

به نام خدا

گزارش تمرین اول درس پایگاه داده پیشرفته



دانشگاه لرستان

: نام و نام خانوادگی

مبینا ترابی 40311415008

: نام استاد

آقای دکتر آرمین رشنو

1 = برنامه ای بنویسید که یک عدد را بگیرد و همه ارقام آن را با علامت * از هم جدا شده اند چاپ کند

ورودی : 822145635

خروجی : 64228

```
1💡 Write a program to get a number and print all its even digits separated by *.
Input: 822145635
Output: 6*4*2*2*8
"""

def evenDigits(number): 1 usage
    even_digits = [digit for digit in str(number) if int(digit) % 2 == 0]
    return '*'.join(even_digits)

num = 822145635
print(evenDigits(num))
```

```
> :
C:\Users\sama\PycharmProjects\ADH\.venv\Scripts
8*2*2*4*6

Process finished with exit code 0
```

توضیح : در اینجا یک تابع به نام `even-digits` تعریف کرده ایم

`Number` یک ارگومان یا پارامتر که قراره تمام ارقام زوج را با `*` از هم جدا کند

در خط دوم : یک لیست `even-digits` ایجاد می کنیم که قراره از `number` به صورت رشته (`str`) استفاده می کند تا بتواند هر رقم را جداگانه بررسی کند. `{حلقه for درون لیست، هر رقم از number را به صورت یک رشته جدا می کند.}` شرط `if` درون حلقه بررسی می کند که آیا رقم مورد نظر زوج است یا خیر (یعنی باقیمانده تقسیم بر ۲ صفر باشد). `{ نتیجه ی نهایی لیستی از ارقام زوج به نام even_digits خواهد بود.}`

در خط سوم : لیست ارقام زوج با استفاده از `*` از هم جدا میشوند

در خط چهارم : متد `num` برابر با مقدار `822145635`

در خط آخر : `even` را با مقدار `num` فراخوانی می کند

2 = برنامه ای بنویسید که عبارت زیر را با 500 جمله کحاسبه کند

```

1      """
2      2💡 Write a program to compute the following expression for 500 sentences:
3      (3! / 2+9) - (5! / 3+7) + (7! / 4+5) - (9! / 5+3) + (11! / 6+1) - (13! / 7-1) ...
4      """
5
6      > import ...
7
8
9      getcontext().prec = 200
10
11      def compute_expression(n): 1 usage
12          result = Decimal(0)
13          sign = -1
14          for i in range(n):
15              sign *= -1
16              numerator = (2 * i) + 3
17              denominator = Decimal(i + 2) + Decimal(9 - (2 * i))
18
19              if denominator == 0:
20                  print(f"division by zero in n = {i+1} and continue.")
21                  continue
22
23              term_value = Decimal(math.factorial(numerator) / denominator)
24              result += Decimal(sign) * term_value
25
26          return result
27
28      n = int(input("Enter n:"))
29      result = compute_expression(n)
30      print("result:", result)
31

```

توضیح : خط اول کتابخانه math برای استفاده تابع فاکتوریل math توابع ریاضی مفیدی را فراهم می کند دو کلاس را از تابع دسیمال وارد می کند getcontext,decimal برای محاسبات اعشاری با دقت بالا است

در خط اول : دقت محاسبات دسیمال را با 200 رقم اعشاری محاسبه می کند

در خط دوم : یک تابع تعریف میکند که ورودی صحیح n بار محاسبه می شود

بطور کلی این کد مجموعه ای از جملات را محاسبه می کند، که هر جمله شامل فاکتوریل در صورت و یک عبارت در مخرج است. علامت های متناوب تضمین می کنند که سری یک

مجموع متناوب است. استفاده از نوع Decimal محاسبات دقیق را حتی هنگام کار با فاکتوریل‌های بزرگ که می‌توانند منجر به خطاهای قابل توجه در نقطه شناور شوند، تضمین می‌کند. بررسی تقسیم بر صفر، کد را قوی‌تر می‌کند. کد ساختار یافته و درک آن آسان است، اگرچه فرمول خود می‌تواند بسته به زمینه ریاضی پیچیده باشد.

[illegible]

3= برنامه ای بنویسید که تمام ارقام 4 رقمی (بین 1000 و 9999) را چاپ کند بطوری که مجموع ارقام اول و دوم با حاصل ضرب رقم سوم و چهارم برابر باشد

```
1.py 2.py 3.py x
1  """
2  3 - Write a program to print all 4 digits numbers (between 1000 and 9999) that the sum of the first
3  and second digits is equal with the product of the third and forth digits.
4  3466: 6 + 6 = 3 * 4
5  Output: 1110, 1101, ..., 2999 , ... , 3466 , ...
6  """
7
8  for num in range(1000, 10000):
9      number = str(num)
10     digit1 = int(number[0])
11     digit2 = int(number[1])
12     digit3 = int(number[2])
13     digit4 = int(number[3])
14     if digit1 + digit2 == digit3 * digit4:
15         print(num)
16
```

```
↑ 1314
↓ 1322
↺ 1341
↻ 1415
↓ 1451
↺ 1516
↻ 1523
↓ 1532
↺ 1561
↓ 1617
↺ 1674
```

