УНИВЕРЗИТЕТ У БЕОГРАДУ

ФАКУЛТЕТ БЕЗБЕДНОСТИ
КАТЕДРА СТУДИЈА БЕЗБЕДНОСТИ

АЛГОРИТАМСКО РАЗМИШЉАЊЕ И АНАЛИЗА ПОДАТАКА ПОМОЋУ ПАЈТОНА

– ДИПЛОМСКИ РАД –

Ментор:
Ана Ковачевић
Проф. др

Студент:
Невенка Ного
19/17

Београд, 2023.

САДРЖАЈ

1. Увод

Британски савет је 2017. године покренуо пилот фазу пројекта „Школе за 21. век“. Иницијално, пројекат је обухватао 135 школа и 26 000 ђака са подручја Западног Балкана. Из године у годину обухват пројекта се ширио, те је данас, 2023. године предвиђено да ће кроз наредне три године пројекат достићи цифру од милион директних корисника. Идеја водиља пројекта је да су вештине критичког мишљења и решавања проблема кључне вештине 21. века, као и да је њихово усвајање задатак образовног система који је најједноставније остварити коришћењем такозваних програмабилних уређаја, попут микробита (British Council, н.д). Исте 2017. године Рачунарство и информатика постаје обавезан предмет за ђаке петог разреда основне школе, а најважнија новина у односу на период пре 2017. године када је Информатика била изборни предмет јесте што званичан план и програм наставе сада укључује и програмирање (Влада Републике Србије, 2017). У међувремену се број пројеката који имају за циљ подизање дигиталних компетенција повећава, Рачунарство и информатика, а тиме и програмирање, консекутивно се уводе у све разреде основних и средњих школа, отварају се одељења ученика са посебним способностима за Рачунарство и информатику при редовним гимназијама широм државе, покрећу се програми ИТ (информационе технологије) преквалификација, развоја дигиталних компетенција у осетљивим друштвеним групама, Влада Републике Србије се све више дигитализује. Чак се и индустрија видео игара укључује у екосистем подсектора ИТ индустрије који улажу у образовање. Тако је на пример, на конференцији Асоцијације индустрије видео игара Србије (SGA – Serbian Gaming Association) под слоганом „За победу!“ (For the Win!) која је одржана у мају месецу 2023. уз панел о екосистему игара региона, одржан и панел о додирним тачкама индустрије видео игара и образовања (Serbian Gamming Association, 2023). А познато је и да је велика компанија која се бави видео играма, Нордеус (Nordeus), основала своју фондацију, која се фокусира на образовање (Nordeus фондација, 2023). Слична атмосфера

постоји и у другим секторима ИТ индустрије. Тако је још један од лепих примера дугорочне посвећености компанија образовању и усавршавању и тачмичење познато као Бабл кап (Bubble Cup) које је годинама организовао Мајкрософтов развојни центар Србија, а које мотивише талентоване средњошколце и студенте да изаберу каријеру у области информационо-комуникационих технологија (Microsoft Development Center Serbia, н.д). Из ових и сличних активности компанија видимо да је образовање у сфери дигиталних технологија постало изузетно важна карика развоја будуће радне снаге. ИТ компаније су схватиле значај квалитетне и образоване радне снаге, те у овој области не мањка прилика за плаћену праксу, посету седишту компаније, помоћи пројектима који за циљ имају унапређење образовања и подршку мотивисаним ђацима.

Важно је истаћи да се у свему овоме не ради искључиво о образовању будућих софтверских инжењера и ИТ стручњака, иако је то дефицитарна радна снага, већ о оспособљавању запослених у свим индустријама за аутоматизацију посла коришћењем дигиталних технологија. Значај раног учења о дигиталним технологимаја су увидели и представници Министарства просвете и Владе Републике Србије који обезбеђују фунансијска средства за пројекте у овој области, али као што је показано и представници једне од најпрофитабилнијих индустрија у држави. Образовни систем у најширем смислу у последњих неколико година уложио је много напора да своје кориснике припреми на послове будућности, и чини се да ће и наставити да их улаже. Већ данас је тешко замислити посао који је могуће обављати без употребе рачунара, а у будућности је питање да ли ће то уопште бити могуће. Постаје јасно да је висок ниво дигиталних компетеција *conditio sine qua non*, те да нас на разговору за посао више не питају да ли, већ колико добро познајемо рад у програмима за табеларна израчунавања и које смо све алате за организацију посла имали прилика да користимо (овде свеска и оловка нису пожељан одговор).

У вези са пословима и тржиштем рада, један потпуно други ниво који постоји у паралели са неопходношћу поседовања дигиталних вештина, а који је неопходан за обављање савремених послова конкретно безбедности или шире за обављање послова из друштвено-хуманистичке сфере тиче се информација. Наиме, чини се да доношење сваке пословне одлуке захтева сумирање свих знања која имамо о одређеној теми, односно покушај предвиђања будућег развоја догађаја. Информације које су нам потребне морају бити потпуне, свеобухватне и проверене. У ери глобализације, подаци се налазе свуда око нас, медији и интернет су нам омогућили приступ мноштву информација, међутим евидентно је да оне нису увек у потпуности тачне или да нису дате у реалном контексту. Да бисмо објективно сагледали мноштво информација којима смо изложени, потребна нам је помоћ дигиталних технологија, односно потребно нам је да уз помоћ технологије вршимо анализу и обраду података.

Сада долазимо до проблема на који овим радом желим да што боље одговорим. Како су генерације које су училе програмирање кроз своје формално образовање већ изузетно близу да закораче на тржиште рада, а поседовање напредних дигиталних компетенција постаје неопходан услов за квалитетно обављање посла поставља се питање како генерације које нису имале програмирање у свом формалном образовању или чак нису слушале Информатику као предмет да остану конкурентне. Врло кратак одговор на такво питање би био да информатику треба да посматрају као алат за рад, чије коришћење се може савладати и ван система формалног образовања. Дужи и свеобухватнији одговор желим да пружим кроз овај рад, као што желим и да објасним да није свако програмирање једнако послу софтверског инжењера, и да није сваки рад са информацијама једнак обавештајно-безбедносном раду, а најважније желим да истакнем да је примена основних концепата програмирања и анализе података, за чије савладавање није неопходна позадина у математичким или електротехничким наукама, у обављању великог скупа послова неопходна, на

сличан начин на који је познавање енглеског језика или поседовање возачке дозволе често неопходно. Радом желим да пружим информације о изворима из којих се може учити и да дам преглед основних алата које је потребно савладати, њихових могућности и међусобне повезаности који издвајају кандидата на тржишту рада. Желим да покажем како се започиње рад у овим важним програмима и окружењима и омогућим даље самостално истраживање.

2. Алгоритамски начин размишљања

У Уводу је већ представљено да су знања из области информатике веома важна за обављање савремених послова. Програми и окружења са којима се упознајемо кроз рачунарске науке представљају својеврстан алат. Коришћење алата за обављање послова није новина. Готово свака професија захтева коришћење прикладног алата, пољопривреда захтева коришћење ашова, мотике, трактора, фризерај фена и четке, молерај ваљка, графички дизајн Илустратора (Illustrator) и/или Фотошопа (Photoshop), архитектура Аутокеда (AutoCad) и слично. Савремени послови најчешће подразумевају коришћење дигиталних алата, (поменути илустратор, фотошоп и аутокед су, на пример, дигитални алати) за чије основно коришћење је довољно бити дигитално писмен, али уз помоћ којих би се могло радити једноставније и брже уз познавање неких основа програмирања, те разумевања како сам алат функционише.. Мало другачије речено, за ефикасно обављање посла коришћењем дигиталног алата често је пожељно поседовати напредне дигиталне компетенције, уместо основне дигиталне писмености. **Напредна дигитална компетенција** најчешће се изједначава са програмирањем почетног нивоа. Сада, да бисмо уопште разумели главне концепте програмирања који чине тај почетни ниво и увидели зашто је њихово познавање од значаја за послове 21. века, морамо да пођемо од тога шта оно заправо представља, односно како је дефинисано. Како се наводи на једном од најпознатијих сајтова за учење програмирања: „Програмирање је ментални процес осмишљавања инструкција које се упућују рачунару“ (Codecademy, н.д). Ово је различито од кодирања, које се тумачи као процес трансформисања идеја у писани језик који је рачунар у стању да разуме (Codecademy, н.д). Код дефиниције програмирања важно је подвући да је у питању ментални процес осмишљавања. Програмирање је, дакле, размишљање о одређеном проблему у одговарајућим оквирима, његово рашчлањивање на мање проблеме и приступање решењу корак по корак. Овако посматрано, програмирање је заправо вештина која је потпуно независна од рачунара и која се може применити

на све области људског деловања. Када желимо да рачунар уместо нас приступа решењу проблема корак по корак, ми њега „испрограмирамо“, а за то је неопходно да рачунар разуме нашу команду. Да бисмо постигли то разумевање користимо програмске језике. Програмски језик је писани језик који је рачунар у стању да разуме или прецизније: програмски језик је скуп свих речи, односно наредби које рачунари разумеју (Фондација Петља, н.д. а). Међутим, исто као што за савладавање природног језика (енглески, италијански, кинески) и комуникацију на њему није довољно научити напамет речник, тако ни за кодирање није довољно научити како се задају наредбе у неком од многобројних програмских језика, већ је важно да познајемо логику рачунара и имамо адекватан приступ решавању проблема, односно важан је спој са програмирањем које посматрамо као гореописану вештину.

За познавање логике рачунара неопходно је да будемо његов корисник, а јако је пожељно да будемо напредан корисник, а аналогији са језиком, изложеност језику умногоме ће олакшати савладавање језика, на исти начин коришћење рачунара ће олакшати савладавање основа програмирања. Напредно коришћење већ испрограмираних софтвера, узмимо за пример оне најпознатије из Мајкрософтовог офис (Microsoft Offce) пакета, подразумева разумевање процеса који се дешавају иза екрана, а који омогућавају резултат који ми видимо. Тако, када као напредни корисник бојимо ћелије у Ексел (Excel) табели ми имамо у виду да исту табелу нећемо моћи једноставно да филтрирамо, односно да постоји шанса да се приликом филтритања података боје поремете. То се дешава зато што Ексел није програмиран да боје тумачи као податке на исти начин као што то чини са текстуалним или нумеричким податком. Једном када то разумемо, проблем ћемо решити врло једноставно, уместо разврставања ћелија по бојама, додаћемо поред њих једну нову колону и уписивати статус онога што нам је потребно да разликујемо алфанумеричким карактерима. Тада ће филтрирање бити исправно. Исто тако, када прелазак на нову страницу Ворд (Word) документа решавамо тако

што притиснемо тастер Ентер (Enter) онолико пута колико је потребно да се курсор премести на нову страницу, може се десити да приликом штампања или отварања документа на туђем рачунару, резултат није онакав какав смо очекивали. Проблем настаје услед тога што Ворд сваки притисак на тастер Ентер тумачи као прелазак у нови ред, а број редова на страници варира у односу на фонт, размак између редова, формат папира и слично. Да бисмо увек, на свим рачунарима, и на сваком формату папира добили очекивани резултат, потребно је да команду преласка у нови ред задајемо коришћењем опције за прелом странице (Insert Page Break). Када овакво разумевање рада рачунара постоји, отворен је пут ка високим дигиталним компетенцијама, укључујући и програмирање.

Основни концепти програмирања су толико једноставни да постоји више организација у свету које њима уче чак и децу предшколског узраста, (Digital schoolhouse, н.д) те не треба осећати страх од те велике речи. Програмирање је, дакле, за свакога, јер се основни концепти програмирања врло једноставно могу приближити чак и детету, за то је потребно само мало креативности. Ево примера како: замислимо да поседујемо робота коме је могуће издати команде природним језиком и да желимо да му кажемо да иде напред 10 корака. Да би извршио наредбу, робот мора да зна шта је корак, како да га направи, која је предвиђена дужина сваког корака, где је напред и слично. **Робота** прво морамо да научимо све што је важно да разуме у вези са корачањем, а затим да му кажемо „Иди напред 10 пута по један корак“. Овиме смо га испрограмирали. Програмирањем ми рачунару говоримо шта желимо да уради у одређеној ситуацији. Односно, задајемо му низ наредби које разуме, а потребно је да изврши одређеним редоследом и одређени број пута. Уколико смо успешни, рачунар ће сваки пут задати проблем решити на исти начин. Начин издавања наредбе је ствар техничке природе, у аналогији са природним језицима, само издавање наредбе би било учење речника, док суштина лежи у тачном схватању сазнања које робот има о томе шта је кретање, односно корак и како се корача и начину на који му је то сазнање дато. Способност

комуникације са роботом када је представљена на овакав начин има свако од нас, укључујући и свако дете. Осим робота, још један пример путем ког је учење основних концепата програмирања на конференцији „За победу!“ представила Шанила Сид (Shahneila Saeed), задужена за образовање у организацији УКИ (UKIE), а који приближава програмирање просечном човеку јесте учење кроз плес. Пример је следећи: замислите да морате да осмислите плесну кореографију и научите своје пријатеље како да је изведу. На располагању имате цртеже са различитим положајима тела и задатак је да их сложите на под, те да на основу сложених цртежа ви и ваши пријатељи одиграте осмишљену кореографију. Прва особа која чита цртеже и изводи покрете показаће да ли је кореографију физички могуће извести, уколико направи грешку, особе из публике ће је исправити, они који пажљиво посматрају моћи ће и да дају мишљење о томе да ли је у питању најбољи редослед покрета, или је нешто могуће побољшати – и у томе већ имате концепт алгоритма, дебаговања, тестирања и оптимизације (Serbian Gamming Association, 2023). Концепти које најчешће везујемо за програмирање уствари су свеприсутни. Алгоритам је низ корака који доводе до решења неког проблема (Фондација Петља, н.д. б). Алгоритамски начин размишљања је, стога, креирање низа корака и затим њихово извршавање одређеним редоследом да би се решио проблем или извршио задатак, и то на такав начин да свако може да га понови. Са алгоритмима се сусрећемо свакодневно. Тако је на пример, најједноставнији рецепт за палачинке алгоритам – у чинију ставити 500г брашна, додати два јаја, 200мл млека, прсохват соли, кашичицу шећера, додати чашу киселе воде и умутити. Ако размислите о палачинкама на мало апстрактнијем нивоу, видећете да ћемо сваки пут када поновимо гореописани низ корака, добити смесу за палачинке. Иста ствар се дешава и са кореографијом коју креирамо и изводимо уз помоћ слагања сличица или било којом другом кореографијом, она је алгоритам – низ поновљених положаја тела одређеним редоследом, који свако може да изведе. На исти начин, рачунар сваки пут када изврши низ корака који смо му задали, добиће исти

резултат. Сам концепт алгоритма, дакле, није сложен. Њега видимо као сложеног јер се често везује за комплексне математичке проблеме. Међутим, чак и у мање свакодневном контексту, односно у ближој вези са математиком и рачунарским наукама, концепт алгоритма може бити показан на начин на који је чак и детету пријемчив, а одрастао човек га свакако мора разумети. Један такав пример алгоритма који је ближи програмирању него што је то прављење палачинки јесте тзв. Ним игрица (Nim Game). Ним игрицу играју два играча која наизменично повлаче потезе. На табли се налази број жетона x (узмимо да је x једнако 15). Сваким својим потезом играч може да склони са табле 1, 2 или 3 жетона. Побеђује играч који склони последњи жетон. Уколико примени прави алгоритам, играч који игра први увек ће победити (Фондација Петља, н.д. в).

