جلسه چهارم – فرزان رحمانی– ۲2/8/۹۹

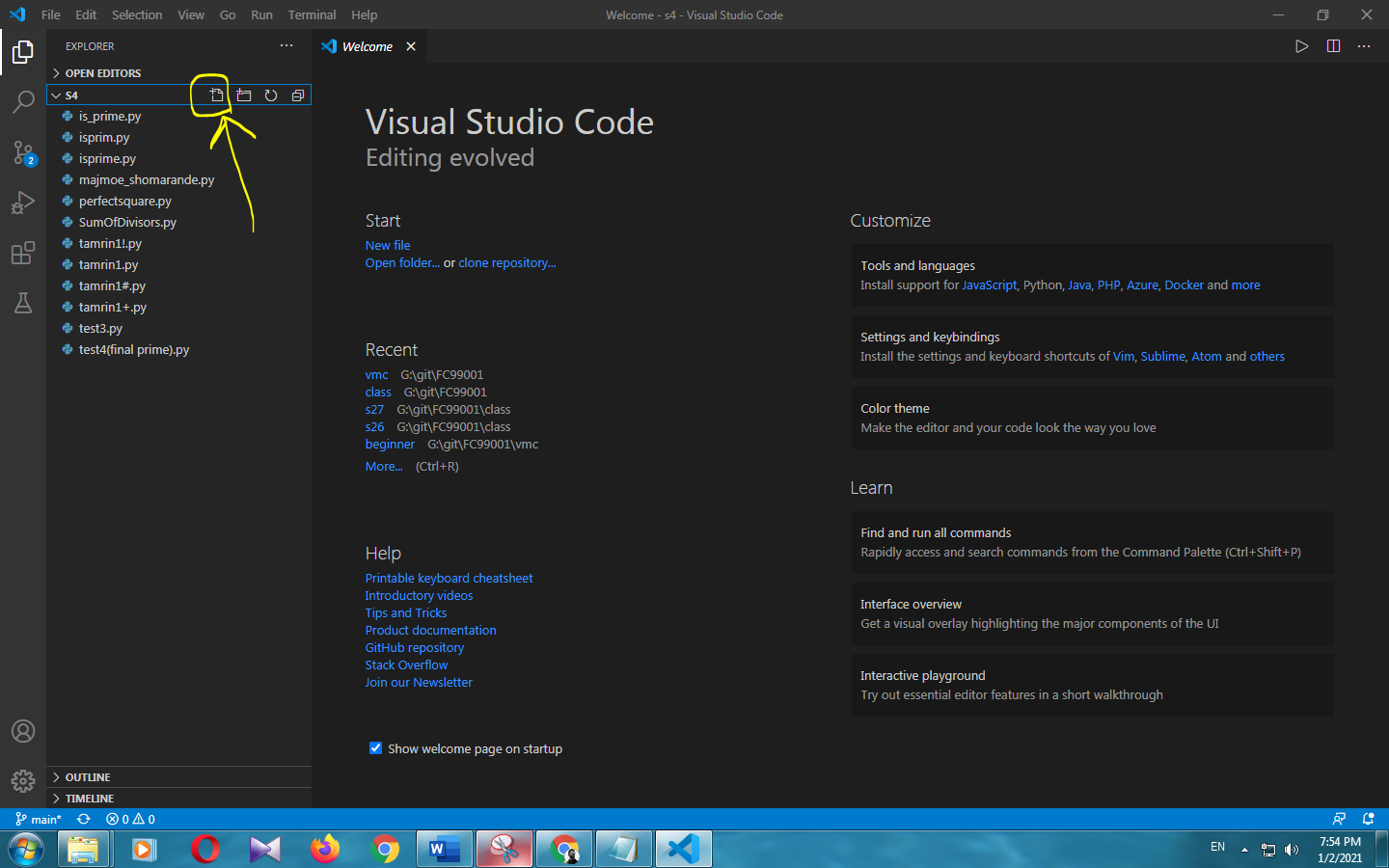
این مستند توسط فرزان رحمانی برای جلسه چهارم مبانی برنامه سازی و کامپیوتر تدوین شده است.امیدوارم که بتوانم تمام مفاهیم را به درستی منتقل کنم و دانش آموزان عزیز نهایت استفاده را از آن ببرند.

