میانگین تاپل ها

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

تاپل دوبعدی به تاپلی می‌گوییم که اعضای آن هم خود، تاپل هستند. در پایین یک تاپل دو بعدی 3×4 را می‌بینید:

```
my_tuple = ((10, 10, 10), (30, 45, 56), (81, 80, 39), (1, 2, 3))
```

تابعی به نام `average_tuple` تعریف کنید که یک تاپل $n \times m$ دریافت کند و یک تاپل یک بعدی با m عضو برگرداند، به نحوی که عضو iم آن میانگین اعضای iم تاپل‌های درونی پارامتر ورودی باشد.

در این صورت با فراخوانی `average_tuple(my_tuple)` مقدار برگشتی تابع به صورت زیر خواهد بود:

```
(30.5, 34.25, 27.0)
```

مثلا عضو اول تاپل برگشتی به صورت زیر محاسبه شده است:

```
( 10 + 30 + 81 + 1 ) / 4 = 30.5
```

راهنمایی:

تابع `zip()` در پایتون تابعی است که چند تاپل یا لیست را در ورودی می‌گیرد و یک شیء zip که آرایه ای از تاپل ها است را در خروجی برمیگرداند؛ که تاپل اول، شامل اعضای اول چند تاپل ورودی، تاپل دوم، شامل دومین عضو های چند تاپل ورودی و ... است. برای دیدن مثالی از آن این لینک را ببینید.

توجه

با کمک تابع های `zip` و `map` و همچنین تابع بدون نام `lambda` و عملگر * برای unpacking باید بتوانید بدنه تابع `average_tuple` را در یک خط تعریف کنید. در غیر این صورت پاسخ شما قابل قبول نخواهد بود.

ورودی

تابع `eval` در پایتون یک عبارت دستوری (expression) را به صورت رشته به عنوان ورودی دریافت می‌کند و مقدار معادل آن آن را برمی‌گرداند. این لینک را ببینید.

ورودی شما در این سوال یک رشته است که حاوی کدهای قابل evaluation پایتون است. در این سوال ممکن است رشته ورودی فراخوانی تابع `average_tuple` با آرگومانهای مختلف باشد.

فرمت کد

فرمت پاسخ ارسالی شما برای این سوال باید به شکل زیر باشد.

```
def average_tuple(two_dimentional_tuple):
    '''write an appropriate docstring here'''
    # write the rest of definition bellow

# your submission must include following lines of codes
if __name__ == '__main__':
    expression_to_evaluate = input()
    print(eval(expression_to_evaluate))
```

مثال

ورودی نمونه ۱

```
average_tuple(((10, 10, 10), (45, 56, 10), (81, 39, 10), (1, 2, 3)))
```

خروجی نمونه ۱

```
(34.25, 26.75, 8.25)
```

XO

پایتون برای برنامه نویسی گرافیکی دارای ابزارهای متعددی هست. یکی از معروفترین ابزارها که به صورت built-in همراه پایتون هست، ماژول tkinter است. در tkinter هرچیزی به صورت یک widget تعریف می شود. مثلا برای قرار دادن متن در صفحه یک widget لازم داریم، برای قرار دادن یک button در صفحه نیاز به widget دیگری داریم و ... .

برای شروع به کار با این ماژول ابتدا باید این ماژول را import کنیم. سپس باید ویجت اصلی یا root را ایجاد کنیم. ویجت اصلی، پنجره اصلی برنامه شما را ایجاد می کند.

خوب، برای نوشتن برنامه مربوط به این سوال خطوط اول کد شما به صورت زیر باید باشد:

```
from tkinter import *
root = Tk()
root.title('Tic-Tac-Toe')
```

حدس زدن اینکه خط سوم چه کاری انجام می دهد خیلی سخت نیست :)

برای اینکه بتوانیم روی صفحه برنامه پیغام هایی نشان بدهیم باید از ویجت messagebox استفاده کنیم. برای همین لازم هست آن را به صورت زیر import کنیم:

```
from tkinter import messagebox
```