Илустрација 1 - Ним игрица; Извор: Фондација Петља

Да бисмо победили у Ним игрици, размишљамо уназад. Побеђује играч који

противнику на табли остави 4 жетона. Противник колико год жетона да узме (сетимо се да је дозвољено узети 1, 2 или 3 жетона) оставља на табли онолико колико први играч може да узме једним потезом. То значи да играч који игра први мора након свог потеза на табли да остави број жетона који је дељив са 4. Након првог потеза, прати колико противник узима и узима онолико колико фали до 4. Овде се примењује алгоритам који је делом линијски (извршава се корак по корак), делом условљен (потези једног играча зависе од потеза другог) и делом цикличан (неки кораци се понављају) (Фондација Петља, н.д. в). Рачунар ће, када се тако испрограмира, уколико игра први, сваки пут победити у Ним игрици. Размишљати алгоритамски значи разумети како рачунар размишља. Осим применљивости високих дигиталних компетенција у свакодневном животу, развој алгоритамског начина размишљања нека је врста менталне хигијене. Из примера Ним игрице јасно је да за њено решење није неопходно познавање програмског језика, већ да је она својеврсна загонетка. Када се алгоритам по ком Ним игрица ради научи, она је увек лако решива. Слично функционишу и Рубикове коцке, чини се да је немогуће сложити их, међутим познато је да постоје људи који их слажу за свега неколико секунди. Њихова тајна је у томе што знају које покрете је потребно колико пута поновити и којим редоследом да би се сваки део Рубикове коцке на одређени начин померио са једног места на друго, односно познају алгоритме за слагање коцке. Алгоритамски начин размишљања омогућава нам да растављањем проблема на ситније потпроблеме и решењем сваког од потпроблема корак по корак приступимо послу систематично и ефикасно га одрадимо. Додатно, порука коју шаље алгоритамски начин размишљања, а која је из чињенице да је слагање Рубикове коцке једнако познавању свега неколико секвенци покрета јасна, јесте да програмирање у некој мери апсолутно јесте за свакога. Оно не захтева генијалност, нити наслеђени таленат, већ захтева искључиво посвећеност и здрав разум. А с друге стране, осим што нам олакшава рад, програмирање нам омогућује да одржавамо менталну кондицију. То је идеја коју Небојша Васиљевић, директор

Фондације Петља, која се бави развојем алгоритамске писмености, често истиче. Тако је на пример и 27. априла 2023. године приликом отварања радионице поводом обележавања Међународног дана девојака у Информационо-комуникационим технологијама (United Nations Development Programme [UNDP], 2023), рекао да је мало програмирања добро за свакога на исти начин на који је мало спорта добро за свакога. Иако можда нећете бити професионални спортиста, бављење спортом вас одржава здравима и даје вам енергију. Исто тако, иако можда нећете бити софтверски инжењер, програмирање вам пружа користан алат за рад и осим тога тренира ваше размишљање.

3. Значај обраде података и употребна вредност алата за обраду подата са примерима из праксе

Програмирање нас, између осталог, чини дигитално писменијима, оспособљава нас за ефикасније обављање послова и тренира наше критичко мишљење и вештине решавања проблема. Сада је међутим, у овом поглављу, направљен заокрет са програмирања и његових значајних страна на обраду и анализу података и тај други паралелни ниво важних вештина за послове безбедности и послове из шире друштвено-хуманистичке сфере. Истакла бих важност анализе и поседовања тачних података. Ту важност данас потврђује и утисак да је најтраженији алат за рад, када посматрамо све професије најшире могуће, програм за рад са подацима – Ексел. Значај Ексела за запошљавање и развој каријере је изузетно велики. Тако на пример, платформа за запошљавање Линктин (LinkedIn) има опцију да на њој тестирате своје вештине коришћења овог програма и добијете „значку“ која потврђује да имате адекватан ниво вештине и налази се на вашем профилу тако да потенцијални послодавац има увид у њу. Притом опсег послова за чије обављање се користи Ексел је изузетно широк. Готово сви послови који подразумевају било какво рачунање, чување и мењање велике количине података, анализу података или логистику било које врсте ослањају се у великој мери на Ексел или други сличан програм. Основно познавање Ексела мора имати на пример сваки запослени који ради на каси, а који ће Ексел користити да би на дневном нивоу у евиденцију могао да унесе пазар, количину робе која је пристигла, број уплаћених карата за превоз или припејд кредита и слично. Основно познавање Ексела готово сигурно има и сваки запослени који се бави организацијом догађаја. Такав запослени вероватно у Ексел или други сличан програм уноси податке о простору где се догађај организује, гостима, распореду седења и доласка високих званица и тако даље. Основни рад у Екселу постаје подразумеван, као што је подразумевано да се буде писмен. Стога није за очекивати да ће нас познавање покретања, навигације кроз табелу, уношења података, формула за основне

рачунске операције, формула за минимум и максимум, просек и слично на било који начин издвојити уколико аплицирамо за послове пројектног менаџера, запосленог у људским ресурсима или менаџера безбедности. Није неопходно да сваку формулу знамо напамет, али је јако важно да разумемо могућности оваквих програма и да препознајемо послове које је могуће аутоматизовати и скратити време њиховог извршавања. Након тога, потребно је да умемо да дођемо до информације како да нешто израчунамо или извучемо из табеле, а најчешће је за то сазнање потребно свега неколико кликова. Тако на пример, уколико је потребно да спојимо две различите табеле које се подударају у само једном критеријуму, након пар минута истраживања на интернету биће нам препоручено да за то користимо функцију XLOOKUP, уколико је потребно да пребројимо колико се пута одређена вредност понавља у неком распону у оквиру ћелија табеле, користићемо функцију COUNTIF, уколико нам је потребно да графички представимо податке које имамо, и то је могуће урадити врло једноставмо у самом Екселу.

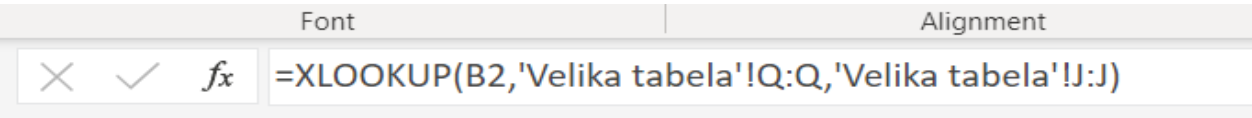
Ове функционалности могу бити јако корисне кад год се ради са подацима из пријава за различите активности или се извештава о општој расподели корисника или клијената по полу, годинама, местима из којих долазе и слично. Наводим само неке од примера употребе ових функција из праксе пројектног менаџмента, али који могу бити врло слични примерима из праксе менаџера безбедности или запосленог у сектору људских ресурса.

3.1. Пример коришћења XLOOKUP функције у Екселу

Организован је програм онлајн подршке учењу за који је расписан јавни позив и отворена пријава. Пријава је отворена коришћењем Мајкрософт формулара (Microsoft Forms). Све што је потребно да корисник уради да би се пријавио на програм јесте да кликне на линк, остави своје податке и одговори на питања организатора. Из угла организатора, све податке прикупљене формуларом

могуће је пребацити на свој рачунар у виду Ексел табеле. Током онлајн програма подршке учењу корисници раде задатке на независној платформи. Резултати рађења задатака преузимају се на рачунар у виду Ексел табеле. Након завршетка програма, корисницима који су приступили рађењу задатака потребно је послати електронском поштом потврде о учешћу на програму. Добро је да свака потврда буде персонализована, односно да садржи име и презиме корисника. Међутим, у табели у којој се налазе задаци могуће је пронаћи само адресе електронске поште корисника. Нешто више од 3000 корисника се пријавило на програм, дакле ручно претраживање сваке адресе електронске поште у табели са пријавама у којој знамо да постоји колона у којој су имена и презимена корисника није опција. Решење по корацима је следеће:

- Спојити обе табеле у један документ, тако да свака од њих чини одвојен радни лист (Excel Spredsheet). Радни лист у коме се налазе подаци преузети са Мајкрософтовог формулара, односно подаци из пријаве, називамо на пример „Велика табела“, док ћемо радни лист са подацима о задацима назвати „Задаци“.
- У оба радна листа пронаћи колону са адресама електронске поште (у неком другачијем примеру тражили бисмо колону за коју знамо да је заједничка обома радним листовима).
- У радном листу „Задаци“ креирати нову колону.
- У прву ћелију нове колоне унети следећу формулу: =XLOOKUP(назив једне ћелије у којој се налази адреса електронске поште у радном листу „Задаци“, назив колоне у којој се налазе све адресе електронске поште из радног листа „Велика табела“, назив колоне из радног листа „Велика табела“ у којој се налазе сва имена и презимена). Називе ћелија и колона потребно је одвојити зарезом и они се најпрецизније добијају тако што се једноставно након куцања формуле кликне на ћелију која нам је потребна, односно означи колона која нам је потребна.



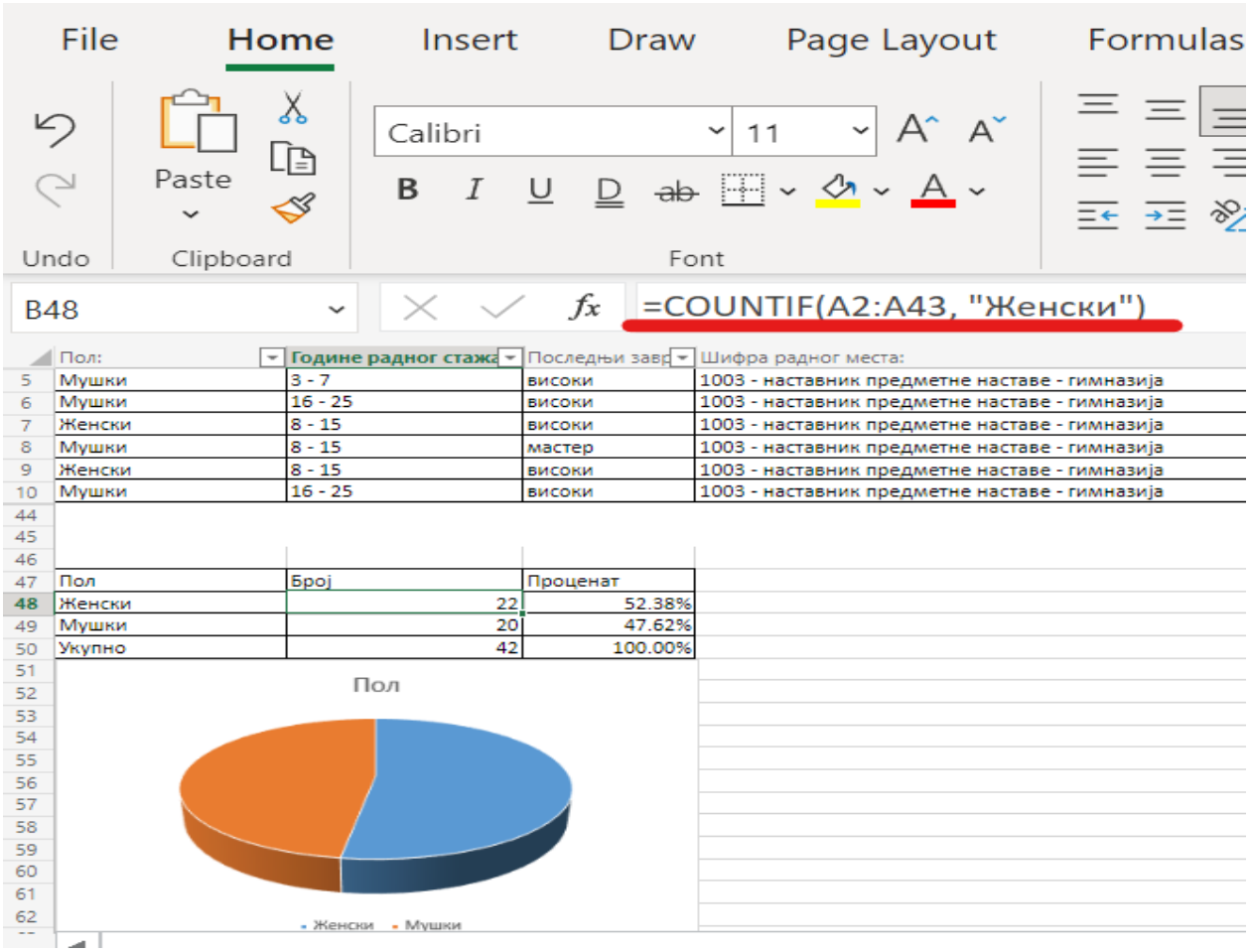
Илустрација 2 - Функција XLOOKUP

- Формулу превући до краја нове колоне.
- Проверити да ли је у неком реду враћена грешка, може да се деси да је на пример неко од корисника приликом пријављивања за програм користио једну адресу електронске поште, а приликом рађења задатака другу. Овакве случајеве уколико их има, решити један по један.