و و

در خط اول : این حلقه for، متغیر num را به هر عدد صحیح در بازه 1000 تا 9999 (بدون احتساب 10000) اختصاص می‌دهد. به عبارت دیگر، از 1000 تا 9999 را به صورت ترتیبی پیمایش می‌کند. در خط دوم : در هر تکرار حلقه، عدد صحیح num به یک رشته (string) با نام number تبدیل می‌شود. این کار برای دسترسی به ارقام جداگانه عدد ضروری است. در خط سوم : این چهار خط، ارقام جداگانه عدد چهار رقمی را استخراج و به اعداد صحیح تبدیل می‌کنند. در if: این شرط، اصل منطق برنامه را نشان می‌دهد. این شرط بررسی می‌کند که آیا مجموع دو رقم اول (digit1 + digit2) برابر

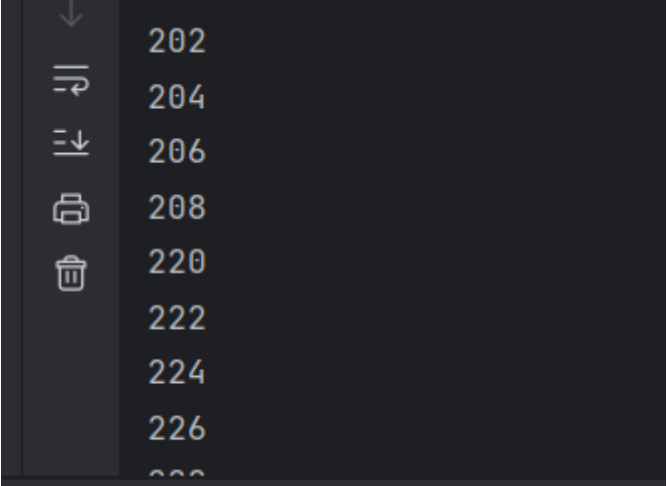
با حاصلضرب دو رقم آخر ($\text{digit3} * \text{digit4}$) است یا خیر. در آخر : اگر شرط در خط قبل برقرار باشد (یعنی مجموع دو رقم اول برابر با حاصلضرب دو رقم آخر باشد)، عدد چهار رقمی num چاپ می‌شود.

4= برنامه ای بنویسید که تمام ارقام سه رقمی (بین 100 تا 999) را چاپ کند که رقم فرد ندارند رقم 0 را به عنوان رقم زوج در نظر بگیرید

خروجی : 200.202.204.206.208.220.222....

```
"""
4💡 Write a program to print all 3 digits numbers (between a 100 and 999) that does not have odd digits.
Consider 0 as even digit.
Output: 200, 202,204, 206,208,220,222,...
"""

for num in range(100, 1000):
    number = str(num)
    digit1 = int(number[0])
    digit2 = int(number[1])
    digit3 = int(number[2])
    if digit1 % 2 == 0 and digit2 % 2 == 0 and digit3 % 2 == 0:
        print(num)
```



202
204
206
208
220
222
224
226
228

PycharmProjects > ADH > ADH-1 > 4

در خط اول : این یک حلقه for است که متغیر num را به هر عدد صحیح در بازه 100 تا 999 (شامل 100، اما نه 1000) اختصاص می‌دهد. به عبارت دیگر، حلقه از 100 شروع می‌شود و تا 999 ادامه می‌یابد. در خط دوم : در هر تکرار حلقه، مقدار عددی متغیر num به یک رشته (string) با نام number تبدیل می‌شود. این کار برای اینکه بتوانیم به ارقام جداگانه عدد دسترسی پیدا کنیم، لازم است.

این سه خط، ارقام عدد سه رقمی را به صورت جداگانه استخراج و به عدد صحیح تبدیل می‌کنند:

رقم اول (صدگان) عدد: digit1

رقم دوم (دهگان) عدد: digit2

رقم سوم (یکان) عدد: digit3

این یک شرط if است که بررسی می‌کند آیا همه سه رقم عدد، زوج هستند یا نه.

$digit1 \% 2 == 0$: این قسمت بررسی می‌کند که آیا باقی‌مانده تقسیم digit1 بر 2 برابر با 0 است یا خیر. این یعنی رقم اول زوج است.

$digit2 \% 2 == 0$: به طور مشابه، بررسی می‌کند که آیا رقم دوم زوج است یا خیر.

$digit3 \% 2 == 0$: بررسی می‌کند که آیا رقم سوم زوج است یا خیر.

and: این عملگر منطقی، هر سه شرط را به هم متصل می‌کند. فقط وقتی هر سه شرط برقرار باشند، کل عبارت شرطی درست خواهد بود.

اگر شرط if در خط قبلی درست باشد (یعنی هر سه رقم عدد زوج باشند)، آنگاه مقدار متغیر num (که همان عدد سه رقمی است) چاپ می‌شود.

5= برنامه ای بنویسید که عدد n را از کاربر دریافت کند و الگوی زیر را چاپ کند

```
1  """
2  5 - Write a program to get n from user and print the following pattern:
3  Input: n=8
4  Output:
5  1
6  2 4
7  3 6 9
8  4 8 12 16
9  5 10 15 20 25
10 6 12 18 24 30 36
11 7 14 21 28 35 42 49
12 8 16 24 32 40 48 56 64
13 """
14
15
16 def pattern(n): 1 usage
17     for i in range(1, n + 1):
18         for j in range(1, i + 1):
19             print(i * j, end=" ")
20         print()
21
22
23 pattern(8)
24
```

این خط، یک تابع به نام pattern تعریف می‌کند که یک پارامتر ورودی به نام n می‌گیرد. این پارامتر، اندازه الگوی خروجی را مشخص می‌کند.