در این سوال می خواهیم بازی معروف XO یا Tic Tac Toe را به صورت گرافیکی پیاده سازی کنیم. در شکل ساده این بازی یک صفحه 3 × 3 است. بنابراین باید برای پیاده سازی گرافیک این بازی 9 تا button روی صفحه ایجاد کنیم. در tkinter یک ویجت به اسم Button وجود دارد. خیلی راحت با استفاده از آن می توانیم یک button تعریف کنیم:

```
b1 = Button(root, text=" ", font=("Helvetica", 20), height=3, width=6,
 bg="white", command=lambda: print('b1 clicked'))
```

سازنده Button با تعدادی آرگومان positional و keyword فراخوانی می شود تا یک button به نام b1 بسازد. در آرگومان اول باید به آن اعلام کنیم این دکمه قرار است در کدام صفحه ایجاد شود (صفحه

root) و بعد با یک تعداد آرگومان keyword تنظیم می کنیم که متنی که قرار است در این button نمایش داده شود چیست، با چه فونت و سایز فونتی نمایش داده شود، طول و عرض این button چه قدر باشد، رنگ آن چه باشد و نهایتا اینکه اگر روی آن کلیک شد چه اتفاقی بیفتد. فعلا اینجا خیلی ساده نوشته ایم که در صورت کلیک کردن روی این button عبارت `b1 clicked` چاپ شود. بعدا شما باید این قسمت را تکمیل کنید.

برای تکمیل این کد لازم هست شما 8 دکمه دیگر با اسمهای b2 تا b9 به همین ترتیب ایجاد کنید.

(البته می توانید با سلیقه خودتان، button ها را به شکل دیگری ایجاد کنید)

بعد از ایجاد این 9 ویجت button ، باید آنها را در یک شبکه 9 تایی (3×3) روی پنجره اصلی جانمایی کنیم. برای اینکار از متود `grid` استفاده می کنیم:

```
b1.grid(row=0, column=0)
```

به این شکل صفحه root به صورت یک شبکه در نظر گرفته می شود که b1 باید در سطر اول و ستون اول آن قرار بگیرد. به همین شکل می توانید 8 دکمه بعدی را هم جانمایی کنید.

تا اینجا شکل کل پنجره اصلی ایجاد شده. برای اینکه این پنجره اصلی روی صفحه باقی بماند باید یک حلقه همیشه درست (while True) داشته باشیم. در tkinter تابعی به نام `mainloop()` این کار را انجام می دهد.

```
root.mainloop()
```

برای تکمیل کد و ساخت این بازی ساده چند مرحله دیگر باقی مانده که باید انجام بدهید:

۱. همان طور که توضیح داده شد باید مقدار آرگومان command در ویجت button را به نحوی تنظیم کنید که هنگام کلیک بر آن عملکر مناسبی داشته باشد. مقدار این آرگومان یک object از نوع function باید باشد که پس از کلیک شدن button آن function اجرا شود. در بازی tic tac toe دو بازیکن داریم. بازی توسط بازیکن X شروع می شود. بنابراین باید به نحوی command را تنظیم کنید که با کلیک روی یک دکمه در صورتی که نوبت بازیکن X است، حرف X و در غیر این صورت حرف O بر روی دکمه درج شود.

برای تغییر متنی که روی یک دکمه نوشته شده است (در حالت اولیه یک space بود) می توانید از دستور زیر استفاده کنید:

```
b1['text'] = 'X'
```

توجه کنید که اگر قبلا یک دکمه کلیک شده است، نباید مقدار آن تغییر کند.

۲. پس از پایان نوبت هر بازیکن، باید چک کنید که آیا این بازیکن برنده شده است یا نه. در صورتی که بازیکن برنده شده است، بازی باید تمام شود و پیغام پایان بازی به کاربر نشان داده شود. همچنین باید همه button ها در وضعیت غیرفعال قرار بگیرند. در غیر این صورت بازی تا کلیک شدن همه button ها ادامه پیدا می کند. در صورتی که بازی تا انتها رفت و برنده ای نداشت، پیغام مناسب را به کاربر نشان دهید.