3.2. Пример коришћења COUNTIF функције у Екселу

Организована је активност обуке наставника који предају у одељењима ученика са посебним способностима за Рачунарство и информатику. Потпуно идентично као пример може послужити било која активност обучавања или усавршавања запослених. Зарад унапређења будућих обука потребно је извући податке о проценту жена које су се пријавиле на обуку, жена које су је савладале, о расподели наставника према радном стажу и последњем завршеном нивоу образовања, те на пример податке о типу средине из које долазе. Као и у претходном примеру, пријаве су пребачене на рачунар у Ексел табелу. Одговори на свако од питања су спаковани у по једну колону. Потребно је пребројити колико има наставника женског пола, колико мушког, колико њих има до 2 године радног стажа, колико има између 3 и 7, колико има између 8 и 15 и тако даље. Да би се резултати што једноставније представили, потребно је припремити визуелни приказ. Решење по корацима на примеру пола које се лако примењује и на остале информације које су нам потребне за унапређење будућих обука је следеће:

- Испод, поред или у другом радном листу табеле коју имамо направити малу табелу која ће садржати три колоне: пол, број и проценат и четири реда. У три реда испод реда који садржи наслов колоне пол биће уписано женски, мушки и укупно.
- У првој слободној ћелији колоне број унети функцију =COUNTIF(опсег из велике табеле из ког пребројавамо колико чега има, под отвореним наводницима текст који тражимо у опсегу. Напомена: потребно је уписати ког је пола особа на идентичан начин на који је то понуђено у великој табели са подацима коју смо извукли из пријава).
- Поновити поступак за све понуђене одговоре. Ово се може аутоматизовати тако што се уместо укуцавања текста који тражимо, позове вредност ћелије која се налази на пример у колони лево од колене у којој радимо, те се за сваки нови ред, формула једноставно превуче. Пре превлачења, потребно је фиксирати опсег у ком се претражује у формули.



Илустрација 3 - Функција COUNTIF

- Сада када смо извукли број, те знамо колико је на обуци било наставница, колико наставника и колико их је било укупно, проценат рачунамо врло једноставно, тако што број наставница поделимо са укупним бројем.
- Остало је још да коришћењем опције за убацивање графикона (Insert) нашим подацима дамо визуелни приказ.

Овим примерима из личне праксе првенствено бих поручила да је Ексел велики савезник сваког менаџера, био он пројектни менаџер или менаџер безбедности. У сваком послу у коме има организације, координације и рада са људима, а самим тиме и са подацима о њима, Ексел је јако моћан алат и добар колега. Додатно, јако је једноставан за коришћење. Једном када знамо које све

могућности пружа, све што је преостало јесте да претражимо на интернету како да их најједноставније применимо на наш конкретан случај. Није потребно знати сваку функцију напамет.

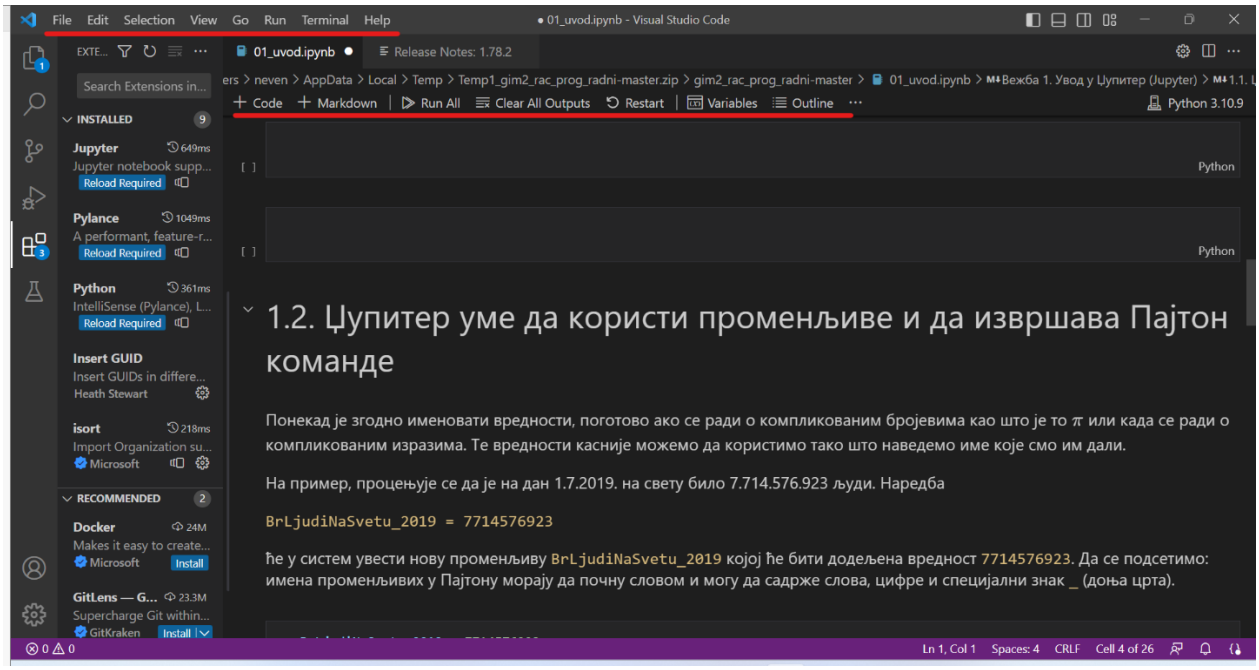
Осим корисности, значај Екселa огледа се и у томе што разумевање свега горенаведеног и неколико успешних покушаја рада са подацима преко популарног Мајкрософтовог софтвера може бити јако добар увод и одскочна даска ка сложенијој обради података. Та обрада података подразумева да из велике количине података можемо да извучемо закључак, пронађемо правилности и можемо да предвидимо резултат одређене радње у будућности или како је то наведено у литератури: „Обрада података јесте процес примене одређених алата и техника са циљем да се информације организују, изучавају, да се из њих изводе закључци и понекад да се на основу информација износе предвиђања“ (McFedries, 2022, стр. 8). Као пример једног од начина, односно алата за обраду података у овом раду биће приказана обрада, односно могућности обраде коришћењем Пајтона (Python), његове библиотеке Пандас (Pandas) и радног окружења Џупитер (Jupyter). Ови алати биће уопштено представљени, са фокусом на њихову инсталацију и покретање. Даље учење обраде података коришћењем било ког алата најефикасније се може реализовати испробавањем, грешењем и исправљањем грешака. Сви ови алати су релативно једноставни за коришћење, али пружају широк спектар могућности. Стога је однос времена потребног да се савладају и квалитета и свеобухватности резултата које је могуће постићи прикладан за прве кораке у обради података. Међутим, пре техничког учења како се користи алат, иако као и у другим професијама, алат нема значај уколико га не користимо на правилан начин, потребно је да умемо да општа правила спустимо на конретан случај који нам је потребан и да покажемо разумевање података на једном апстрактнијем нивоу. Важно је схватати како се одређени податак у односу на контекст може другачије протумачити и како је, поред обраде података, и комуникација резултата одређене анализе података јако значајна. Морамо

приликом анализе података имати у виду да се на основу наше презентације резултата анализе могу доносити важне одлуке. Самим тиме, потребно је анализу радити тако да у њене резултате увек будемо сигурни и увек располажемо аргументима који ће потврдити наш став у вези са одређеним питањем, односно морамо податак током процеса анализе ставити у контекст и врло пажљиво и уредно водити евиденције које имамо на располагању. Као што је за разумевање функционисања система важно знати у каквом се односу налазе подсистеми у оквиру датог система, као и какав је однос система са окружењем, тако је и за разумевање резултата анализе података важно разумети да они зависе од контекста у ком се анализа дешава. А контекст даје онај ко податак обрађује. Курс „Буди Data Driven“ Фондације Петља то показује кроз навод да је „податак сирова вредност настала као резултат мерења или регистровања неког својства, те да таква вредност не говори пуно о самом објекту или феномену који истражујемо све док је не обрадимо заједно са другим подацима“. Односно, истиче се да се податак разликује од информације и то тако што је информација заправо обрађен скуп података, смислено интерпретиран и смештен у одговарајући контекс (Фондација Петља, н.д. г). За човека је поседовање информације увек значајније од поседовања податка. Наводим пример у ком податак сам по себи, као што је већ истакнуто, не говори пуно: податак је да је тренутна температура ваздуха 14 степени. Без других података, да је напољу суво, ведро и сунчано без ветра, као и да је место мерења температуре парк, у хладовини, и да се ради о мајском јутру у Републици Србији, на Балканском полуострву, не можемо да закључимо да ли је у питању некаква аномалија или не, не можемо чак да закључимо ни да ли је потребно да обучемо мајицу, џемпер или јакну да би нам напољу било пријатно. Стога, да бисмо од податка добили информацију, која за нас има тежину, односно вредност, која нам поможе да донесемо адекватне одлуке, потребно је да имамо на располагању још неке податке и да можемо да свему дамо контекст, односно креирамо информацију.

Важност информација истичу и професори Факултета Безбедности, проф. др Горан Мандић и доц. др Ненад Путник, када податак и информацију, сврставају у највредније корпоративне ресурсе, тј. она добра која доприносе способности корпорације да води пословање и стиче профит. Такође на сличан начин као претходно наведени извори објашњавају разлику између податка и информације. Истичу да анализом и обрадом податка, те његовом употребом, тај податак добија квалитативну вредност и тиме постаје информација. Информација, дакле, настаје, процесом обраде податка од стране корисника (Мандић и сар., 2017, стр. 20). Из овога видимо да је рад са информацијама неизбежан, те да информације представљају велику вредност сваке организације која их има на располагању и велику опасност за оне организације које их немају. Информације нам помажу да донесемо одлуке, предвидимо будућа дешавања и правовремено одреагујемо на ризике и искористимо могућности. Уз помоћ информација можемо унапредити своје радне процедуре, исправити грешке и одреаговати на потребе тржишта. А да бисмо у бескрајном мноштву информација којима смо окружени пронашли баш онај смисао који задовољава наше потребе, неопходно је да се послужимо обрадом и анализом података и информација. Затим, да би таква обрада дала довољно добре резултате, важно је да користимо дигиталне алате за обраду и анализу података и информација јер једино помоћу дигиталних алата можемо довољно брзо (пре него што се информације промене или се њихов значај умањи) добити резултате обраде велике количине података.

4. Увод у Џупитер

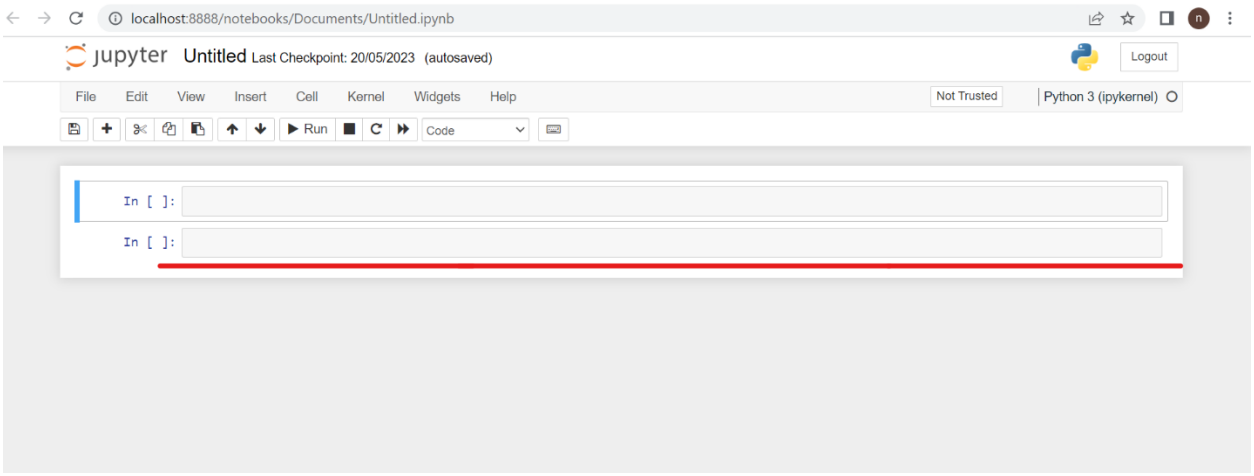
Да заокренемо назад на програмирање и напредне дигиталне компетенције. Чини се да то стереотипно схватање програмирања као изузетно тешког и немогућег за разумети произилази из замисли црних екрана са откуцаним словима у бесконачним редовима из којих не разазнајемо ништа. Када помислимо на програмирање, најчешће помишљамо на приказивање програмера у популарној култури, филмовима и серијама. Видимо програмирање као бесконачно велику командну линију у којој се дешава наука или чак научна фантастика коју нисмо у стању да поимимо. Међутим, реалност је, захваљујући развојним окружењима за извршавање кода, доста другачија. Ова окружења креирана су тако да сналажење корисника у њима буде врло једноставно и интуитивно, најчешће имају уграђену неку врсту асистенције за писање кода, слично као када на телефону постоји аутокорект (autocorrect) који исправља словне грешке. Постоји велики број окружења за извршавање кода, а међу најпознатијама су Спајдер (Spider), ВС Код (Visual Studio Code), Интегрисано окружење за развој и учење (IDLE – Integrated Development and Learning Environment) и тако даље. Свако од поменутих окружења одликује се одређеним карактеристикама, али оно што је јако важно нагласити јесте да се сва окружења константно надограђују тако да постају све једноставнија за коришћење. Осим тога, главне могућности свих окружења сличне су главним могућностима било ког другог програма, омогућавају да нешто сачувате, отворите ново, копирате, налепите, уметнете итд. На фотографији испод приказано је окружење ВС Код, подвучена је командна линија која изузетно подсећа на командну линију у Ворду. Ова окружења нам, дакле, нису толико страна.



Илустрација 4 - Окружење ВС Код

Као и фотографисани ВС Код, и Џупитер је окружење у коме се могу извршавати Пајтон програми. У Џупитеру, анализе су организоване у радне свеске, које представљају фајлове (files) са екстензијом .ipynb. На горњој фотографији, једна Џупитер радна свеска отворена је у окружењу ВС Код. Ова окружења су дакле компатабилна једно са другим. Исти код који се може отворити у Интегрисаном окружењу за развој и учење, може се отворити на пример и у ВС Коду.