# نوشتن برنامه محاسبه‌ اعداد اول

در این برنامه اهداف مفید و کاربردی ای را دنبال می کنیم.همچنین چندین مفهوم و تعریف جدید نیز یاد خواهیم گرفت.

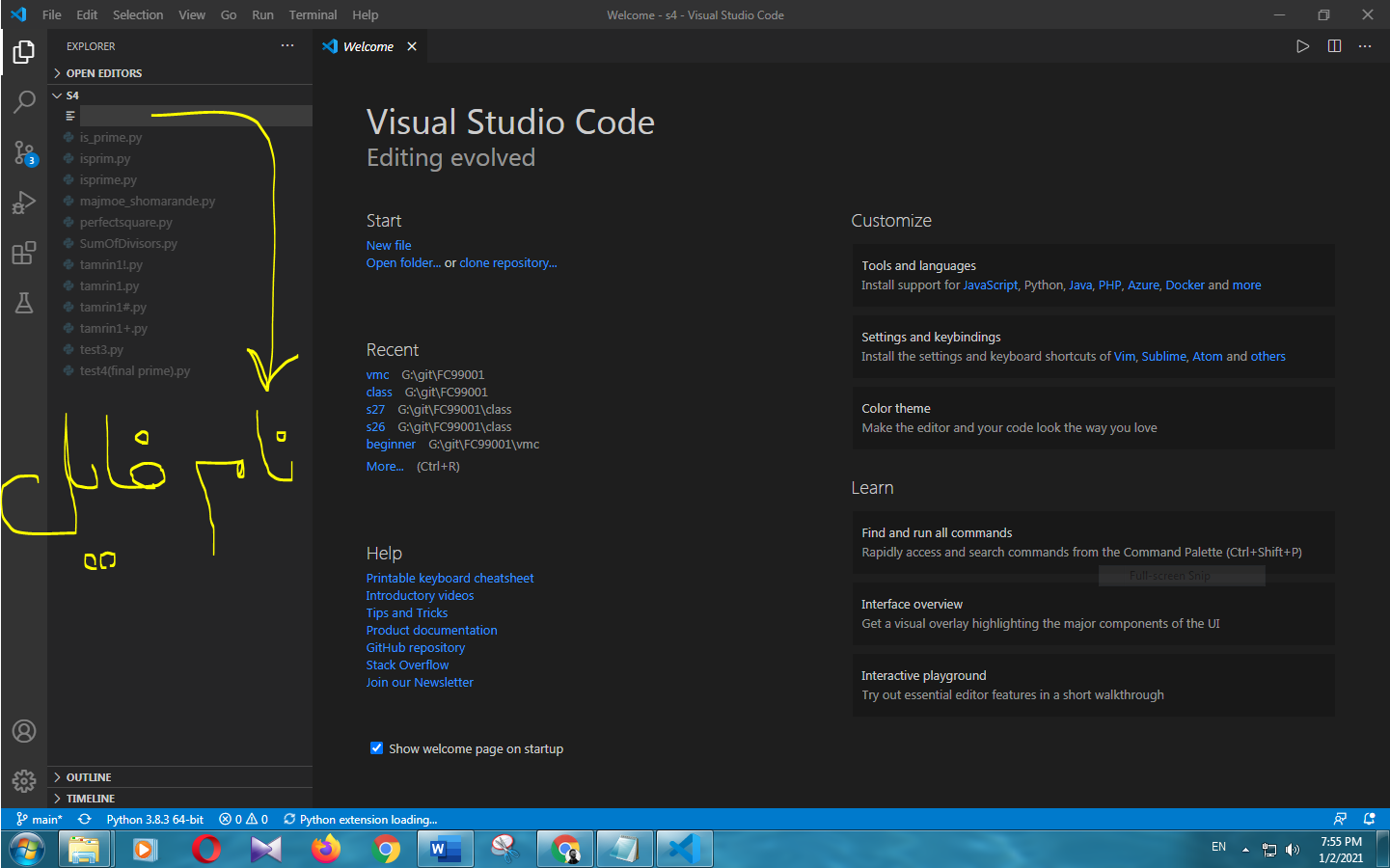
## ساختن فایل در نرم افزار Visual Studio Code

اولین گام ایجاد فایل prime.py در نرم افزار VS Codeمانند شکل 1- ایجاد فایل می باشد.