خط دوم : این یک حلقه for بیرونی است. متغیر i از 1 تا n تکرار می‌شود. این حلقه، تعداد سطرهای الگوی خروجی را مشخص می‌کند. هر بار که حلقه بیرونی اجرا می‌شود، یک سطر جدید در الگو چاپ می‌شود.

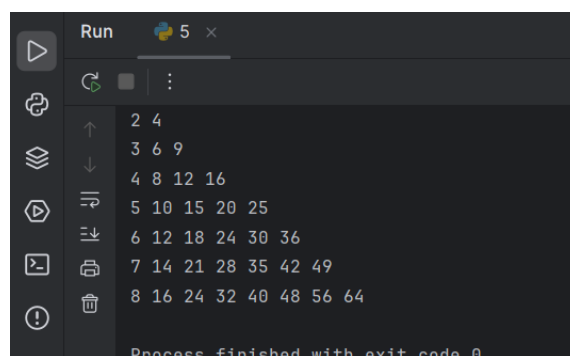
خط سوم : این یک حلقه for درونی است که داخل حلقه بیرونی قرار دارد.

. متغیر j از 1 تا i تکرار می‌شود. تعداد دفعات اجرای حلقه درونی در هر سطر، به مقدار i در آن سطر بستگی دارد. این حلقه، اعداد موجود در هر سطر را چاپ می‌کند.

خط آخر : این خط، حاصل ضرب i و z را چاپ می‌کند. i شماره سطر و z شماره عدد در آن سطر است. `"=end"` باعث می‌شود که اعداد در یک سطر، با یک فاصله از هم جدا شوند و به خط بعد نروند.

پرینت : این خط بعد از اتمام حلقه درونی و چاپ تمام اعداد یک سطر اجرا می‌شود. این خط، یک خط جدید در خروجی ایجاد می‌کند و باعث می‌شود که اعداد سطر بعدی در خط پایین‌تری چاپ شوند.

پترن : این خط، تابع `pattern` را با مقدار $n = 8$ فراخوانی می‌کند. این کار، باعث می‌شود که الگوی مورد نظر با 8 سطر چاپ شود.



```
Run 5 x
2 4
3 6 9
4 8 12 16
5 10 15 20 25
6 12 18 24 30 36
7 14 21 28 35 42 49
8 16 24 32 40 48 56 64
Process finished with exit code 0
```

=6 برنامه ای بنویسید تا n را دریافت کرده و سپس n عدد را از کاربر بگیرد و حداکثر حداقل و میانگین و انحراف معیار را محاسبه کند برنامه خود را برای 5 ورودی مختلف آزمایش کنید

```
Enter number 1:12
Enter number 2:15
Enter number 3:9
Enter number 4:11
Enter number 5:84

output:
Maximum is: 84
Minimum is: 9
Average is: 26.2
Standard Deviation is: 28.96

"""

numbers = []
n = int(input("Enter the n:"))
for i in range(n):
    number = int(input(f"Enter number {i+1}:"))
    numbers.append(number)

numbers = np.array(numbers)
maximum = np.max(numbers)
minimum = np.min(numbers)
average = np.mean(numbers)
std = np.std(numbers) # standard deviation

print("Maximum is:", maximum)
print("Minimum is:", minimum)
print("Average is:", average)
print(f"Standard Deviation is: {std:.2f}")
```

```
C:\Users\sama\PycharmProjects\ADH
Enter the n:4
Enter number 1:12
Enter number 2:3
Enter number 3:4
Enter number 4:9
Maximum is: 12
Minimum is: 3
Average is: 7.0
Standard Deviation is: 3.67
armProjects > ADH > ADH-1 > 6.py
```

در خط اول : این خط یک لیست خالی به نام `numbers` تعریف می‌کند که قرار است اعداد ورودی کاربر در آن ذخیره شوند. در خط دوم : این خط از کاربر درخواست می‌کند که تعداد اعداد `n` ه می‌خواهد وارد کند را وارد کند. ورودی کاربر به صورت رشته `string` دریافت می‌شود و سپس به نوع عدد صحیح `int` تبدیل می‌شود و در متغیر `n` ذخیره می‌شود.