Свака Џупитер радна свеска се састоји од низа ћелија (подвучена црвеном линијом на доњој фотографији), док свака ћелија може да садржи текст, математички израз или Пајтон наредбе.



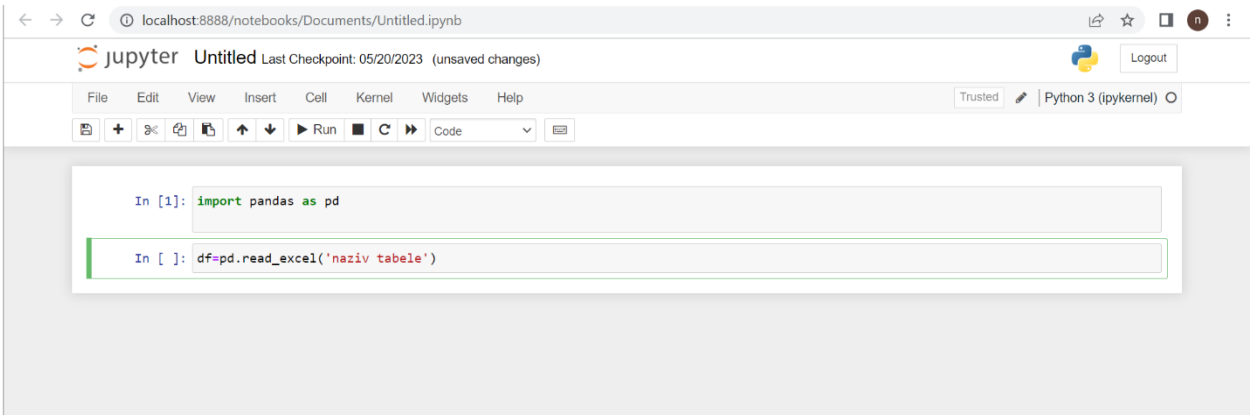
Илустрација 5 - Ћелија у Џупитер радној свесци

У питању је окружење које се може користити колаборативно, и подржава различите програмске језике (али се најчешће повезује са Пајтоном). Важно је истаћи да су Пајтон, све његове библиотеке и Џупитер потпуно бесплатни. Свако може без икакве надокнаде да их инсталира и користи на свом рачунару (Jupyter, н. д). С обзиром на то да је у контексту рада са подацима већ помињан Ексел сада ће се упоредити и истаћи узајамну везу ова два алата. Познато је да се Ексел плаћа, те ту предност Џупитера нема потребе додатно образлагати. Међутим, без обзира на комерцијалну цену, не треба занемарити вредност Ексела као алата за складиштење и обраду података. Он дозвољава манипулацију подацима, филтрирање, претраживање, израчунавања. Штавише, омогућава нам креирање такозваних макроа (Macro), који представљају запамћени сет корака који се може именовати и користити по потреби. Ексел има јако широку примену и потпуно је читљив највећем делу популације, а такође је једноставно интегрисати га у друге алате. То смо већ видели из гореописаних примера када податке прикупљене Мајкрософтовим формуларом можемо изузетно једноставно преузети на свој рачунар у виду Ексел табеле. Међутим, код Џупитера се, осим цене, издвајају још две важне погодности, а то су:

- Јасно видљива процедура за обраду података – У великим табелама није лако установити која формула зависи од које ћелије, нити којим ће се редом израчунавати формуле. Лако је поделити саму табелу, али није лако поделити процес који табела имплементира (Фондација Петља, н.д. д). Ова мана класичних програма за табеларна израчунавања се у пракси врло примећује у тренуцима када је потребно упознати нову особу са табелом на којој је неко претходно радио. Насупрот томе, обрада података коришћењем језика попут Пајтона омогућава да се процес врло једноставно подели са колегама. Важан допринос овоме представљају коментари које можемо да оставимо у самом коду. У коментару може стајати објашњење шта која линија кода ради, или која логика стоји иза конкретног приступа обраде података. У коментару може стајати и охрабрујућа порука колеги који наставља са радом на неком пројекту или препорука за добру књигу. У коментару може писати апсолутно све, јер је коментар део, у нашем случају Пајтон кода, обележен # који се не извршава. Овако, путем коментара, се може реконструисати целокупан процес обраде података, што је за колаборативан рад јако важно јер омогућава правовремено уочавање и исправљање грешака, као и прилагођавање већ постојећег кода новим потребама.
- Флексибилност – Ова предност Џупитера у односу на Ексел се дефинитивно може одразити на крајњег корисника. Главна разлика је у томе што се надоградња Пајтона огледа у константном креирању нових библиотека, које се затим врло једноставно импортују у постојећи програм. То значи да се корисничко искуство не мења, већ корисник једноставно повремено добија могућност да импортује нове пакете функционалности. Да ли ће то урадити или не никако не утиче на његове претходне пројекте, нити ремети колаборацију. Корисници који изаберу да импортују различите библиотеке и даље без икаквих проблема могу да читају сваки код. С друге стране, надоградња Ексел-а подразумева поновно инсталирање целог програма

(Фондација Петља, н.д. д). Дешава се да услед поседовања различитих верзија програма, један корисник не може да отвори документ који му је други послао, или уколико успе да га отвори, приказани подаци код њега изгледају другачије.

Поред предности и мана једног у односу на други, важно је истаћи и узајамну повезаност Екселa и Џупитера, а која се огледа у чињеници да коришћењем библиотеке Пандас подаци које чувамо у Ексел табели могу да се учитају у Џупитер радно окружење. То значи да податке које бисмо иначе добили у Екселу (попут података из пријава из гореописаних примера из праксе) у сваком случају имамо могућност да прочитамо и у Џупитеру. Две ћелије на фотографији испод импортују Пандас библиотеку, а затим налажу читање табеле у Џупитеру. Услов да овај код ради јесте да се табела која нам је потребна налази у истом директоријуму као и радна свеска. У случају да се не налазе у истом директоријуму, уместо имена табеле потребно је дати путању ка њој, на пример: df=pd.read_excel('C:/Users/neven/OneDrive/Desktop/UNDP/tabela.xlsx')

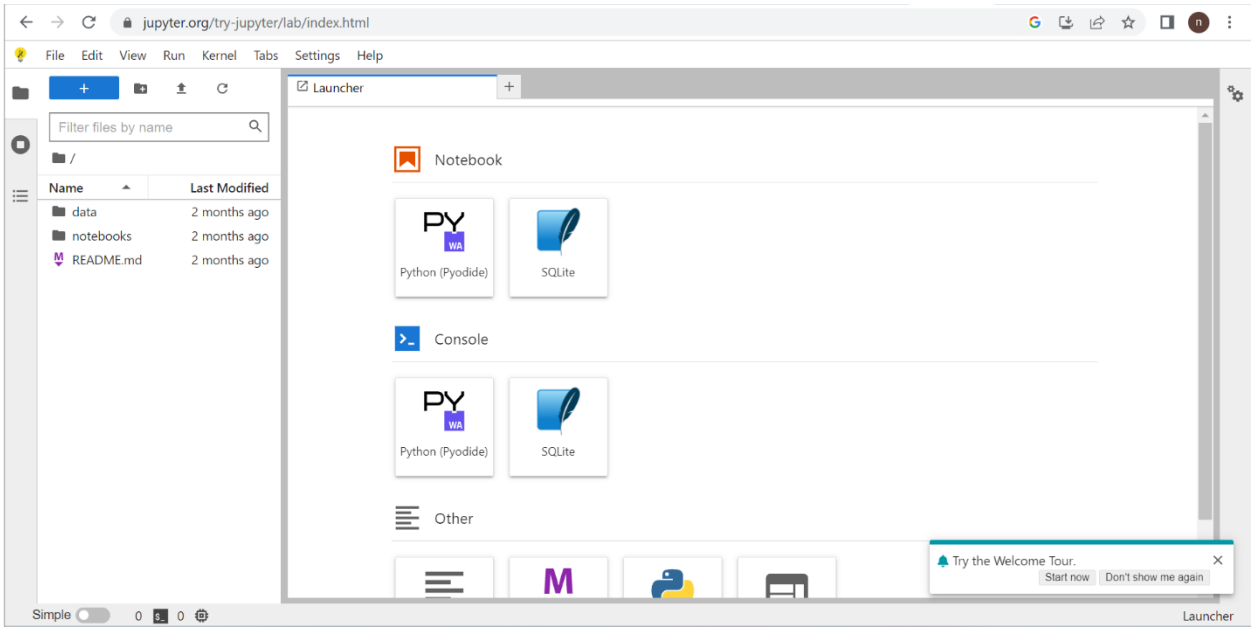


Илустрација 6 - Читање Ексел табеле помоћу Пандас библиотеке

Осим могућности повезивања Екселa и Џупитера, Пандас пружа богат скуп функција за манипулацију подацима, као што су филтрирање, сортирање, груписање, спајање табела итд. Међутим, пре него што прикажемо главне

функције Пандаса, потребно је да се вратимо неколико корака уназад и прикажемо на који начин се покрећу Џупитер и његове радне свеске.

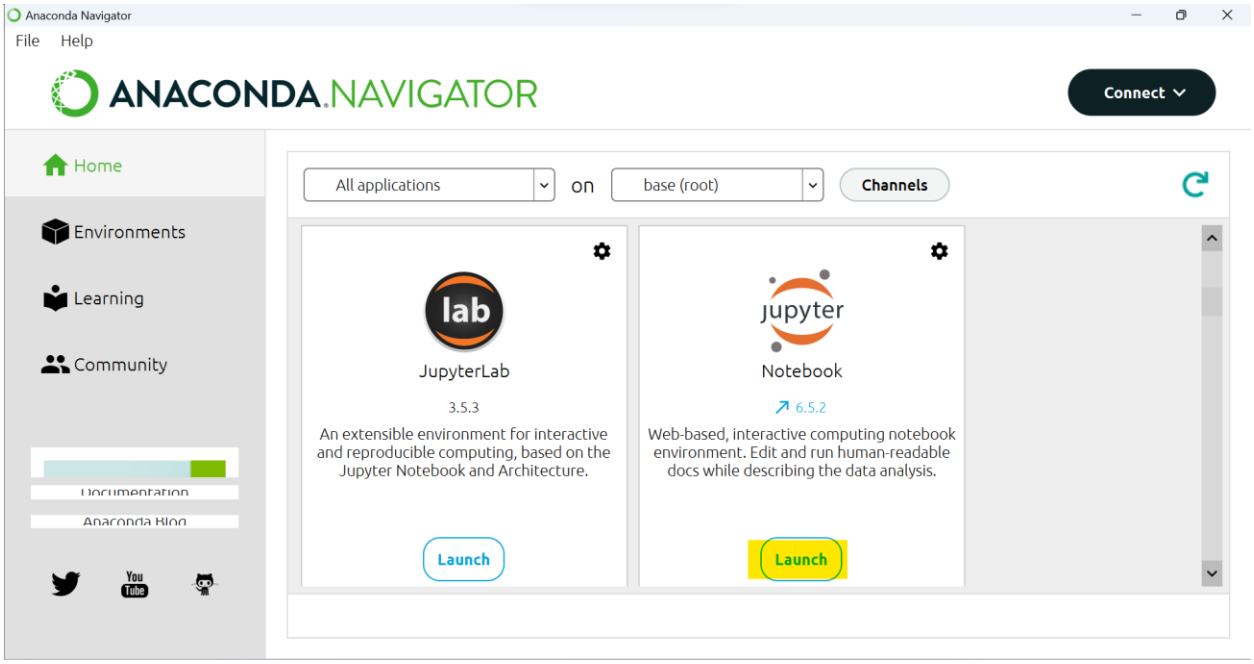
Џупитер радну свеску можемо покренути из интернет прегледача (Browser) и она се тада назива Џупитер Лајт (JupyterLite) или са свог рачунара. Џупитер Лајт покрећемо тако што у интернет претраживач укуцамо „JupyterLite“.