شکل 1- ایجاد فایل

بعد از زدن کلید ایجاد فایل در قسنت سمت چپ برنامه می توانیم نام فایل مورد نظر را بزنیم.سپس پسوند فایل مطلوب که نوع آن را مشخص می کند بعد از یک نقطه می گذاریم.مثلا در اینجا چون می خواهیم برنامه اعداد اول را بنویسیم ابتدا نام فایل را مثلا prime می نویسیم. بعد از آن یک نقطه می گذاریم. و در آخر چون می خواهیم فایل مورد نظر پایتون باشد از پسوند py استفاده می کنیم. بذا در قسمت سمت چپ می نویسیم prime.py و سپس بازدن دکمه enter فایل مطلوب در پوشه مورد نظر ساحته می شود و ما می توانیم از آن استفاده کنیم و در آن تغییرات ایجاد کنیم.این مطالب را می توانید در شکل 2 – نام فایل مشاهده کنید.



شکل 2 – نام فایل

## نوشتن دستور ها در برنامه VS Code

برای نوشتن برنامه اصلی ابتدا نیاز یک سری از تعاریف و دستور های اولیه و اساسی داریم که در این قسمت آن ها را می بینیم.

### تعریفی ابتدایی از for

حلقه یا تکراری از یک متغیر که میتوان شروع (start)،پایان(end) و گام یا فاصله یا مقدار زیاد شدن در هر حلقه(step) را به آن داد. اگر شروع را ندهیم خودش به طور پیش فرض صفر در نظر میگیرد و اگر مقدار زیاد شدن را ندهیم به طور پیش فرض 1 در نظر میگیرد. در شکل 3 – شکلی ساده از for می توانید نمونه ای از ان را دید.

1. #  variable    start  end+1    step
2. for i in range(  1  ,   10  ,   2   ):

شکل 3 – شکلی ساده از for

در هر حلقه می توان از continue یا break استفاده کرد که تفاوت آن ها با هم این است که break کلا حلقه را تموم میکند و می رود سراغ دستور بعدی ولی continue بر می گردد به داخل for و مقدار متغیر داخل for را زیاد میکند. در ادامه در شکل 4 - break and continue می توانیم نوع عملکرد این دو را در یک حلقه ببینیم.

1. for i in range(10):
2. if i == 5 :
3. break----
4. |
5. |
6. <----------------
8. --<--<------<----
9. |               ^
10. |               |
11. <i=6>             |
12. ^
13. for i in range(10):  |
14. if i == 5 :      |
15. continue-->---

شکل 4 - break and continue

### تعریف متغیر از نوع boolean

همان طور که در شکل 5 - boolean دیده می شود تعریف متغیر از نوع درست یا غلط (Boolean 🡪 True or False) را می توانیم به راحتی انجام بدهیم.

1. #     boolean  variable
2. x   =   True
3. y   =   False

شکل 5 - boolean

### تعریف کردن تابع

با استفاده از دستور def و سپس اسم تابع می توانیم تابع تعریف کنیم. بعد عملیات می رود نظر را قرار می دهیم. در آخر نیز خروجی تابع با استفاده از return تایین می کنیم. در شکل 6 – تعریف تابع می توانیم ببینیم.

1. #def=define   |  function\_input == ورودی
2. def function\_name(       x     ):
3. #operations
4. return y
5. # function\_output == خروجی

شکل 6 – تعریف تابع

### تعریف کردن شرط بوسیله if

در شکل 7 – نوشتن شرط تعریف شرط را یادآوری کرده ایم.

