django requires the use of a View Class