Илустрација 7 - Џупитер Лајт

Овај начин не захтева додатну инсталацију, међутим има своја ограничења. На пример, унете промене неће бити трајно запамћене, извршавање кода је спорије и може се десити да не ради све на исти начин на који ради у инсталираном окружењу (Фондација Петља, н.д. ђ). А како ни покретање Џупитера са рачунара није нимало компликовано, препоручује се други начин. Само прво покретање Џупитера захтева мало времена и то искључиво зато што Џупитер са рачунара покрећемо тек након што имамо инсталиране Пајтон и Анаконду (Anaconda). Исталација Пајтона и Анаконде је врло интуитивна. Потребно је само у интернет претраживач укуцати та два појма и након тога, програми се врло једноставно пребацују на наш рачунар и покрећу. За Пајтон смо већ напоменули да је

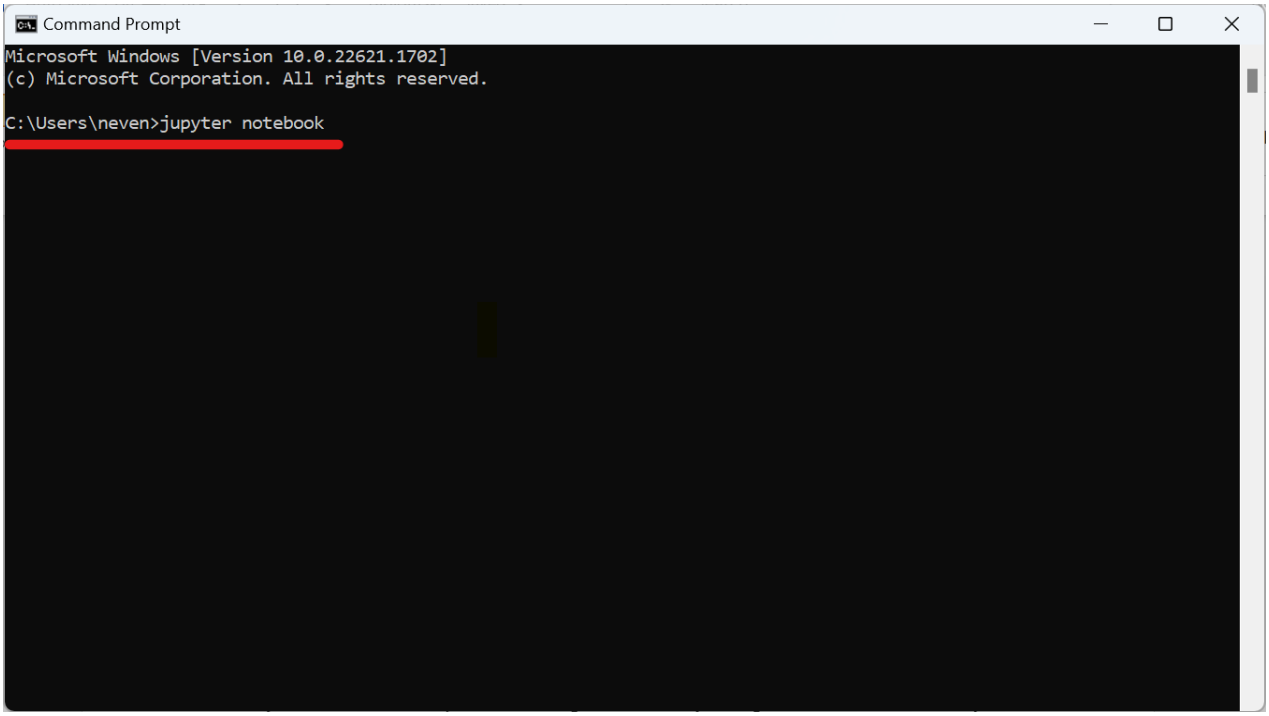
програмски језик, док је Анаконда програмско окружење за развој софтвера које се користи за анализу података и машинско учење. Она садржи скуп алата, библиотека и окружења (међу којима и Пандас) и олакшава управљање пакетима и виртуелним окружењем. Долази са већ преинсталираним Џупитер радним свескама, те се коришћењем Анаконде њихово покретање своди на један клик. Након инсталације Анаконде Џупитер је могуће покренути из Анаконде кликом на дугме за „лансирање“ пакета. На фотографији испод то дугме је подвучено жутим маркером.



Илустрација 8 - Покретање Џупитера из Анаконде

Овде је једино потребно пронаћи пакет који нам је потребан, а с обзиром на то да се у Анаконди налазе многобројни различити пакети то може бити спорији начин. Значајно бржа алтернатива, када већ имамо инсталирану Анаконду је покретање Џупитера из командне линије, које се ради тако што у командну линију (cmd) укуцамо „jupyter notebook“. Командна линија може се покренути из поља за претраживање. На фотографији испод приказана је тек отворена командна линија у коју је укуцана команда за отварање Џупитер радник свески. Кликом на дугме

Ентер, рачунар би нас аутоматски пребацио на прозор са отвореном радном свеском.

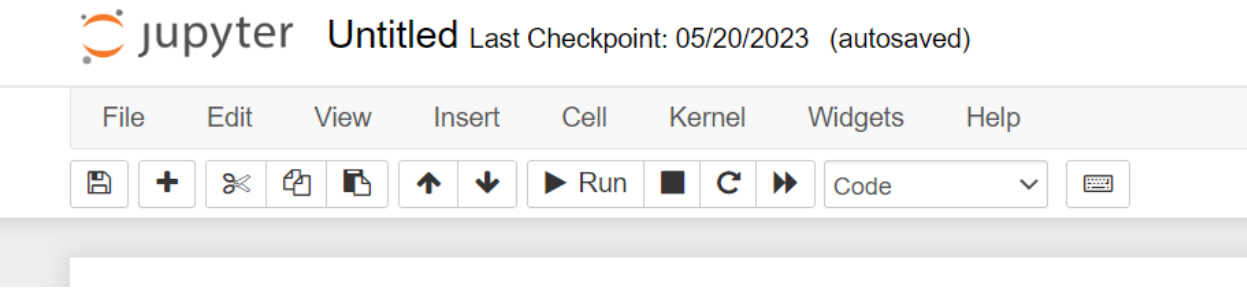


Илустрација 9 - Покретање Џупитера из командне линије

Инсталирање Анаконде није препоручено искључиво зарад олакшаног покретања Џупитера, већ Анаконда такође олакшава управљање пакетима и окружењима. Она има могућност креирања одвојених виртуелних окружења за различите пројекте, што спречава конфликте између библиотека. Док заједничко коришћење Џупитера и Анаконде омогућава велику флексибилност у обради података, јер у истом документу можете да извршите код по деловима, прикажете резултате корак по корак, комбинујете код, текст и визуализацију и све то делите са колегама.

Једном када смо покренули Џупитер, постаје врло интуитивно користити га. Линија са командама доста подсећа на линију са командама у програмима које имамо прилике свакодневно да користимо, попут Ворда или Екселa, тако иконица маказица означава команду сеци (cut), односно покреће премештање ћелије на неко

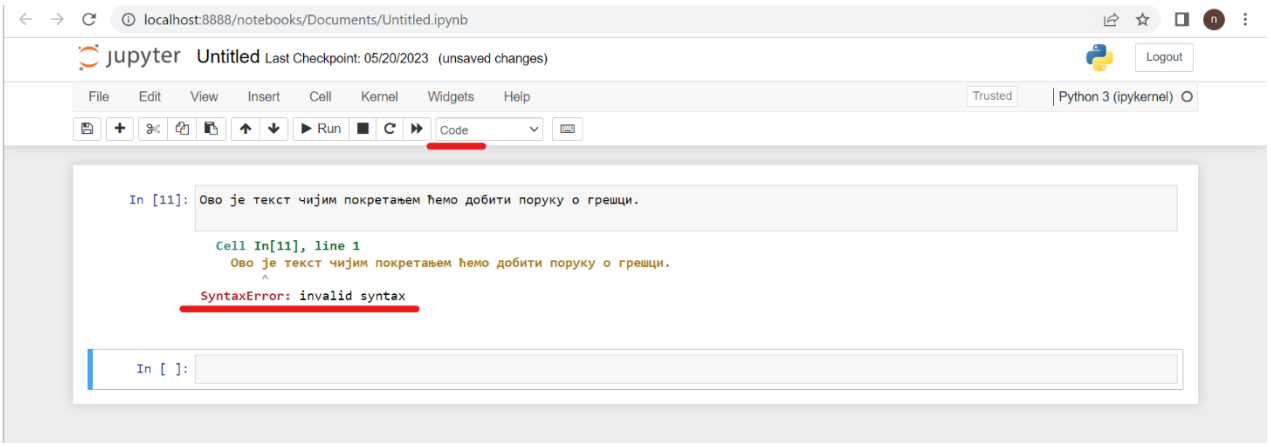
друго место, команда за копирање (copy) и команда „налепи“ (paste) су такође врло интуитивне. Стрелице на горе и доле померају ћелију у оквиру радне свеске, а дугме које нас највероватније асоцира на дугме за паузу на камери прекинуће извршавање програма.



Илустрација 10 - Изглед Џупитера

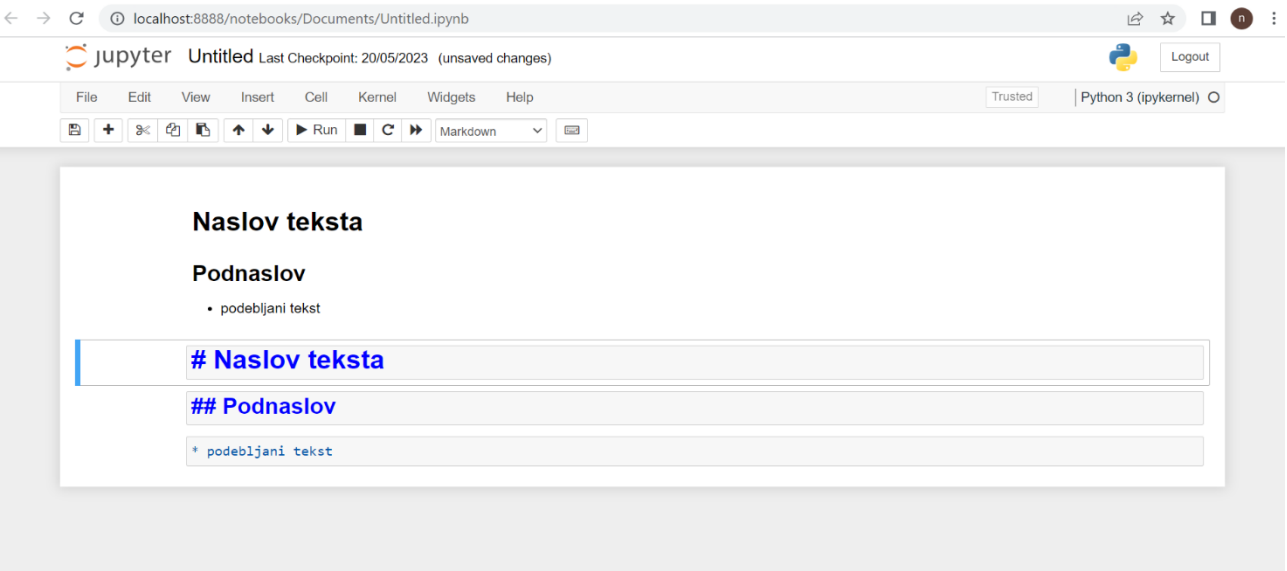
Прекидање извршавања програма почетницима у коришћењу Џупитера може бити збуњујуће. Поставља се питање зашто бисмо желели да прекинемо програм. Међутим, могућност прекида извршавања програма је неопходна, и то пре свега зато што се може десити да се програм предуго извршава или да је програм у бесконачној петљи. Уколико је програм у бесконачној петљи, он се никада неће до краја извршити, а заузимаће радну меморију рачунара. Прекидање програма не треба да нас брине, у сваком тренутку коришћењем команде за покретање (run) можемо поново да покренемо ћелије. Џупитер пружа могућност покретања целе радне свеске, покретања појединачне ћелије, као и покретања одређеног низа ћелија. Ако је у питању ћелија са кодом, он ће се покретањем ћелије извршити, уколико је у питању ћелија са текстом, он ће се исписати (Фондација Петља, н.д. ж). Некада је потребно означити ћелију као ћелију са кодом или текстом. На тај начин Џупитер разуме да смо можда желели код само да прикажемо, у свху учења на пример, а не и да извршимо. Сличан принцип постоји и у програмима са којима смо се већ сусрели, те је у Екселу ћелије које чувају податак попут ЈМБГ-а или броја телефона потребно означити као текстуалне ћелије, јер ће у супротном Ексел да их тумачи као нумеричке и „прогутаће“ сваку нулу која се налази на

почетној позицији. У Екселу се обележавање ћелије као текстуалне постиже врло једноставном стављањем апострофа испред текста који уносимо, или форматирањем ћелије. У Џупитеру ово постижемо бирањем да ли ће ћелија бити текстуалног типа (Markdown) или типа ћелије са кодом (Code). На фотографији горе можемо видети да на крајњој десној страни траке имамо падајући мени из код бирамо једну од ове две опције. У случају да у ћелију са кодом унесемо текст и покушамо да га покренемо, добићемо поруку о грешци.



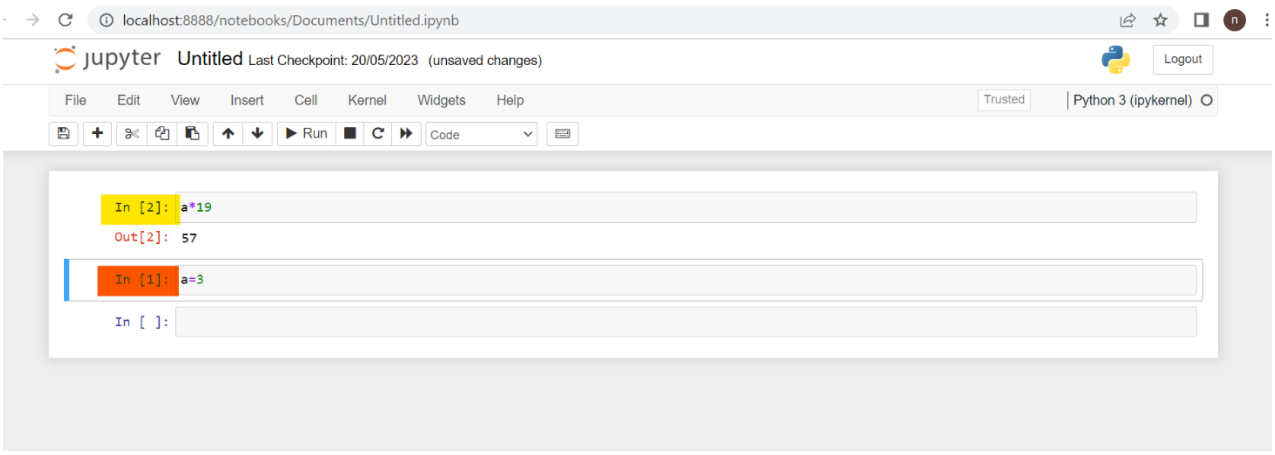
Илустрација 11 - Порука о грешци у синтакси

Ово се дешава зато што Џупитер покушава да сваку ћелију означену као ћелију са кодом изврши у одређеном програмском језику, те увиђа да наш текст није у складу са правилима, односно синтаксом језика. Осим правила програмског језика, чак и када желимо да испишемо текст у ћелијама и изаберемо опцију да је ћелија текстуалног типа постоје одређена правила којих се треба придржавати да би се текст приказао онако како је замишљено. На фотографији испод приказано је у како је текст исписан, а затим у ћелијама које нису активиране како је исти тај текст унет у Џупитер окружење.