در خط سوم : این یک حلقه `for` است که از 0 تا `n-1` تکرار می‌شود. یعنی به تعداد `n` بار اجرا می‌شود. در هر تکرار، متغیر `i` به شماره تکرار فعلی شماره اندیس اختصاص داده می‌شود. در خط چهارم : در هر تکرار این حلقه، از کاربر درخواست می‌شود تا یک عدد وارد کند. از آنجا که شماره ورود از 1 شروع می‌شود، `i+1` به نمایش می‌گذارد به عنوان مثال، وقتی که `i` برابر با 0 باشد، پیام "Enter number 1:" نمایش داده می‌شود و ورودی کاربر به نوع عدد صحیح `int` تبدیل و در متغیر `number` ذخیره می‌شود. در خط پنجم : این خط عدد ورودی `number` را به لیست `numbers` اضافه می‌کند. در نتیجه، پس از این حلقه، لیست `numbers` شامل `n` عدد ورودی خواهد بود. در خط ششم : این خط لیست `numbers` را به یک آرایه `NumPy` تبدیل می‌کند. `NumPy` یک کتابخانه قدرتمند در پایتون برای محاسبات عددی و کار با آرایه‌ها است. این تبدیل باعث می‌شود که بتوانیم از توابع ریاضی و آماری `NumPy` استفاده کنیم. در خط هفتم : این خط حداکثر مقدار در آرایه `numbers` را محاسبه می‌کند و در متغیر `maximum` ذخیره می‌کند. در خط هشتم: این خط حداقل مقدار در آرایه `numbers` را محاسبه می‌کند و در متغیر `minimum` ذخیره می‌کند. در خط نهم : این خط میانگین (میانگین حسابی) اعداد در آرایه `numbers` را

محاسبه می‌کند و در متغیر average ذخیره می‌کند. در خط دهم : این خط انحراف معیار اعداد در آرایه numbers را محاسبه می‌کند و در متغیر std ذخیره می‌کند.

این خط مقدار حداکثر را چاپ می‌کند.

این خط مقدار حداقل را چاپ می‌کند.

این خط مقدار میانگین را چاپ می‌کند.

این خط مقدار انحراف معیار را چاپ می‌کند. در اینجا از فرمت f2. استفاده شده است تا عدد به صورت اعشاری با دو رقم بعد از ویرگول نمایش داده شود.

7= تابعی برای سوال 6 بنویسید 5 عدد از کاربر در تابع اصلی دریافت کنید اعداد را به توابع ave1,min1,max1,sid1 ارسال کنید n max1 عدد را می‌گیرد و حداکثر را بر می‌گرداند تابع min1 n عدد را می‌گیرد و حداقل را بر می‌گرداند توابع ave1 و sid1 عدد را می‌گیرد و میانگین و انحراف معیار را در بدنه خود چاپ می‌کند و هیچ مقداری را باز نمی‌گرداند برنامه خود را برای 5 ورودی مختلف آزمایش کنید

```

1 > import ...
3
4 """
5 7 - Write 4 functions for question 6. Get 5 numbers from user in main.
6 Send numbers for Max1, Min1, Ave1 and STD1 functions. Max1 gives n numbers and return maximum.
7 Min1 gives n number and return minimum. Ave1 and STD1 give n number and print average and
8 standard deviation in their own body and do not return any value. Test your program for 5 different inputs.
9 """
10
11
12 def Max1(numbers): 1usage
13     maximum = numbers[0]
14     for num in numbers:
15         if num > maximum:
16             maximum = num
17     print("The Maximum is: ", maximum)
18
19
20 def Min1(numbers): 1usage
21     minimum = numbers[0]
22     for num in numbers:
23         if num < minimum:
24             minimum = num
25     print("The Minimum is: ", minimum)
26
27
28 def Ave1(numbers): 1usage
29     total = 0
30     for num in numbers:
31         total += num
32     average = total / len(numbers)

```

```
7.py x
28 def Ave1(numbers): 1 usage
30     for num in numbers:
31         total += num
32     average = total / len(numbers)
33     print("The Average is: ", average)
34
35
36 def STD1(numbers): 1 usage
37     ave = mean(numbers)
38     total = 0
39     for num in numbers:
40         total += ((num - ave) ** 2)
41
42     std = math.sqrt(total / len(numbers))
43     print(f"The Standard Deviation is: {std:.2f}")
44
45
46 nums = []
47 n = int(input("Enter the n:"))
48 for i in range(n):
49     number = int(input(f"Enter number {i + 1}:"))
50     nums.append(number)
51
52 Max1(nums)
53 Min1(nums)
54 Ave1(nums)
55 STD1(nums)
56
```

```
C:\Users\sama\PycharmProjects\ADH\.venv\Script
Enter the n:5
Enter number 1:10
Enter number 2:12
Enter number 3:9
Enter number 4:8
Enter number 5:3
The Maximum is: 12
The Minimum is: 3
The Average is: 8.4
The Standard Deviation is: 3.01

Process finished with exit code 0
PycharmProjects > ADH > ADH-1 > 7.py
```

این خط یک تابع به نام Max1 تعریف می‌کند که یک آرایه یا لیست از اعداد به نام numbers را به عنوان ورودی می‌گیرد.

در خط دوم : در این خط، اولین عدد در لیست numbers به عنوان مقدار حداکثر اولیه (maximum) در نظر گرفته می‌شود. در خط سوم : این یک حلقه for است که در آن هر عدد num از لیست numbers به طور تکراری بررسی می‌شود. در خط چهارم : در اینجا، بررسی می‌شود که آیا مقدار num بزرگتر از maximum فعلی است یا خیر.