1. # شرط    اگر
2. if   condition:
3. operation
4. # اگر نه
5. else:
6. operation

شکل 7 – نوشتن شرط

### تعریف دقیق اعداد اول

عدد اول: عددی که تنها شمارنده هایش خودش و عدد یک است.. پس کافی است با استفاده از یک حلقه تمام اعداد قبل از آن عدد به جز یک را چک کنیم اگر بخش پذیر نبودند (باقی مانده تقسیمشان صفر نبود) اول هستند.لذا تابعی مینویسیم که به عنوان ورودی یک عدد میگیرد و اگر اول باشد درست خارج میشود و اگر اول نباشد نادرست خارج میشود. در شکل 8 – عدد اول می توانیم کد آن را ببینیم.

1. def is\_prime(num):
2. is\_prime=True
3. for in range(num):
4. if n>2 and num % n == 0:
5. is\_prime=False
6. if is\_prime and num>1 :
7. return True
8. return False

شکل 8 – عدد اول

## استفاده از تابع عدد اول برای چاپ اعداد اول یک تا صد

با ترکیب کردن دستور های بالا مثل for می توانیم برنامه نهایی را بنویسیم.

### چاپ اعداد اول یک تا صد

می توانیم از یک حلقه استفاده کنیم که اعداد 1 تا صد را به تابعی که تعریف کردیم بدهد. سپس اگر اول بودند آن هارا چاپ کنیم. می توانیم در شکل 9 – چاپ کردن اعداد اول یک تا صد ببینیم ک چجوری دستور ها را ترکیب کرده ایم.

1. #num = ورودی تابع   return = خروجی تابع
2. def is\_prime(num):
3. is\_prime=True
4. for n in range(num):
5. if n>1 and num % n == 0 :
6. is\_prime=False
7. if is\_prime and num>1 :
8. return True
9. return False
10. for i in range(101):
11. is\_i\_prime = is\_prime(i) #is\_i\_prime yek moteghayere az noe boolean
12. if is\_i\_prime:
13. print(i)