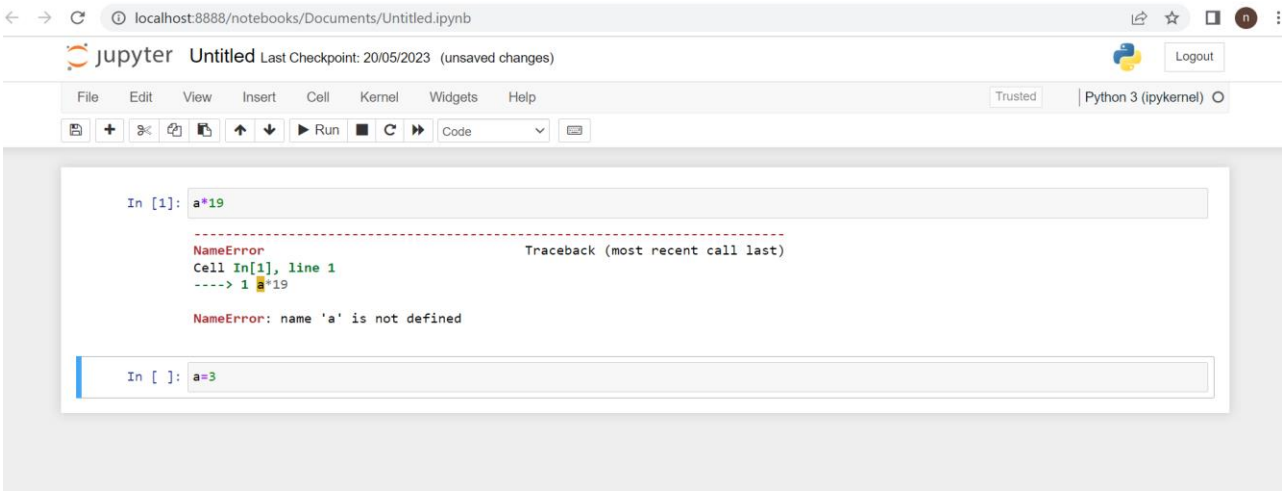
Илустрација 12 - Обележавање текста у Јупитеру

На истој фотографији примећујемо још нешто. Када је ћелија означена као ћелија текстуалног типа са њене леве стране нема угластих заграда које су иначе присутне. У тим угластим заграда испред ћелије у току извођења програма можемо видети број који означава редослед извршавања ћелија. Како код текстуалних ћелија нема извршења, тај број нам није потребан, па тиме нису ни угласте заграде. Овај редослед извршења означен бројем не мора се подударати са редоследом по коме су ћелије физички поређане у радној свесци. Ми можемо да унесемо у неку ћелију вредност за коју нам је потребно да је програм запамти. Рецимо да желимо да израчунамо неки израз. Можемо након ћелије у коју је унето на пример: a*19 (звездица у Пајтону означава множење) да унесемо a=3. Само уколико прво извршимо накнадно унету ћелију у којој је записано a=3, добићемо резултат 57. На фотографији испод наранџастим маркером је означена ћелија која је извршена прва, а жутим ћелија која је извршена након ње. Ћелију по ћелију извршавамо тако што се поставимо на ћелију коју желимо да извршимо и притиснемо дугме Ентер или кликнемо на команду за покретање ћелије (run), а затим исти поступак поновимо са ћелијом коју желимо накнадно да извршимо.



Илустрација 13 - Правилан редослед извршавања ћелија

У случају да смо у овом нашем примеру извршавали ћелије редом којим су позициониране у Џупитер радној свесци, добили бисмо поруку о грешци.



Илустрација 14 - Грешка у редоследу извршавања ћелија

Такође, уколико угласта заграда остане празна, то значи да се ћелија није извршила, што нас упућује да потражимо грешку у свом коду. Када имамо велики код који се из неког разлога не извршава, потребно је да прво потражимо прву ћелију у низу која се није извршила, односно поред које је угласта заграда остала празна. Велика је вероватноћа да ћемо пронаћи грешку у ћелији изнад те. Џупитер нам дакле помаже у проналаску грешака. У програмирању појава грешака је подразумејућа,

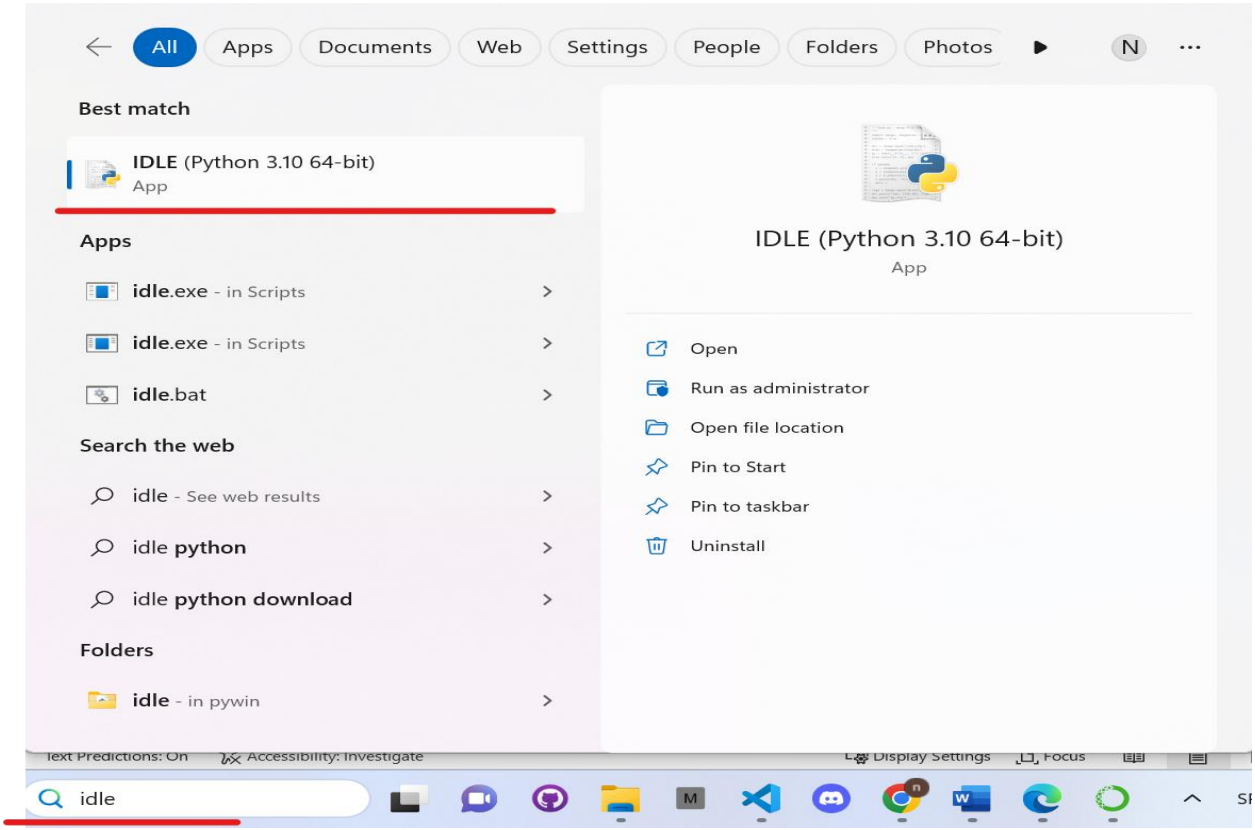
и не треба никако да буде обесхрабрујућа. А да бисмо, једном када пронађемо грешку, знали како да је исправимо, потребно нам је познавање логике и синтакте програмског језика у коме радимо, овога пута, Пајтона.

5. Основе Пајтона и Пандас библиотека

5.1. Основе Пајтона

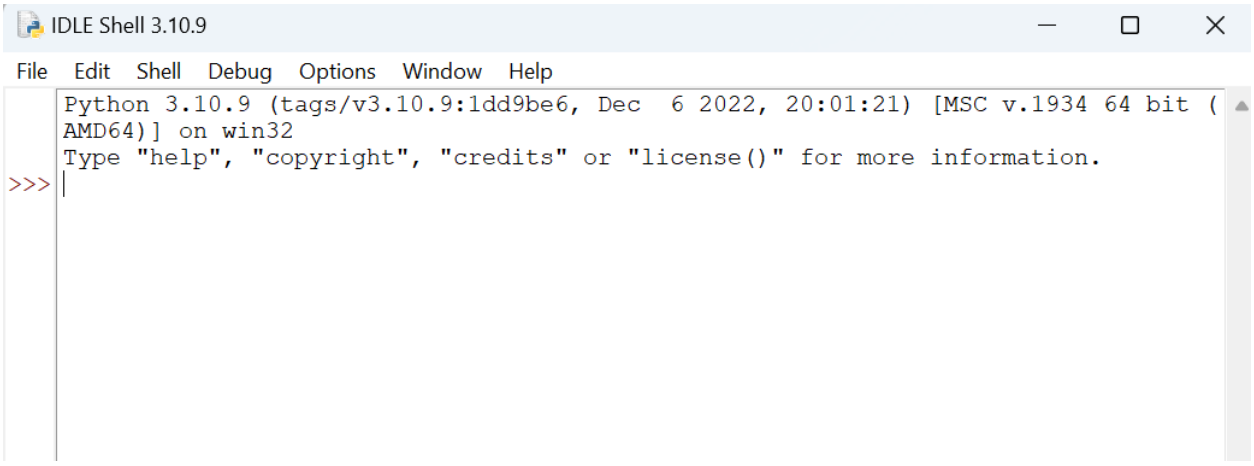
Пајтон је веома популаран програмски језик опште намене. Сматра се веома једноставним и лаким за учење те се, с једне стране, препоручује почетницима у програмирању, а са друге, често га користе они који се у свом послу не баве примарно програмирањем (Фондација Петља, н.д. з). Међутим, то никако не умањује значај Пајтона за програмирање на професионалном нивоу, поготово не зато што Пајтон омогућава брзу аутоматизацију посла, те је од значаја и софтверским инжењерима. Као и Џупитер, и он је у потпуности бесплатан за коришћење, а како је у питању производ отвореног кода око њега је настала заједница људи који константно раде на његовом развоју. За коришћење Пајтона, и извршавање Пајтон програма потребан нам је „тумач“ језика, односно Пајтон интерпретер (Python interpreter) (Фондација Петља, н.д. з). Окружење у коме се извршава овај интерпретер назива се „шкољка“ (Shell). Самом инсталацијом Пајтона, добијамо и Интегрисано окружење за развој и учење које је већ поменуто, а које у себи садржи шкољку. Чини се да су креатори Пајтона успели у томе да га учине доступним свима, без обзира на ниво техничког знања. То се види из чињенице да, као и код Џупитера, и код Пајтона постоји начин да његово коришћење буде могуће без инсталације више различитих програма, те је доступна и онлајн шкољка за коју је неопходно само да имамо приступ интернету. Овиме су све веће препреке за покретање Пајтона отклоњене.

Након што смо инсталирали Пајтон, шкољку можемо покренути из претраживача на рачунару.



Илустрација 15 - Покретање школьке из претраживача на рачунару

Када покренемо школьку, знаци >>> које видимо означавају да је окружење спремно да прими наредбу.



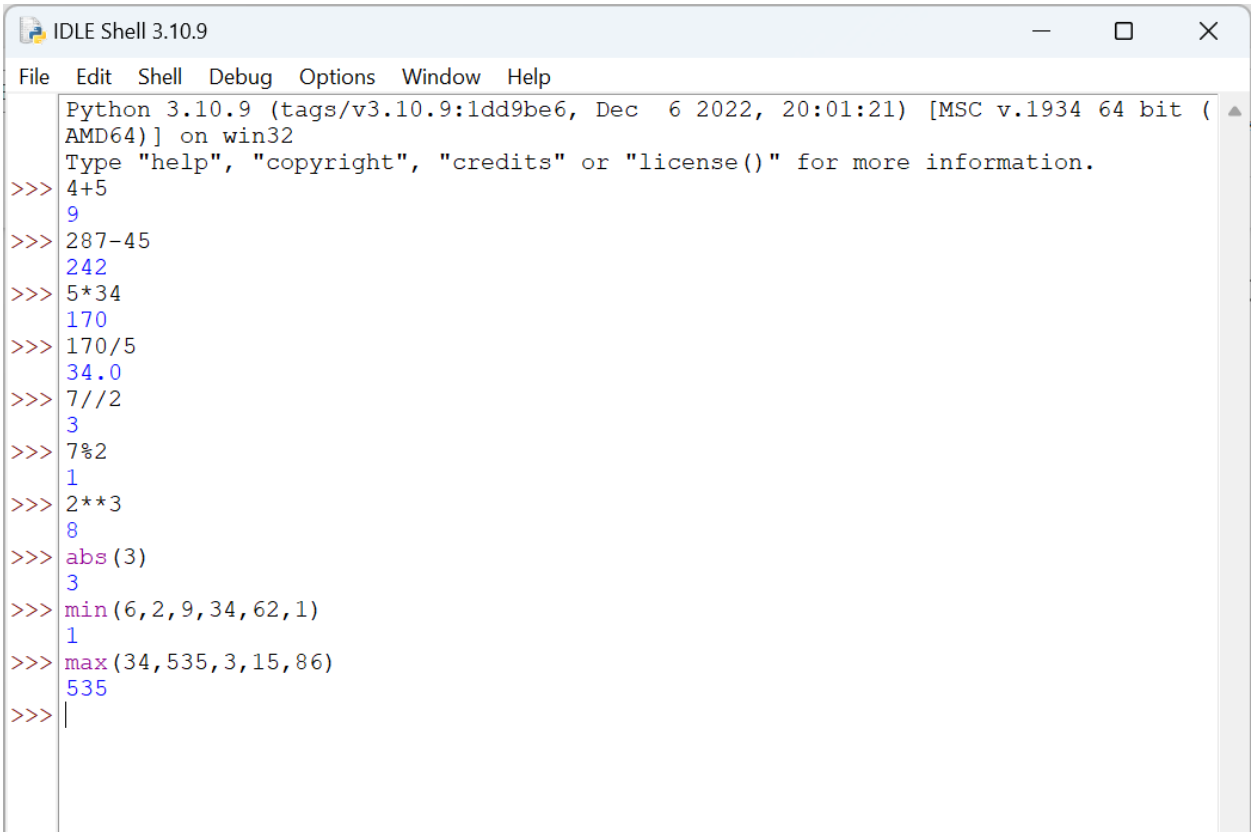
Илустрација 16 - Покренута школька

За прве кораке у коришћењу Пајтона препоручује се коришћење програма као калкуратора. На овај начин корисник врло једноставно може да се упозна са начином рада самог програма, окружења у коме се извршава и са основним синтаксичким правилима. Поступак је изузетно поједностављен: укуцамо израз и добијемо резултат. За основне рачунарске операције у Пајтону (као и у већини програмских језика) користе се следећи симболи:

- Сабирање +
- Одузимање –
- Множење *
- Дељење /

Осим основних операција, за обраду података, могу нам бити потребне још неке, попут:

- Дељења целог дела количника // (нпр израз 7//2=3)
- Остатка при дељењу целих бројева % (нпр 7%2=1)
- Степеновања ** (2**3=8)
- Функције минимум (min) која издваја најмању вредност из групе
- Функције максимум (max) која издваја највећу вредност из групе
- Функције абсолутне вредности (abs) која одређује колико је одређени број удаљен од нуле (Фондација Петља, н.д. и).