در خط پنجم : اگر شرط قبل درست باشد، مقدار maximum به مقدار num به‌روزرسانی می‌شود. در خط ششم : پس از اتمام حلقه، حداکثر مقدار در لیست چاپ می‌شود. هفتم : مشابه تابع Max1, این خط یک تابع جدید به نام Min1 تعریف می‌کند که کار مشابهی برای محاسبه حداقل انجام می‌دهد. هشتم : در اینجا همنند همانند تابع Max1, ابتدا حداقل مقدار با اولین عدد از numbers اولیه‌سازی می‌شود. نهم ال آخر : این حلقه هر عدد num را از لیست numbers دوباره بررسی می‌کند. 10:

بررسی می‌شود که آیا عدد num کوچکتر از minimum فعلی است. 11: اگر شرط درست باشد، مقدار minimum به num به‌روزرسانی می‌شود. 12: اگر شرط درست باشد، مقدار minimum به num به‌روزرسانی می‌شود. 16:

هر عدد به مقدار کل (total) اضافه می‌شود. 17: میانگین با تقسیم مجموع بر تعداد اعداد محاسبه می‌شود. 18: میانگین محاسبه شده چاپ می‌شود. 19: تابع آخر STD1 برای محاسبه انحراف معیار تعریف شده است.

20 : در این خط، میانگین اعداد با استفاده از تابع mean منبعی محاسبه می‌شود (لطفاً توجه داشته باشید که این تابع باید از یک کتابخانه خاص، معمولاً NumPy، وارد شود). 21: یک متغیر total که برای محاسبه مجموع مربعات انحرافات از میانگین استفاده می‌شود. 22: دوباره از اعداد عبور می‌کند. 23: در اینجا، مربع انحراف هر عدد از میانگین محاسبه و به مقدار total اضافه می‌شود. 24: انحراف معیار با محاسبه ریشه مربع متوسط مربعات انحرافات بدست می‌آید. 25: انحراف معیار به دو رقم اعشار format شده و چاپ می‌شود. 26: لیست nums برای ذخیره اعداد ورودی از کاربر تعریف می‌شود. 27: از کاربر خواسته می‌شود که تعداد اعدادی که قرار است وارد کند را مشخص کند. 28: این حلقه برای دریافت n عدد اجرا می‌شود. 29: در هر تکرار، از کاربر خواسته می‌شود که عددی وارد کند و این عدد به نوع عدد صحیح تبدیل می‌شود. 30: عدد وارد شده به لیست nums اضافه می‌شود. 31:

تابع Max1 با ورودی nums (لیست اعداد) فراخوانی می‌شود. 32: تابع Min1 با ورودی nums فراخوانی می‌شود. 33: تابع Ave1 با ورودی nums فراخوانی می‌شود. 34: تابع STD1 با ورودی nums فراخوانی می‌شود.

=8 یک عدد 5 رقمی را در تابع اصلی (main) دریافت کنید. اگر عدد وارد شده 5 رقمی نباشد، به صورت مکرر از کاربر بخواهید که یک عدد معتبر وارد کند. عدد را برای تابع F1 ارسال کنید تا بزرگترین رقم عدد را پیدا و برگرداند. در مرحله بعد، بزرگترین رقم و عدد ورودی را برای تابع F2 ارسال کنید تا بزرگترین رقم را از عدد حذف کرده و آن را برگرداند. در نهایت، خروجی نهایی را در تابع اصلی به صورت مورد نظر چاپ کنید.

```
7
8
9 def F1(number): 1 usage
10     digits = []
11     while number > 0:
12         digit = number % 10
13         digits.append(digit)
14         number //= 10
15     maxDigit = max(digits)
16     return maxDigit
17
18
19 def F2(number, digit): 1 usage
20     number = str(number).replace(_old: f"{digit}", _new: "").strip()
21     return number
22
23
24 num = input("Enter a 5 digits number:")
25
26 while True:
27     if len(num) == 5 and num.isdigit():
28         break
29     else:
30         print("It is not a 5 digits number.")
31         num = input("Please enter a valid number:")
32
33 num = int(num)
34 max_digit = F1(num)
35 print("The Maximum digit is:", max_digit)
36 print(f"The Final Digit without {max_digit} is:", F2(num, max_digit))
37
```

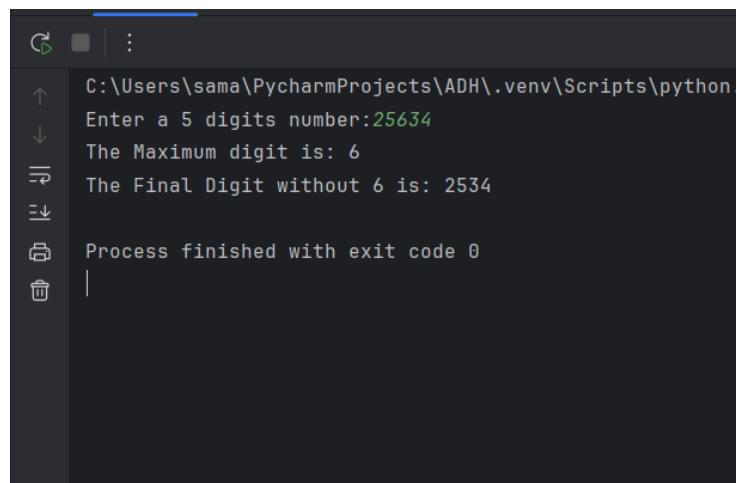
تابع F1(number): این تابع بزرگترین رقم یک عدد را پیدا می‌کند. یک عدد صحیح number. یک لیست خالی به نام digits برای ذخیره ارقام عدد تعریف می‌شود. با استفاده از یک حلقه while، عدد را به ارقام آن تجزیه می‌کند: $number \% 10$ باقیمانده تقسیم عدد بر 10 را محاسبه می‌کند که آخرین رقم عدد را به دست می‌دهد. این رقم به لیست digits اضافه می‌شود. سپس عدد با استفاده از $number //= 10$ (تقسیم صحیح) به‌روز می‌شود. با این کار ما آخرین رقم را حذف کرده و به رقم‌های

