نمونه سوالات آزمون میانترم

مبانی برنامه‌سازی (پایتون)
استاد درس: مرضیه صدری
گردآورندگان: امین هاشمی، پارسا نادری

دانشگاه صنعتی شریف
نیمسال: بهار ۱۴۰۴
دانشکده: مهندسی کامپیوتر

پرسش ۰ [ساختار ماشین]

(آ) کامپیوتر از چند واحد منطقی تشکیل شده است؟ نام تمامی این واحدها را ذکر کرده و هر کدام را توضیح دهید.

پاسخ. کامپیوتر از شش واحد منطقی اصلی تشکیل شده است که هر کدام نقش خاصی در پردازش داده‌ها و اجرای برنامه‌ها دارند:

واحد ورودی (Input Unit) وظیفه: دریافت داده‌ها و دستورها از محیط بیرونی (کاربر یا سایر سیستم‌ها) و انتقال آن‌ها به کامپیوتر برای پردازش. مثال‌ها: صفحه‌کلید، ماوس، میکروفون، اسکنر. توضیح: این واحد داده‌های قابل خواندن توسط انسان را به فرم قابل فهم برای ماشین تبدیل می‌کند.

واحد خروجی (Output Unit) وظیفه: دریافت داده‌های پردازش‌شده و ارائه آن‌ها به صورت قابل درک برای انسان یا ارسال به سیستم‌های دیگر. مثال‌ها: نمایشگر، چاپگر، بلندگو. توضیح: نتایج را به کاربر نمایش می‌دهد یا برای دستگاه‌ها و شبکه‌های دیگر ارسال می‌کند.

واحد حافظه (Memory Unit) وظیفه: ذخیره موقت داده‌ها، دستورها و نتایج میانی برای دسترسی سریع حین اجرا. توضیح: اطلاعات آن با قطع برق از بین می‌رود (حافظه فرّار). سرعت بالا و ظرفیت محدود دارد. شامل RAM و cache می‌باشد.

واحد حساب و منطق (Arithmetic and Logic Unit - ALU) وظیفه: انجام عملیات ریاضی (جمع، تفریق، ضرب و تقسیم) و عملیات منطقی (مقایسه‌ها مانند بزرگتر، مساوی، کوچکتر). توضیح: این واحد نقش مغز محاسبات کامپیوتر را دارد و تصمیم‌گیری‌های منطقی را انجام می‌دهد.

واحد پردازش مرکزی (Central Processing Unit - CPU) وظیفه: هماهنگ‌سازی و نظارت بر عملکرد دیگر واحدها. توضیح: به عنوان "مغز کامپیوتر" شناخته می‌شود. شامل ALU و واحد کنترل است. مدیریت زمان ورود داده، انجام محاسبات و ارسال خروجی را بر عهده دارد.

واحد ذخیره‌سازی ثانویه (Secondary Storage Unit) وظیفه: نگهداری طولانی‌مدت داده‌ها و برنامه‌هایی که در حال حاضر استفاده نمی‌شوند. مثال‌ها: هارد دیسک، SSD، CD، DVD، فلش مموری. توضیح: حافظه‌ای دائمی و غیرفرّار است (اطلاعات با خاموش شدن کامپیوتر از بین نمی‌روند). ظرفیت بالا و سرعت کمتر نسبت به حافظه اصلی دارد.

(ب) واحد حساب و منطق (ALU) و واحد پردازش مرکزی (CPU) را تعریف کرده، تفاوت‌های آن‌ها را بیان کنید. همچنین یک نکته در مورد ساختار این دو در کامپیوترهای امروزی بنویسید.

پاسخ. **واحد حساب و منطق** (ALU - Arithmetic and Logic Unit) کارکرد: این واحد وظیفه انجام محاسبات ریاضی (مثل جمع، تفریق، ضرب، تقسیم) و همچنین عملیات منطقی (مثل مقایسه، بزرگ‌تر، مساوی، کوچکتر و...) را بر عهده دارد. نقش: مانند بخش "تولیدی" در یک کارخانه، عملیات اصلی پردازش داده‌ها را انجام می‌دهد.