Илустрација 17 - Основне рачунске операције у Пајтону

На фотографији изнад јасно се види да иза знака >>> следи команда коју задајемо. Притиском дугмета Ентер, команда се извршава и резултат нам се приказује у реду испод плавом бојом. Омдах након што добијемо резултат рачунања програм је спреман за наредну команду. Када су у питању функције минимум, максимум и апсолутна вредност, након увођења функције потребно је вредности унети између заграда. Код функција минимум и максимум очекивано је унети низ вредности зато што те две функције из задатог низа издвајају најмању, односно највећу вредности. Хијерархија поменутих операција иста је као и у математици, односно множење и дељење имају предност у односу на сабирање и одузимање, вредности у заградама се рачунају пре вредности ван заграда и тако даље. Важност поменутих математичких операција у обради података је евидентна. Замислимо још једном да посао којим се бавимо подразумева организацију обуке или усавршавања

наставника или запослених из неке области. На нама је да након што је сам програм обуке припремљен, запослене позовемо на обуку и затим да пратимо колико запослених се пријавило, колико је међу њима колегиница, а колико колега, који запослени међу пријављенима има највише или најмање стажа у компанији, затим, који проценат пријављених је успешно савладао обуку и каква је структура те групе запослених, те међу онима који нису успешно савладали обуку, шта је заједнички именилац и тако даље. Све ове информације постају јако релевантне оног тренутка када треба да организујемо нову обуку. Важно је да знамо на које групе запослених у ком делу процеса обучавања треба да обратимо посебну пажњу. Јасно је да су редовне едукације у области на пример корпоративне безбедности, а поготово у области заштите података, јако важне зато што се новине у таквим и сличним областима дешавају често. Прикупљање ових информација, односно њихово извођење из сирових података подразумева коришћење основних математичких операција. Употреба програмског језика за израчунавање умногоме олакшава потребну статистичку анализу.

С обзиром на то да је вредности могуће рачунати и коришћењем обичног калкулатора, чак и рачунањем из главе, поставља се питање зашто је коришћење програмског језика препоручљивије. Наиме, програмски језик нам дозвољава да вредности које је потребно израчунати променимо без да мењамо формулу по којој их рачунамо. На тај начин, могуће је да једном припремљен програм за израчунавање које захтева један тип посла применимо на сваку ситуацију у којој нам је иста врста рачунања потребна. У примеру са усавршавањем запослених ово би значило да је само први пут када координишемо усавршавањем и из истог изводимо закључак о успешности потребно припремити програм, док је сваки следећи пут довољно у исти програм унети нове вредности. У вези са овиме, на фотографији испод дат је пример рачунања са „непознатим“. Свакој непознатој (a, b, c, d, h) додељена је вредност, након чега је исписана формула коришћењем непознатих. Програм дозвољава да променимо вредности које смо додели

непознатима, те да се поновним активирањем реда кода са формулом добије нови резултат. Због могућности мењања вредности, ове непознате у програмирају се називају променљивим.

```
>>> a=3
>>> b=76
>>> c=2
>>> d=9
>>> h=34
>>> a+(b**c)/d-h
610.7777777777778
>>>
```

Илустрација 18 - Рачунање са променљивима у Пајтону

Променљиве се користе да би програм могао да упамти вредности које им додељујемо. Пајтон спада у групу динамичких типизираних језика, што значи да променљиве добијају тип вредности коју садрже, а могу да прихвате вредност било ког типа (Фондација Петља, н.д. j).

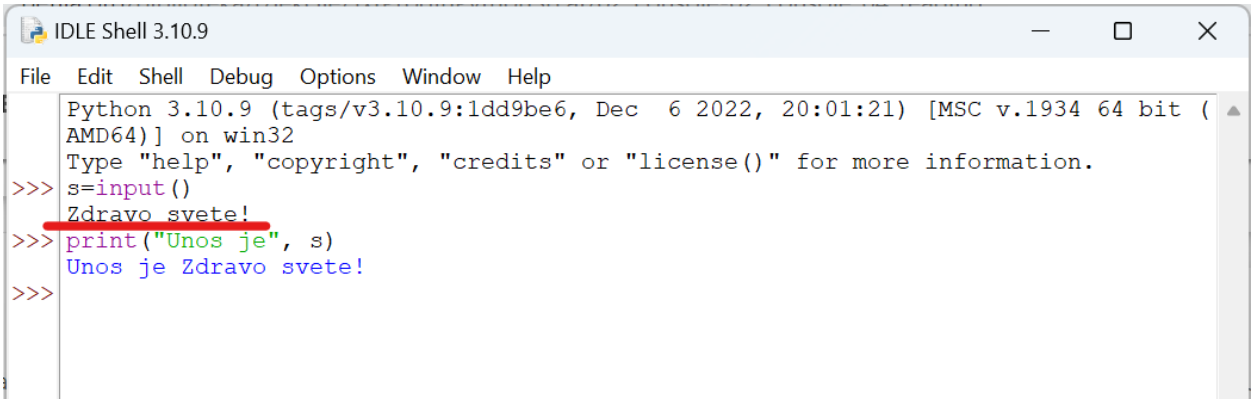
За прве кораке у Пајтону потребно је да знамо да постоји неколико типова вредности, односно података које можемо да користимо. За сваки од типа података који користимо у Пајтону постоји ознака којом обавештавамо рачунар о коме се типу података ради. Испод су наведени основни типови података, њихове ознаке, које морају бити на енглеском, као и примери:

- Цео број – int - 1234
- Реалан број – float – 3.141
- Текст (ниска карактера) – str – „Здраво свете“
- Логичка вредност – bool – True (означава да је вредност тачна) или False (означава да вредности није тачна)

У Пајтону целобројне променљиве могу да чувају врло велике и врло мале бројеве, док код реалних бројева постоје извесна ограничења у распону и прецизности у складу са стандардом који користе и други програмски језици (Фондација Петља, н.д. j). Када се променљивој први пут у програму додељује

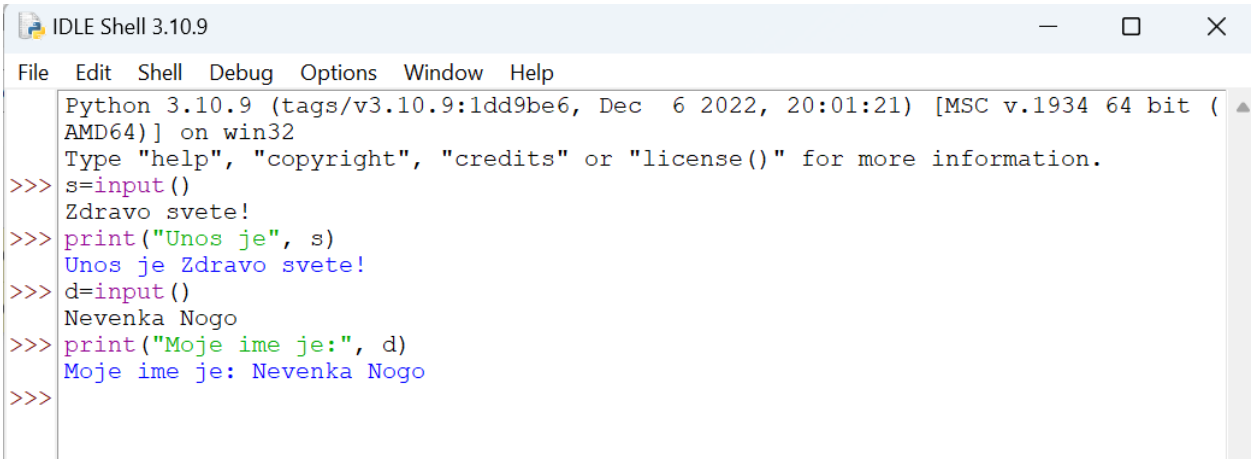
вредност, та променљива се дефинише. То значи да је променљивој придружен неки простор у меморији рачунара и да је у тај простор уписана вредност променљиве. Уколико нам је потребно да једну од низа променљивих фиксирамо, њој се могу доделити и неке конкретне вредности, које се тада називају константе (Фондација Петља, н.д. ј).

Осим прилагођавања програма коришћењем променљивих, још једна важна могућност коју Пајтон пружа јесте унос података у току рада програма. На примеру са усавршавањем запослених, ово значи да бисмо уместо мењања делова програма (на горњој фотографији a=3 бисмо морали да променимо у на пример a=16), програм могли од почетка да конципирамо тако да он од нас захтева да унесемо вредности. Ово се постиже функцијом за унос (input). Ова функција чека да корисник унесе на пример текст, а као резултат га исписује.



Илустрација 19 - Унос у Пајтону

На фотографији изнад црвеном бојом је подвучен ред у коме је од корисника очекивано да унесе текст за испис. У реду испод тог, програму је дата команда да унос корисника испише иза речи „Унос је“. Уместо речи „Унос је“ може писати све оно што се програму зада. На пример, на овај начин могу се тражити подаци о имену и презимену, адреси или броју телефона.



Илустрација 20 - Унос и испис у Пајтону

Као што је на двема претходним фотографијама приказано, податке је потребно исписати, односно приказати, што радимо помоћу функције за испис (print). Без обзира на тип података, сви се исписују истом функцијом. Овде је потребно скренути пажњу на један детаљ, текстуални подаци се у програмима записују под једноструким или двоструким наводницима, док се реални бројеви записују тако што се децимала одваја тачком, а не зарезом. Дакле, прати се енглески правопис (Фондација Петља, н.д. к).

Приказано је неколико основних функција и могућности Пајтона, њих укупно има значајно више. Осим многобројних функција које програм садржи у себи, његова важна карактеристика огледа се у томе што он омогућава кориснику да самостално дефинише своје функције. Дефинисање функција спречава понављање истих линија кода сваки пут када имамо неку ситуацију специфичну за оно што нам је потребно да програм уради. Овиме се проблем декомпозује на мање проблеме и доприноси бољој организацији кода. Декомпозиција проблема на мање проблеме је једна од најважнијих вештина како у програмирању тако и шири, и такође једна од вештина којом нас програмирање најдиректније учи. У овоме лежи суштина алгоритамског начина размишљања, оно није искључиво једнако

напредним дигиталним компетенцијама, већ доприноси развоју индивидуе као што то чини вежба концентрације или загонетка.

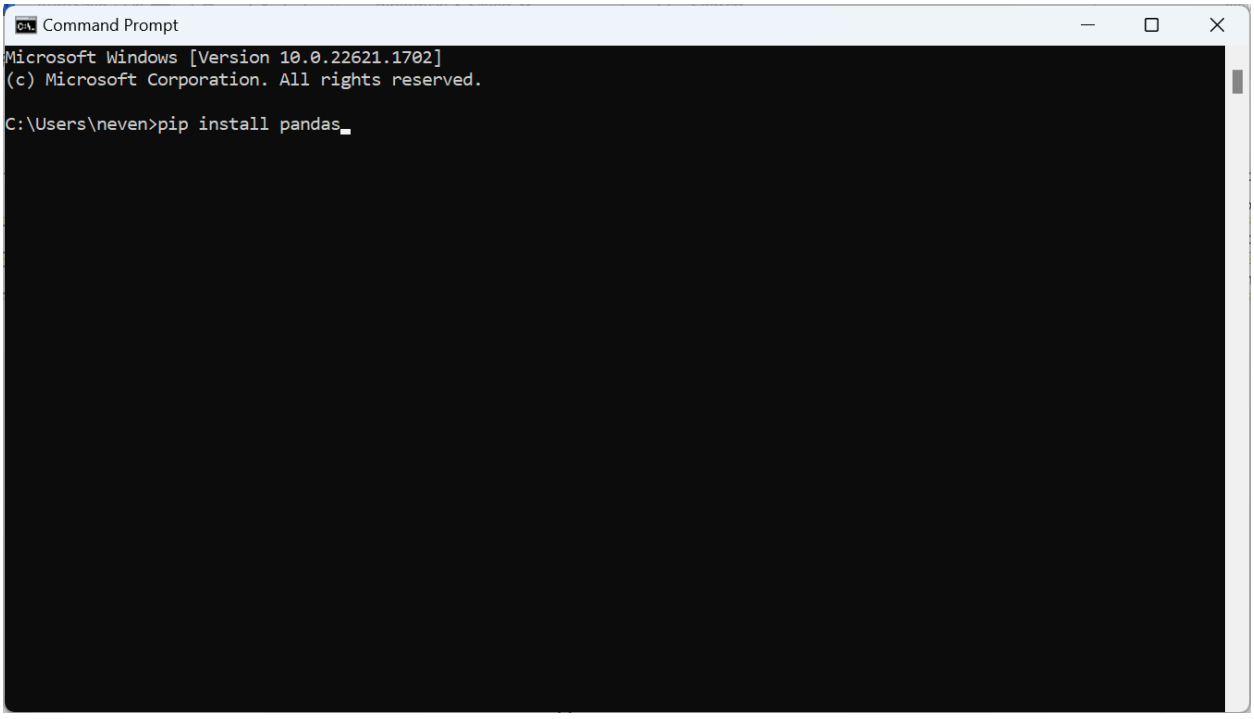
5.2. Пандас библиотека и њене могућности

Осим основних могућности рада у Пајтону које су приказане у претходном поглављу, на располагању је и мноштво такозваних библиотека. Библиотека представља својеврсну колекцију модула, функција и метода које пружају додатне могућности и алате за рад у Пајтону. Она омогућава коришћење функционалсности и решења која је неко други већ изградио (Geeks for geeks, н.д). Најчешће су решења која постоје у одређеној библиотеци у вези са једном сфером примере, те су тако неке од Пајтонових библиотека фокусиране на област обраде података, машинског учења, развоја веба, визуализације података и сличног. Једна од таквих је и Пандас библиотека која омогућава анализу, манипулацију и обраду структурираних података. Она је отвореног кода и подразумева машинско учење, што значи да садржи некакав сет инструкција намењен рачунару који му помаже да боље разуме податке. Представља ефикасан и једноставан алат за рад са различитим типовима података, односно формама у којима подаци могу бити „упаковани“ . То значи да Пандас може да чита податке из различитих извора попут табеле, текстуалног документа у коме су подаци одвојени запетама (CSV file – Comma-separated values file), базе података (на пример SQL), као и да може да их представља у различитим формама. Пружа јако пуно могућности за индексирање, филтрирање и издвајање података.