بعدی می‌رویم. پس از پایان حلقه، تابع `max(digits)`، بزرگترین رقم را در لیست `digits` پیدا می‌کند. این مقدار بزرگترین رقم به عنوان خروجی بازگشت داده می‌شود.

تابع `F2(number, digit)` : این تابع یک رقم خاص (`digit`) را از عدد (`number`) حذف می‌کند و نتیجه را به عنوان یک رشته برمی‌گرداند. `number`: عددی که قرار است از آن رقم حذف شود `digit`: رقمی که باید از عدد حذف شود. عدد ورودی به نوع رشته‌ای (`str`) تبدیل می‌شود با استفاده از متد `replace()`، تمام ارقام `digit` در عدد حذف می‌شوند. همچنین از `strip()` برای حذف فضاها در ابتدای و انتهای نتیجه استفاده می‌شود. رشته حاصل، که حالا رقم خاص را ندارد، به عنوان خروجی برمی‌گردد.

بزرگترین رقم با پیام مناسب چاپ می‌شود.

سپس تابع `F2` برای حذف بزرگترین رقم از عدد فراخوانی شده و نتیجه نهایی چاپ می‌شود.



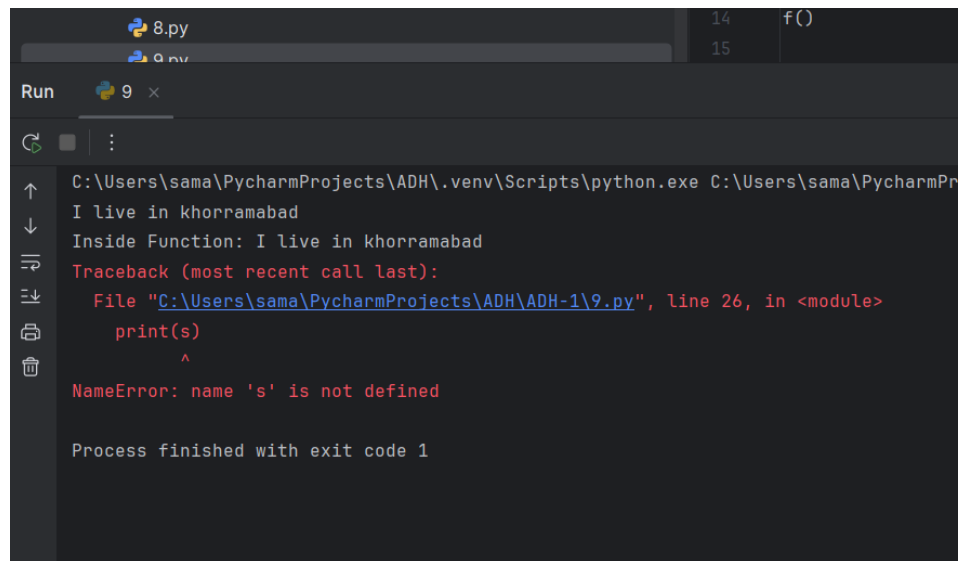
```
C:\Users\sama\PycharmProjects\ADH\.venv\Scripts\python
Enter a 5 digits number:25634
The Maximum digit is: 6
The Final Digit without 6 is: 2534

Process finished with exit code 0
```

9= متغیرهای جهانی و محلی: کدهای پی‌وی ون نوبت به پی‌سی را اجرا کنید و خروجی‌ها را گزارش کنید. دلایل خروجی‌ها را به‌صورت دقیق توضیح دهید.

```
7.py 8.py 10.py 9.py x
1  """
2  9-Global and Local variables: Run codes P1, P2, P3 and P4 and report outputs.
3  Explain reasons for their outputs in details.
4  """
5
6  # # P1
7  # def f():
8  #     # local variable
9  #     s = "I live in khorrabad"
10 #     print(s)
11 #
12 #
13 # # Main
14 # f()
15
16
17 # # P2
18 # def f():
19 #     # local variable
20 #     s = "I live in khorrabad"
21 #     print("Inside Function:", s)
22 #
23 #
24 # # Main
25 # f()
26 # print(s)
27
28
29 # # P3
30 # def f():
31 #     global s
```

```
27
28
29 # # P3
30 # def f():
31 #     global s
32 #     s = 'I live in khorrabad.'
33 #     print(s)
34 #
35 #
36 # # Main: Global Scope
37 # s = 'I live in iran.'
38 # f()
39 # print(s)
40
41
42 # # P4
43 # # This function uses global variable s
44 # def f():
45 #     s = "I live in khorrabad."
46 #     print(s)
47 #
48 #
49 # # Main
50 # s = "I live in iran."
51 # f()
52 # print(s)
53
```



```
8.py 14 f()
9.py 15
Run 9 x
C:\Users\sama\PycharmProjects\ADH\.venv\Scripts\python.exe C:\Users\sama\PycharmPr
I live in khorramabad
Inside Function: I live in khorramabad
Traceback (most recent call last):
  File "C:\Users\sama\PycharmProjects\ADH\ADH-1\9.py", line 26, in <module>
    print(s)
    ^
NameError: name 's' is not defined

Process finished with exit code 1
```