برای قرار دادن یک button (مثلا b1) در وضعیت غیر فعال از دستور زیر استفاده کنید:

```
b1['status'] = DISABLED
```

۳. پیغامهای لازم را در طول بازی می توانید با ویجت messagebox نمایش دهید. مثلا اگر بازی هیچ برنده ای نداشت، پیغام زیر را نمایش دهید.

```
messagebox.showinfo("Tic Tac Toe", "It's a tie!!\nNo one wins!\n Reset the ga
```

۴. تا همین جا برای پاسخ به این سوال کافی است. برای تکمیل کد می توانید بخشهای دیگری به کدتان اضافه کنید. مثلا اینکه زمان سپری شده از ابتدای بازی نمایش داده شود، رنگ خانه های بازیکن پیروز، پس از پیروزی تغییر کند. ابعاد صفحه قابل تغییر باشد، مشخص باشد که نوبت چه بازیکنی هست، دو بازیکن بتوانند به تعداد مختلف بازی را انجام بدهند و امتیاز بگیرند، وجود یک دکمه reset که بازی از ابتدا شروع شود و ... پیاده سازی هر یک از این موارد اضافه تر دارای نمره امتیازی خواهد بود.

۵. یک فایل py برای پاسخ این سوال ارسال کنید. پاسخ های شما به صورت دستی بررسی می شوند. در ابتدای کد، یک کامنت چند خطی بنویسید و توضیح دهید بازی شما دارای چه option های اضافه تری هست.

تمرین مفهوم closure

سوال اول:

می خواهیم یک تابع averager تعریف کنیم به نحوی که بتواند میانگین چند عدد را حساب کند. نحوه کار این تابع باید به نحوی باشد که هر بار که این تابع با یک مقدار جدید فراخوانی می شود، مقدار میانگین به روز شود و مقدار جدید برگردانده شود.

به قطعه کد زیر توجه کنید:

```
numbers = []
def averager(number):
    global numbers
    numbers.append(number)
    total = sum(numbers)
    count = len(numbers)
    return total / count
```

سعی کنید یک تابع به نام averager بنویسید که همین کار را با استفاده از closure انجام دهد.

سوال دوم:

قطعه کد زیر زمان سپری شده پس از ایجاد یک object را نگهداری می‌کند.

```
import time
class Timer:
    def __init__(self):
        self.start = time.perf_counter()

    def elapsed(self):
        return round(time.perf_counter() - self.start)
```

برای دیدن نحوه کار کلاس Timer می توانید قطعه کد زیر را امتحان کنید.

```
x = Timer()
for i in range(5):

```

```
    print(x.elapsed())
    time.sleep(2)
```

می دانیم closure می تواند جایگزین مناسبی برای کلاسهای ساده و کوچک باشد. یک closure به اسم timer بنویسید که کاری مشابه کار کلاس Timer انجام بدهد. یعنی به ازای اجرای قطعه کد زیر باید خروجی مشابه به حالت فوق باشد:

```
import time
x = timer()
for i in range(5):
    time.sleep(2)
    print(x())
```

سوال سوم:

توضیح دهید closure چیست و چه کاربردی در برنامه نویسی دارد. (می توانید به صورت کامنت در فایل ارسالی بنویسید)

این سوال دارای تست کیس نیست.

حافظه نیما

بخش انتهایی این سوال در ساعت 1:00 پیش از ظهر روز 1400/12/5 به روز شده است.

نیما، یک دانشجوی سال اولی است که می خواهد 504 کلمه کتاب معروف زبان را حفظ کند.

در مرحله اول کلمات زیر را حفظ می کند:

```
abandon : leave behind, desert
keen : sharp, eager, sensitive
```

در مرحله دوم کلمات دیگری را حفظ می کند:

```
tact : ability to say the right thing
vacant : empty, not filled
hardship : difficulty
```

اما حافظه نیما یک اشکالی دارد. همین که سری دوم کلمات را حفظ می کند، کلمات سری قبلی از یادش می رود! او می خواهد حافظه خودش را تمرین بدهد تا بتواند به نحوی این مشکل را برطرف کند. تمرین او به این شکل است که هر بار که کلمات جدیدی را حفظ می کند، باید کلماتی را که سری قبل حفظ کرده است را به یاد بیاورد.

حالا شما سعی کنید با استفاده از `closure` یک تابع به نام memorize بنویسید که تعدادی کلمه و معانی آنها را به صورت keyword بکیرد و در هر بار فراخوانی بگوید دفعه قبل با چه آرگومانهایی فراخوانی شده است.

