* Potreba pridania balickov pomocou prikazu v cmd(ako spravca): **py -m pip install requests**
* Inštalácia virtual enviromentu: **py -m pip install virtualenv**
* Vytvorím priecinok, v ktorom budu vsetky pouzite baliky atd, je to dobre ak rozne verzie pythonu instalujem, takto si jednoducho aktivujem danu verziu ktoru potrebujem a budem mat vsetko na pracu pripravene(treba byt ako spravca,inak to nepojde): **py -m virtualenv env1(alebo “ucivo”)**
* v danom balicku si otvorim bat subor s activate, takto dany priecinok aktivujem:

**env1/Script/activate** (ak nefunguje tak pridat cd na zaciatok)

* pokial sa nechcem starat o aktivavovanie a prepinanie priecinkov, je mozne nainstalovat wrapper, ktory sa o to postara za nas: **py -m pip install virtualenvwrapper**
* Ukaze vsetky nainstalovane baliky v danom prostredi, ktory mam aktivovany: **pip freeze**
* Zhrnutie:
* Použitie:
* pip <príkaz> [extra možnosti]
* Príkazy:
* install Nainštaluje balík
* uninstall Odinštaluje balík
* list Vypíše zoznam nainštalovaných balíkov
* freeze Vypíše zoznam nainštalovaných balíkov vo formáte pre pip
* show Vypíše informácie o balíčku
* search Prehladávanie PyPI či sa tam vyskytuje želaný balík
* help Vypíše pomoc pre <príkaz>
* pip install django # nainštaluje django
* pip show django # ukáže podrobnosti o balíčku django
* pip uninstall django # odinštaluje \*\*iba\*\* django balíček
* pip freeze > requirements.txt # uloží všetky nainštalované balíky do súboru
* pip install -r requirements.txt # nainštaluje všetky balíčky definované v pip requirements formáte
* virtualenvwrapper je množina pomocných skriptov na zjednodušenie práce
* s virtualenv.
* Niektoré vybrané príkazy:
* lssitepackages: vylistuje nainštalované balíky v site-packages
* lsvirtualenv: vylistuje všetky virtuálne prostredia
* mkproject: vytvorí nový projekt, jeho priečinok a k nemu prislúchajúci virtualenv
* mktmpenv: vytvorí dočasný virtualenv
* mkvirtualenv: vytvorí nový virtualenv v domácom priečinku používateľa
* rmvirtualenv: vymaže virtualenv
* wipeenv: vymaže všetky balíčky nainštalované v danom virtualenv
* workon: zmení aktívny virtualenv
* Python package index: vyhladavanie **PYPI** (zoznam a informacie o balikoch), spustenie pomocou **pip install**
* **py -m pip install ipython(cez admina)** -instalacia ipython, nasledne spustenie prikazom **ipython**

# Dátové typy:

* **type**(3) -prikaz pre zistenie datoveho typu(vysledok int)
* **help**(int) - pomocka pre zistenie argumentu ,typu atd, klavesa Q vracia do konzoly

# Prikazy:

* meno = **input**(“Ako sa volas?“) - tento prikaz ti vypise text v zatvorke a do premennej „meno“ ti ulozi text z konzoly ako String, konverziu treba spravit rucne pomocou castingu
* **int**(vek)\*2 - **casting** na integer z datoveho typu string, nasledne vynasobenie cislom
* **str(**3.14**) -** konverzia na string
* int ("FF", **base**=16) - konverzia FF zo 16-kovej sustavy do integeru
* **print**(“Ahoj, ako sa mas“) - vypis textu
* **bin(4) -** prevod do binarneho tvaru, rovnako prikaz aj hex
* **abs(-5)** - absolutna hodnota
* **round(3.5) -** zaokruhlovanie, pozor, ine je vo verzii 3, ako v starsej, v python 3 sa zaokruhluje na najblizsie parne cislo, 6.5 zaokruhli na 6, 7.5 zaokruhli na 8
* **A, B =**  B, A- tento prikaz vymeni hodnoty A a B medzi sebou
* **Viacriadkové stringy** pomocou “““ troch uvodzoviek na zaciatok aj na koniec
* **len**(“nejaky retazec“) - vrati dlzku retazca
* print(“Ahoj {0}”.**format**(meno)) - dosadzanie do stringu pomocou zatvoriek {}
* “{**:<**30}“.format(“left aligned”) - tento prikaz **zarovna** text na 30 znakov **dolava**
* “{**:^**30}“.format(“left aligned”) - **zarovnanie** textu “left aligned” na **stred**
* “{**:\*<**30}“.format(“left aligned”) - prazdne znaky nahradi hviezdickami \*
* ‘{**:\_**}‘.format(123582693) - formatovanie cisla, tisicky **oddeli** **podtrznikom**
* “desiatkova = {0**:d**}; hex = {0**:#x**}; osm = {0**:#o**}; bin = {0**:#b**}“.format(42) - cislo 42 zapise do roznych sustav

# Grafika:

* Spustenie prikazov pre zaciatok grafickeho prostredia v idle3(okno kde je napis Shell):
  + - 1. import **tkinter**
      2. canvas = tkinter.**Canvas**(width=640, height=480) - vyska a sirka sa nemusi udavat
      3. canvas.**pack**()
* canvas.create\_**line**(x1,y1,x2,y2...., fill=“red“, width=3) - kreslenie ciary, ciara bude cervena s hrubkou 3, farbu sa da zadavat aj v tvare “#00FF00“
* canvas.create\_**rectangle**(x1,y1,x2,y2,**fill**=“farba“,**outline**=“farba“,**width**=hrubka) - obdlznik urcitej farby a farby obrisu, povinne su len suradnice, dalsie parametre su doplnkove
* canvas.create\_**oval**(x1,y1,x2,y2) - elipsa
* canvas.create\_**polygon**(x1,y1,x2,y2...) - uzavrety viacuholnik
* canvas.create\_**text**(stredX,stredY,text=“lubovolny text“, font=“Monaco 30“, fill=“green“ )- vlozenie textu do grafickeho prostredia
* obr = tkinter.**PhotoImage**(file=“Arsenal.png“) - prilepenie obrazka, musi byt png alebo gif
* canvas.create\_**image**(320, 240, image = obr) - vykreslenie obrazka
* canvas.**config**(width = 900, bg=“gray“) - dodatocna konfiguracia widgetu, bg je pozadie
* canvas["bg"]="white"- iny sposob zapisu
* canvas.**itemconfig**(2, fill = “yellow“) - configuracia podla ID (2), toto ID napise na obrazovku po vytvoreni objektu
* canvas.**move**(2, -4, 0) - posunutie objektu s ID=2 o x=-4 a y=0
* canvas.**coords**(1, 100, 100) - objekt s ID=1 premiestni na suradnice [100, 100]
* canvas.**delete**(1) - vymazanie objektu s ID=1