P1: در این تابع، متغیر `s` به عنوان یک متغیر محلی تعریف شده و مقدار `"I live in khorramabad"` به آن اختصاص یافته است. وقتی تابع `f()` اجرا می‌شود، متغیر `s` به درستی چاپ می‌شود. پس از خروج از تابع، متغیر `s` دیگر قابل دسترسی نیست و هیچ خطا یا پیامی دیگری تولید نمی‌شود.

P2: شبیه P1، در اینجا مجدداً متغیر محلی `s` در تابع تعریف شده است و مقدار آن چاپ می‌شود. با این حال، در بخش اصلی (Main) وقتی سعی می‌شود `print(s)` انجام شود، خطا به وجود می‌آید زیرا `s` خارج از تابع تعریف شده و قابل دسترسی نیست. پس از اجرای کد، پیام خطا به صورت زیر خواهد بود:

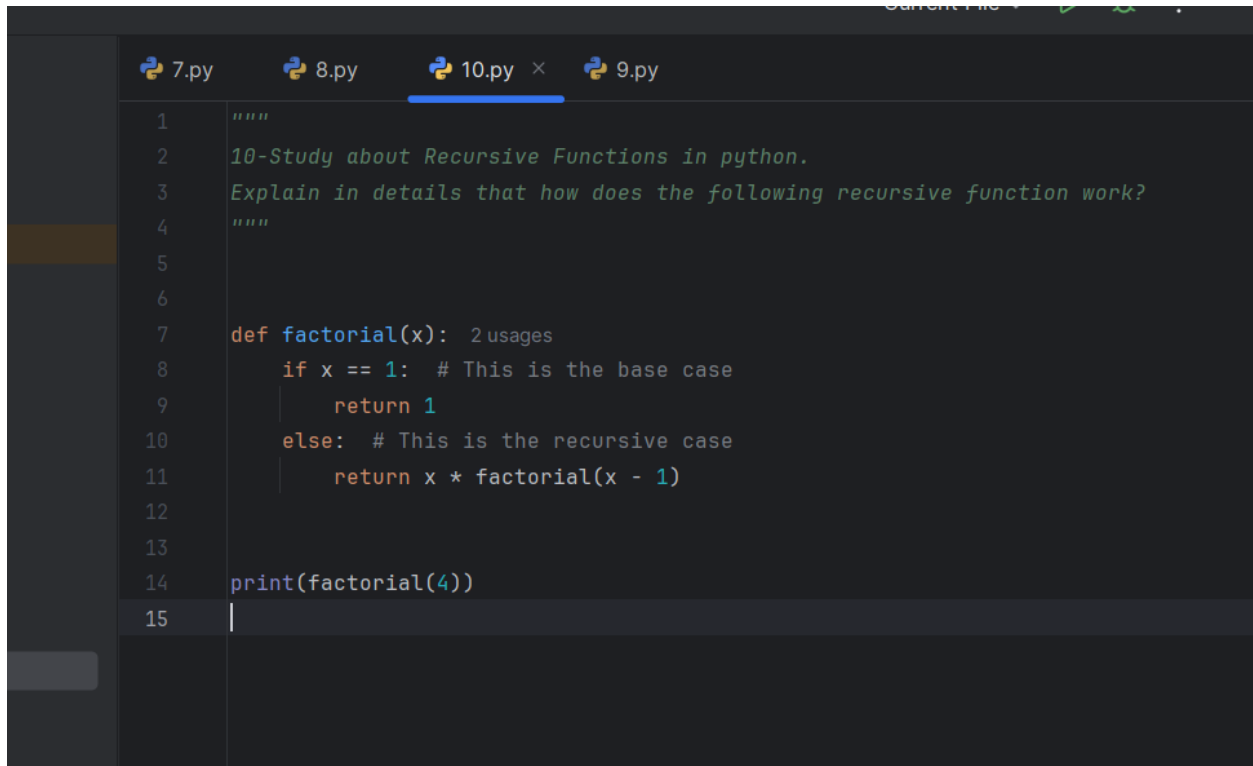
P3: در اینجا، استفاده از کلمه کلیدی `global` به تابع اجازه می‌دهد تا به متغیر `s` که در فضای جهانی (Global Scope) تعریف شده است، دسترسی داشته باشد. ابتدا، `s` در فضای جهانی به `"I live in iran"` تنظیم شده است و سپس در تابع، مقدار آن به `"I live in khorramabad"` تغییر می‌کند و چاپ می‌شود. پس از فراخوانی تابع، `s` در محیط جهانی نیز به `"I live in khorramabad"` تغییر می‌یابد و وقتی که در بخش اصلی چاپ می‌شود، این مقدار نیز نمایش داده می‌شود.