مثلا اگر این تابع را به ترتیب زیر فراخوانی کنیم:

```
memorize(abandon = 'leave behind, desert',
         keen = 'sharp, eager, sensitive')

memorize(tact = 'ability to say the right thing',
         vacant = 'empty, not filled',
         hardship = 'difficulty')

```

```
8
memorize(gallant = 'brave')
```

خروجی باید به شکل زیر باشد:

```
None

abandon: desert, leave behind
keen: eager, sharp, sensitive

hardship: difficulty
tact: ability to say the right thing
vacant: empty, not filled
```

نکات پیاده سازی:

۱. تابع شما باید با تعداد دلخواه آرگومان قابل فراخوانی باشد.
۲. برای نمایش خروجی دقت کنید که کلمات هر سری باید به ترتیب حروف الفبا چاپ شوند، در صورتی که یک کلمه دارای چند معنی است، معانی در اولویت اول به ترتیب تعداد کاراکترها و در اولویت دوم بر اساس ترتیب حروف الفبا مرتب شوند.
۳. همه آرگومانها باید از نوع keyword باشند، مقدار تمام آرگومانها باید از نوع رشته باشد. معانی مختلف با یک کاما از هم جدا شوند.
۴. در خروجی پس از چاپ کلمات و معانی هر سری یک خط خالی وجود داشته باشد.
۵. در انتهای کد شما باید قطعه کد زیر وجود داشته باشد.

```
1 if __name__ == '__main__':
2  exec(input().replace('\\n', '\n'))
```

محاسبه ی sin(x)

امین در درس ریاضی1 با بسط تیلور توابع مثلثاتی آشنا شده است. او قصد دارد با کمک شما بسط تیلور برای محاسبه ی سینوس زوایا را پیاده سازی کند.

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!}$$

نکته: درصورت استفاده از توابع آماده نمره ی این تمرین را دریافت نخواهید کرد. در راه حل خود باید قطعه کد زیر را کامل کنید:

```
def factorial(n):
    #your code here
def sinus(n):
    def inner(x):
        #your code here
    return inner
```

ورودی

ورودی تنها شامل یک خط است که در آن عدد طبیعی n و عدد اعشاری x با فاصله از هم آمده است. که n تعداد جملات بسط است که باید محاسبه کنید و x زاویه ی مورد نظر به درجه است. با توجه به مجاز نبودن به استفاده از $math$ عدد پی را 3.14 درنظر بگیرید.

خروجی

در تنها خط خروجی مقدار sin چاپ میشود. قبل از چاپ کردن نتیجه پاسخ را تا 6 رقم اعشار گرد کنید.

مثال

ورودی نمونه

```
5 90
```

خروجی نمونه

```
1.000003
```

محاسبه ی Arcsin(x)

در این سوال شما باید تابع Arcsin(X) در ریاضیات را با استفاده از روش جست و جوی باینری بدست آورید. همانطور که میدانید این تابع معکوس سینوس است. Arcsin یک تابع صعودی در بازه ی دامنه ی -1 تا 1 است و برد آن در محدوده ی $\pi/2$ تا $-\pi/2$ میباشد. روش حل به این صورت است که شما یک بازه ی جست و جو درنظر میگیرید و سینوس نقطه ی وسط این بازه را بدست آورده و با x مقایسه میکنید و با توجه به نتیجه ی مقایسه بازه ی جست و جوی شما محدود تر میشود و نقطه ی ابتدا یا انتهای آن تغییر میکند. جست و جو را آن قدر ادامه میدهید تا طول بازه کمتر از e شود. هر چقدر عدد e کوچکتر باشد، جواب دقیق تر است. نکته: برای عدد پی و محاسبه ی سینوس از کتابخانه ی $math$ استفاده نمایید.

```python
import math
def Arcsinus(e):
    def inner(x):
        #your code here
    return inner
#your code here
Arcsin_e = Arcsinus(e)
print(Arcsin_e(x))
```

ورودی

در تنها خط ورودی e دقت محاسبه و x مقداری که باید $Arcsin$ آن را محاسبه کنید، داده میشود و هر دو میتوانند اعشاری باشند.