واحد پردازش مرکزی (CPU - Central Processing Unit) کارکرد: این واحد به‌عنوان "مغز کامپیوتر" شناخته می‌شود و تمام فعالیت‌های دیگر بخش‌ها (ورودی، خروجی، حافظه و ALU) را هماهنگ و کنترل می‌کند. نقش: تصمیم می‌گیرد که چه داده‌ای، چه زمانی و به کجا منتقل شود؛ چه محاسبه‌ای انجام شود و چه خروجی‌ای تولید گردد.

نکته: در کامپیوترهای مدرن، واحد ALU به‌طور معمول بخشی از CPU است. یعنی CPU درون خود ALU را به‌عنوان یکی از اجزای اصلی دارد و از آن برای انجام عملیات ریاضی و منطقی استفاده می‌کند.

(ج) واحد حافظه (Memory Unit) و واحد ذخیره‌سازی ثانویه (Secondary Storage Unit) را تعریف کرده، تفاوت‌های آن‌ها را توضیح دهید.

پاسخ. **واحد حافظه** (Memory Unit): حافظه‌ی اصلی یا اولیه‌ی کامپیوتر است که داده‌ها و دستورهایی را که توسط کاربر وارد شده‌اند به‌صورت موقتی نگهداری می‌کند. اطلاعات موجود در این حافظه فرّار (volatile) هستند، یعنی با خاموش شدن کامپیوتر از بین می‌روند. این حافظه دسترسی سریع دارد ولی ظرفیت آن نسبت به حافظه ثانویه کمتر است. گاهی به آن حافظه اصلی (Primary Memory) نیز گفته می‌شود.

واحد ذخیره‌سازی ثانویه (Secondary Storage Unit): جایی برای ذخیره‌سازی دائمی داده‌ها و برنامه‌هایی است که در حال حاضر مورد استفاده قرار نمی‌گیرند. اطلاعات در این حافظه پایدار (persistent) هستند، یعنی با خاموش شدن سیستم از بین نمی‌روند. دسترسی به این حافظه کندتر از حافظه اصلی است ولی ظرفیت آن بسیار بیشتر و هزینه‌اش پایین‌تر است. مثال‌ها: هارد دیسک، SSD، CD، DVD، فلش مموری.

(د) با اینکه حافظه اصلی (Primary Memory) بسیار سریع‌تر از حافظه ثانویه (Secondary Storage) است، چرا در کامپیوترها از حافظه ثانویه نیز استفاده می‌شود و نمی‌توان فقط از حافظه اصلی استفاده کرد؟

پاسخ. با وجود اینکه حافظه اصلی سرعت بسیار بالایی دارد، اما دلایل زیر باعث می‌شود نتوان از آن به‌تنهایی استفاده کرد:

- **ظرفیت پایین:** حافظه اصلی معمولاً ظرفیت محدودی دارد و نمی‌تواند حجم بالایی از داده‌ها و برنامه‌ها را در خود جای دهد.
- **فرّار بودن:** اطلاعات موجود در حافظه اصلی با خاموش شدن کامپیوتر از بین می‌رود، بنابراین برای نگهداری دائمی اطلاعات مناسب نیست.

- **هزینه بالا:** ساخت حافظه‌های سریع مانند RAM نسبت به حافظه‌های ثانویه هزینه بسیار بیشتری دارد.

در نتیجه در معماری کامپیوتر از ترکیب حافظه اصلی و حافظه ثانویه استفاده می‌شود تا هم **سرعت** (در زمان اجرا)، هم **پایداری** (در ذخیره‌سازی طولانی‌مدت) و هم **صرفه‌جویی اقتصادی** تأمین شود.

(ه) چند نوع زبان برنامه‌نویسی وجود دارد؟ نام ببرید و هر یک را توضیح دهید.

پاسخ. بر اساس دسته‌بندی رایج، سه نوع اصلی زبان برنامه‌نویسی وجود دارد:

زبان ماشین (Machine Language)**:** تعریف: زبان طبیعی سخت‌افزار کامپیوتر است و مستقیماً توسط پردازنده قابل درک می‌باشد. ویژگی‌ها:

- از رشته‌های صفر و یک تشکیل شده است.
- وابسته به معماری پردازنده (غیر قابل حمل بین سیستم‌ها).
- نوشتن آن برای انسان بسیار سخت و ناخوانا است.