P4: در اینجا، `s` در تابع به عنوان یک متغیر محلی تعریف شده است. بنابراین، این `s` هیچ ارتباطی با متغیر `s` در فضای جهانی ندارد. در نتیجه، مقدار `"I live in khorramabad"` چاپ می‌شود، اما پس از آن که تابع اجرا می‌شود و در بخش اصلی `print(s)` انجام می‌شود، مقدار `s` که در فضای جهانی تعریف شده (`"I live in iran"`) نمایش داده می‌شود. بنابراین در این مورد هیچ تغییری در مقدار جهانی `s` ایجاد نمی‌شود.

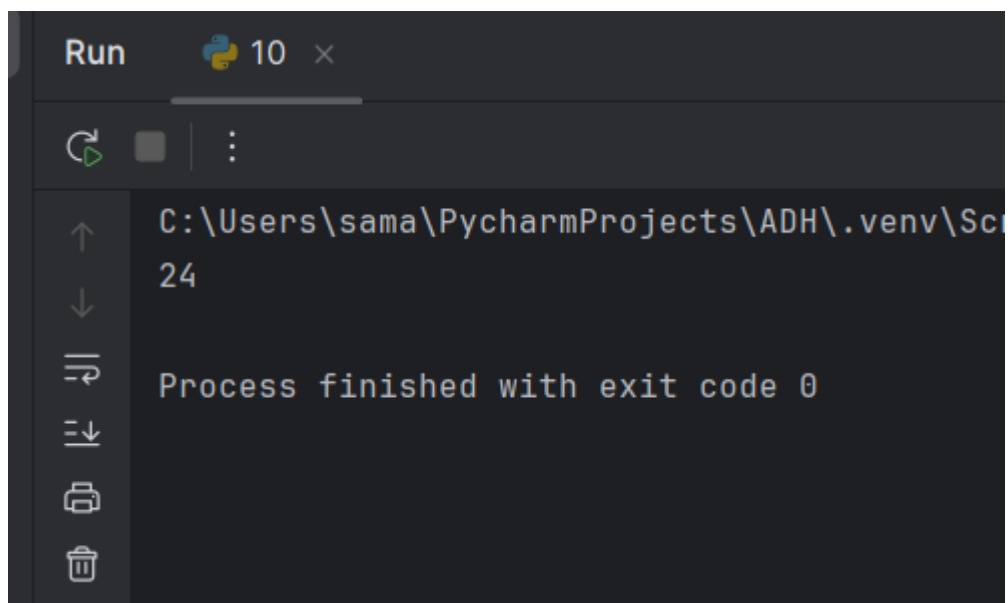
با `ctrl /` از حالت کامنت خارج می‌شود

10. = در مورد توابع بازگشتی در پایتون مطالعه کنید

به صورت دقیق توضیح دهید که تابع بازگشتی زیر چگونه کار می کند؟



```
1 """
2 10-Study about Recursive Functions in python.
3 Explain in details that how does the following recursive function work?
4 """
5
6
7 def factorial(x): 2 usages
8     if x == 1: # This is the base case
9         return 1
10    else: # This is the recursive case
11        return x * factorial(x - 1)
12
13
14 print(factorial(4))
15
```



```
Run 10 x
C:\Users\sama\PycharmProjects\ADH\.venv\Sc
24
Process finished with exit code 0
```

این تابع یک عدد صحیح x را به عنوان ورودی می‌گیرد. هدف این تابع محاسبه فاکتوریل x است (حاصل ضرب تمام اعداد صحیح مثبت تا x).

$if\ x == 1$: این خط بررسی می‌کند که آیا x برابر 1 است یا خیر. $return\ 1$: اگر x برابر 1 باشد، تابع مقدار 1 را برمی‌گرداند. این حالت، مورد پایه (Base Case) است، یعنی جایی که بازگشت متوقف می‌شود. فاکتوریل 1 برابر 1 است. $else$: اگر x برابر 1 نباشد، کد داخل این بلوک اجرا می‌شود. $return\ x * factorial(x - 1)$: در این خط، دو اتفاق مهم رخ می‌دهد: تابع خودش را با ورودی $x - 1$ فراخوانی می‌کند ($factorial(x - 1)$). این همان بازگشت است. نتیجه فراخوانی بازگشتی در x ضرب می‌شود. این عمل باعث می‌شود که هر بار که تابع خود را فراخوانی می‌کند، عدد ورودی در نتیجه ضرب شود تا در نهایت فاکتوریل محاسبه شود. تابع $factorial$ با ورودی 4 فراخوانی می‌شود.

چون $4 \neq 1$ است، تابع $factorial * 4$ (3) را برمی‌گرداند.

چون $3 \neq 1$ است، تابع $factorial * 3$ (2) را برمی‌گرداند.

چون $2 \neq 1$ است، تابع $factorial * 2$ (1) را برمی‌گرداند.

چون $1 == 1$ است (مورد پایه)، تابع مقدار 1 را برمی‌گرداند.

$factorial$ (2) مقدار $2 * 1 = 2$ را برمی‌گرداند.

$factorial$ (3) مقدار $3 * 2 = 6$ را برمی‌گرداند.

$factorial$ (4) مقدار $4 * 6 = 24$ را برمی‌گرداند.