خروجی

در تنهاخط خروجی مقدار $Arcsin$ بعد از 4 رقم گرد شدن چاپ میشود.

مثال

ورودی نمونه

```
0.001 1
```

خروجی نمونه

```
1.57
```

ثبت نام در سایت

رضا قصد دارد یک سایت فروشگاهی طراحی کند. شما باید برای بخش ثبت نام و ورود کاربران در سایت به او کمک کنید. مشخصات مورد نیاز هر کاربر برای ثبت نام شامل نام کاربری و رمز ورود میباشد که شرایط لازم برای هر یک به صورت زیر است:

- باید حداقل 8 کاراکتر داشته باشد.
- باید شامل اعداد و حروف کوچک و حروف بزرگ انگلیسی باشد.

در غیر این صورت قابل قبول نیست.

در این برنامه سه کلیدواژه ی `sign up` (برای ثبت نام)، `log in` (برای ورود به حساب کاربری) و `exit` (برای خروج و اتمام برنامه) استفاده میشود. شما باید برای هر کدام از این عملیات ها یک تابع بنویسید. قطعه کد زیر برای اینکار به شما کمک میکند.

```
def has_num():
    def inner():
        pass
    return inner

def has_uppercase():
    def inner():
        pass
    return inner

def has_lowercase():
    def inner():
        pass
    return inner

def length_bigger_than_8():
    def inner():
        pass
    return inner

@length_bigger_than_8
@has_lowercase
@has_uppercase
```

```
@has_num
def validation_username():
    pass

@length_bigger_than_8
@has_lowercase
@has_uppercase
@has_num
def validation_password():
    pass
```

نکاتی که باید رعایت شود:

- برای ثبت نام کاربر باید چک کنید که نام کاربری و رمز عبور شرایط خواسته شده را دارند. برای این کار تابع `validation_username` و `validation_password` را کامل کنید.
- اگر هرکدام شرایط لازم رانداشتند باید به کاربر با پیغام زیر اطلاع دهید:

```
username have to be at least 8 characters!

password have to be at least 8 characters!

username should have at least one capital letter!

password should have at least one capital letter!

username should have at least one lowercase letter!

password should have at least one lowercase letter!

username should have at least one number!

password should have at least one number!
```

- اگر ثبت نام موفقیت آمیز بود باید پیغام زیر را نمایش دهید:

```
Username signed up successfully!
```

- برای استفاده از log in باید کاربر ثبت نام کرده باشد و نام کاربری و رمز عبورش موجود باشد، در غیر این صورت پیغام زیر را نمایش دهید:

```
you should sign up first!
```

- باید نام کاربری و رمز عبور ورودی را با نام کاربری و رمز عبور ثبت شده مقایسه کنید. اگر قبلا ثبت نام کرده بود و رمز عبور و نام کاربری اشتباه بود پیغام زیر را نشان دهید:

```
username or password is not correct!
```

- اگر هر دو درست وارد شده بودند پیغام زیر را وارد کنید:

```
Username logged in successfully!
```

ورودی

هر خط ورودی میتواند دو حالت داشته باشد:

```
Sign up username password
```

```
Log in username password
```

تضمین میشود همیشه آخرین خط ورودی `exit` است.

خروجی

پیغام های خروجی به طور مفصل در بالا شرح داده شده است.

مثال

ورودی نمونه 1

```
Sign up ali 12345678
exit
```

خروجی نمونه 1

```
username have to be at least 8 characters!
username should have at least one capital letter!
```

```
username should have at least one number!
password should have at least one capital letter!
password should have at least one lowercase letter!
```

ورودی نمونه 2

```
Log in Ali123456 Ali123456
exit
```

خروجی نمونه 2

```
you should sign up first!
```

ورودی نمونه 3

```
Sign up Ali123456 Ali123456
Log in Ali123456 Ali123456
exit
```

خروجی نمونه 3

```
Ali123456 signed up successfully!
Ali123456 logged in successfully!
```

ماشین حساب

ماشین حساب calculator

تابعی به نام calculator بنویسید. این تابع دارای سه پارامتر زیر است:

- *args

یک تعداد عدد است که تعداد آنها حتما باید بیش از یک باشد.