شکل 9 – چاپ کردن اعداد اول یک تا صد

### بهبود عملکرد تابع

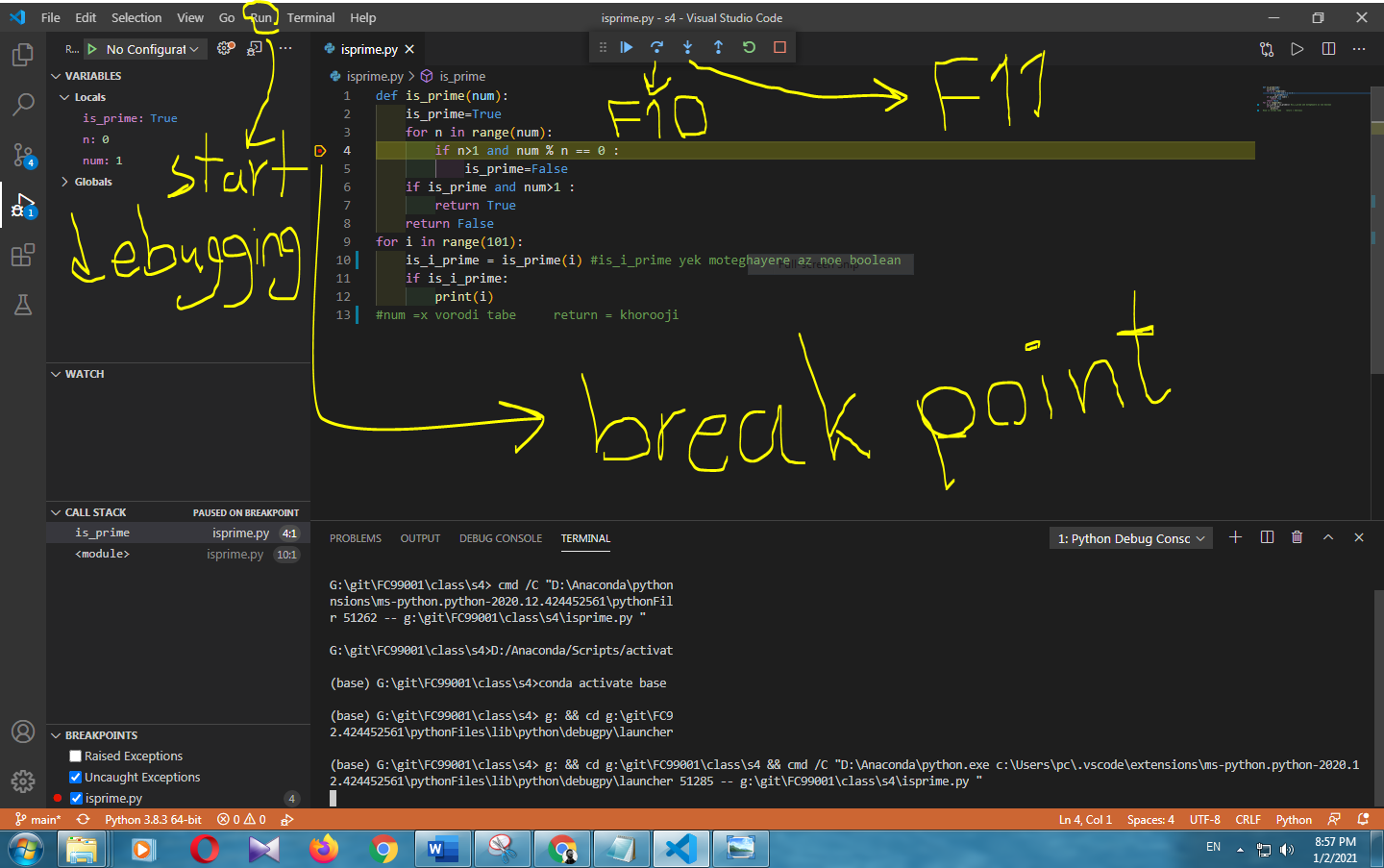
اگر دقت کنید ما در بالا متغیر is\_prime را تعریف کردیم ولی بدون اضافه کردن متغیر اضافی هم میتوانستیم به هدف مان برسیم که این طوری هم تعداد خط های مان کمتر و هم فهم آن آسان تر بود. به علاوه اینکه برای مطمئن شدن از اول بودن یک تابع نیازی به چک کردن تمام اعداد از 2 تا n-1 نداریم بلکه فقط تا جذر n کافی است پس با بهتر کردن تابع به هدف نهایی خواهیم رسید. در ادامه از int استفاده می کنیم که مانند جزء صحیح یا براکت در ریاضی است که کف صحیح یک عدد را محاسبه میکند. در شکل 10 – تابع بهینه شده می توانیم شکل نهایی را ببینیم.

1. def is\_prime(x):
2. for n in range(2,int(x \*\* 0.5 )+1): # chon ta end-1 mire ma +1 kardim int(x\*\*2) ro
3. if x % n == 0:
4. return False
5. if x <=1 :
6. return False
7. return True
8. for i in range(1,101):
9. if is\_prime(i):
10. print(i)

شکل 10 – تابع بهینه شده

## دیباگ کردن

دیباگ کردن نکته ای بسیار مهم و کاربردی است که ضمن پیدا کردن خطا تابع به فهم بهتر از آن هم کمک شایانی می کند. با گذاشتن چندین break point و زدن کلید F5 or start debugging و استفاده از F10(step over) and F11(step into) که به ترتیب داخل تابع و خط بعد می روند می توان این کار را انجام داد. اگر تابع اجرا نمی شد یا اینکه اجرا می شد ولی به درستی کار نی کرد می توان با این کار مشکل آن را پیدا کرد و آن را از میان برد. همچنین حتی اگر تابع درست هم کار کند می توان با استفاده از آن به نحوه دقیق کارکرد تابع پی برد. با این کار در سمت چپ برنامه چندین قسمت می بینیم. در قسمت variables می توانیم متغیر ها را ببینیم. در قسمت watch می توانیم متغیر های دلخواه را اضافه کنیم و به طور ویژه آن ها زیر نظر بگیریم. در قسمت call stack تابع های در حال اجرا را می بینیم . در قسمت breakpoints هم نقاط ایستادن برنامه را مشاهده می کنیم. در شکل 11 – دیباگ کردن تصویری از آن را مشاهده می کنیم.



شکل 11 – دیباگ کرد فاصله‌گذاری