زبان اسمبلی (Assembly Language)**:** تعریف: زبان سطح پایین که با استفاده از علائم و کلمات اختصاری انگلیسی‌مانند، عملیات پایه را مشخص می‌کند. ویژگی‌ها:

- نسبت به زبان ماشین قابل فهم‌تر است.
- نیاز به اسمبلر (Assembler) برای تبدیل به زبان ماشین دارد.
- همچنان وابسته به معماری سخت‌افزار است (غیر قابل حمل).

زبان‌های سطح بالا (High-Level Languages)**:** تعریف: زبان‌هایی نزدیک به زبان انسان که برای برنامه‌نویسی راحت و قابل فهم طراحی شده‌اند. ویژگی‌ها:

- قابل حمل بین پلتفرم‌های مختلف (معمولاً مستقل از پردازنده).
- از کامپایلر یا مفسر برای اجرا استفاده می‌کنند.
- برای وظایف پیچیده تنها با چند خط کد قابل استفاده هستند.

مثال‌ها: Fortran Basic، Pascal، Java، C، Python،

(و) کامپایلر (Compiler) و مفسر (Interpreter) را تعریف کرده، تفاوت آن‌ها را بیان کنید. پایتون در کدام دسته قرار می‌گیرد؟ همچنین این دو روش را از نظر سرعت اجرا و سادگی در debug مقایسه کنید.

پاسخ. **کامپایلر :(Compiler)** برنامه‌ای است که کل کد منبع (source code) را یک‌باره به زبان ماشین ترجمه کرده و فایل اجرایی تولید می‌کند. ویژگی‌ها:

- ترجمه کامل قبل از اجرا انجام می‌شود.
- سرعت اجرای برنامه بالا است چون کد نهایی مستقیم اجرا می‌شود.
- خطاها فقط هنگام کامپایل گزارش می‌شوند (نه حین اجرا).

مفسر :(Interpreter) برنامه‌ای است که کد منبع را خط‌به‌خط می‌خواند، ترجمه می‌کند و همان لحظه اجرا می‌کند. ویژگی‌ها:

- نیازی به فایل اجرایی ندارد.
- سرعت اجرا پایین‌تر است چون ترجمه و اجرا هم‌زمان انجام می‌شود.
- خطاها به‌صورت زنده (real-time) شناسایی می‌شوند، که دیباگ کردن را آسان‌تر می‌کند.

پایتون در کدام دسته است؟ پایتون یک زبان **مفسری (Interpreted)** است، یعنی توسط مفسر اجرا می‌شود. البته در عمل ترکیبی از کامپایل و تفسیر دارد، اما رفتار آن بیشتر شبیه به زبان‌های مفسری است.

پرسش ۱ [شرط]

(آ) خروجی عبارت زیر چه مقداری خواهد بود؟

```
print((6-2)/(5*(1+4))+3)
```

`3.16` √ `12.0` ◯ `5.0` ◯ `50.0` ◯

(ب) خروجی عبارت مقایسه‌ای `"small" > "big"` کدام یک از موارد زیر خواهد بود؟

`True` ◯ **`False`** √ `big` ◯ `small` ◯

(ج) ارزش عبارت مقایسه‌ای در `if` چیست؟

```
n = 4
if n*6 > n**2 or n%2 == 0:
    print("Check")
```

`True` √ $24 > 16$ `or` 0 ◯ `Check` ◯ `False` ◯

(د) عبارت `result` بعد از اجرای کد زیر چیست؟

```
x = 6
result = "High" if x**2 > 40 else "Equal" if x**2 == 36 else "Low"
```

`Error` ◯ `Low` ◯ **`Equal`** √ `High` ◯

(ه) خروجی قطعه‌کد زیر چیست؟

```
x = 10
if x > 15:
    print("Greater than 15")
elif x > 5:
    print("between 5 and 15")
else:
    print("less than 5")
```