- output_format

مقدار آن می‌تواند int یا float باشد که به صورت پیش فرض float است.

- operation

مقدار operation می تواند یکی از مقادیر ADD یا SUB یا MUL یا DIV به ترتیب به معنای جمع، تفریق، ضرب و تقسیم باشد. تابع باید عملیات مشخص شده با operation را به ترتیب روی مقادیر args انجام دهد. مقدار پیش فرض آن باید ADD باشد.

تابع باید عملیات مورد نظر را بر روی پارامترهای ورودی انجام دهد و نتیجه را با فرمت مشخص شده در output_format برگرداند. (اگر فرمت خواسته شده عدد صحیح است، عدد باید گرد شود).

درصورتی که هر کدام از پارامترهای ارسال شده به تابع نامعتبر هستند، باید منجر به ایجاد استثنای ValueError شود :

- اگر تنها یک عدد به تابع ارسال شود، هیچ عملیاتی نباید روی تابع صورت بگیرد و اگر هیچ عددی ارسال نشود، باید برنامه منجر به خطا (ValueError) بشود و پیغام At least one number must be entered چاپ شود.
- در صورتی که عملیات خواسته شده معتبر نبود، پیغام Operation must be ADD, SUB, MUL or DIV چاپ شود.
- در صورتی که فرمت خروجی نامعتبر بود، پیغام Format must be float or int چاپ شود.

گزارش log

یک decorator به نام `log` بنویسید که اگر بر یک تابع اعمال شود، نام تابع، تعداد آرگومانهای positional، آرگومانهای keyword و مقدار برگشتی تابع را چاپ کند. همچنین در صورتی که اجرای تابع کمتر از یک ثانیه طول می کشد عبارت less than a second و در غیر این صورت more than a second را چاپ کند.

توجه داشته باشید، در صورتی که فراخوانی تابع اصلی منجر به exception شود، باید نام تابع، پیغام خطا، مقدار برگشتی None و زمان اجرای تابع چاپ شود. (به ورودی و خروجی نمونه 3 توجه کنید)

ورودی

تابع `exec` در پایتون یک رشته کد را به عنوان ورودی دریافت می‌کند و آن را اجرا می‌کند. برای آشنایی بیشتر با این تابع این لینک را ببینید.

ورودی شما در این سوال یک رشته است که حاوی کدهای قابل اجرای پایتون است. در این سوال رشته کد ورودی ممکن است فراخوانی تابع `calculator` یا `log` با آرگومانهای مختلف باشد. (به ورودیهای نمونه توجه کنید.)

فرمت کد

فرمت پاسخ ارسالی شما برای این سوال باید به شکل زیر باشد.

```
def log(original_function):
    '''write an appropriate docstring here'''
    # write the rest of definition bellow

@log
def calculator(*args, operation=ADD, output_format='float'):
    '''write an appropriate docstring here'''
    # write the rest of definition bellow

# your submission must include following lines of codes
if __name__ == '__main__':
    code_to_execute = input()
    exec(code_to_execute)
```

ورودی نمونه ۱

```
calculator(1,2,3)
```

خروجی نمونه ۱

```
Function Name: 'calculator'
Number of Positional Arguments: 3
Keyword Arguments: {}
return value: 6.0
Time Consumption: less than a second
```

ورودی نمونه ۲

```
calculator(*range(100000), output_format = 'float')
```

خروجی نمونه ۲

```
Function Name: 'calculator'
Number of Positional Arguments: 100000
Keyword Arguments: {'output_format': 'float'}
return value: 4999950000.0
Time Consumption: less than a second
```

ورودی نمونه ۳

```
calculator()
```

خروجی نمونه ۳

```
Function Name: 'calculator'
Value Error: At least one number must be entered.
return value: None
Time Consumption: less than a second
```

ورودی نمونه ۴

```
calculator(*range(5000000), operation = 'SUB', output_format = 'int')
```

خروجی نمونه ۴

```
Function Name: 'calculator'
Number of Positional Arguments: 5000000
Keyword Arguments: {'operation': 'SUB', 'output_format': 'int'}
return value: -12499997500000
Time Consumption: more than a second
```