Затим, на основу различитих услова или критеријума, уз помоћ Пандаса, могуће је приступити појединачним вредностима, редовима или колонама наших података. Тим истим подацима могуће је манипулисати, те можемо да их додамо, изменимо или избришемо. Можемо да променимо облик податка, преименујемо редове или колоне, групишемо податке или над њима вршимо статистичку

анализу. На основу заједничких кључева, можемо да спојимо податке из више различитих извора. Податке можемо да комбинујемо, да уклонимо дупле вредности и обликујемо их за сопствене потребе. Све ове могућности Пандас библиотеке олакшавају различита мерења која се врше над подацима и помажу у извлачењу објективних закључака.

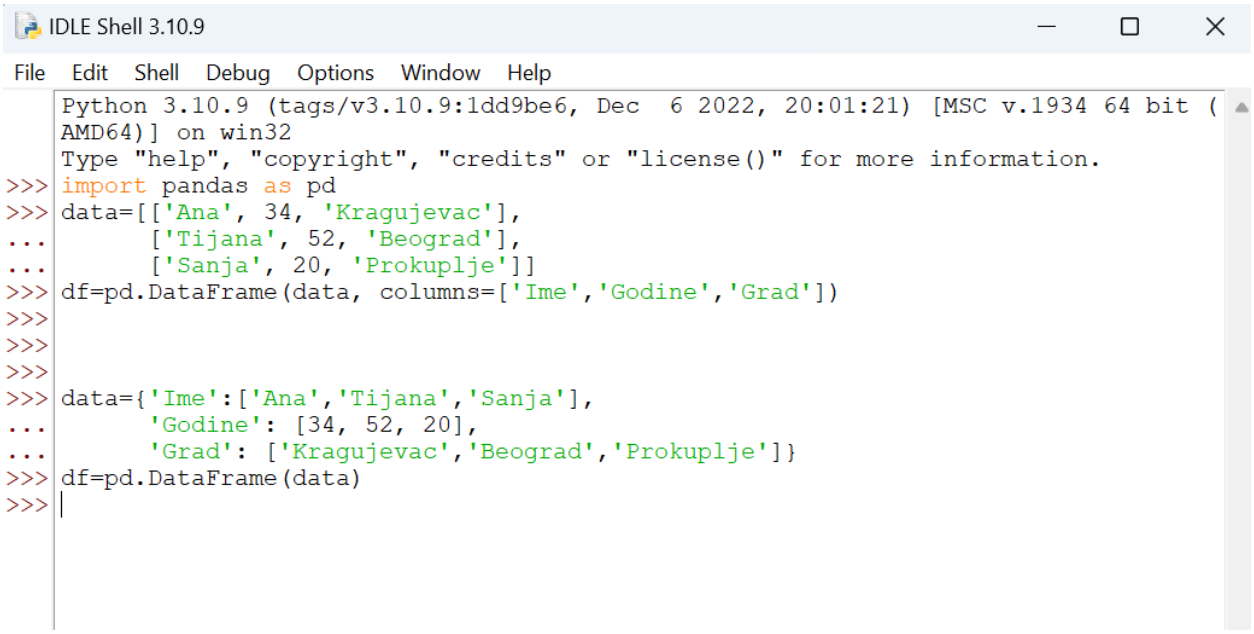
Сам рад у библиотекама не разликује се много од рада у Пајтону, међутим потребно је инсталирати одређену библиотеку, а затим је импортовати у код. Инсталација Пандаса након што постоји инсталиран Пајтон је врло једноставна. Потребно је отворити командну линију и у њу укуцати: „pip install pandas“. Добро је знати да је „pip install“ општа команда којом се инсталирају различите додатне компонентне које Пајтон подржава.



Илустрација 21 - Инсталација Пандас библиотеке

Након инсталације библиотеке, она се интегрише у код командом „import pandas“. На фотографији испод приказана је школька са командом „import pandas as pd“, овде се уводи и скраћеница „pd“ која је општеприхваћена скраћеница за Пандас и

она убрзава куцање кода јер ако импортујемо Пандас као „pd“, при сваком позивању на библиотеку биће довољно да укуцамо „pd“ уместо „pandas“, што се такође може видети на фотографији. Да бисмо користили могућности библиотеке и рачунар разумео да желимо да користимо могућности библиотеке, а не неке друге могућности, потребно је да се за сваку функционалност коју желимо позовемо на библиотеку користећи „pandas.funkcija“ односно с обзиром на то да смо већ увели скраћеницу користећи „pd.funkcija“.



Илустрација 22 - Интеграција библиотеке Пандас у код и креирање табеле у библиотеци Пандас

Након инсталирања и импортовања Пандаса у код, важно је разумети како се у овој библиотеци тумаче подаци. Основна структура података у Пандасу је „Оквир података“ (Data Frame). Он представља дводимензионалну табелу са редовима и колонама. На фотографији изнад приказана су два једноставна начина за креирање табеле у Пандасу. Већ на први поглед је јасно да се разликују по томе да ли се подаци уносе према реду или према колони. Као што је поменуто у поглављу о Џупитеру, Пандас нам такође омогућава да податке пребацујемо из

Оквира података у Ексел табелу или обрнуто. Међутим, пре коришћења функционалности за рад са Ексел фајловима, односно функционалности која ће податке које смо ручно унели у Пандас пребацити у Ексел потребно је инсталирати на исти начин као и Пандас још једну библиотеку под називом „openpyxl". Оквир података је основа рада са подацима коришћењем Пандаса. Он представља начин да из табеларно упамћених података можемо да извучемо више него што бисмо могли простим коришћењем Екселa. Олакшава рад са подацима који су размештени у више различитих табела, убрзава манипулацију подацима и олакшава проналазак информација које су нам потребне.

Даљи рад у Пандасу, као и у Пајтону уопште подразумева покушаје и погрешке. Програмирање се учи програмирањем. Као и код сваке друге вештине, за напредак је потребан одређени број понављања, вежбе и наравно мотивације.

6. Закључак

Алгоритамски начин размишљања данас се позиционира као кључна вештина за решавање проблема и ефикасно обављање савремених послова. Подразумева структурисан приступ размишљању, идентификацију и разумевање проблема кроз рашчлањење на мање сегменте проблема и њихово решавање корак по корак. Такође, алгоритамски начин размишљања доприноси бољем разумевању начина на који технологија функционише, што нам омогућава да експлоатишемо могућности које технологија пружа. У времену у коме је за ефикасно обављање сваког посла, па и оних послова из друштвено-хуманистичке сфере делатности неопходно бити дигитално писмен, и у коме, с друге стране, на тржиште рада излазе генерације које су чак и кроз формални систем образовања училе да програмирају, да би се остао конкурентан општа дигитална писменост треба да се подигне на један ниво више. Неколико различитих области рачунарских наука могу представљати тај виши ниво, међутим чини се да је спој информатике и рада са подацима за сферу послова безбедности, или шире друштвено-хуманистичких послова, најочигледнији избор. Подаци и њихово разумевање у контексту, односно њихово претварање у информацију која има своју вредност, одлика су великог броја савремених послова. Насупрот томе, неадекватно коришћење података услед презасићености подацима или непостојање алгоритамског приступа размишљању може довести до неефикасности, губитка конкурентске предности и пропуштања пословних прилика. Улагање у развој сопствених дигиталних компетенција и подизање свести о значају дигиталних компетенција је један од могућих начина функционисања на тржишту које се брзо мења.

Чини се да је најједноставнији начин да се компетенције подигну на виши ниво да се пође од нечега што је добро познато и свакодневно, попут Екселa, а да се након тога приступи коришћењу окружења попут Џупитера, где ће се додатно разумети да за куцање кода и одређену врсту програмирања нису неопходни

генијалност, предодређеност, нити посебан таленат. У вези са идејом да програмирање јесте за свакога ко је довољно мотивисан да научи да програмира је и схватање да је Пајтон веома поједностављен програмски језик, односно прецизније, да је његова синтакса ослобођења компликованих формалности. Стога се чини да је Пајтон природни први корак након откривања могућности које пружају окружења за извршавање кода. Исто тако, након савладавања основа Пајтона, отворен је пут ка различитим областима програмирања, само је потребно бити помало радознао.

Литература

British Council. (н.д.). *About 21st Century Schools.* Преузето 10. маја 2023, са https://www.britishcouncil.rs/en/programmes/education/21st-century-schools/about

Влада Републике Србије. (31. август 2017). *Информатика и рачунарство обавезан предмет за ученике петог разреда.* Преузето 10. маја 2023, са https://www.srbija.gov.rs/vest/300384/informatika-i-racunarstvo-obavezan-predmet-za-ucenike-petog-razreda.php

Geeks for geeks. (н.д.). *Libraries in Python.* Преузето 25. маја 2023, са https://www.geeksforgeeks.org/libraries-in-python/

DIGITAL SCHOOLHOUSE. (н.д.). *Free computing workshops.* Преузето 30. маја 2023, са https://www.digitalschoolhouse.org.uk/computing-at-home

Jupyter. (н.д.). *Jupyter – Free software, open standards, and web services for interactive computing across all programming languages.* Преузето 02. јуна 2023, са https://jupyter.org/

Codecademy. (н.д.). *What is Programming?.* Преузето 30. маја 2023, са https://www.codecademy.com/article/what-is-programming

Мандић, Г., Путник, Н. и Милошевић, М. (2017). *Заштита података и социјални инжењеринг – правни, организациони и безбедносни аспекти.* Београд: Универзитет у Београду – Факултет Безбедности

Microsoft Development Center Serbia (MDCS). (н.д.). *MDCS News.* Преузето 27. маја 2023, са https://www.microsoft.com/en-rs/mdcs/mdcs-news

McFedries, P. (2022). *Microsoft Excel Data Analysis For Dummies* (5. изд.). Wiley, Преузето 01. јуна 2023, са https://www.wiley.com/en-ie/Excel+Data+Analysis+For+Dummies%2C+5th+Edition-p-9781119844471

Nordeus fondacija. (н.д.). *Zašto postojimo.* Преузето 27. маја 2023, са https://nordeusfondacija.org/

Serbian Gamming Association. (26. мај 2023.). *FOR THE WIN 2023_Final conference schedule.* Преузето 27. маја 2023, са https://drive.google.com/file/d/11znMp9cVVhSOHTjJu2Mo0m4Nv6hvI0jO/view?pli=1

UNPD. (28. Април 2023.). *Devojke treba da budu kreatorke u digitalnom svetu.* Преузето 30. маја 2023, са https://www.undp.org/sr/serbia/announcements/devojke-treba-da-budu-kreatorke-u-digitalnom-svetu

Фондација Петља. (н.д.) а. *Шта су то програми и наредбе.* Преузето 30. маја 2023, са https://petlja.org/biblioteka/r/HoC/Naredbe

Фондација Петља. (н.д.) б. *Програмирање у Скречу 2.0, приручник за пети разред: Алгоритамски начин размишљања.* Преузето 30. маја 2023, са https://petlja.org/biblioteka/r/lekcije/scratch-support/algoritamski-nacin-razmisljanja

Фондација Петља. (н.д.) в. *Програмирање у Скречу за пети разред: Алгоритамски начин размишљања.* Преузето 30. маја 2023, са https://petlja.org/kurs/351/2/5494

Фондација Петља. (н.д.) г. *Буди Data Driven: Анализа и визуализација података – Говори статистички да те цео свет разуме.* Преузето 30. маја 2023, са https://petlja.org/kurs/6173/7/3686

Фондација Петља. (н.д.) д. *Анализа података у Џупитеру и технике програмирања за други разред гимназије: Џупитер и Ексел.* Преузето 03. јуна 2023, са https://petlja.org/kurs/478/10/6172

Фондација Петља. (н.д.) ђ. *Анализа података у Џупитеру и технике програмирања за други разред гимназије: Покретање Џупитер радних свески.* Преузето 03. јуна 2023, са https://petlja.org/kurs/478/10/6141

Фондација Петља. (н.д.) ж. *Буди Data Driven: Анализа и визуализација података – Кратак увод у Џупитер окружење.* Преузето 04. јуна 2023, са https://petlja.org/kurs/6173/2/3651

Фондација Петља. (н.д.) з. *Програмирање на Пајтону, приручник: Основе програмирања у Пајтону – Увод у Пајтон.* Преузето 30. маја 2023, са https://petlja.org/biblioteka/r/lekcije/TxtProgInPythonSrLat/02_console-02_console_01_about_python

Фондација Петља. (н.д.) и. *Решавање алгоритамских задатака на Пајтону (ниво А1): Израчунавања – Рачунање са целим бројевима.* Преузето 30. маја 2023, са https://petlja.org/kurs/9732/3/7458

Фондација Петља. (н.д.) ј. *Методичка збирка задатака из основа програмирања (Пајтон): Аритметика.* Преузето 01. јуна 2023, са https://petlja.org/biblioteka/r/Zbirka-python/01%20Aritmetika

Фондација Петља. (н.д.) к. *Решавање алгоритамских задатака на Пајтону (ниво А1): Увод - Прве наредбе*. Преузето 30. маја 2023, са https://petlja.org/kurs/9732/2/7452