◯ بزرگتر از ۱۵ √ بین ۵ تا ۱۵ ◯ کوچکتر یا مساوی ۵ ◯ خطا دارد

(و) خروجی کد زیر چه مقداری خواهد بود؟

```
trait = "wise"
house = "Gryffindor" if trait == "brave" else "Hufflepuff" if trait == "loyal"
    else "Ravenclaw" if trait == "wise" else "Slytherin"
print(house)
```

`Slytherin` ◯ **`Ravenclaw`** √ `Hufflepuff` ◯ `Gryffindor` ◯

(ز) برنامه‌ای بنویسید که دو عدد صحیح را از کاربر بگیرد و تعیین کند که آیا یکی از آن‌ها مضرب دیگری هست یا خیر. اگر هست، "مضرب هستند" و اگر نه، "مضرب نیستند" چاپ کند.

پاسخ.

```
a = int(input())
b = int(input())

if a % b == 0 or b % a == 0:
    print("divisible")
else:
    print("not divisible")
```

(ح) برنامه‌ای بنویسید که یک عدد صحیح از کاربر گرفته و بررسی کند عدد مثبت است، منفی است یا صفر. سپس تعیین کند که فرد است یا زوج، و همه نتایج را چاپ کند.

پاسخ.

```
n = int(input())

if n > 0:
    print("positive")
elif n < 0:
    print("negative")
else:
    print("zero")

if n % 2 == 0:
    print("even")
else:
    print("odd")
```

پرسش ۲ [حلقه]

(آ) خروجی کد زیر کدام گزینه است؟

```
s = "HelLo WORld"
old, new = "l", "o"

s = s.replace(old, new).lower()
result = ''
i = 0
while i < len(s):
    result += s[i].upper() if i % 2 == 0 else s[i]
    i += 1
print(result)
```

HeOlO WoRoD √ hEoLo wOrOd ◯ HEOLO WOROD ◯ HeOlO WoRlD ◯

(ب) برنامه‌ی زیر چه کاری را انجام می‌دهد؟

```
num = int(input())

result = 0

while num:
    r = num % 10
    result = result * 10 + r
    num = num // 10

print(result)

```

پاسخ. یک عدد را مقلوب می‌کند. (برعکس نمیکند!)

(ج) توضیح دهید چرا در زبان برنامه نویسی پایتون، حلقه `while` کند تر از حلقه `for` اجرا می‌شود.

پاسخ. در حلقه‌ی `while` هر دوی شمارنده و بررسی کننده‌ی حلقه در پایتون نوشته شده‌اند اما در حلقه‌ی `for` شمارنده در زبان C نوشته شده.

(د) کد زیر به درستی اجرا نمی‌شود. خطا(های) آنرا پیدا کنید و آن(ها) را اصلاح کنید. همچنین مشخص کنید خطاهای فعلی، منجر به رخ‌دادن چه نوع پیغام خطایی می‌شوند.

```
i = 1
Num = input()
sum == 0
for i in range(1, Num):
    sum += i
    if(Num % 2 == 0):
        print(Num * i)
    else:
        print(Num + i)
```

پاسخ. در خط ۳، از علامت مقایسه به جای انتساب استفاده شده که منجر به `NameError` خواهد شد.
همچنین متغیر `Num` از نوع رشته است اما با آن به عنوان عدد برخورد شده که منجر به خطای `TypeError` خواهد شد.
کد صحیح شده به صورت زیر است:

```
i = 1
Num = int(input())
sum = 0
for i in range(1, Num):
    sum += i
    if(Num % 2 == 0):
        print(Num * i)
    else:
        print(Num + i)
```

پرسش ۳ [تابع]

(آ) تابعی به نام `grade_status` بنویسید که یک نمره بین ۰ تا ۲۰ دریافت کرده و با توجه به جدول زیر، وضعیت را چاپ کند:

- نمره کمتر از ۱۰: مردود

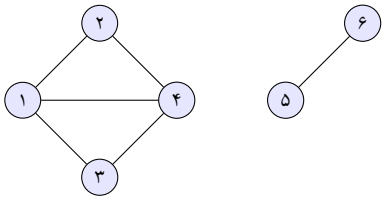
- از ۱۰ تا ۱۴: قابل قبول
- از ۱۴ تا ۱۷: خوب
- بالای ۱۷: عالی

پاسخ.

```
def grade_status(score):
    if score < 10:
        print("failed")
    elif score < 14:
        print("passed")
    elif score < 17:
        print("good")
    else:
        print("perfect")
```

(ب) نقشه‌ی شهرها (گره) و جاده‌های (یال) سرزمین **خیلی خیلی دور** به صورت یک گراف با نمایش لیست مجاورت به شما داده شده است. مشخص کنید حداقل چه تعداد جاده‌ی جدید لازم است کشیده شود تا تضمین شود از هر شهر به شهر دیگر راهی وجود دارد. مثالی از لیست مجاورت را در زیر مشاهده می‌کنید:

```
adjacency_list = {
    1: [2, 3, 4], 2: [1, 4], 3: [1, 4],
    4: [1, 2, 3], 5: [6], 6: [5]
}
```



پاسخ. برای این سؤال، می‌خواهیم حداقل تعداد جاده‌های جدید لازم برای متصل کردن تمام شهرها را پیدا کنیم. این مسئله به یافتن تعداد **مولفه‌های همبند** گراف مربوط است. برای این کار، با استفاده از DFS تعداد مولفه‌های همبند را می‌شماریم. اگر گراف دارای k مولفه همبند باشد، حداقل $k-1$ جاده جدید نیاز است.

شبه کد

```
Input: adjacency_list (a dictionary where keys are nodes, and values are lists of connected nodes)
Output: Minimum number of new roads to connect all cities

1. Initialize a set `visited` to keep track of visited cities (nodes).
2. Define a helper function `dfs(node)`:
   a. Mark `node` as visited.
   b. For each neighbor of `node` in adjacency_list:
      i. If the neighbor is not visited, call `dfs(neighbor)` recursively.
3. Initialize `connected_components` to 0.
4. For each city (node) in adjacency_list:
   a. If the city is not visited:
      i. Increment `connected_components`.
      ii. Call `dfs(city)`.
5. Return `connected_components - 1`.
```

کد زیر الگوریتم فوق را پیاده‌سازی می‌کند:

```python
def minimum_roads_to_connect(adjacency_list):
    visited = []

    def dfs(node):
        visited.append(node)
        for neighbor in adjacency_list.get(node, []):
            if neighbor not in visited:
                dfs(neighbor)

    connected_components = 0
    for city in adjacency_list:
        if city not in visited:
            connected_components += 1
            dfs(city)

    return connected_components - 1
```

پ.ن: استفاده از BFS نیز بلامانع است. سعی کنید خودتان با BFS پیاده سازی کنید.

(ج) راهنمای نگارش مستندات PEP 257 استاندارد پایتون برای نوشتن داک‌استرینگ‌ها است که برای مستندسازی ماژول‌ها، کلاس‌ها و توابع استفاده می‌شود. یک داک‌استرینگ بین یک رشته چند خطی که با سه علامت نقل قول شروع می‌شود قرار می‌گیرد و باید دارای بخش‌های ساختار یافته‌ای برای کاربرد تابع، آرگومان‌های تابع و مقادیر بازگشتی باشد. در این سوال، ابتدا شما برای تابع زیر یک داک‌استرینگ نوشته و سپس به چند پرسش دیگر پاسخ خواهید داد.

```python
def mystery_function(a=10, b=20, *args, flag=True, **kwargs):
    """
    Goal:

    Parameters:
        a (     ):
        b (     ):
        *args:
        flag (      ):
        **kwargs: Named arguments for customization.

    Returns:
        (      or      ):
    """
    if flag:
        result = a + b + sum(args)
    else:
        result = a * b
        for num in args:
            result *= num

    scale = kwargs.get('scale', 1)
    result *= scale

    if kwargs.get('format') == 'hex':
        return hex(result)

    return result
```

پرسمان ۱ در یک خط، مشخص کنید هدف از کد فوق چیست. (خط سوم کد را کامل کنید.)

پرسمان ۲ برای مستندسازی پارامترها، ابتدا نام متغیر را ذکر کنید، سپس نوع داده (type) را مشخص نمایید و هدف آن پارامتر را به‌طور خلاصه توضیح دهید. اگر مقدار پیش‌فرضی دارد، آن را نیز ذکر کنید. (خط ۶ تا ۹ کد.)

پرسمان ۳ برای مستندسازی مقادیر بازگشتی، نوع داده (type) مقدار بازگشتی را مشخص کرده و توضیح دهید که این مقدار یا مقادیر چه اطلاعاتی را ارائه می‌دهد و در چه شرایطی بازگردانده می‌شود. (خط ۱۳ کد را کامل کنید)

پرسمان ۴ خروجی `mystery_function(2, flag=False, format='hex', scale=3)` چیست؟

پرسمان ۵ خطایی در کد پیدا کنید که منجر به `TypeError` شود. تابع را طوری فراخوانی کنید که خطای بیان شده رخ دهد. (راهنمایی: `TypeError` زمانی رخ می‌دهد که در یک عملیات، از نوع داده‌های نامناسب استفاده کنید.)

پرسش ۴ [رشته]

با کامل کردن جای خالی، قطعه کد مربوط به هر سوال را در یک خط کامل کنید. (راهنمایی: با استفاده از `()join` می‌توانید نتیجه‌ی نهایی را به صورت یک رشته خروجی بگیرید)

(آ) چگونه می‌توانیم تمامی حروف کوچک یک رشته `s` را به حروف بزرگ و حروف بزرگ را به حروف کوچک تبدیل کنیم؟

```python
swapped = ''.join(c.lower() if c.isupper() else c.upper() for c in s)
```

(ب) چگونه می‌توانیم حروف صدادار (`'aeiou'`) را از یک رشته `s` حذف کنیم؟

```python
no_vowels = ''.join(c for c in s if c not in 'aeiou')
```

(ج) چگونه می‌توانیم بررسی کنیم که آیا یک رشته‌ی `s` واروخوانه (palindrome) هست یا خیر؟ (نوشتن عبارت شرطی کفایت می‌کند)

```python
_____s_____ == ____s[::-1]____
```

(د) چگونه می‌توانیم فاصله‌های اضافی ابتدا و انتهای رشته‌ی `s` را حذف کرده، و سپس تمامی فاصله‌های وسط را با یک خط‌تیره جایگزین کنیم؟

```python
formatted = s.strip().replace(' ', '-')
```

(ه) (درست/نادرست) رشته‌ی `s = "hello"` با `()s.capitalize` به `"HELLO"` تبدیل می‌شود ◯ درست √ نادرست

(و) برای بررسی اینکه آیا یک رشته‌ی `s` فقط از حروف کوچک تشکیل شده، کدام شرط صحیح است؟

```python
___s.islower()___
```

(ز) چگونه می‌توانیم تمام کاراکترهای رشته‌ی `s` را همراه با شماره‌ی ایندکس‌شان در قالب `(index, char)` لیست کنیم؟

```python
indexed = list(enumerate(s))
```

(ح) اگر `s = "abc"` و بنویسیم `s*3`، نتیجه چیست؟
◯ "aabbcc" √ **"abcabcabc"** ◯ "abc3" ◯ خطا می‌دهد

(ط) با استفاده از اسلایس، چگونه می‌توان از رشته‌ی s که طول آن حداقل ۶ است، حروف زوج ایندکس را از ایندکس ۲ تا ۶ استخراج کرد؟

```
part = ____s[2:6:2]____
```

(ی) (درست/نادرست) برای رشته‌ی `s = 'a'`، عبارت `s*0` یک رشته‌ی شامل `'0'` بازمی‌گرداند.
√ نادرست ◯ درست

(ک) یک تابع بنویسید که یک رشته را به عنوان ورودی بگیرد و کاراکترهای آن را بر اساس تکرار آنها به ترتیب نزولی مرتب کند. در صورتی که دو کاراکتر دارای تعداد تکرار یکسانی باشند، باید به ترتیب ظاهر شدن در رشته مرتب شوند. مثال) ورودی:"programming"، خروجی: "rgmpoain"

پاسخ.

```
def sortCount(inString):
    countDict = {}
    for c in inString:
        if c in countDict:
            countDict[c] += 1
        else:
            countDict[c] = 1

    outString = ""
    for _ in range(len(countDict)):
        max_value = 0
        tempC = ""
        for count in countDict:
            if countDict[count] > max_value:
                tempC = count
                max_value = countDict[count]

        outString += tempC
        del countDict[tempC]

    return outString
```

پرسش ۵ [ساختمان داده]

(آ) کدام گزینه خروجی قطعه کد زیر را به درستی نمایش می‌دهد؟

```
a = (10, 8, 16)
print(type(sorted(a)))
```

√ `list` ◯ `tuple` ◯ `TypeError` ◯ `(8, 10, 16)`

(ب) درستی و نادرستی عبارات زیر را با ذکر دلیل مشخص کنید.

۱. اگر از متد `()pop` روی یک لیست خالی استفاده شود، یک مقدار پیش‌فرض صفر برگردانده می‌شود.
۲. تابع `map` در پایتون بلافاصله لیستی از مقادیر را برمی‌گرداند.
۳. استفاده از کلیدهای mutable در دیکشنری مجاز نیست.
۴. در پایتون متد `append` برای اضافه کردن چند عنصر به یک لیست به صورت همزمان استفاده می‌شود.
۵. عملیات `pop` همیشه آخرین مقدار اضافه شده را حذف می‌کند.

پاسخ.

۱ نادرست. IndexError

۲ نادرست.

۳ درست.

۴ نادرست.

۵ نادرست.

(ج) می‌دانیم تاپل یک داده ساختار immutable است. می‌خواهیم یک کپی از تاپل a داشته باشیم. به همین خاطر، آن را به b نسبت می‌دهیم. در قدم بعد، b را تغییر می‌دهیم اما متوجه می‌شویم a نیز تغییر کرده است با اینکه immutable بود!

```
a = ([1, 20], [0, 25])
b = a
b[1][0] = 2025
print(a, b)
```

۱. مقدار نهایی a , b را بعد از اجرای کد بنویسید.

پاسخ.

```
a,b = ([1, 20], [2025, 25]), ([1, 20], [2025, 25])
```

۲. توضیح دهید چرا a تغییر کرده است.

پاسخ. در پایتون، زمانی که a را به b انتساب می‌دهیم، در حقیقت یک نسخه از تاپل را کپی نمی‌کنیم، بلکه یک مرجع جدید به همان محل حافظه‌ای که تاپل اصلی a در آن قرار دارد ایجاد می‌کنیم. این به این معنی است که هم a و هم b به همان شیء تاپل در حافظه اشاره می‌کنند. در حالی که تاپل‌ها خود تغییرناپذیر (immutable) هستند، می‌توانند اشیاء تغییرپذیر مانند لیست‌ها را درون خود داشته باشند. از آنجا که b[1] به همان لیست اشاره می‌کند که a[1]، تغییر مقدار b[1][0] باعث تغییر a[1][0] نیز می‌شود.

۳. راه حلی ارائه دهید تا در انتهای اجرای کد فوق، محتوای a, b یکسان نباشد. (راهنمایی: استفاده از توابع بازگشتی را برای مدیریت اشیای mutable در ساختارهای immutable در نظر بگیرید. ممکن است لازم باشد نوع هر عنصر را بررسی کنید تا نحوه‌ی کپی کردن آن را تعیین کنید.)

پاسخ.

```
def deep_copy(obj):
    if type(obj) == tuple:
        return tuple(deep_copy(x) for x in obj)
    elif type(obj) == list:
        return [deep_copy(x) for x in obj]
    elif type(obj) == dict:
        return {key: deep_copy(value) for key, value in obj.items()}
    else:
        return obj
```

استفاده از ماژول copy و تابع deepcopy پایتون نیز بلامانع است